Fortify Audit Workbench

Developer Workbook

WebGoat-main

Table of Contents

© Copyright [2008-2024] Micro Focus or one of its affiliates.

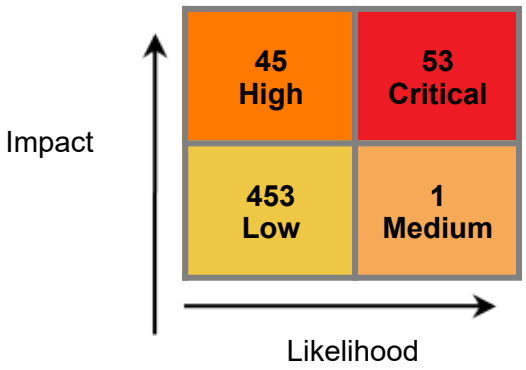
Executive Summary

This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the WebGoat-main project audit. The information contained in this workbook is targeted at project managers and developers.

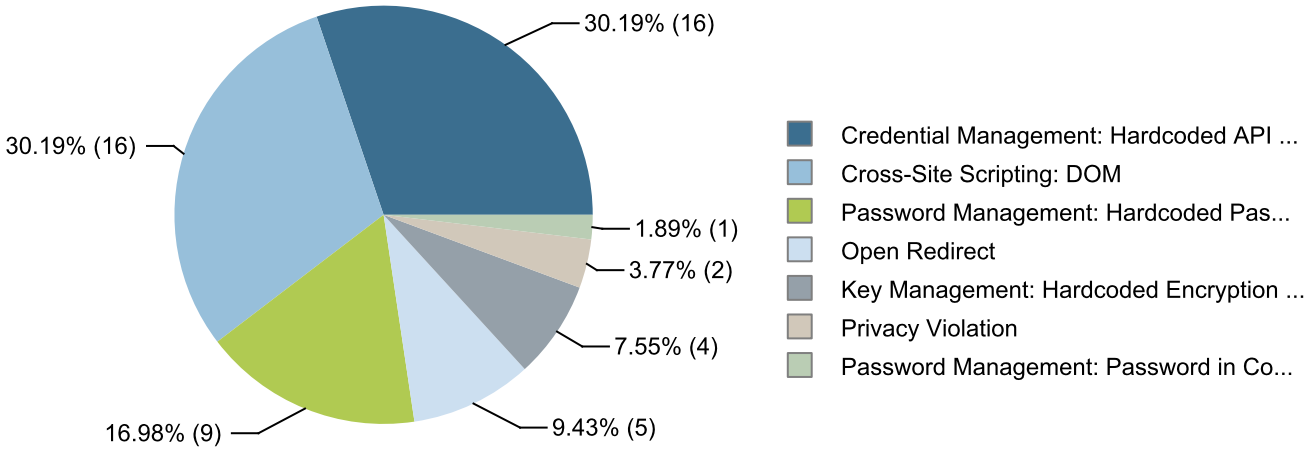
This section provides an overview of the issues uncovered during analysis.

Project Name:	WebGoat-main
Project Version:	
SCA:	Results Present
WebInspect:	Results Not Present
WebInspect Agent:	Results Not Present
Other:	Results Not Present

Issues by Priority



Top Ten Critical Categories



© Copyright [2008-2024] Micro Focus or one of its affiliates.

Project Description

This section provides an overview of the Fortify scan engines used for this project, as well as the project meta-information.

SCA

Date of Last Analysis:	Nov 2, 2024 10:05 AM	**Engine Version:**	22.2.1.0004
Host Name:	hunter	**Certification:**	VALID
Number of Files:	651	**Lines of Code:**	37,894

Rulepack Name	Rulepack Version
Fortify Secure Coding Rules, Community, Cloud	2022.4.0.0009
Fortify Secure Coding Rules, Community, Universal	2022.4.0.0009
Fortify Secure Coding Rules, Core, Android	2022.4.0.0009
Fortify Secure Coding Rules, Core, Annotations	2022.4.0.0009
Fortify Secure Coding Rules, Core Cloud	2022.4.0.0009
Fortify Secure Coding Rules, Core, Java	2022.4.0.0009
Fortify Secure Coding Rules, Core, JavaScript	2022.4.0.0009
Fortify Secure Coding Rules, Core, SQL	2022.4.0.0009
Fortify Secure Coding Rules, Core, Universal	2022.4.0.0009
Fortify Secure Coding Rules, Extended, Configuration	2022.4.0.0009
Fortify Secure Coding Rules, Extended, Content	2022.4.0.0009
Fortify Secure Coding Rules, Extended, Java	2022.4.0.0009
Fortify Secure Coding Rules, Extended, JavaScript	2022.4.0.0009
Fortify Secure Coding Rules, Extended, JSP	2022.4.0.0009
Fortify Secure Coding Rules, Extended, SQL	2022.4.0.0009

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Issue Breakdown by Fortify Categories

The following table depicts a summary of all issues grouped vertically by Fortify Category. For each category, the total number of issues is shown by Fortify Priority Order, including information about the number of audited issues.

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
Code Correctness: Byte Array to String Conversion	0	0	0	0 / 10	0 / 10
Code Correctness: Constructor Invokes Overridable Function	0	0	0	0 / 6	0 / 6
Code Correctness: Erroneous String Compare	0	0	0	0 / 1	0 / 1
Code Correctness: Incorrect Serializable Method Signature	0	0	0	0 / 1	0 / 1
Code Correctness: readObject() Invokes Overridable Function	0	0	0	0 / 1	0 / 1
Command Injection	0	0	0	0 / 1	0 / 1
Credential Management: Hardcoded API Credentials	0 / 16	0 / 1	0	0	0 / 17
Cross-Site Request Forgery	0	0	0	0 / 31	0 / 31
Cross-Site Scripting: DOM	0 / 16	0	0	0	0 / 16
Cross-Site Scripting: Self	0	0	0	0 / 2	0 / 2
Dead Code: Unused Field	0	0	0	0 / 9	0 / 9
Dead Code: Unused Method	0	0	0	0 / 27	0 / 27
Denial of Service	0	0	0	0 / 1	0 / 1
Dockerfile Misconfiguration: Default User Privilege	0	0 / 1	0	0	0 / 1
Dockerfile Misconfiguration: Dependency Confusion	0	0 / 1	0	0	0 / 1
HTML5: Overly Permissive Message Posting Policy	0	0	0	0 / 1	0 / 1
Insecure Randomness	0	0 / 14	0	0	0 / 14
Insecure Randomness: User-Controlled Seed	0	0 / 1	0	0	0 / 1
J2EE Bad Practices: JVM Termination	0	0	0	0 / 3	0 / 3
J2EE Bad Practices: Leftover Debug Code	0	0	0	0 / 4	0 / 4
Key Management: Hardcoded Encryption Key	0 / 4	0 / 2	0	0	0 / 6
Log Forging	0	0	0	0 / 3	0 / 3
Missing Check against Null	0	0	0	0 / 3	0 / 3
Often Misused: Authentication	0	0	0	0 / 1	0 / 1
Open Redirect	0 / 5	0	0	0	0 / 5
Password Management: Empty Password	0	0 / 1	0	0	0 / 1
Password Management: Hardcoded Password	0 / 9	0 / 8	0	0	0 / 17
Password Management: Password in Comment	0	0	0	0 / 7	0 / 7
Password Management: Password in Configuration File	0 / 1	0 / 1	0	0	0 / 2
Path Manipulation	0	0 / 3	0	0	0 / 3
Path Manipulation: Zip Entry Overwrite	0	0	0 / 1	0	0 / 1
Poor Error Handling: Empty Catch Block	0	0	0	0 / 4	0 / 4
Poor Error Handling: Overly Broad Catch	0	0	0	0 / 35	0 / 35
Poor Error Handling: Overly Broad Throws	0	0	0	0 / 217	0 / 217
Poor Logging Practice: Use of a System Output Stream	0	0	0	0 / 9	0 / 9
Poor Style: Confusing Naming	0	0	0	0 / 5	0 / 5
Poor Style: Identifier Contains Dollar Symbol ($)	0	0	0	0 / 1	0 / 1
Poor Style: Value Never Read	0	0	0	0 / 21	0 / 21
Portability Flaw: Locale Dependent Comparison	0	0	0	0 / 6	0 / 6
Privacy Violation	0 / 2	0	0	0	0 / 2
Privacy Violation: Autocomplete	0	0 / 2	0	0	0 / 2
Race Condition	0	0 / 2	0	0	0 / 2
Resource Injection	0	0	0	0 / 1	0 / 1
SQL Injection	0	0	0	0 / 16	0 / 16

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
System Information Leak	0	0	0	0 / 8	0 / 8
System Information Leak: External	0	0	0	0 / 2	0 / 2
System Information Leak: Internal	0	0	0	0 / 10	0 / 10
Unchecked Return Value	0	0	0	0 / 4	0 / 4
Unreleased Resource: Database	0	0 / 2	0	0	0 / 2
Unreleased Resource: Files	0	0 / 1	0	0	0 / 1
Unreleased Resource: Streams	0	0 / 4	0	0	0 / 4
Weak Cryptographic Hash	0	0	0	0 / 1	0 / 1
XML Entity Expansion Injection	0	0	0	0 / 1	0 / 1
XML External Entity Injection	0	0 / 1	0	0	0 / 1

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Results Outline

Code Correctness: Byte Array to String Conversion (10 issues)

Abstract

Converting a byte array into a `String` may lead to data loss.

Explanation

When data from a byte array is converted into a `String`, it is unspecified what will happen to any data that is outside of the applicable character set. This can lead to data being lost, or a decrease in the level of security when binary data is needed to ensure proper security measures are followed. **Example 1:** The following code converts data into a String in order to create a hash.

```
...
FileInputStream fis = new FileInputStream(myFile);
byte[] byteArr = byte[BUFSIZE];
...
int count = fis.read(byteArr);
...
String fileString = new String(byteArr);
String fileSHA256Hex = DigestUtils.sha256Hex(fileString);
// use fileSHA256Hex to validate file
...
```

Assuming the size of the file is less than `BUFSIZE`, this works fine as long as the information in `myFile` is encoded the same as the default character set, however if it's using a different encoding, or is a binary file, it will lose information. This in turn will cause the resulting SHA hash to be less reliable, and could mean it's far easier to cause collisions, especially if any data outside of the default character set is represented by the same value, such as a question mark.

Recommendation

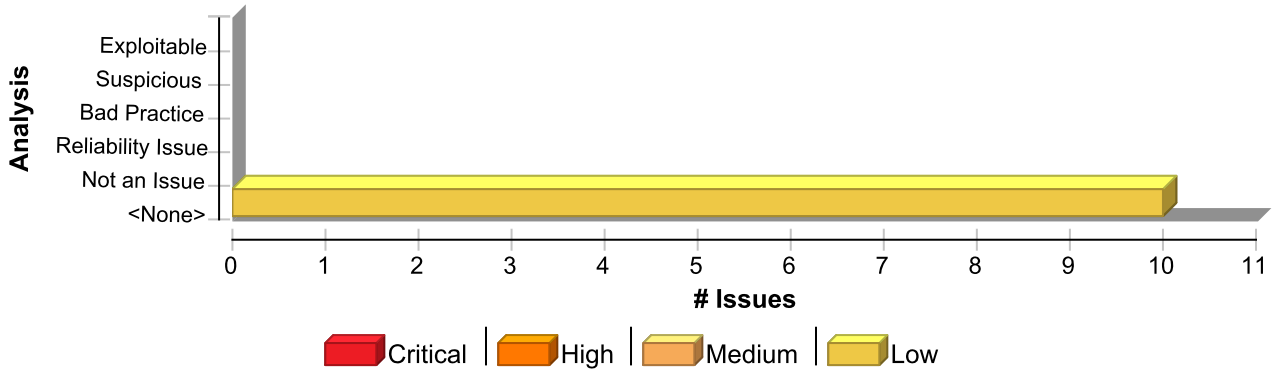
Generally speaking, a byte array potentially containing noncharacter data should never be converted into a `String` object as it may break functionality, but in some cases this can cause much larger security concerns. In a lot of cases there is no need to actually convert a byte array into a String, but if there is a specific reason to be able to create a `String` object from binary data, it must first be encoded in a way such that it will fit into the default character set. **Example 2:** The following uses a different variant of the API in `Example 1` to prevent any validation problems.

```
...
FileInputStream fis = new FileInputStream(myFile);
byte[] byteArr = byte[BUFSIZE];
...
int count = fis.read(byteArr);
...
byte[] fileSHA256 = DigestUtils.sha256(byteArr);
// use fileSHA256 to validate file, comparing hash byte-by-byte.
...
```

In this case, it is straightforward to rectify, since this API has overloaded variants including one that accepts a byte array, and this could be simplified even further by using another overloaded variant of `DigestUtils.sha256()` that accepts a `FileInputStream` object as its argument. Other scenarios may need careful consideration as to whether it's possible that the byte array could contain data outside of the character set, and further refactoring may be required.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Byte Array to String Conversion	10	0	0	10
Total	**10**	**0**	**0**	**10**

Code Correctness: Byte Array to String Conversion — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 60 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

Sink: String()
Enclosing Method: checkAssignment2()
File: src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java:60
Taint Flags:

```
57  .extract()
58  .asString();
59  basicEncoding = basicEncoding.substring("Authorization: Basic ".length());
60  String decodedString = new String(Base64.getDecoder().decode(basicEncoding.getBytes()));
61  String answer_user = decodedString.split(":")[0];
62  String answer_pwd = decodedString.split(":")[1];
63  Map<String, Object> params = new HashMap<>();
```

src/it/java/org/owasp/webgoat/ChallengeIntegrationTest.java, line 31 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Byte Array to String Conversion — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/ChallengeIntegrationTest.java, line 31 (Code Correctness: Byte Array to String Conversion) — Low

Sink: String()
Enclosing Method: testChallenge1()
File: src/it/java/org/owasp/webgoat/ChallengeIntegrationTest.java:31
Taint Flags:

```
28  .extract()
29  .asByteArray();
30
31  String pincode = new String(Arrays.copyOfRange(resultBytes, 81216, 81220));
32  Map<String, Object> params = new HashMap<>();
33  params.clear();
34  params.put("username", "admin");
```

src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 140 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

Sink: String()
Enclosing Method: resetVotes()
File: src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java:140
Taint Flags:

```
137  .cookie("access_token");
138
139  String header = accessToken.substring(0, accessToken.indexOf("."));
140  header = new
String(Base64.getUrlDecoder().decode(header.getBytes(Charset.defaultCharset())));
141
142  String body = accessToken.substring(1 + accessToken.indexOf("."),
accessToken.lastIndexOf("."));
143  body = new String(Base64.getUrlDecoder().decode(body.getBytes(Charset.defaultCharset())));
```

src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 178 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Byte Array to String Conversion — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 178 (Code Correctness: Byte Array to String Conversion) — Low

Sink: String()
Enclosing Method: buyAsTom()
File: src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java:178
Taint Flags:

```
175  private void buyAsTom() throws IOException {
176
177  String header =
178  new String(
179  Base64.getUrlDecoder()
180  .decode("eyJhbGciOiJIUzUxMiJ9".getBytes(Charset.defaultCharset())));
181
```

src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 143 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

Sink: String()
Enclosing Method: resetVotes()
File: src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java:143
Taint Flags:

```
140  header = new
String(Base64.getUrlDecoder().decode(header.getBytes(Charset.defaultCharset())));
141
142  String body = accessToken.substring(1 + accessToken.indexOf("."),
accessToken.lastIndexOf("."));
143  body = new String(Base64.getUrlDecoder().decode(body.getBytes(Charset.defaultCharset())));
144
145  ObjectMapper mapper = new ObjectMapper();
146  JsonNode headerNode = mapper.readTree(header);
```

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 144 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Byte Array to String Conversion — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 144 (Code Correctness: Byte Array to String Conversion) — Low

Sink: String()
Enclosing Method: checkAssignmentDefaults()
File: src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java:144
Taint Flags:

```
141  private void checkAssignmentDefaults() {
142
143  String text =
144  new String(
145  Base64.getDecoder()
146  .decode(
147  "TGVhdmluZyBwYXNzd29yZHMgaW4gZG9ja2VyIGltYWdlcyBpcyBub3Qgc28gc2VjdXJl"
```

src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 198 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

Sink: String()
Enclosing Method: buyAsTom()
File: src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java:198
Taint Flags:

```
195  String replacedToken =
196  new String(Base64.getUrlEncoder().encode(headerNode.toString().getBytes()))
197  .concat(".")
198  .concat(new String(Base64.getUrlEncoder().encode(body.getBytes())).toString())
199  .concat(".")
200  .replace("=", "");
201
```

src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 183 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

Sink: String()
Enclosing Method: buyAsTom()
File: src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java:183
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Byte Array to String Conversion — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 183 (Code Correctness: Byte Array to String Conversion) — Low

```
180   .decode("eyJhbGciOiJIUzUxMiJ9".getBytes(Charset.defaultCharset())));
181
182   String body =
183   new String(
184   Base64.getUrlDecoder()
185   .decode(
186   "eyJhZG1pbiI6ImZhbHNlIiwidXNlciI6IkplcnJ5In0"
```

Package: org.owasp.webgoat.lessons.spoofcookie.encoders

src/main/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDec.java, line 86 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

Sink: String()
Enclosing Method: base64Decode()
File: src/main/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDec.java:86
Taint Flags:

```
83
84   private static String base64Decode(final String value) {
85   byte[] decoded = Base64.getDecoder().decode(value.getBytes());
86   return new String(decoded);
87   }
88 }
89
```

src/main/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDec.java, line 77 (Code Correctness: Byte Array to String Conversion) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Sink Details

Sink: String()
Enclosing Method: hexDecode()
File: src/main/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDec.java:77
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Byte Array to String Conversion	Low
Package: org.owasp.webgoat.lessons.spoofcookie.encoders	
src/main/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDec.java, line 77 (Code Correctness: Byte Array to String Conversion)	Low

```
74
75  private static String hexDecode(final String value) {
76  byte[] decoded = Hex.decode(value);
77  return new String(decoded);
78  }
79
80  private static String base64Encode(final String value) {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Constructor Invokes Overridable Function (6 issues)

Abstract

A constructor of the class calls a function that can be overridden.

Explanation

When a constructor calls an overridable function, it may allow an attacker to access the `this` reference prior to the object being fully initialized, which can in turn lead to a vulnerability. **Example 1:** The following calls a method that can be overridden.

```
...
class User {
  private String username;
  private boolean valid;
  public User(String username, String password){
    this.username = username;
    this.valid = validateUser(username, password);
  }
  public boolean validateUser(String username, String password){
    //validate user is real and can authenticate
    ...
  }
  public final boolean isValid(){
    return valid;
  }
}
```

Since the function `validateUser` and the class are not `final`, it means that they can be overridden, and then initializing a variable to the subclass that overrides this function would allow bypassing of the `validateUser` functionality. For example:

```
...
class Attacker extends User{
  public Attacker(String username, String password){
    super(username, password);
  }
  public boolean validateUser(String username, String password){
    return true;
  }
}
...
class MainClass{
  public static void main(String[] args){
    User hacker = new Attacker("Evil", "Hacker");
    if (hacker.isValid()){
      System.out.println("Attack successful!");
    }else{
      System.out.println("Attack failed");
    }
  }
}
```

The code in `Example 1` prints "Attack successful!", since the `Attacker` class overrides the `validateUser()` function that is called from the constructor of the superclass `User`, and Java will first look in the subclass for functions called from the constructor.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

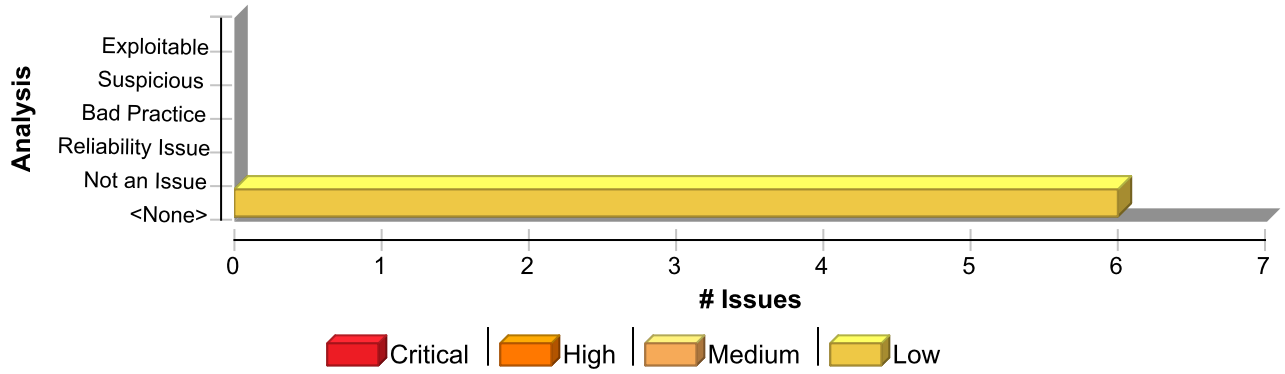
Recommendation

Constructors should not call functions that can be overridden, either by specifying them as `final`, or specifying the class as `final`. Alternatively if this code is only ever needed in the constructor, the `private` access specifier can be used, or the logic could be placed directly into the constructor of the superclass.
Example 2: The following makes the class `final` to prevent the function from being overridden elsewhere.

```
...
final class User {
  private String username;
  private boolean valid;
  public User(String username, String password){
    this.username = username;
    this.valid = validateUser(username, password);
  }
  private boolean validateUser(String username, String password){
    //validate user is real and can authenticate
    ...
  }
  public final boolean isValid(){
    return valid;
  }
}
```

This example specifies the class as `final`, so that it cannot be subclassed, and changes the `validateUser()` function to `private`, since it is not needed elsewhere in this application. This is programming defensively, since at a later date it may be decided that the `User` class needs to be subclassed, which would result in this vulnerability reappearing if the `validateUser()` function was not set to `private`.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Constructor Invokes Overridable Function	6	0	0	6
Total	**6**	**0**	**0**	**6**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Constructor Invokes Overridable Function — Low

Package: org.owasp.webgoat.container.session

src/main/java/org/owasp/webgoat/container/session/UserSessionData.java, line 13 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: setValue
Enclosing Method: UserSessionData()
File: src/main/java/org/owasp/webgoat/container/session/UserSessionData.java:13
Taint Flags:

```
10  public UserSessionData() {}
11
12  public UserSessionData(String key, String value) {
13  setValue(key, value);
14  }
15
16  // GETTERS & SETTERS
```

Package: org.owasp.webgoat.container.users

src/main/java/org/owasp/webgoat/container/users/WebGoatUser.java, line 40 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: createUser
Enclosing Method: WebGoatUser()
File: src/main/java/org/owasp/webgoat/container/users/WebGoatUser.java:40
Taint Flags:

```
37  this.username = username;
38  this.password = password;
39  this.role = role;
40  createUser();
41  }
42
43  public void createUser() {
```

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Constructor Invokes Overridable Function — Low

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: reset
Enclosing Method: MD5()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java:34
Taint Flags:

```
31  * @since ostermillerutils 1.00.00
32  */
33  public MD5() {
34  reset();
35  }
36
37  /**
```

Package: org.owasp.webgoat.lessons.missingac

src/main/java/org/owasp/webgoat/lessons/missingac/DisplayUser.java, line 47 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: genUserHash
Enclosing Method: DisplayUser()
File: src/main/java/org/owasp/webgoat/lessons/missingac/DisplayUser.java:47
Taint Flags:

```
44  this.admin = user.isAdmin();
45
46  try {
47  this.userHash = genUserHash(user.getUsername(), user.getPassword(), passwordSalt);
48  } catch (Exception ex) {
49  this.userHash = "Error generating user hash";
50  }
```

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java, line 66 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Constructor Invokes Overridable Function — Low

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java, line 66 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: initDefaultComments
Enclosing Method: CommentsCache()
File: src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java:66
Taint Flags:

```
63
64  public CommentsCache(WebSession webSession) {
65  this.webSession = webSession;
66  initDefaultComments();
67  }
68
69  void initDefaultComments() {
```

Package: org.owasp.webgoat.webwolf.user

src/main/java/org/owasp/webgoat/webwolf/user/WebGoatUser.java, line 52 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: createUser
Enclosing Method: WebGoatUser()
File: src/main/java/org/owasp/webgoat/webwolf/user/WebGoatUser.java:52
Taint Flags:

```
49  public WebGoatUser(String username, String password) {
50  this.username = username;
51  this.password = password;
52  createUser();
53  }
54
55  public void createUser() {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Erroneous String Compare (1 issue)

Abstract

Strings should be compared with the `equals()` method, not == or `!=`.

Explanation

This program uses == or `!=` to compare two strings for equality, which compares two objects for equality, not their values. Chances are good that the two references will never be equal. **Example 1:** The following branch will never be taken.

```
if (args[0] == STRING_CONSTANT) {
    logger.info("miracle");
}
```

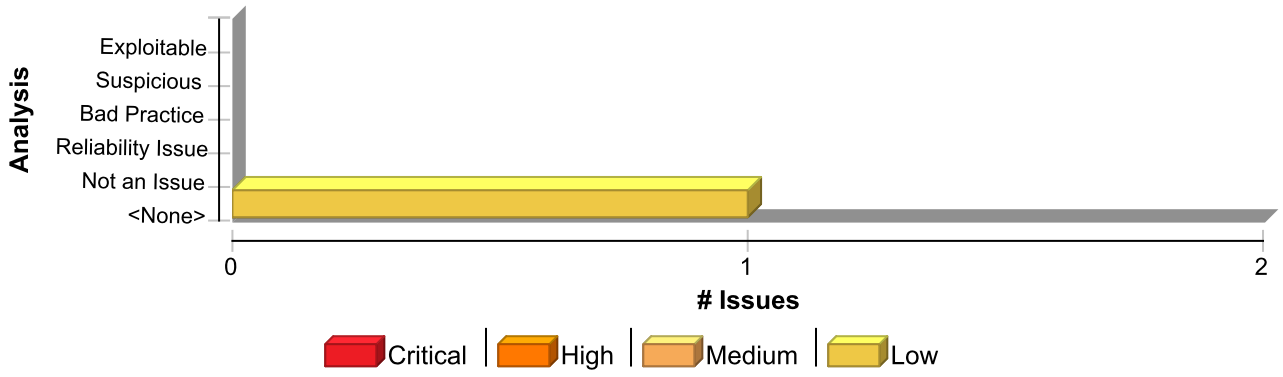
The == and `!=` operators will only behave as expected when they are used to compare strings contained in objects that are equal. The most common way for this to occur is for the strings to be interned, whereby the strings are added to a pool of objects maintained by the `String` class. Once a string is interned, all uses of that string will use the same object and equality operators will behave as expected. All string literals and string-valued constants are interned automatically. Other strings can be interned manually be calling `String.intern()`, which will return a canonical instance of the current string, creating one if necessary.

Recommendation

Use `equals()` to compare strings. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
if (STRING_CONSTANT.equals(args[0])) {
    logger.info("could happen");
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Erroneous String Compare	1	0	0	1
Total	**1**	**0**	**0**	**1**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Erroneous String Compare	Low
Package: org.owasp.webgoat.lessons.csrf	
src/main/java/org/owasp/webgoat/lessons/csrf/ForgedReviews.java, line 110 (Code Correctness: Erroneous String Compare)	**Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: createNewReview()
File: src/main/java/org/owasp/webgoat/lessons/csrf/ForgedReviews.java:110
Taint Flags:

```
107  return failed(this).feedback("csrf-you-forgot-something").build();
108  }
109  // we have the spoofed files
110  if (referer != "NULL" && refererArr[2].equals(host)) {
111  return failed(this).feedback("csrf-same-host").build();
112  } else {
113  return success(this)
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Incorrect Serializable Method Signature (1 issue)

Abstract

Using the incorrect method signature on a method used in serialization may lead to it never being called.

Explanation

Code Correctness: Incorrect Serializable Method Signature issues occur when a serializable class creates a serialization or deserialization function but does not follow the correct signatures:

```
  private void writeObject(java.io.ObjectOutputStream out) throws IOException;
  private void readObject(java.io.ObjectInputStream in) throws IOException,
ClassNotFoundException;
  private void readObjectNoData() throws ObjectStreamException;
```

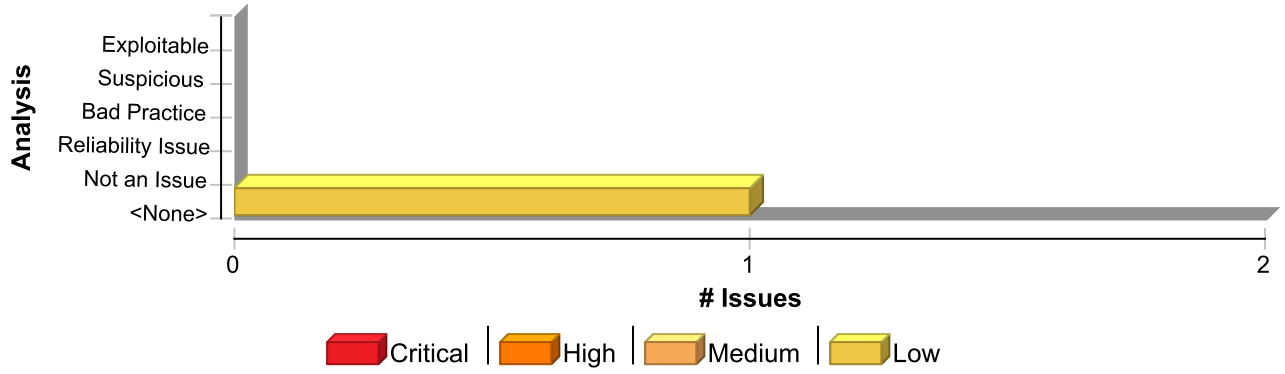
Deviating from the method signatures that serialization requires may mean that the method is never called during serialization/deserialization, leading to incomplete serialization/deserialization, or could mean that untrusted code could gain access to the objects. In the case that there are exceptions that are not thrown, it may mean that serialization/deserialization fails and crashes the application or potentially even fails quietly such that objects may be only partially constructed correctly, leading to flaws that can be extremely difficult to debug. The caller should catch these exceptions such that incorrect serialization/deserialization can be handled properly without a crash or partially constructed objects.

Recommendation

When using serialization in Java for classes that require special handling, the `writeObject()`, `readObject()` and `readObjectNoData` methods must have the exact signatures:

```
  private void writeObject(java.io.ObjectOutputStream out) throws IOException;
  private void readObject(java.io.ObjectInputStream in) throws IOException,
ClassNotFoundException;
  private void readObjectNoData() throws ObjectStreamException;
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Incorrect Serializable Method Signature	1	0	0	1
Total	**1**	**0**	**0**	**1**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: Incorrect Serializable Method Signature — Low

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 44 (Code Correctness: Incorrect Serializable Method Signature) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: readObject
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:44
Taint Flags:

```
41  *
42  * @author stupid develop
43  */
44  private void readObject(ObjectInputStream stream) throws Exception {
45  // unserialize data so taskName and taskAction are available
46  stream.defaultReadObject();
47
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Code Correctness: readObject() Invokes Overridable Function (1 issue)

Abstract

The `readObject()` method within the class calls a function that may be overridden.

Explanation

During deserialization, `readObject()` acts like a constructor, so object initialization is not complete until this function ends. Therefore when a `readObject()` function of a `Serializable` class calls an overridable function, this may provide the overriding method access to the object's state prior to it being fully initialized. **Example 1:** The following `readObject()` function calls a method that can be overridden.

```
  ...
  private void readObject(final ObjectInputStream ois) throws IOException,
ClassNotFoundException {
    checkStream(ois);
    ois.defaultReadObject();
  }

  public void checkStream(ObjectInputStream stream){
    ...
  }
```

Since the function `checkStream()` and its enclosing class are not `final` and public, it means that the function can be overridden, which may mean that an attacker may override the `checkStream()` function in order to get access to the object during deserialization.

Recommendation

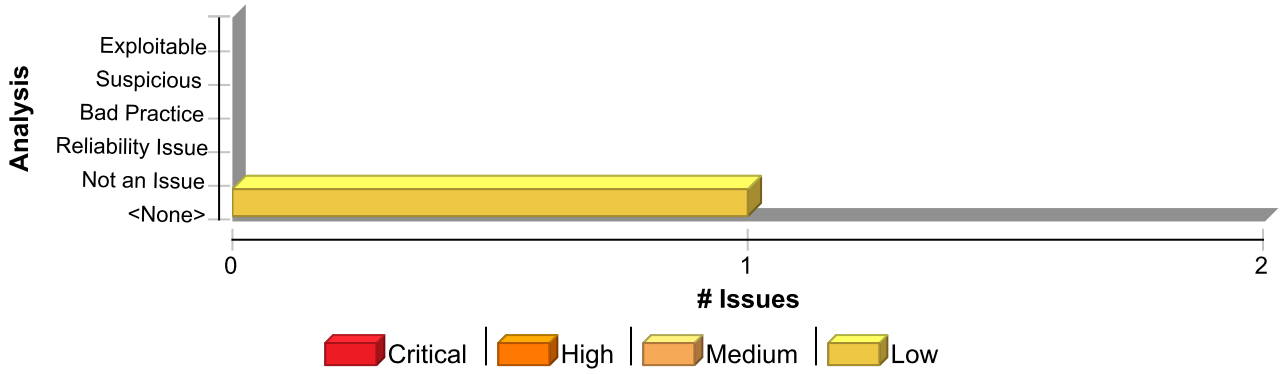
During deserialization, `readObject` should not call functions that can be overridden, either by specifying them as `final`, or specifying the class as `final`. Alternatively if this code is only ever needed in the `readObject()` function, the `private` access specifier can be used, or the logic could be placed directly into the `readObject()` method itself. **Example 2:** The following prevents the `checkStream()` method from being overridden.

```
  ...
  private void readObject(final ObjectInputStream ois) throws IOException,
ClassNotFoundException {
    checkStream(ois);
    ois.defaultReadObject();
  }

  public final void checkStream(ObjectInputStream stream){
    ...
  }
```

In `Example 2`, `checkStream` was set to `final`, so that it cannot be overridden by a subclass.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: readObject() Invokes Overridable Function	1	0	0	1
Total	**1**	**0**	**0**	**1**

Code Correctness: readObject() Invokes Overridable Function — Low

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 56 (Code Correctness: readObject() Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: toString
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:56
Taint Flags:

```
53  && (requestedExecutionTime.isBefore(LocalDateTime.now().minusMinutes(10))
54  || requestedExecutionTime.isAfter(LocalDateTime.now()))) {
55  // do nothing is the time is not within 10 minutes after the object has been created
56  log.debug(this.toString());
57  throw new IllegalArgumentException("outdated");
58  }
59
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Command Injection (1 issue)

Abstract

Executing commands that include unvalidated user input can cause an application to execute malicious commands on behalf of an attacker.

Explanation

Command injection vulnerabilities take two forms: - An attacker can change the command that the program executes: the attacker explicitly controls what the command is. - An attacker can change the environment in which the command executes: the attacker implicitly controls what the command means. In this case, we are primarily concerned with the second scenario, the possibility that an attacker may be able to change the meaning of the command by changing an environment variable or by putting a malicious executable early in the search path. Command injection vulnerabilities of this type occur when: 1. An attacker modifies an application's environment. 2. The application executes a command without specifying an absolute path or verifying the binary being executed. 3. By executing the command, the application gives an attacker a privilege or capability that the attacker would not otherwise have. **Example:** The following code is from a web application that provides an interface through which users can update their password on the system. Part of the process for updating passwords in certain network environments is to run a `make` command in the `/var/yp` directory.

```
...
System.Runtime.getRuntime().exec("make");
...
```

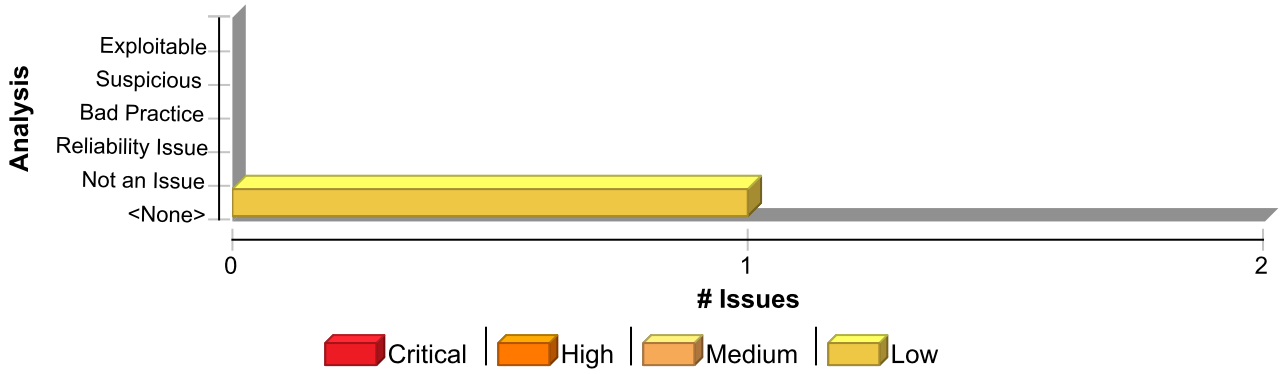
The problem here is that the program does not specify an absolute path for make and fails to clean its environment prior to executing the call to `Runtime.exec()`. If an attacker can modify the `$PATH` variable to point to a malicious binary called `make` and then execute the application in their environment, the malicious binary will be loaded instead of the one intended. Because of the nature of the application, it runs with the privileges necessary to perform system operations, which means the attacker's `make` will now be run with these privileges, possibly giving them complete control of the system.

Recommendation

An attacker may indirectly control commands executed by a program by modifying the environment in which they are executed. The environment should not be trusted and precautions should be taken to prevent an attacker from using some manipulation of the environment to perform an attack. Whenever possible, commands should be controlled by the application and executed using an absolute path. In cases where the path is not known at compile time, such as for cross-platform applications, an absolute path should be constructed from trusted values during execution. Command values and paths read from configuration files or the environment should be sanity-checked against a set of invariants that define valid values. Other checks can sometimes be performed to detect if these sources may have been tampered with. For example, if a configuration file is world-writable, the program might refuse to run. In cases where information about the binary to be executed is known in advance, the program may perform checks to verify the identity of the binary. If a binary should always be owned by a particular user or have a particular set of access permissions assigned to it, these properties can be verified programmatically before the binary is executed. In the end it may be impossible for a program to fully protect itself from an imaginative attacker bent on controlling the commands the program executes. You should strive to identify and protect against every conceivable manipulation of input values and the environment. The goal should be to shut down as many attack vectors as possible.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Command Injection	1	0	0	1
Total	**1**	**0**	**0**	**1**

Command Injection — Low

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 65 (Command Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: exec(0)
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:65
Taint Flags:

```
62  && taskAction.length() < 22) {
63  log.info("about to execute: {}", taskAction);
64  try {
65  Process p = Runtime.getRuntime().exec(taskAction);
66  BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
67  String line = null;
68  while ((line = in.readLine()) != null) {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials (17 issues)

Abstract

Hardcoded API credentials can compromise system security in a way that is not easy to remedy.

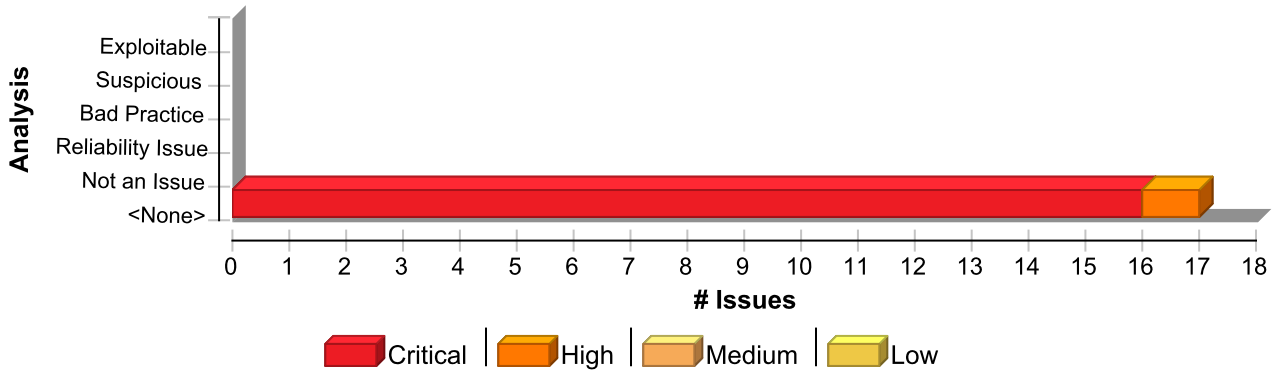
Explanation

Never hardcode credentials, including usernames, passwords, API keys, API secrets, and API Tokens. Not only are hardcoded credentials visible to all of the project developers, they are extremely difficult to update. After the code is in production, the credentials cannot be changed without patching the software. If the credentials are compromised, the organization must choose between security and system availability.

Recommendation

Make sure that API credentials are either loaded from a configuration file that is only available in the runtime environment or from environment variables.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Credential Management: Hardcoded API Credentials	17	0	0	17
Total	**17**	**0**	**0**	**17**

Credential Management: Hardcoded API Credentials	Critical
Package: .github.workflows	
.github/workflows/test.yml, line 65 (Credential Management: Hardcoded API Credentials)	**Critical**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: ConfigPair
File: .github/workflows/test.yml:65
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials — Critical

Package: .github.workflows

.github/workflows/test.yml, line 65 (Credential Management: Hardcoded API Credentials) — Critical

```
62  - name: Send report to commit
63  if: github.repository != 'WebGoat/WebGoat' && github.event_name == 'push'
64  uses: joonvena/robotframework-reporter-action@v2.2
65  with:
66  gh_access_token: ${{ secrets.GITHUB_TOKEN }}
67  report_path: 'robotreport'
68
```

Package: .org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 121 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:121
Taint Flags:

```
118  @Test
119  void checkoutWithTomsTokenFromAccessLogShouldFail() throws Exception {
120  String accessTokenTom =
121
"eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsInVzZX
bjm7Q";
122  mockMvc
123  .perform(
124  MockMvcRequestBuilders.post("/JWT/refresh/checkout")
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 133 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:133
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials — Critical

Package: .org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 133 (Credential Management: Hardcoded API Credentials) — Critical

```
130  @Test
131  void checkoutWitRandomTokenShouldFail() throws Exception {
132  String accessTokenTom =
133  "eyJhbGciOiJIUzUxMiJ9.eyJpLXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsInVzZX
bjm7Q";
134  mockMvc
135  .perform(
136  MockMvcRequestBuilders.post("/JWT/refresh/checkout")
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 75 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:75
Taint Flags:

```
72  // Now create a new refresh token for Tom based on Toms old access token and send the
refresh
73  // token of Jerry
74  String accessTokenTom =
75  "eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsInVzZX
bjm7Q";
76  Map<String, Object> refreshJson = new HashMap<>();
77  refreshJson.put("refresh_token", refreshToken);
78  result =
```

Package: .org.owasp.webgoat.lessons.jwt.claimmisuse

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java, line 24 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java:24
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials — Critical

Package: .org.owasp.webgoat.lessons.jwt.claimmisuse

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java, line 24 (Credential Management: Hardcoded API Credentials) — Critical

```
21 public class JWTHeaderKIDEndpointTest extends LessonTest {
22
23  private static final String TOKEN_JERRY =
24 "eyJraWQiOiJ3ZWJnb2F0X2tleSIsImFsZyI6IkhTNTEyIn0.eyJhdWQiOiJ3ZWJnb2F0Lm9yZyIsImVtYWlsIjoiamVycnl
25
26  @BeforeEach
27  public void setup() {
```

Package: .org.owasp.webgoat.webwolf.jwt

src/test/java/org/owasp/webgoat/webwolf/jwt/JWTTokenTest.java, line 44 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/test/java/org/owasp/webgoat/webwolf/jwt/JWTTokenTest.java:44
Taint Flags:

```
41  void decodeValidSignedToken() {
42  var token =
43  JWTToken.decode(
44  "eyJhbGciOiJIUzI1NiJ9.eyJ0ZXN0IjoidGVzdCJ9.KOobRHDYyaesV_doOk11XXGKSONwzllraAaqqM4VFE4",
45  "test");
46
47  assertThat(token.getHeader()).contains("\"alg\" : \"HS256\"");
```

src/test/java/org/owasp/webgoat/webwolf/jwt/JWTTokenTest.java, line 30 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/test/java/org/owasp/webgoat/webwolf/jwt/JWTTokenTest.java:30
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials — Critical

Package: .org.owasp.webgoat.webwolf.jwt

src/test/java/org/owasp/webgoat/webwolf/jwt/JWTTokenTest.java, line 30 (Credential Management: Hardcoded API Credentials) — Critical

```
27
28  assertThat(token.getEncoded())
29  .isEqualTo(
30  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZXN0IjoidGVzdCJ9.axNp9BkswwK_YRF2URJ5P1UejQNYZbK4qYcMn
31  }
32
33  @Test
```

src/test/java/org/owasp/webgoat/webwolf/jwt/JWTTokenTest.java, line 55 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/test/java/org/owasp/webgoat/webwolf/jwt/JWTTokenTest.java:55
Taint Flags:

```
52  void decodeInvalidSignedToken() {
53  var token =
54  JWTToken.decode(
55  "eyJhbGciOiJIUzI1NiJ9.eyJ0ZXsdfdfsaasfddfasN0IjoidGVzdCJ9.KOobRHDYyaesV_doOk11XXGKSONwzllraAaqqM
56  "");
57
58  assertThat(token.getHeader()).contains("\"alg\" : \"HS256\"");
```

Package: robot

robot/goat.robot, line 116 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: robot/goat.robot:116
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials — Critical

Package: robot

robot/goat.robot, line 116 (Credential Management: Hardcoded API Credentials) — Critical

```
113  Go To ${ENDPOINT_WOLF}/jwt
114  Click Element token
115  Wait Until Element Is Enabled token 5s
116  Input Text token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0Ijo
117  Click Element secretKey
118  Input Text secretKey none
119  Sleep 2s # Pause before reading the result
```

Package: src.main.resources.lessons.jwt.documentation

src/main/resources/lessons/jwt/documentation/JWT_libraries.adoc, line 28 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/resources/lessons/jwt/documentation/JWT_libraries.adoc:28
Taint Flags:

```
25
26 [source]
27 ----
28 var token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0Ij
29
30 Jwts.parser().setSigningKey("test").parseClaimsJws(token);
31 ----
```

src/main/resources/lessons/jwt/documentation/JWT_libraries.adoc, line 40 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/resources/lessons/jwt/documentation/JWT_libraries.adoc:40
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials — Critical

Package: src.main.resources.lessons.jwt.documentation

src/main/resources/lessons/jwt/documentation/JWT_libraries.adoc, line 40 (Credential Management: Hardcoded API Credentials) — Critical

```
37
38 [source]
39 ----
40 var token = "
eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0Ijox
41
42 Jwts.parser().setSigningKey("test").parseClaimsJws(token);
43 ----
```

src/main/resources/lessons/jwt/documentation/JWT_decode.adoc, line 8 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/resources/lessons/jwt/documentation/JWT_decode.adoc:8
Taint Flags:

```
5
6 [source]
7 ----
8
 eyJhbGciOiJIUzI1NiJ9.ew0KICAiYXV0aG9yaXRpZXMiIDogWyAiUk9MRV9BRE1JTiIsICJST0xFX1VTRVIiIF0sDQogIC
zKDSntJQyHPpJ2mZAbnWRfel99iI
9 ----
10
11 Copy and paste the following token and decode the token, can you find the user inside the
token?
```

src/main/resources/lessons/jwt/documentation/JWT_signing_solution.adoc, line 35 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/resources/lessons/jwt/documentation/JWT_signing_solution.adoc:35
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials — Critical

Package: src.main.resources.lessons.jwt.documentation

src/main/resources/lessons/jwt/documentation/JWT_signing_solution.adoc, line 35 (Credential Management: Hardcoded API Credentials) — Critical

```
32 ----
33 GET http://localhost:8080/WebGoat/JWT/votings/login?user=Tom HTTP/1.1
34
35  access_token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE2MDgxMjg1NjYsImFkbWluIjoiZmFsc2UiLCJ1c2VyIjoiVG9tI
X6FPf34Qly9SMpqIUSP8jykedJbjOBNlM3_CTjgk1SvUv48Pz8zIzA
36 ----
37
38 Decoding the token gives:
```

Package: src.main.resources.lessons.jwt.html

src/main/resources/lessons/jwt/html/JWT.html, line 322 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/resources/lessons/jwt/html/JWT.html:322
Taint Flags:

```
319  <div class="assignment-success"><i class="fa fa-2 fa-check hidden" aria-hidden="true"></
i></div>
320  <form class="attack-form" accept-charset="UNKNOWN"
321  method="POST"
322  action="JWT/final/delete?
token=eyJ0eXAiOiJKV1QiLCJqa3UiOiJodHRwczovL2NvZ25pdG8taWRwLnVzLWVhc3QtMS5hbWF6b25hd3MuY29tL3dlYm
bwPqRvxmaShPYtG4BBM_wOGlg-BYTTuws-6yISMfTB5U1WBDwLr6dLU123TGO26wCVBgTKbA0KKG94-
ToOcneWLOTEacEfQQOlIQ">
323  <div class="container-fluid">
324  <div id="toast"></div>
325  <div class="col-sm-6 col-md-4 col-lg-3 mt-4">
```

src/main/resources/lessons/jwt/html/JWT.html, line 388 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/resources/lessons/jwt/html/JWT.html:388
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials — Critical

Package: src.main.resources.lessons.jwt.html

src/main/resources/lessons/jwt/html/JWT.html, line 388 (Credential Management: Hardcoded API Credentials) — Critical

```
385  <div class="assignment-success"><i class="fa fa-2 fa-check hidden" aria-hidden="true"></
i></div>
386  <form class="attack-form" accept-charset="UNKNOWN"
387  method="POST"
388  action="JWT/kid/delete?
token=eyJ0eXAiOiJKV1QiLCJraWQiOiJ3ZWJnb2F0X2tleSIsImFsZyI6IkhTMjU2In0.eyJpc3MiOiJXZWJHb2F0IFRva2
389  <div class="container-fluid">
390  <div id="toast"></div>
391  <div class="col-sm-6 col-md-4 col-lg-3 mt-4">
```

Package: src.main.resources.lessons.jwt.images

src/main/resources/lessons/jwt/images/logs.txt, line 2 (Credential Management: Hardcoded API Credentials) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/resources/lessons/jwt/images/logs.txt:2
Taint Flags:

```
1
2 194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "GET /JWT/refresh/checkout?
token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsI
bjm7Q HTTP/1.1" 401 242 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0" "-"
3 194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "POST /JWT/refresh/moveToCheckout HTTP/1.1"
200 12783 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0"
"-"
4 194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "POST /JWT/refresh/login HTTP/1.1" 200 212
"-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" "-"
5 194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "GET /JWT/refresh/addItems HTTP/1.1" 404 249
"-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" "-"
6
7 undefined
```

Credential Management: Hardcoded API Credentials — High

Package: src.main.resources.lessons.jwt.i18n

src/main/resources/lessons/jwt/i18n/WebGoatLabels.properties, line 22 (Credential Management: Hardcoded API Credentials) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Credential Management: Hardcoded API Credentials	High
Package: src.main.resources.lessons.jwt.i18n	
src/main/resources/lessons/jwt/i18n/WebGoatLabels.properties, line 22 (Credential Management: Hardcoded API Credentials)	High

Sink Details

File: src/main/resources/lessons/jwt/i18n/WebGoatLabels.properties:22
Taint Flags:

```
19 jwt-refresh-hint1=Look at the access log you will find a token there
20 jwt-refresh-hint2=The token from the access log is no longer valid, can you find a way to refresh it?
21 jwt-refresh-hint3=The endpoint for refreshing a token is 'JWT/refresh/newToken'
22 jwt-refresh-hint4=Use the found access token in the Authorization: Bearer header and use your own refresh token
23 jwt-refresh-not-tom=User is not Tom but {0}, please try again
24 jwt-refresh-alg-none=Nicely found! You solved the assignment with 'alg: none' can you also solve it by using the refresh token?
25
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery (31 issues)

Abstract

HTTP requests must contain a user-specific secret to prevent an attacker from making unauthorized requests.

Explanation

A cross-site request forgery (CSRF) vulnerability occurs when: 1. A web application uses session cookies. 2. The application acts on an HTTP request without verifying that the request was made with the user's consent. A nonce is a cryptographic random value that is sent with a message to prevent replay attacks. If the request does not contain a nonce that proves its provenance, the code that handles the request is vulnerable to a CSRF attack (unless it does not change the state of the application). This means a web application that uses session cookies has to take special precautions to ensure that an attacker can't trick users into submitting bogus requests. Imagine a web application that allows administrators to create new accounts as follows:

```
  var req = new XMLHttpRequest();
  req.open("POST", "/new_user", true);
  body = addToPost(body, new_username);
  body = addToPost(body, new_passwd);
  req.send(body);
```

An attacker might set up a malicious web site that contains the following code.

```
  var req = new XMLHttpRequest();
  req.open("POST", "http://www.example.com/new_user", true);
  body = addToPost(body, "attacker");
  body = addToPost(body, "haha");
  req.send(body);
```

If an administrator for `example.com` visits the malicious page while she has an active session on the site, she will unwittingly create an account for the attacker. This is a CSRF attack. It is possible because the application does not have a way to determine the provenance of the request. Any request could be a legitimate action chosen by the user or a faked action set up by an attacker. The attacker does not get to see the Web page that the bogus request generates, so the attack technique is only useful for requests that alter the state of the application. Applications that pass the session identifier in the URL rather than as a cookie do not have CSRF problems because there is no way for the attacker to access the session identifier and include it as part of the bogus request. CSRF is entry number five on the 2007 OWASP Top 10 list.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

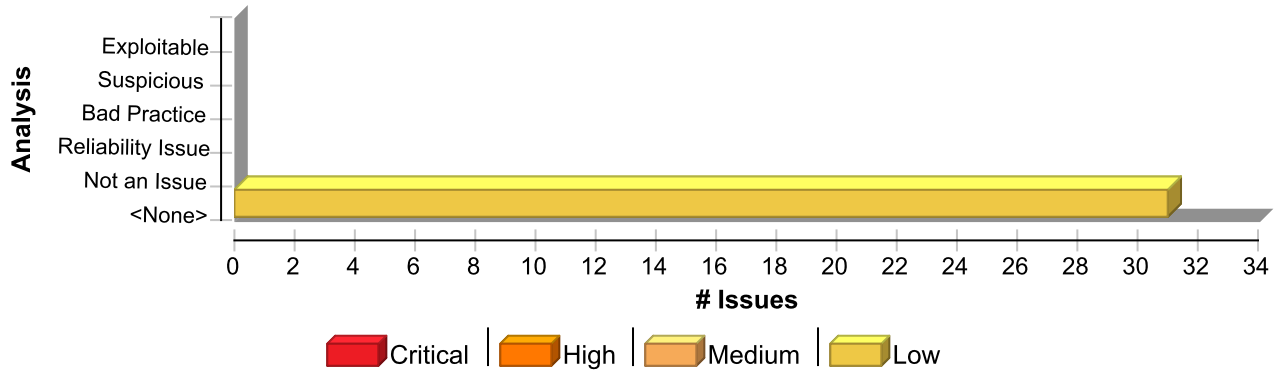
Recommendation

Applications that use session cookies must include some piece of information in every form post that the back-end code can use to validate the provenance of the request. One way to do that is to include a random request identifier or nonce, as follows:

```
RequestBuilder rb = new RequestBuilder(RequestBuilder.POST, "/new_user");
body = addToPost(body, new_username);
body = addToPost(body, new_passwd);
body = addToPost(body, request_id);
rb.sendRequest(body, new NewAccountCallback(callback));
```

Then the back-end logic can validate the request identifier before processing the rest of the form data. When possible, the request identifier should be unique to each server request rather than shared across every request for a particular session. As with session identifiers, the harder it is for an attacker to guess the request identifier, the harder it is to conduct a successful CSRF attack. The token should not be easily guessed and it should be protected in the same way that session tokens are protected, such as using SSLv3. Additional mitigation techniques include: **Framework protection:** Most modern web application frameworks embed CSRF protection and they will automatically include and verify CSRF tokens. **Use a Challenge-Response control:** Forcing the customer to respond to a challenge sent by the server is a strong defense against CSRF. Some of the challenges that can be used for this purpose are: CAPTCHAs, password re-authentication and one-time tokens. **Check HTTP Referer/Origin headers:** An attacker won't be able to spoof these headers while performing a CSRF attack. This makes these headers a useful method to prevent CSRF attacks. **Double-submit Session Cookie:** Sending the session ID Cookie as a hidden form value in addition to the actual session ID Cookie is a good protection against CSRF attacks. The server will check both values and make sure they are identical before processing the rest of the form data. If an attacker submits a form in behalf of a user, he won't be able to modify the session ID cookie value as per the same-origin-policy. **Limit Session Lifetime:** When accessing protected resources using a CSRF attack, the attack will only be valid as long as the session ID sent as part of the attack is still valid on the server. Limiting the Session lifetime will reduce the probability of a successful attack. The techniques described here can be defeated with XSS attacks. Effective CSRF mitigation includes XSS mitigation techniques.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cross-Site Request Forgery	31	0	0	31
Total	**31**	**0**	**0**	**31**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.authbypass.js

src/main/resources/lessons/authbypass/js/bypass.js, line 11 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: AssignmentStatement
Enclosing Method: onViewProfile()
File: src/main/resources/lessons/authbypass/js/bypass.js:11
Taint Flags:

```
8 var onViewProfile = function () {
9  console.warn("on view profile activated")
10  webgoat.customjs.jquery.ajax({
11  method: "GET",
12  url: "IDOR/profile",
13  contentType: 'application/json; charset=UTF-8'
14  }).then(webgoat.customjs.idorViewProfile);
```

Package: src.main.resources.lessons.challenges.js

src/main/resources/lessons/challenges/js/challenge8.js, line 46 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: doVote()
File: src/main/resources/lessons/challenges/js/challenge8.js:46
Taint Flags:

```
43
44 function doVote(stars) {
45  $("#voteResultMsg").hide();
46  $.get("challenge/8/vote/" + stars, function (result) {
47  if (result["error"]) {
48  $("#voteResultMsg").addClass('alert-danger alert-dismissable');
49  } else {
```

src/main/resources/lessons/challenges/js/challenge8.js, line 26 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.challenges.js

src/main/resources/lessons/challenges/js/challenge8.js, line 26 (Cross-Site Request Forgery) — Low

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: average()
File: src/main/resources/lessons/challenges/js/challenge8.js:26
Taint Flags:

```
23 }
24
25 function average() {
26  $.get("challenge/8/votes/average", function (average) {
27  for (var i = 1; i <= 5; i++) {
28  var number = average["average"];
29  $("#star" + i).removeClass('btn-warning');
```

src/main/resources/lessons/challenges/js/challenge8.js, line 7 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: loadVotes()
File: src/main/resources/lessons/challenges/js/challenge8.js:7
Taint Flags:

```
4 })
5
6 function loadVotes() {
7  $.get("challenge/8/votes/", function (votes) {
8  var totalVotes = 0;
9  for (var i = 1; i <= 5; i++) {
10  totalVotes = totalVotes + votes[i];
```

Package: src.main.resources.lessons.clientsidefiltering.js

src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js, line 17 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.clientsidefiltering.js

src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js, line 17 (Cross-Site Request Forgery) — Low

Sink: FunctionPointerCall: get
Enclosing Method: ajaxFunction()
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js:17
Taint Flags:

```
14 }
15
16 function ajaxFunction(userId) {
17  $.get("clientSideFiltering/salaries?userId=" + userId, function (result, status) {
18  var html = "<table border = '1' width = '90%' align = 'center'";
19  html = html + '<tr>';
20  html = html + '<td>UserID</td>';
```

src/main/resources/lessons/clientsidefiltering/js/clientSideFilteringFree.js, line 41 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: lambda()
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFilteringFree.js:41
Taint Flags:

```
38  })
39  $(".checkoutCode").on("blur", function () {
40  var checkoutCode = $(".checkoutCode").val();
41  $.get("clientSideFiltering/challenge-store/coupons/" + checkoutCode, function (result,
status) {
42  var discount = result.discount;
43  if (discount > 0) {
44  $('#discount').text(discount);
```

Package: src.main.resources.lessons.csrf.js

src/main/resources/lessons/csrf/js/csrf-review.js, line 35 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.csrf.js

src/main/resources/lessons/csrf/js/csrf-review.js, line 35 (Cross-Site Request Forgery) — Low

Sink: FunctionPointerCall: get
Enclosing Method: getChallenges()
File: src/main/resources/lessons/csrf/js/csrf-review.js:35
Taint Flags:

```
32
33  function getChallenges() {
34  $("#list").empty();
35  $.get('csrf/review', function (result, status) {
36  for (var i = 0; i < result.length; i++) {
37  var comment = html.replace('USER', result[i].user);
38  comment = comment.replace('DATETIME', result[i].dateTime);
```

Package: src.main.resources.lessons.hijacksession.templates

src/main/resources/lessons/hijacksession/templates/hijackform.html, line 3 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Content)

Sink Details

File: src/main/resources/lessons/hijacksession/templates/hijackform.html:3
Taint Flags:

```
1 <div class="row">
2  <div class="col-md-4">
3  <form class="attack-form" accept-charset="UNKNOWN" method="POST"
4  action="HijackSession/login">
5  <div style="padding: 20px;" id="password-login">
6  <h4 style="border-bottom: 1px solid #c5c5c5;">Account Access</h4>
7  <fieldset>
```

Package: src.main.resources.lessons.idor.js

src/main/resources/lessons/idor/js/idor.js, line 14 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: AssignmentStatement
Enclosing Method: onViewProfile()
File: src/main/resources/lessons/idor/js/idor.js:14
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.idor.js

src/main/resources/lessons/idor/js/idor.js, line 14 (Cross-Site Request Forgery) — Low

```
11 var onViewProfile = function () {
12   console.warn("on view profile activated")
13   webgoat.customjs.jquery.ajax({
14   method: "GET",
15   url: "IDOR/profile",
16   contentType: 'application/json; charset=UTF-8'
17   }).then(webgoat.customjs.idorViewProfile);
```

Package: src.main.resources.lessons.insecurelogin.js

src/main/resources/lessons/insecurelogin/js/credentials.js, line 3 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: submit_secret_credentials()
File: src/main/resources/lessons/insecurelogin/js/credentials.js:3
Taint Flags:

```
1 function submit_secret_credentials() {
2   var xhttp = new XMLHttpRequest();
3   xhttp['open']('POST', 'InsecureLogin/login', true);
4   //sending the request is obfuscated, to descourage js reading
5   var
_0xb7f9=["\x43\x61\x70\x74\x61\x69\x6E\x4A\x61\x63\x6B","\x42\x6C\x61\x63\x6B\x50\x65\x61\x72\x6
(JSON[_0xb7f9[2]]({username:_0xb7f9[0],password:_0xb7f9[1]}))
6 }
7
```

Package: src.main.resources.lessons.jwt.js

src/main/resources/lessons/jwt/js/jwt-voting.js, line 43 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: getVotings()
File: src/main/resources/lessons/jwt/js/jwt-voting.js:43
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.jwt.js

src/main/resources/lessons/jwt/js/jwt-voting.js, line 43 (Cross-Site Request Forgery) — Low

```
40
41 function getVotings() {
42  $("#votesList").empty();
43  $.get("JWT/votings", function (result, status) {
44  for (var i = 0; i < result.length; i++) {
45  var voteTemplate = html.replace('IMAGE_SMALL', result[i].imageSmall);
46  if (i === 0) {
```

Package: src.main.resources.lessons.lessontemplate.js

src/main/resources/lessons/lessontemplate/js/idor.js, line 14 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: AssignmentStatement
Enclosing Method: onViewProfile()
File: src/main/resources/lessons/lessontemplate/js/idor.js:14
Taint Flags:

```
11 var onViewProfile = function () {
12  console.warn("on view profile activated")
13  webgoat.customjs.jquery.ajax({
14  method: "GET",
15  url: "IDOR/profile",
16  contentType: 'application/json; charset=UTF-8'
17  }).then(webgoat.customjs.idorViewProfile);
```

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 46 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: profileUploadCallbackRemoveUserInput()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:46
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 46 (Cross-Site Request Forgery) — Low

```
43 }
44
45 webgoat.customjs.profileUploadCallbackRemoveUserInput = function () {
46  $.get("PathTraversal/profile-picture", function (result, status) {
47  document.getElementById("previewRemoveUserInput").src = "" + result;
48  });
49 }
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 13 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: profileUploadCallback()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:13
Taint Flags:

```
10 }
11
12 webgoat.customjs.profileUploadCallback = function () {
13  $.get("PathTraversal/profile-picture", function (result, status) {
14  document.getElementById("preview").src = "" + result;
15  });
16 }
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 53 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: profileUploadCallbackRetrieval()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:53
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 53 (Cross-Site Request Forgery) — Low

```
50
51
52 webgoat.customjs.profileUploadCallbackRetrieval = function () {
53  $.get("PathTraversal/profile-picture", function (result, status) {
54  document.getElementById("previewRetrieval").src = "" + result;
55  });
56 }
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 29 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: profileUploadCallbackFix()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:29
Taint Flags:

```
26 }
27
28 webgoat.customjs.profileUploadCallbackFix = function () {
29  $.get("PathTraversal/profile-picture", function (result, status) {
30  document.getElementById("previewFix").src = "" + result;
31  });
32 }
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 59 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: newRandomPicture()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:59
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 59 (Cross-Site Request Forgery) — Low

```
56 }
57
58 function newRandomPicture() {
59  $.get("PathTraversal/random-picture", function (result, status) {
60  document.getElementById("randomCatPicture").src = "" + result;
61  });
62 }
```

Package: src.main.resources.lessons.spoofcookie.templates

src/main/resources/lessons/spoofcookie/templates/spoofcookieform.html, line 3 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Content)

Sink Details

File: src/main/resources/lessons/spoofcookie/templates/spoofcookieform.html:3
Taint Flags:

```
1 <div class="row">
2  <div class="col-md-4">
3  <form class="attack-form" accept-charset="UNKNOWN" method="POST"
4  action="SpoofCookie/login">
5  <div style="padding: 20px;" id="password-login">
6  <h4 style="border-bottom: 1px solid #c5c5c5;">Account Access</h4>
7  <fieldset>
```

Package: src.main.resources.lessons.sqlinjection.js

src/main/resources/lessons/sqlinjection/js/assignment13.js, line 43 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: getServers()
File: src/main/resources/lessons/sqlinjection/js/assignment13.js:43
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.sqlinjection.js

src/main/resources/lessons/sqlinjection/js/assignment13.js, line 43 (Cross-Site Request Forgery) — Low

```
40  '</tr>';
41
42 function getServers(column) {
43  $.get("SqlInjectionMitigations/servers?column=" + column, function (result, status) {
44  $("#servers").empty();
45  for (var i = 0; i < result.length; i++) {
46  var server = html.replace('ID', result[i].id);
```

Package: src.main.resources.lessons.ssrf.js

src/main/resources/lessons/ssrf/js/credentials.js, line 3 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: submit_secret_credentials()
File: src/main/resources/lessons/ssrf/js/credentials.js:3
Taint Flags:

```
1 function submit_secret_credentials() {
2  var xhttp = new XMLHttpRequest();
3  xhttp['open']('POST', '#attack/307/100', true);
4  //sending the request is obfuscated, to descourage js reading
5  var
_0xb7f9=["\x43\x61\x70\x74\x61\x69\x6E\x4A\x61\x63\x6B","\x42\x6C\x61\x63\x6B\x50\x65\x61\x72\x6
(JSON[_0xb7f9[2]]({username:_0xb7f9[0],password:_0xb7f9[1]}))
6 }
7
```

Package: src.main.resources.lessons.xss.js

src/main/resources/lessons/xss/js/stored-xss.js, line 35 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: getChallenges()
File: src/main/resources/lessons/xss/js/stored-xss.js:35
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.lessons.xss.js

src/main/resources/lessons/xss/js/stored-xss.js, line 35 (Cross-Site Request Forgery) — Low

```
32
33  function getChallenges() {
34  $("#list").empty();
35  $.get('CrossSiteScriptingStored/stored-xss', function (result, status) {
36  for (var i = 0; i < result.length; i++) {
37  var comment = html.replace('USER', result[i].user);
38  comment = comment.replace('DATETIME', result[i].dateTime);
```

Package: src.main.resources.lessons.xxe.js

src/main/resources/lessons/xxe/js/xxe.js, line 72 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: getComments()
File: src/main/resources/lessons/xxe/js/xxe.js:72
Taint Flags:

```
69  '</li>';
70
71 function getComments(field) {
72  $.get("xxe/comments", function (result, status) {
73  $(field).empty();
74  for (var i = 0; i < result.length; i++) {
75  var comment = html.replace('USER', result[i].user);
```

Package: src.main.resources.webgoat.static.js

src/main/resources/webgoat/static/js/quiz.js, line 15 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/main/resources/webgoat/static/js/quiz.js:15
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.webgoat.static.js

src/main/resources/webgoat/static/js/quiz.js, line 15 (Cross-Site Request Forgery) — Low

```
12  var json = "";
13  var client = new XMLHttpRequest();
14  var quiz_id = document.getElementById("quiz_id").getAttribute("data-quiz_id");
15  client.open('GET', 'lesson_js/questions_' + quiz_id + '.json');
16  client.onreadystatechange = function() {
17  if (this.readyState == 4 && this.status == 200) {
18  json += client.responseText;
```

Package: src.main.resources.webgoat.static.js.goatApp

src/main/resources/webgoat/static/js/goatApp/goatApp.js, line 21 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: getJSON
Enclosing Method: initApp()
File: src/main/resources/webgoat/static/js/goatApp/goatApp.js:21
Taint Flags:

```
18  return {
19  initApp: function () {
20  var locale = localStorage.getItem('locale') || 'en';
21  $.getJSON('service/labels.mvc', function(data) {
22  window.polyglot = new Polyglot({phrases: data});//i18n polyglot labels
23  asyncErrorHandler.init();
24  console.log('about to create app router');//default js
```

Package: src.main.resources.webgoat.static.js.goatApp.controller

src/main/resources/webgoat/static/js/goatApp/controller/LessonController.js, line 145 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: AssignmentStatement
Enclosing Method: restartLesson()
File: src/main/resources/webgoat/static/js/goatApp/controller/LessonController.js:145
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.webgoat.static.js.goatApp.controller

src/main/resources/webgoat/static/js/goatApp/controller/LessonController.js, line 145 (Cross-Site Request Forgery) — Low

```
142  var self=this;
143  $.ajax({
144  url:'service/restartlesson.mvc',
145  method:'GET'
146  }).done(function(lessonLink) {
147  self.loadLesson(self.name);
148  self.updateMenu();
```

Package: src.main.resources.webgoat.static.js.goatApp.support

src/main/resources/webgoat/static/js/goatApp/support/GoatUtils.js, line 56 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: showLessonCookiesAndParams()
File: src/main/resources/webgoat/static/js/goatApp/support/GoatUtils.js:56
Taint Flags:

```
53  },
54
55  showLessonCookiesAndParams: function() {
56  $.get(goatConstants.cookieService, {}, function(reply) {
57  $("#lesson_cookies").html(reply);
58  }, "html");
59  },
```

Package: src.main.resources.webgoat.static.js.goatApp.view

src/main/resources/webgoat/static/js/goatApp/view/GoatRouter.js, line 68 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: AssignmentStatement
Enclosing Method: phoneHome()
File: src/main/resources/webgoat/static/js/goatApp/view/GoatRouter.js:68
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.webgoat.static.js.goatApp.view

src/main/resources/webgoat/static/js/goatApp/view/GoatRouter.js, line 68 (Cross-Site Request Forgery) — Low

```
65  webgoat.customjs.phoneHome = function (e) {
66  console.log('phoneHome invoked');
67  webgoat.customjs.jquery.ajax({
68  method: "POST",
69  url: "CrossSiteScripting/phone-home-xss",
70  data: {param1: 42, param2: 24},
71  headers: {
```

Package: src.main.resources.webgoat.static.js.jquery_form

src/main/resources/webgoat/static/js/jquery_form/jquery.form.js, line 245 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: ajaxSubmit()
File: src/main/resources/webgoat/static/js/jquery_form/jquery.form.js:245
Taint Flags:

```
242  // hack to fix Safari hang (thanks to Tim Molendijk for this)
243  // see: http://groups.google.com/group/jquery-dev/browse_thread/thread/36395b7ab510dd5d
244  if (options.closeKeepAlive) {
245  $.get(options.closeKeepAlive, function() {
246  jqxhr = fileUploadIframe(a);
247  });
248  }
```

Package: src.main.resources.webgoat.static.js.libs

src/main/resources/webgoat/static/js/libs/text.js, line 270 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: get()
File: src/main/resources/webgoat/static/js/libs/text.js:270
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery — Low

Package: src.main.resources.webgoat.static.js.libs

src/main/resources/webgoat/static/js/libs/text.js, line 270 (Cross-Site Request Forgery) — Low

```
267  text.createXhr())) {
268  text.get = function (url, callback, errback, headers) {
269  var xhr = text.createXhr(), header;
270  xhr.open('GET', url, true);
271
272  //Allow plugins direct access to xhr headers
273  if (headers) {
```

src/main/resources/webgoat/static/js/libs/jquery.form.js, line 245 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: ajaxSubmit()
File: src/main/resources/webgoat/static/js/libs/jquery.form.js:245
Taint Flags:

```
242  // hack to fix Safari hang (thanks to Tim Molendijk for this)
243  // see: http://groups.google.com/group/jquery-dev/browse_thread/thread/36395b7ab510dd5d
244  if (options.closeKeepAlive) {
245  $.get(options.closeKeepAlive, function() {
246  jqxhr = fileUploadIframe(a);
247  });
248  }
```

src/main/resources/webgoat/static/js/libs/ace.js, line 4157 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: get()
File: src/main/resources/webgoat/static/js/libs/ace.js:4157
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Request Forgery	Low
Package: src.main.resources.webgoat.static.js.libs	
src/main/resources/webgoat/static/js/libs/ace.js, line 4157 (Cross-Site Request Forgery)	**Low**

```
4154
4155 exports.get = function (url, callback) {
4156   var xhr = new XMLHttpRequest();
4157   xhr.open('GET', url, true);
4158   xhr.onreadystatechange = function () {
4159   if (xhr.readyState === 4) {
4160   callback(xhr.responseText);
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM (16 issues)

Abstract

Sending unvalidated data to a web browser can result in the browser executing malicious code.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Explanation

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-site scripting (XSS) vulnerabilities occur when: 1. Data enters a web application through an untrusted source. In the case of DOM-based XSS, data is read from a URL parameter or other value within the browser and written back into the page with client-side code. In the case of reflected XSS, the untrusted source is typically a web request, while in the case of persisted (also known as stored) XSS it is typically a database or other back-end data store. 2. The data is included in dynamic content that is sent to a web user without validation. In the case of DOM-based XSS, malicious content is executed as part of DOM (Document Object Model) creation, whenever the victim's browser parses the HTML page. The malicious content sent to the web browser often takes the form of a JavaScript segment, but can also include HTML, Flash or any other type of code that the browser executes. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data such as cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site. **Example 1:** The following JavaScript code segment reads an employee ID, `eid`, from a URL and displays it to the user.

```
<SCRIPT>
var pos=document.URL.indexOf("eid=")+4;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
```

Example 2: Consider the HTML form:

```
  <div id="myDiv">
    Employee ID: <input type="text" id="eid"><br>
    ...
    <button>Show results</button>
  </div>
  <div id="resultsDiv">
    ...
  </div>
```

The following jQuery code segment reads an employee ID from the form, and displays it to the user.

```
  $(document).ready(function(){
    $("#myDiv").on("click", "button", function(){
      var eid = $("#eid").val();
      $("resultsDiv").append(eid);
      ...
    });
  });
```

These code examples operate correctly if the employee ID from the text input with ID `eid` contains only standard alphanumeric text. If `eid` has a value that includes metacharacters or source code, then the code will be executed by the web browser as it displays the HTTP response. **Example 3:** The following code shows an example of a DOM-based XSS within a React application:

```
let element = JSON.parse(getUntrustedInput());
ReactDOM.render(<App>
    {element}
</App>);
```

In `Example 3`, if an attacker can control the entire JSON object retrieved from `getUntrustedInput()`, they may be able to make React render `element` as a component, and therefore can pass an object with `dangerouslySetInnerHTML` with their own controlled value, a typical cross-site scripting attack. Initially these might not appear to be much of a vulnerability. After all, why would someone provide input containing malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use email or social engineering tricks to lure victims into visiting a link to the URL. When victims click the link, they unwittingly reflect the malicious content through the vulnerable web application back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS. As the example demonstrates, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim: - Data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering

© Copyright [2008-2024] Micro Focus or one of its affiliates.

malicious content is to include it as a parameter in a URL that is posted publicly or emailed directly to victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that might include session information, from the user's machine to the attacker or perform other nefarious activities. - The application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Persistent XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. - A source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

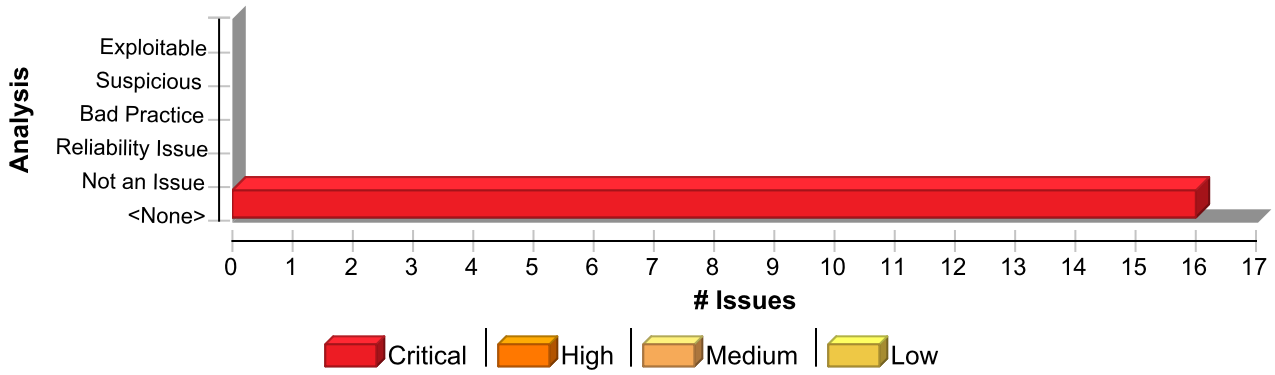
© Copyright [2008-2024] Micro Focus or one of its affiliates.

The solution to prevent XSS is to ensure that validation occurs in the required places and that relevant properties are set to prevent vulnerabilities. Because XSS vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating dynamic content, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for XSS. Web applications must validate all input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for XSS is generally relatively easy. Despite its value, input validation for XSS does not take the place of rigorous output validation. An application might accept input through a shared data store or other trusted source, and that data store might accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means that the best way to prevent XSS vulnerabilities is to validate everything that enters the application and leaves the application destined for the user. The most secure approach to validation for XSS is to create an allow list of safe characters that can appear in HTTP content and accept input composed exclusively of characters in the approved set. For example, a valid username might only include alphanumeric characters or a phone number might only include digits 0-9. However, this solution is often infeasible in web applications because many characters that have special meaning to the browser must be considered valid input after they are encoded, such as a web design bulletin board that must accept HTML fragments from its users. A more flexible, but less secure approach is to implement a deny list, which selectively rejects or escapes potentially dangerous characters before using the input. To form such a list, you first need to understand the set of characters that hold special meaning for web browsers. Although the HTML standard defines which characters have special meaning, many web browsers try to correct common mistakes in HTML and might treat other characters as special in certain contexts. This is why we do not recommend the use of deny lists as a means to prevent XSS. The CERT(R) Coordination Center at the Software Engineering Institute at Carnegie Mellon University provides the following details about special characters in various contexts [1]: In the content of a block-level element (in the middle of a paragraph of text): - "<" is special because it introduces a tag. - "&" is special because it introduces a character entity. - ">" is special because some browsers treat it as special, on the assumption that the author of the page intended to include an opening "<", but omitted it in error. The following principles apply to attribute values: - In attribute values enclosed in double quotes, the double quotes are special because they mark the end of the attribute value. - In attribute values enclosed in single quote, the single quotes are special because they mark the end of the attribute value. - In attribute values without any quotes, white-space characters, such as space and tab, are special. - "&" is special when used with certain attributes, because it introduces a character entity. In URLs, for example, a search engine might provide a link within the results page that the user can click to re-run the search. This can be implemented by encoding the search query inside the URL, which introduces additional special characters: - Space, tab, and new line are special because they mark the end of the URL. - "&" is special because it either introduces a character entity or separates CGI parameters. - Non-ASCII characters (that is, everything greater than 127 in the ISO-8859-1 encoding) are not allowed in URLs, so they are considered to be special in this context. - The "%" symbol must be filtered from input anywhere parameters encoded with HTTP escape sequences are decoded by server-side code. For example, "%" must be filtered if input such as "%68%65%6C%6C%6F" becomes "hello" when it appears on the web page. Within the body of a : - Semicolons, parentheses, curly braces, and new line characters must be filtered out in situations where text could be inserted directly into a pre-existing script tag. Server-side scripts: - Server-side scripts that convert any exclamation characters (!) in input to double-quote characters (") on output might require additional filtering. Other possibilities: - If an attacker submits a request in UTF-7, the special character '<' appears as '+ADw-' and might bypass filtering. If the output is included in a page that does not explicitly specify an encoding format, then some browsers try to intelligently identify the encoding based on the content (in this case, UTF-7). After you identify the correct points in an application to perform validation for XSS attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. If special characters are not considered valid input to the application, then you can reject any input that contains special characters as invalid. A second option is to remove special characters with filtering. However, filtering has the side effect of changing any visual representation of the filtered content and might be unacceptable in circumstances where the integrity of the input must be preserved for display. If input containing special characters must be accepted and displayed accurately, validation must

© Copyright [2008-2024] Micro Focus or one of its affiliates.

encode any special characters to remove their significance. A complete list of ISO 8859-1 encoded values for special characters is provided as part of the official HTML specification [2]. Many application servers attempt to limit an application's exposure to cross-site scripting vulnerabilities by providing implementations for the functions responsible for setting certain specific HTTP response content that perform validation for the characters essential to a cross-site scripting attack. Do not rely on the server running your application to make it secure. For any developed application, there are no guarantees about which application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will continue to stay in sync.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cross-Site Scripting: DOM	16	0	0	16
Total	**16**	**0**	**0**	**16**

Cross-Site Scripting: DOM	**Critical**
Package: src.main.resources.lessons.challenges.js	
src/main/resources/lessons/challenges/js/challenge8.js, line 52 (Cross-Site Scripting: DOM)	**Critical**

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/challenges/js/challenge8.js:46

```
43
44 function doVote(stars) {
45   $("#voteResultMsg").hide();
46   $.get("challenge/8/vote/" + stars, function (result) {
47   if (result["error"]) {
48   $("#voteResultMsg").addClass('alert-danger alert-dismissable');
49   } else {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.challenges.js

src/main/resources/lessons/challenges/js/challenge8.js, line 52 (Cross-Site Scripting: DOM) — Critical

Sink Details

Sink: ~JS_Generic.html()
Enclosing Method: lambda()
File: src/main/resources/lessons/challenges/js/challenge8.js:52
Taint Flags: WEB, XSS

```
49  } else {
50  $("#voteResultMsg").addClass('alert-success alert-dismissable');
51  }
52  $("#voteResultMsg").html(result["message"]);
53  $("#voteResultMsg").show();
54  })
55  loadVotes();
```

src/main/resources/lessons/challenges/js/challenge8.js, line 18 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/challenges/js/challenge8.js:7

```
4 })
5
6 function loadVotes() {
7  $.get("challenge/8/votes/", function (votes) {
8  var totalVotes = 0;
9  for (var i = 1; i <= 5; i++) {
10  totalVotes = totalVotes + votes[i];
```

Sink Details

Sink: ~JS_Generic.html()
Enclosing Method: lambda()
File: src/main/resources/lessons/challenges/js/challenge8.js:18
Taint Flags: WEB, XSS

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.challenges.js

src/main/resources/lessons/challenges/js/challenge8.js, line 18 (Cross-Site Scripting: DOM) — Critical

```
15  console.log(percent);
16  var progressBar = $('#progressBar' + i);
17  progressBar.width(Math.round(percent) * 2 + '%');
18  $("#nrOfVotes" + i).html(votes[i]);
19
20  }
21  }
```

Package: src.main.resources.lessons.clientsidefiltering.js

src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js, line 38 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js:17

```
14 }
15
16 function ajaxFunction(userId) {
17  $.get("clientSideFiltering/salaries?userId=" + userId, function (result, status) {
18  var html = "<table border = '1' width = '90%' align = 'center'";
19  html = html + '<tr>';
20  html = html + '<td>UserID</td>';
```

Sink Details

Sink: Assignment to newdiv.innerHTML
Enclosing Method: lambda()
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js:38
Taint Flags: WEB, XSS

```
35  html = html + '</tr></table>';
36
37  var newdiv = document.createElement("div");
38  newdiv.innerHTML = html;
39  var container = document.getElementById("hiddenEmployeeRecords");
40  container.appendChild(newdiv);
41  });
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.csrf.js

src/main/resources/lessons/csrf/js/csrf-review.js, line 41 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/csrf/js/csrf-review.js:35

```
32
33  function getChallenges() {
34  $("#list").empty();
35  $.get('csrf/review', function (result, status) {
36  for (var i = 0; i < result.length; i++) {
37  var comment = html.replace('USER', result[i].user);
38  comment = comment.replace('DATETIME', result[i].dateTime);
```

Sink Details

Sink: ~JS_Generic.append()
Enclosing Method: lambda()
File: src/main/resources/lessons/csrf/js/csrf-review.js:41
Taint Flags: WEB, XSS

```
38  comment = comment.replace('DATETIME', result[i].dateTime);
39  comment = comment.replace('COMMENT', result[i].text);
40  comment = comment.replace('STARS', result[i].stars)
41  $("#list").append(comment);
42  }
43
44  });
```

Package: src.main.resources.lessons.jwt.js

src/main/resources/lessons/jwt/js/jwt-voting.js, line 63 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/jwt/js/jwt-voting.js:43

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.jwt.js

src/main/resources/lessons/jwt/js/jwt-voting.js, line 63 (Cross-Site Scripting: DOM) — Critical

```
40
41 function getVotings() {
42  $("#votesList").empty();
43  $.get("JWT/votings", function (result, status) {
44  for (var i = 0; i < result.length; i++) {
45  var voteTemplate = html.replace('IMAGE_SMALL', result[i].imageSmall);
46  if (i === 0) {
```

Sink Details

Sink: ~JS_Generic.append()
Enclosing Method: lambda()
File: src/main/resources/lessons/jwt/js/jwt-voting.js:63
Taint Flags: WEB, XSS

```
60  hidden = (result[i].average === undefined ? 'hidden' : '');
61  voteTemplate = voteTemplate.replace(/HIDDEN_VIEW_RATING/g, hidden);
62
63  $("#votesList").append(voteTemplate);
64  }
65  })
66 }
```

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 14 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:13

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 14 (Cross-Site Scripting: DOM) — Critical

```
10 }
11
12 webgoat.customjs.profileUploadCallback = function () {
13  $.get("PathTraversal/profile-picture", function (result, status) {
14  document.getElementById("preview").src = "" +
result;
15  });
16 }
```

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:14
Taint Flags: WEB, XSS

```
11
12 webgoat.customjs.profileUploadCallback = function () {
13  $.get("PathTraversal/profile-picture", function (result, status) {
14  document.getElementById("preview").src = "" + result;
15  });
16 }
17
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 30 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:29

```
26 }
27
28 webgoat.customjs.profileUploadCallbackFix = function () {
29  $.get("PathTraversal/profile-picture", function (result, status) {
30  document.getElementById("previewFix").src = "" +
result;
31  });
32 }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 30 (Cross-Site Scripting: DOM) — Critical

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:30
Taint Flags: WEB, XSS

```
27
28 webgoat.customjs.profileUploadCallbackFix = function () {
29  $.get("PathTraversal/profile-picture", function (result, status) {
30  document.getElementById("previewFix").src = "" + result;
31  });
32 }
33
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 60 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:59

```
56 }
57
58 function newRandomPicture() {
59  $.get("PathTraversal/random-picture", function (result, status) {
60  document.getElementById("randomCatPicture").src = "" + result;
61  });
62 }
```

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:60
Taint Flags: WEB, XSS

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 60 (Cross-Site Scripting: DOM) — Critical

```
57
58 function newRandomPicture() {
59  $.get("PathTraversal/random-picture", function (result, status) {
60  document.getElementById("randomCatPicture").src = "" + result;
61  });
62 }
63
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 47 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:46

```
43 }
44
45 webgoat.customjs.profileUploadCallbackRemoveUserInput = function () {
46  $.get("PathTraversal/profile-picture", function (result, status) {
47  document.getElementById("previewRemoveUserInput").src = "" + result;
48  });
49 }
```

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:47
Taint Flags: WEB, XSS

```
44
45 webgoat.customjs.profileUploadCallbackRemoveUserInput = function () {
46  $.get("PathTraversal/profile-picture", function (result, status) {
47  document.getElementById("previewRemoveUserInput").src = "" + result;
48  });
49 }
50
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 54 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:53

```
50
51
52 webgoat.customjs.profileUploadCallbackRetrieval = function () {
53  $.get("PathTraversal/profile-picture", function (result, status) {
54  document.getElementById("previewRetrieval").src = "" + result;
55  });
56 }
```

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:54
Taint Flags: WEB, XSS

```
51
52 webgoat.customjs.profileUploadCallbackRetrieval = function () {
53  $.get("PathTraversal/profile-picture", function (result, status) {
54  document.getElementById("previewRetrieval").src = "" + result;
55  });
56 }
57
```

Package: src.main.resources.lessons.sqlinjection.js

src/main/resources/lessons/sqlinjection/js/assignment13.js, line 57 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/sqlinjection/js/assignment13.js:43

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.sqlinjection.js

src/main/resources/lessons/sqlinjection/js/assignment13.js, line 57 (Cross-Site Scripting: DOM) — Critical

```
40  '</tr>';
41
42 function getServers(column) {
43  $.get("SqlInjectionMitigations/servers?column=" + column, function
(result, status) {
44  $("#servers").empty();
45  for (var i = 0; i < result.length; i++) {
46  var server = html.replace('ID', result[i].id);
```

Sink Details

Sink: ~JS_Generic.append()
Enclosing Method: lambda()
File: src/main/resources/lessons/sqlinjection/js/assignment13.js:57
Taint Flags: WEB, XSS

```
54  server = server.replace('IP', result[i].ip);
55  server = server.replace('MAC', result[i].mac);
56  server = server.replace('DESCRIPTION', result[i].description);
57  $("#servers").append(server);
58  }
59
60  });
```

Package: src.main.resources.lessons.xss.js

src/main/resources/lessons/xss/js/stored-xss.js, line 40 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/xss/js/stored-xss.js:35

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.xss.js

src/main/resources/lessons/xss/js/stored-xss.js, line 40 (Cross-Site Scripting: DOM) — Critical

```
32
33  function getChallenges() {
34  $("#list").empty();
35  $.get('CrossSiteScriptingStored/stored-xss', function (result, status) {
36  for (var i = 0; i < result.length; i++) {
37  var comment = html.replace('USER', result[i].user);
38  comment = comment.replace('DATETIME', result[i].dateTime);
```

Sink Details

Sink: ~JS_Generic.append()
Enclosing Method: lambda()
File: src/main/resources/lessons/xss/js/stored-xss.js:40
Taint Flags: WEB, XSS

```
37  var comment = html.replace('USER', result[i].user);
38  comment = comment.replace('DATETIME', result[i].dateTime);
39  comment = comment.replace('COMMENT', result[i].text);
40  $("#list").append(comment);
41  }
42
43  });
```

Package: src.main.resources.lessons.xxe.js

src/main/resources/lessons/xxe/js/xxe.js, line 78 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/xxe/js/xxe.js:72

```
69  '</li>';
70
71 function getComments(field) {
72  $.get("xxe/comments", function (result, status) {
73  $(field).empty();
74  for (var i = 0; i < result.length; i++) {
75  var comment = html.replace('USER', result[i].user);
```

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.lessons.xxe.js

src/main/resources/lessons/xxe/js/xxe.js, line 78 (Cross-Site Scripting: DOM) — Critical

Sink: ~JS_Generic.append()
Enclosing Method: lambda()
File: src/main/resources/lessons/xxe/js/xxe.js:78
Taint Flags: WEB, XSS

```
75  var comment = html.replace('USER', result[i].user);
76  comment = comment.replace('DATETIME', result[i].dateTime);
77  comment = comment.replace('COMMENT', result[i].text);
78  $(field).append(comment);
79  }
80
81  });
```

Package: src.main.resources.webgoat.static.js.goatApp.support

src/main/resources/webgoat/static/js/goatApp/support/GoatUtils.js, line 57 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/webgoat/static/js/goatApp/support/GoatUtils.js:56

```
53  },
54
55  showLessonCookiesAndParams: function() {
56  $.get(goatConstants.cookieService, {}, function(reply) {
57  $("#lesson_cookies").html(reply);
58  }, "html");
59  },
```

Sink Details

Sink: ~JS_Generic.html()
Enclosing Method: lambda()
File: src/main/resources/webgoat/static/js/goatApp/support/GoatUtils.js:57
Taint Flags: WEB, XSS

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.webgoat.static.js.goatApp.support

src/main/resources/webgoat/static/js/goatApp/support/GoatUtils.js, line 57 (Cross-Site Scripting: DOM) — Critical

```
54
55  showLessonCookiesAndParams: function() {
56  $.get(goatConstants.cookieService, {}, function(reply) {
57  $("#lesson_cookies").html(reply);
58  }, "html");
59  },
60
```

Package: src.main.resources.webgoat.static.js.jquery_form

src/main/resources/webgoat/static/js/jquery_form/jquery.form.js, line 346 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read href
From: ajaxSubmit
File: src/main/resources/webgoat/static/js/jquery_form/jquery.form.js:115

```
112  action = options.url || this.attr2('action');
113
114  url = (typeof action === 'string') ? $.trim(action) : '';
115  url = url || window.location.href || '';
116  if (url) {
117  // clean url (don't include hash vaue)
118  url = (url.match(/^([^#]+)/)||[])[1];
```

Sink Details

Sink: ~JS_Generic.ajax()
Enclosing Method: fileUploadXhr()
File: src/main/resources/webgoat/static/js/jquery_form/jquery.form.js:346
Taint Flags: VALIDATED_OPEN_REDIRECT, WEB, XSS

```
343  beforeSend.call(this, xhr, o);
344  }
345  };
346  return $.ajax(s);
347  }
348
349  // private function for handling file uploads (hat tip to YAHOO!)
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: DOM — Critical

Package: src.main.resources.webgoat.static.js.libs

src/main/resources/webgoat/static/js/libs/jquery.form.js, line 346 (Cross-Site Scripting: DOM) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read href
From: ajaxSubmit
File: src/main/resources/webgoat/static/js/libs/jquery.form.js:115

```
112  action = options.url || this.attr2('action');
113
114  url = (typeof action === 'string') ? $.trim(action) : '';
115  url = url || window.location.href || '';
116  if (url) {
117  // clean url (don't include hash vaue)
118  url = (url.match(/^([^#]+)/)||[])[1];
```

Sink Details

Sink: ~JS_Generic.ajax()
Enclosing Method: fileUploadXhr()
File: src/main/resources/webgoat/static/js/libs/jquery.form.js:346
Taint Flags: VALIDATED_OPEN_REDIRECT, WEB, XSS

```
343  beforeSend.call(this, xhr, o);
344  }
345  };
346  return $.ajax(s);
347  }
348
349  // private function for handling file uploads (hat tip to YAHOO!)
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: Self (2 issues)

Abstract

Sending unvalidated data to a web browser can result in the browser executing malicious code.

Explanation

Cross-site scripting (XSS) vulnerabilities occur when: 1. Data enters a web application through an untrusted source. In the case of self-XSS, data is read from a text box or other value that can be controlled from the DOM and written back into the page using client-side code. 2. The data is included in dynamic content that is sent to a web user without validation. In the case of self-XSS, malicious content is executed as part of DOM (Document Object Model) modification. The malicious content in the case of self-XSS takes the form of a JavaScript segment, or any other type of code that the browser executes. As self-XSS is primarily an attack on oneself, it is often considered unimportant, but should be treated the same as a standard XSS weakness if one of the following can occur: - A Cross-Site Request Forgery vulnerability is identified on your website. - A social engineering attack can convince a user to attack their own account, compromising their session. **Example 1:** Consider the HTML form:

```
<div id="myDiv">
  Employee ID: <input type="text" id="eid"><br>
  ...
  <button>Show results</button>
</div>
<div id="resultsDiv">
  ...
</div>
```

The following jQuery code segment reads an employee ID from the text box, and displays it to the user.

```
$(document).ready(function(){
  $("#myDiv").on("click", "button", function(){
    var eid = $("#eid").val();
    $("resultsDiv").append(eid);
    ...
  });
});
```

These code examples operate correctly if the employee ID from the text input with ID `eid` contains only standard alphanumeric text. If `eid` has a value that includes metacharacters or source code, then after the user clicks the button, the code is added to the DOM for the browser to execute. If an attacker can convince a user to input malicious input into the text input, then this is simply a DOM-based XSS.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

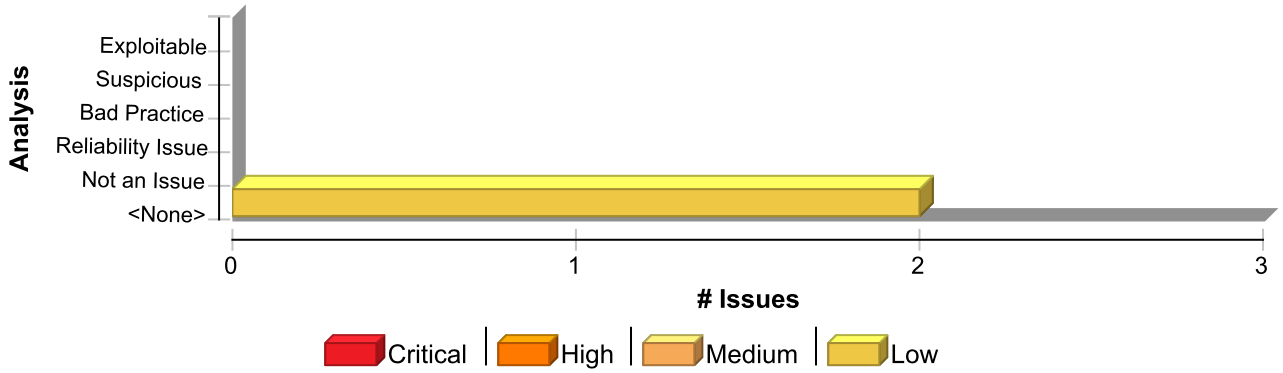
© Copyright [2008-2024] Micro Focus or one of its affiliates.

The solution to prevent XSS is to ensure that validation occurs in the required places and that relevant properties are set to prevent vulnerabilities. Because XSS vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application (or just before rendered, if DOM-based). However, because web applications often have complex and intricate code for generating dynamic content, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for XSS. Web applications must validate all input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for XSS is generally relatively easy. Despite its value, input validation for XSS does not take the place of rigorous output validation. An application might accept input through a shared data store or other trusted source, and that data store might accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means that the best way to prevent XSS vulnerabilities is to validate everything that enters the application and leaves the application destined for the user. The most secure approach to validation for XSS is to create an allow list of safe characters that can appear in HTTP content and accept input composed exclusively of characters in the approved set. For example, a valid username might only include alphanumeric characters or a phone number might only include digits 0-9. However, this solution is often infeasible in web applications because many characters that have special meaning to the browser must be considered valid input after they are encoded, such as a web design bulletin board that must accept HTML fragments from its users. A more flexible, but less secure approach is to implement a deny list, which selectively rejects or escapes potentially dangerous characters before using the input. To form such a list, you first need to understand the set of characters that hold special meaning for web browsers. Although the HTML standard defines which characters have special meaning, many web browsers try to correct common mistakes in HTML and might treat other characters as special in certain contexts. This is why we do not recommend the use of deny lists as a means to prevent XSS. The CERT(R) Coordination Center at the Software Engineering Institute at Carnegie Mellon University provides the following details about special characters in various contexts [1]: In the content of a block-level element (in the middle of a paragraph of text): - "<" is special because it introduces a tag. - "&" is special because it introduces a character entity. - ">" is special because some browsers treat it as special, on the assumption that the author of the page intended to include an opening "<", but omitted it in error. The following principles apply to attribute values: - In attribute values enclosed in double quotes, the double quotes are special because they mark the end of the attribute value. - In attribute values enclosed in single quote, the single quotes are special because they mark the end of the attribute value. - In attribute values without any quotes, white-space characters, such as space and tab, are special. - "&" is special when used with certain attributes, because it introduces a character entity. In URLs, for example, a search engine might provide a link within the results page that the user can click to re-run the search. This can be implemented by encoding the search query inside the URL, which introduces additional special characters: - Space, tab, and new line are special because they mark the end of the URL. - "&" is special because it either introduces a character entity or separates CGI parameters. - Non-ASCII characters (that is, everything greater than 127 in the ISO-8859-1 encoding) are not allowed in URLs, so they are considered to be special in this context. - The "%" symbol must be filtered from input anywhere parameters encoded with HTTP escape sequences are decoded by server-side code. For example, "%" must be filtered if input such as "%68%65%6C%6C%6F" becomes "hello" when it appears on the web page. Within the body of a : - Semicolons, parentheses, curly braces, and new line characters must be filtered out in situations where text could be inserted directly into a pre-existing script tag. Server-side scripts: - Server-side scripts that convert any exclamation characters (!) in input to double-quote characters (") on output might require additional filtering. Other possibilities: - If an attacker submits a request in UTF-7, the special character '<' appears as '+ADw-' and might bypass filtering. If the output is included in a page that does not explicitly specify an encoding format, then some browsers try to intelligently identify the encoding based on the content (in this case, UTF-7). After you identify the correct points in an application to perform validation for XSS attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. If special characters are not considered valid input to the application, then you can reject any input that contains special characters as invalid. A second option is to remove special characters with filtering. However, filtering has the side effect of changing any visual representation of the filtered content and might be unacceptable in circumstances where the integrity of the input must be preserved for display. If input

© Copyright [2008-2024] Micro Focus or one of its affiliates.

containing special characters must be accepted and displayed accurately, validation must encode any special characters to remove their significance. A complete list of ISO 8859-1 encoded values for special characters is provided as part of the official HTML specification [2]. Many application servers attempt to limit an application's exposure to cross-site scripting vulnerabilities by providing implementations for the functions responsible for setting certain specific HTTP response content that perform validation for the characters essential to a cross-site scripting attack. Do not rely on the server running your application to make it secure. For any developed application, there are no guarantees about which application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will continue to stay in sync.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cross-Site Scripting: Self	2	0	0	2
Total	**2**	**0**	**0**	**2**

Cross-Site Scripting: Self	**Low**
Package: src.main.resources.lessons.clientsidefiltering.js	
src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js, line 17 (Cross-Site Scripting: Self)	**Low**

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read value
From: fetchUserData
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js:12

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: Self — Low

Package: src.main.resources.lessons.clientsidefiltering.js

src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js, line 17 (Cross-Site Scripting: Self) — Low

```
9  function fetchUserData() {
10   if (!dataFetched) {
11   dataFetched = true;
12   ajaxFunction(document.getElementById("userID").value);
13   }
14 }
15
```

Sink Details

Sink: ~JS_Generic.get()
Enclosing Method: ajaxFunction()
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js:17
Taint Flags: SELF_XSS, WEB

```
14 }
15
16 function ajaxFunction(userId) {
17  $.get("clientSideFiltering/salaries?userId=" + userId, function (result, status) {
18  var html = "<table border = '1' width = '90%' align = 'center'";
19  html = html + '<tr>';
20  html = html + '<td>UserID</td>';
```

src/main/resources/lessons/clientsidefiltering/js/clientSideFilteringFree.js, line 41 (Cross-Site Scripting: Self) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: ~JS_Generic.val()
From: lambda
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFilteringFree.js:40

```
37   calculate();
38   })
39   $(".checkoutCode").on("blur", function () {
40   var checkoutCode = $(".checkoutCode").val();
41   $.get("clientSideFiltering/challenge-store/coupons/" + checkoutCode,
function (result, status) {
42   var discount = result.discount;
43   if (discount > 0) {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Cross-Site Scripting: Self	Low
Package: src.main.resources.lessons.clientsidefiltering.js	
src/main/resources/lessons/clientsidefiltering/js/clientSideFilteringFree.js, line 41 (Cross-Site Scripting: Self)	Low

Sink Details

Sink: ~JS_Generic.get()
Enclosing Method: lambda()
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFilteringFree.js:41
Taint Flags: SELF_XSS, WEB

```
38  })
39  $(".checkoutCode").on("blur", function () {
40  var checkoutCode = $(".checkoutCode").val();
41  $.get("clientSideFiltering/challenge-store/coupons/" + checkoutCode, function (result,
status) {
42  var discount = result.discount;
43  if (discount > 0) {
44  $('#discount').text(discount);
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Field (9 issues)

Abstract

This field is never used.

Explanation

This field is never accessed, except perhaps by dead code. Dead code is defined as code that is never directly or indirectly executed by a public method. It is likely that the field is simply vestigial, but it is also possible that the unused field points out a bug. **Example 1:** The field named `glue` is not used in the following class. The author of the class has accidentally put quotes around the field name, transforming it into a string constant.

```
public class Dead {

  String glue;

  public String getGlue() {
    return "glue";
  }

}
```

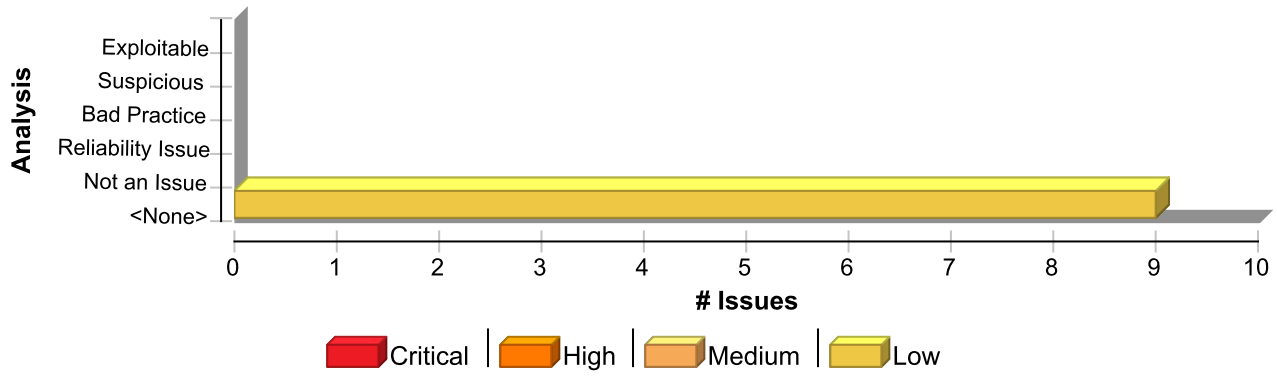
Example 2: The field named `glue` is used in the following class, but only from a method that is never called.

```
public class Dead {

  String glue;

  private String getGlue() {
    return glue;
  }

}
```

Recommendation

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary



© Copyright [2008-2024] Micro Focus or one of its affiliates.

Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Unused Field	9	0	0	9
Total	**9**	**0**	**0**	**9**

Dead Code: Unused Field — Low

Package: org.owasp.webgoat.container.plugins

src/test/java/org/owasp/webgoat/container/plugins/LessonTest.java, line 48 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: clientRegistrationRepository
File: src/test/java/org/owasp/webgoat/container/plugins/LessonTest.java:48
Taint Flags:

```
45  @MockBean protected WebSession webSession;
46  @MockBean private Language language;
47
48  @MockBean private ClientRegistrationRepository clientRegistrationRepository;
49
50  @Value("${webgoat.user.directory}")
51  protected String webGoatHomeDirectory;
```

Package: org.owasp.webgoat.container.service

src/test/java/org/owasp/webgoat/container/service/HintServiceTest.java, line 29 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: assignment
File: src/test/java/org/owasp/webgoat/container/service/HintServiceTest.java:29
Taint Flags:

```
26  private MockMvc mockMvc;
27  @Mock private WebSession websession;
28  @Mock private Lesson lesson;
29  @Mock private Assignment assignment;
30
31  @BeforeEach
32  void setup() {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Field — Low

Package: org.owasp.webgoat.lessons.authbypass

src/main/java/org/owasp/webgoat/lessons/authbypass/VerifyAccount.java, line 53 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: webSession
File: src/main/java/org/owasp/webgoat/lessons/authbypass/VerifyAccount.java:53
Taint Flags:

```
50 })
51 public class VerifyAccount extends AssignmentEndpoint {
52
53  @Autowired private WebSession webSession;
54
55  @Autowired UserSessionData userSessionData;
56
```

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignment.java, line 53 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: webGoatHomeDirectory
File: src/main/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignment.java:53
Taint Flags:

```
50  };
51
52  @Value("${webgoat.server.directory}")
53  private String webGoatHomeDirectory;
54
55  @Autowired private WebSession webSession;
56  @Autowired private CommentsCache comments;
```

src/main/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignment.java, line 55 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Field — Low

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignment.java, line 55 (Dead Code: Unused Field) — Low

Sink Details

Sink: Field: webSession
File: src/main/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignment.java:55
Taint Flags:

```
52  @Value("${webgoat.server.directory}")
53  private String webGoatHomeDirectory;
54
55  @Autowired private WebSession webSession;
56  @Autowired private CommentsCache comments;
57
58  @PostMapping(path = "xxe/content-type")
```

src/main/java/org/owasp/webgoat/lessons/xxe/SimpleXXE.java, line 64 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: webGoatHomeDirectory
File: src/main/java/org/owasp/webgoat/lessons/xxe/SimpleXXE.java:64
Taint Flags:

```
61  };
62
63  @Value("${webgoat.server.directory}")
64  private String webGoatHomeDirectory;
65
66  @Value("${webwolf.landingpage.url}")
67  private String webWolfURL;
```

src/main/java/org/owasp/webgoat/lessons/xxe/SimpleXXE.java, line 67 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: webWolfURL
File: src/main/java/org/owasp/webgoat/lessons/xxe/SimpleXXE.java:67
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Field — Low

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/SimpleXXE.java, line 67 (Dead Code: Unused Field) — Low

```
64  private String webGoatHomeDirectory;
65
66  @Value("${webwolf.landingpage.url}")
67  private String webWolfURL;
68
69  @Autowired private CommentsCache comments;
70
```

Package: org.owasp.webgoat.webwolf.mailbox

src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java, line 61 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: userService
File: src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java:61
Taint Flags:

```
58  @MockBean private MailboxRepository mailbox;
59
60  @MockBean private ClientRegistrationRepository clientRegistrationRepository;
61  @MockBean private UserService userService;
62  @Autowired private ObjectMapper objectMapper;
63
64  @JsonIgnoreProperties("time")
```

src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java, line 60 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: clientRegistrationRepository
File: src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java:60
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Field	Low
Package: org.owasp.webgoat.webwolf.mailbox	
src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java, line 60 (Dead Code: Unused Field)	Low

```
57  @Autowired private MockMvc mvc;
58  @MockBean private MailboxRepository mailbox;
59
60  @MockBean private ClientRegistrationRepository clientRegistrationRepository;
61  @MockBean private UserService userService;
62  @Autowired private ObjectMapper objectMapper;
63
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method (27 issues)

Abstract

This method is not reachable from any method outside the class.

Explanation

This method is never called or is only called from other dead code. **Example 1:** In the following class, the method `doWork()` can never be called.

```
public class Dead {
  private void doWork() {
    System.out.println("doing work");
  }
  public static void main(String[] args) {
    System.out.println("running Dead");
  }
}
```

Example 2: In the following class, two private methods call each other, but since neither one is ever invoked from anywhere else, they are both dead code.

```
public class DoubleDead {
  private void doTweedledee() {
    doTweedledumb();
  }
  private void doTweedledumb() {
    doTweedledee();
  }
  public static void main(String[] args) {
    System.out.println("running DoubleDead");
  }
}
```

(In this case it is a good thing that the methods are dead: invoking either one would cause an infinite loop.)

Recommendation

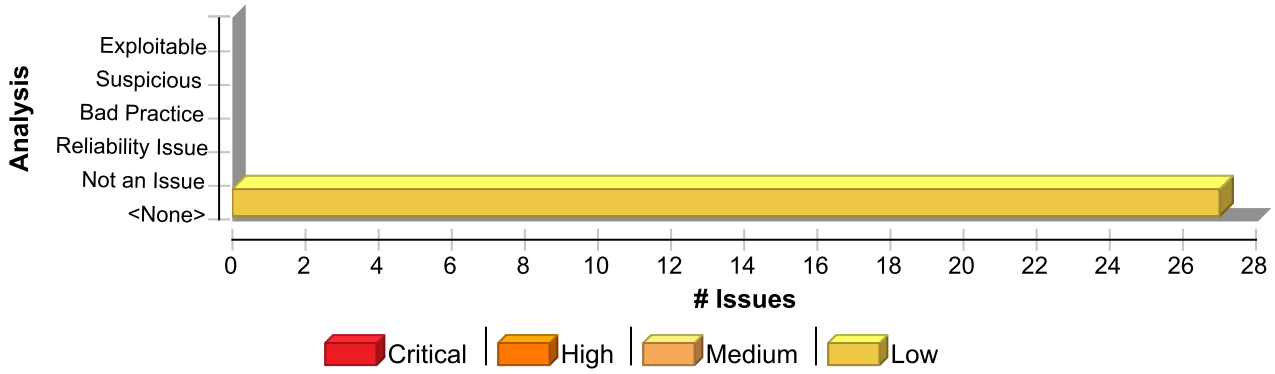
A dead method may indicate a bug in dispatch code. **Example 3:** If method is flagged as dead named `getWitch()` in a class that also contains the following dispatch method, it may be because of a copy-and-paste error. The 'w' case should return `getWitch()` not `getMummy()`.

```
public ScaryThing getScaryThing(char st) {
  switch(st) {
    case 'm':
      return getMummy();
    case 'w':
      return getMummy();
    default:
      return getBlob();
  }
}
```

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Unused Method	27	0	0	27
Total	**27**	**0**	**0**	**27**

Dead Code: Unused Method — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 114 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: callTrickHtml
Enclosing Method: callTrickHtml()
File: src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java:114
Taint Flags:

```
111   .asString();
112   }
113
114   private String callTrickHtml(String htmlName) {
115   String result =
116   RestAssured.given()
117   .when()
```

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 175 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 175 (Dead Code: Unused Method) — Low

Sink: Function: checkAssignment7
Enclosing Method: checkAssignment7()
File: src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java:175
Taint Flags:

```
172  assertEquals(true, result);
173  }
174
175  private void checkAssignment7(String goatURL) {
176  Map<String, Object> params = new HashMap<>();
177  params.put(
178  "{\"name\":\"WebGoat\",\"email\":\"webgoat@webgoat.org\",\"content\":\"WebGoat is the"
```

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 204 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: checkAssignment8
Enclosing Method: checkAssignment8()
File: src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java:204
Taint Flags:

```
201  checkAssignment(url("csrf/feedback"), params, true);
202  }
203
204  private void checkAssignment8(String goatURL) {
205
206  // first make sure there is an attack csrf- user
207  registerCSRFUser();
```

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 275 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: registerCSRFUser
Enclosing Method: registerCSRFUser()
File: src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java:275
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 275 (Dead Code: Unused Method) — Low

```
272  }
273
274  /** Try to register the new user. Ignore the result. */
275  private void registerCSRFUser() {
276
277  RestAssured.given()
278  .when()
```

src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java, line 82 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: assignment3
Enclosing Method: assignment3()
File: src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java:82
Taint Flags:

```
79  CoreMatchers.is(true));
80  }
81
82  private void assignment3() throws IOException {
83  MatcherAssert.assertThat(
84  RestAssured.given()
85  .when()
```

src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java, line 118 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: assignment5
Enclosing Method: assignment5()
File: src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java:118
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java, line 118 (Dead Code: Unused Method) — Low

```
115  true);
116  }
117
118  private void assignment5() throws IOException {
119  var webGoatHome = webGoatServerDirectory() + "PathTraversal/" + this.getUser();
120  webGoatHome =
121  webGoatHome.replaceAll("^[a-zA-Z]:", ""); // Remove C: from the home directory on Windows
```

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 152 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: checkAssignment4
Enclosing Method: checkAssignment4()
File: src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java:152
Taint Flags:

```
149  checkAssignment(url("csrf/confirm-flag-1"), params, true);
150  }
151
152  private void checkAssignment4(String goatURL) {
153
154  Map<String, Object> params = new HashMap<>();
155  params.clear();
```

src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java, line 66 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: assignment2
Enclosing Method: assignment2()
File: src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java:66
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java, line 66 (Dead Code: Unused Method) — Low

```
63  CoreMatchers.is(true));
64  }
65
66  private void assignment2() throws IOException {
67  MatcherAssert.assertThat(
68  RestAssured.given()
69  .when()
```

src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java, line 50 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: assignment1
Enclosing Method: assignment1()
File: src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java:50
Taint Flags:

```
47  dynamicTest("assignment 5 - zip slip", () -> assignment5()));
48  }
49
50  private void assignment1() throws IOException {
51  MatcherAssert.assertThat(
52  RestAssured.given()
53  .when()
```

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 133 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: checkAssignment3
Enclosing Method: checkAssignment3()
File: src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java:133
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 133 (Dead Code: Unused Method) — Low

```
130  return result;
131  }
132
133  private void checkAssignment3(String goatURL) {
134  String flag =
135  RestAssured.given()
136  .when()
```

src/it/java/org/owasp/webgoat/IDORIntegrationTest.java, line 36 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: loginIDOR
Enclosing Method: loginIDOR()
File: src/it/java/org/owasp/webgoat/IDORIntegrationTest.java:36
Taint Flags:

```
33  checkResults("/IDOR");
34  }
35
36  private void loginIDOR() {
37
38  Map<String, Object> params = new HashMap<>();
39  params.put("username", "tom");
```

src/it/java/org/owasp/webgoat/IDORIntegrationTest.java, line 45 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: profile
Enclosing Method: profile()
File: src/it/java/org/owasp/webgoat/IDORIntegrationTest.java:45
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/IDORIntegrationTest.java, line 45 (Dead Code: Unused Method) — Low

```
42  checkAssignment(url("IDOR/login"), params, true);
43  }
44
45  private void profile() {
46
47  // View profile - assignment 3a
48  MatcherAssert.assertThat(
```

src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java, line 100 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: assignment4
Enclosing Method: assignment4()
File: src/it/java/org/owasp/webgoat/PathTraversalIntegrationTest.java:100
Taint Flags:

```
97  CoreMatchers.is(true));
98  }
99
100  private void assignment4() throws IOException {
101  var uri = "PathTraversal/random-picture?id=%2E%2E%2F%2E%2E%2Fpath-traversal-secret";
102  RestAssured.given()
103  .urlEncodingEnabled(false)
```

Package: org.owasp.webgoat.container.lessons

src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java, line 95 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: methodReturnTypeIsOfTypeAttackResult
Enclosing Method: methodReturnTypeIsOfTypeAttackResult()
File: src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java:95
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat.container.lessons

src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java, line 95 (Dead Code: Unused Method) — Low

```
92  + " 'ResponseEntity<AttackResult>' please consider adding one");
93  }
94
95  private boolean methodReturnTypeIsOfTypeAttackResult(Method m) {
96  if (m.getReturnType() == AttackResult.class) {
97  return true;
98  }
```

src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java, line 79 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getPath
Enclosing Method: getPath()
File: src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java:79
Taint Flags:

```
76  .toList();
77  }
78
79  private String getPath(Class<? extends AssignmentEndpoint> e) {
80  for (Method m : e.getMethods()) {
81  if (methodReturnTypeIsOfTypeAttackResult(m)) {
82  var mapping = getMapping(m);
```

src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java, line 107 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getMapping
Enclosing Method: getMapping()
File: src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java:107
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat.container.lessons

src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java, line 107 (Dead Code: Unused Method) — Low

```
104  return false;
105  }
106
107  private String getMapping(Method m) {
108  String[] paths = null;
109  // Find the path, either it is @GetMapping("/attack") of GetMapping(path = "/attack")
both are
110  // valid, we need to consider both
```

src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java, line 137 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getHints
Enclosing Method: getHints()
File: src/main/java/org/owasp/webgoat/container/lessons/CourseConfiguration.java:137
Taint Flags:

```
134  }
135  }
136
137  private List<String> getHints(Class<? extends AssignmentEndpoint> e) {
138  if (e.isAnnotationPresent(AssignmentHints.class)) {
139  return List.of(e.getAnnotationsByType(AssignmentHints.class)[0].value());
140  }
```

Package: org.owasp.webgoat.container.users

src/main/java/org/owasp/webgoat/container/users/Scoreboard.java, line 70 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: toLessonTitle
Enclosing Method: toLessonTitle()
File: src/main/java/org/owasp/webgoat/container/users/Scoreboard.java:70
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat.container.users

src/main/java/org/owasp/webgoat/container/users/Scoreboard.java, line 70 (Dead Code: Unused Method) — Low

```
67  .toList();
68  }
69
70  private String toLessonTitle(String id) {
71  String titleKey =
72  course.getLessons().stream()
73  .filter(l -> l.getId().equals(id))
```

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5Test.java, line 43 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: providedForMD5Values
Enclosing Method: providedForMD5Values()
File: src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5Test.java:43
Taint Flags:

```
40  assertThat(out).isEqualTo(MD5.getHashString(in.getBytes()));
41  }
42
43  private static Stream<Arguments> providedForMD5Values() {
44  return Stream.of(
45  Arguments.of("", "d41d8cd98f00b204e9800998ecf8427e"),
46  Arguments.of("a string", "3a315533c0f34762e0c45e3d4e9d525c"));
```

Package: org.owasp.webgoat.lessons.hijacksession.cas

src/test/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProviderTest.java, line 119 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: authenticationForCookieValues
Enclosing Method: authenticationForCookieValues()
File: src/test/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProviderTest.java:119
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat.lessons.hijacksession.cas

src/test/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProviderTest.java, line 119 (Dead Code: Unused Method) — Low

```
116  assertThat(provider.getSessionsSize(),
is(HijackSessionAuthenticationProvider.MAX_SESSIONS));
117  }
118
119  private static Stream<Arguments> authenticationForCookieValues() {
120  return Stream.of(
121  Arguments.of((Object) null),
122  Arguments.of(Authentication.builder().name("any").credentials("any").build()),
```

Package: org.owasp.webgoat.lessons.jwt.claimmisuse

src/main/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpoint.java, line 60 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: JWTHeaderKIDEndpoint
Enclosing Method: JWTHeaderKIDEndpoint()
File: src/main/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpoint.java:60
Taint Flags:

```
57
58  private final LessonDataSource dataSource;
59
60  private JWTHeaderKIDEndpoint(LessonDataSource dataSource) {
61  this.dataSource = dataSource;
62  }
63
```

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 199 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: providedCookieValues
Enclosing Method: providedCookieValues()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:199
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 199 (Dead Code: Unused Method) — Low

```
196   .andExpect(cookie().value(COOKIE_NAME, ""));
197   }
198
199   private static Stream<Arguments> providedCookieValues() {
200   return Stream.of(
201   Arguments.of("NjI2MTcwNGI3YTQxNGE1OTUNzQ2ZDZmNzQ="),
202   Arguments.of("NjI2MTcwNGI3YTQxNGE1OTU2NzQ3NDYxNmY2NzYyNjU3Nw=="),
```

Package: org.owasp.webgoat.lessons.spoofcookie.encoders

src/test/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDecTest.java, line 74 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: providedForEncValues
Enclosing Method: providedForEncValues()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDecTest.java:74
Taint Flags:

```
71   assertThat(EncDec.decode(null)).isNull();
72   }
73
74   private static Stream<Arguments> providedForEncValues() {
75   return Stream.of(
76   Arguments.of("webgoat", "YxNmY2NzYyNjU3Nw=="),
77   Arguments.of("admin", "2ZTY5NmQ2NDYx"),
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDecTest.java, line 81 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: providedForDecValues
Enclosing Method: providedForDecValues()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDecTest.java:81
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat.lessons.spoofcookie.encoders

src/test/java/org/owasp/webgoat/lessons/spoofcookie/encoders/EncDecTest.java, line 81 (Dead Code: Unused Method) — Low

```
78  Arguments.of("tom", "2ZDZmNzQ="));
79  }
80
81  private static Stream<Arguments> providedForDecValues() {
82  return Stream.of(
83  Arguments.of("webgoat", "NjI2MTcwNGI3YTQxNGE1OTU2NzQ3NDYxNmY2NzYyNjU3Nw=="),
84  Arguments.of("admin", "NjI2MTcwNGI3YTQxNGE1OTU2NzQ2ZTY5NmQ2NDYx"),
```

Package: org.owasp.webgoat.webwolf.requests

src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java, line 97 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: toJsonString
Enclosing Method: toJsonString()
File: src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java:97
Taint Flags:

```
94  return t.getRequest().getUri().getPath();
95  }
96
97  private String toJsonString(HttpExchange t) {
98  try {
99  return objectMapper.writeValueAsString(t);
100  } catch (JsonProcessingException e) {
```

src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java, line 93 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: path
Enclosing Method: path()
File: src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java:93
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dead Code: Unused Method — Low

Package: org.owasp.webgoat.webwolf.requests

src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java, line 93 (Dead Code: Unused Method) — Low

```
90  return allowed;
91  }
92
93  private String path(HttpExchange t) {
94  return t.getRequest().getUri().getPath();
95  }
96
```

src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java, line 77 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: allowedTrace
Enclosing Method: allowedTrace()
File: src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java:77
Taint Flags:

```
74  return model;
75  }
76
77  private boolean allowedTrace(HttpExchange t, String username) {
78  HttpExchange.Request req = t.getRequest();
79  boolean allowed = true;
80  /* do not show certain traces to other users in a classroom setup */
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Denial of Service (1 issue)

Abstract

An attacker could cause the program to crash or otherwise become unavailable to legitimate users.

Explanation

Attackers may be able to deny service to legitimate users by flooding the application with requests, but flooding attacks can often be defused at the network layer. More problematic are bugs that allow an attacker to overload the application using a small number of requests. Such bugs allow the attacker to specify the quantity of system resources their requests will consume or the duration for which they will use them. **Example 1:** The following code allows a user to specify the amount of time for which a thread will sleep. By specifying a large number, an attacker may tie up the thread indefinitely. With a small number of requests, the attacker may deplete the application's thread pool.

```
int usrSleepTime = Integer.parseInt(usrInput);
Thread.sleep(usrSleepTime);
```

Example 2: The following code reads a String from a zip file. Because it uses the `readLine()` method, it will read an unbounded amount of input. An attacker may take advantage of this code to cause an `OutOfMemoryException` or to consume a large amount of memory so that the program spends more time performing garbage collection or runs out of memory during some subsequent operation.

```
InputStream zipInput = zipFile.getInputStream(zipEntry);
Reader zipReader = new InputStreamReader(zipInput);
BufferedReader br = new BufferedReader(zipReader);
String line = br.readLine();
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

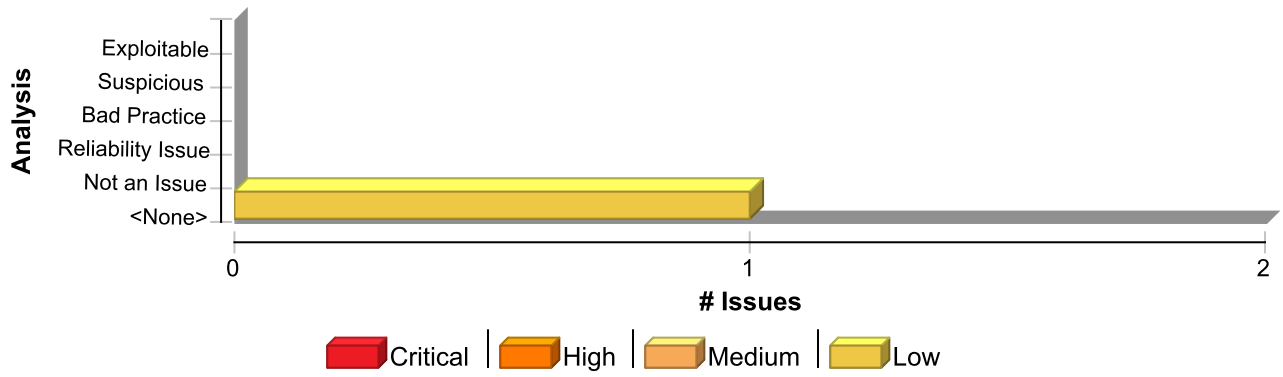
Validate user input to ensure that it will not cause inappropriate resource utilization. **Example 3:** The following code allows a user to specify the amount of time for which a thread will sleep just as in `Example 1`, but only if the value is within reasonable bounds.

```
  int usrSleepTime = Integer.parseInt(usrInput);
  if (usrSleepTime >= SLEEP_MIN &&
      usrSleepTime <= SLEEP_MAX) {
    Thread.sleep(usrSleepTime);
  } else {
    throw new Exception("Invalid sleep duration");
  }
}
```

Example 4: The following code reads a String from a zip file just as in `Example 2`, but the maximum string length it will read is `MAX_STR_LEN` characters.

```
  InputStream zipInput = zipFile.getInputStream(zipEntry);
  Reader zipReader = new InputStreamReader(zipInput);
  BufferedReader br = new BufferedReader(zipReader);
  StringBuffer sb = new StringBuffer();
  int intC;
  while ((intC = br.read()) != -1) {
    char c = (char) intC;
    if (c == '\n') {
      break;
    }
    if (sb.length() >= MAX_STR_LEN) {
      throw new Exception("input too long");
    }
    sb.append(c);
  }
  String line = sb.toString();
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Denial of Service	1	0	0	1
Total	**1**	**0**	**0**	**1**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Denial of Service	Low
Package: org.dummy.insecure.framework	
src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 68 (Denial of Service)	**Low**

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: readLine()
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:68
Taint Flags:

```
65  Process p = Runtime.getRuntime().exec(taskAction);
66  BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
67  String line = null;
68  while ((line = in.readLine()) != null) {
69  log.info(line);
70  }
71  } catch (IOException e) {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dockerfile Misconfiguration: Default User Privilege (1 issue)

Abstract

The Dockerfile does not specify a USER, so it defaults to running with a root user.

Explanation

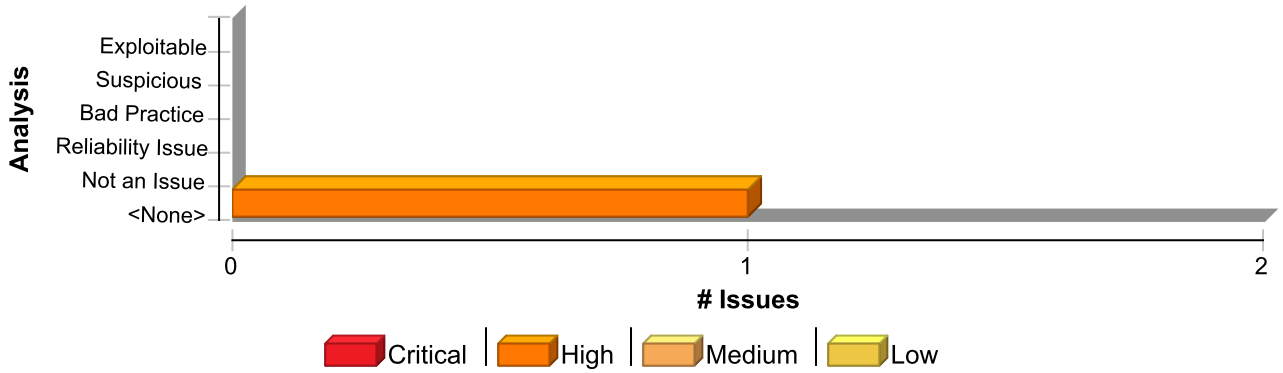
When a `Dockerfile` does not specify a USER, Docker containers run with super user privileges by default. These super user privileges are propagated to the code running inside the container, which is usually more permission than necessary. Running the Docker container with super user privileges broadens the attack surface which might enable attackers to perform more serious forms of exploitation.

Recommendation

It is good practice to run your containers as a non-root user when possible. To modify a docker container to use a non-root user, the Dockerfile needs to specify a different user, such as:

```
RUN useradd myLowPrivilegeUser
USER myLowPrivilegeUser
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dockerfile Misconfiguration: Default User Privilege	1	0	0	1
Total	**1**	**0**	**0**	**1**

Dockerfile Misconfiguration: Default User Privilege — High

Package: <none>

Dockerfile_desktop, line 1 (Dockerfile Misconfiguration: Default User Privilege) — High

Issue Details

Kingdom: Environment
Scan Engine: SCA (Configuration)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dockerfile Misconfiguration: Default User Privilege — High

Package: <none>

Dockerfile_desktop, line 1 (Dockerfile Misconfiguration: Default User Privilege) — High

Sink: FROM
File: Dockerfile_desktop:1
Taint Flags:

```
1 FROM lscr.io/linuxserver/webtop:ubuntu-xfce
2 LABEL NAME = "WebGoat: A deliberately insecure Web Application"
3 LABEL maintainer = "WebGoat team"
4
5
6
7
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dockerfile Misconfiguration: Dependency Confusion (1 issue)

Abstract

Retrieving build dependencies using a non-specific version can leave the build system vulnerable to malicious binaries or cause the system to experience unexpected behavior.

Explanation

Dockerfiles can specify an unbound range of versions for dependencies and base images. If an attacker is able to add malicious versions of dependencies to a repository or trick the build system into downloading dependencies from a repository under the attacker's control, if docker is configured without specific versions of dependencies, then docker will silently download and run the compromised dependency. This type of weakness would be exploitable as a result of a supply chain attack where attackers can leverage misconfiguration by developers, typosquatting and can add malicious packages to open source repositories. An attack of this type exploits the trust in the published packages to gain access and exfiltrate data. In docker, the `latest` tag automatically indicates the version level of an image that doesn't use a digest or unique tag to provide a version for it. Docker automatically assigns the `latest` tag as mechanism to point to the most recent image manifest file. Because tags are mutable, an attacker can replace an image or layer using a `latest` (or weak tags such as `imagename-lst, imagename-last, myimage`).
Example 1: The following configuration instructs Docker to pick the base image using the latest version of `ubuntu`.

```
    FROM ubuntu:Latest
    ...
```

Docker does not validate whether the repository configured to support the package manager is trustworthy.
Example 2: The following configuration instructs the package manager `zypper` to retrive the latest version of the given package.

```
...
zypper install package
...
```

In `Example 2`, if the repository is compromised, an attacker could simply upload a version that meets the dynamic criteria and cause `zypper` to download a malicious version of the dependency.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

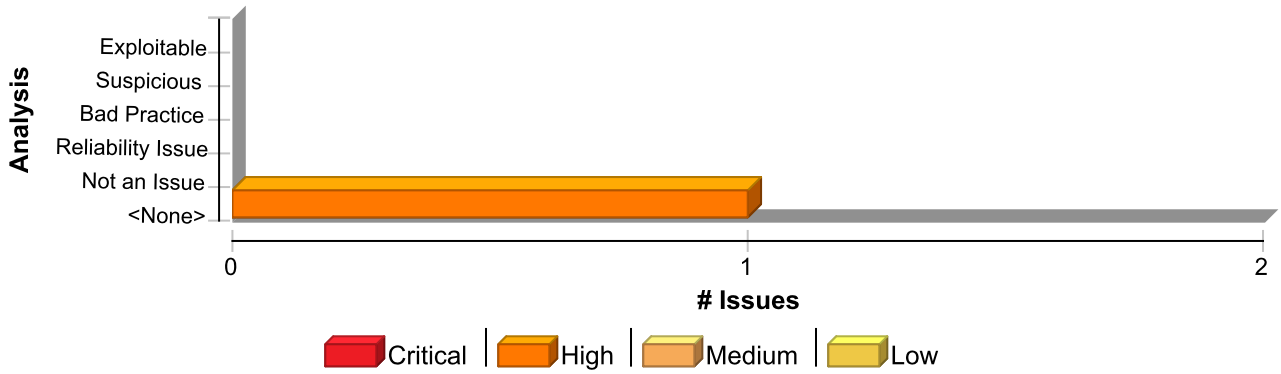
Recommendation

Perform version pinning or simple pinning. Version pinning explicitly specifies the version of images, libraries and support packages an application or system depends on. The primary goal of pinning is to ensure system stability to achieve repeatable deployments. Pinning ensures that end-users, developers, and testers all use the same code base. Pinning can additionally ensure the use of safe dependencies; those which have gone through the rigorous process of application security validation and malware detection. When you invoke zypper (or other package managers) from docker, use the following formats:

```
RUN zypper install <package_name>=<version> \
RUN gem install <package_name> --version <version>
RUN gem install <package_name> -v <version>
RUN apk add <package_name>=<version>
RUN apt-get update && apt-get install -y  \
  <package_name>=<version>  \
  <package_name>=<version> \
  <package_name>=<version> \
  && rm -rf /var/lib/apt/lists/*
```

Where is the name of the dependency to install and is the exact version or release the application should use. Fortify also recommends: - Ensure the repositories that the package managers use are trustworthy or that they are properly kept, and there is no install package substitution possible, including the addition of malicious code onto the package. - Avoid using public or untrusted repositories. - Scan packages for malware and security vulnerabilities prior to executing any regression tests. - Use digitally signed images. - Avoid using image tags such as `latest` or weak tags such as`imagename-lst, imagename-last, myimage` for deployments in production environments. - Stick to more stable tags, like specific version tags, although there is no guarantee that these cannot mutate either. - Do not create mutant tags. - Implement strict control over the source of images and their layers.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dockerfile Misconfiguration: Dependency Confusion	1	0	0	1
Total	**1**	**0**	**0**	**1**

Dockerfile Misconfiguration: Dependency Confusion — **High**

Package: <none>

Dockerfile_desktop, line 12 (Dockerfile Misconfiguration: Dependency Confusion) — **High**

Issue Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Dockerfile Misconfiguration: Dependency Confusion — High

Package: <none>

Dockerfile_desktop, line 12 (Dockerfile Misconfiguration: Dependency Confusion) — High

Kingdom: Environment
Scan Engine: SCA (Configuration)

Sink Details

Sink: RUN
File: Dockerfile_desktop:12
Taint Flags:

```
9  COPY config/desktop/start_zap.sh /config/start_zap.sh
10 COPY config/desktop/WebGoat.txt /config/Desktop/
11
12 RUN \
13  apt-get update && \
14  apt-get --yes install vim nano gzip
15
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

HTML5: Overly Permissive Message Posting Policy (1 issue)

Abstract

The program posts a cross-document message with an overly permissive target origin.

Explanation

One of the new features of HTML5 is cross-document messaging. The feature allows scripts to post messages to other windows. The corresponding API allows the user to specify the origin of the target window. However, caution should be taken when specifying the target origin because an overly permissive target origin will allow a malicious script to communicate with the victim window in an inappropriate way, leading to spoofing, data theft, relay, and other attacks. **Example 1:** The following example uses a wildcard to programmatically specify the target origin of the message to be sent.

```
o.contentWindow.postMessage(message, '*');
```

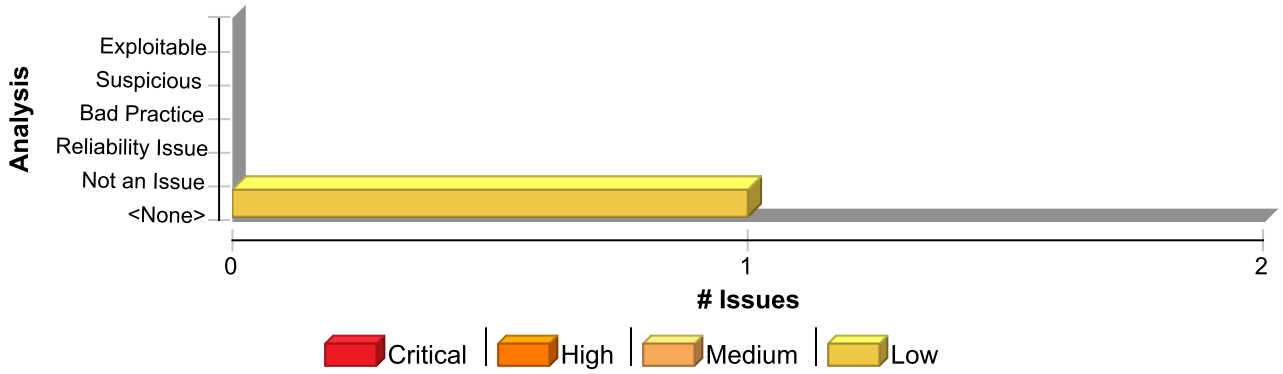
Using the * as the value of the target origin indicates that the script is sending a message to a window regardless of its origin.

Recommendation

Do not use the * as the value of the target origin. Instead, provide a specific target origin. **Example 2:** The following code provides a specific value for the target origin.

```
o.contentWindow.postMessage(message, 'www.trusted.com');
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
HTML5: Overly Permissive Message Posting Policy	1	0	0	1
Total	**1**	**0**	**0**	**1**

HTML5: Overly Permissive Message Posting Policy	**Low**
Package: src.main.resources.webgoat.static.js.libs	
src/main/resources/webgoat/static/js/libs/ace.js, line 1740 (HTML5: Overly Permissive Message Posting Policy)	**Low**
Issue Details	

© Copyright [2008-2024] Micro Focus or one of its affiliates.

HTML5: Overly Permissive Message Posting Policy	Low
Package: src.main.resources.webgoat.static.js.libs	
src/main/resources/webgoat/static/js/libs/ace.js, line 1740 (HTML5: Overly Permissive Message Posting Policy)	**Low**

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: postMessage
Enclosing Method: nextTick()
File: src/main/resources/webgoat/static/js/libs/ace.js:1740
Taint Flags:

```
1737   };
1738
1739   exports.addListener(win, "message", listener);
1740   win.postMessage(messageName, "*");
1741   };
1742 }
1743
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness (14 issues)

Abstract

Standard pseudorandom number generators cannot withstand cryptographic attacks.

Explanation

Insecure randomness errors occur when a function that can produce predictable values is used as a source of randomness in a security-sensitive context. Computers are deterministic machines, and as such are unable to produce true randomness. Pseudorandom Number Generators (PRNGs) approximate randomness algorithmically, starting with a seed from which subsequent values are calculated. There are two types of PRNGs: statistical and cryptographic. Statistical PRNGs provide useful statistical properties, but their output is highly predictable and form an easy to reproduce numeric stream that is unsuitable for use in cases where security depends on generated values being unpredictable. Cryptographic PRNGs address this problem by generating output that is more difficult to predict. For a value to be cryptographically secure, it must be impossible or highly improbable for an attacker to distinguish between the generated random value and a truly random value. In general, if a PRNG algorithm is not advertised as being cryptographically secure, then it is probably a statistical PRNG and should not be used in security-sensitive contexts, where its use can lead to serious vulnerabilities such as easy-to-guess temporary passwords, predictable cryptographic keys, session hijacking, and DNS spoofing. **Example:** The following code uses a statistical PRNG to create a URL for a receipt that remains active for some period of time after a purchase.

```
String GenerateReceiptURL(String baseUrl) {
    Random ranGen = new Random();
    ranGen.setSeed((new Date()).getTime());
    return (baseUrl + ranGen.nextInt(400000000) + ".html");
}
```

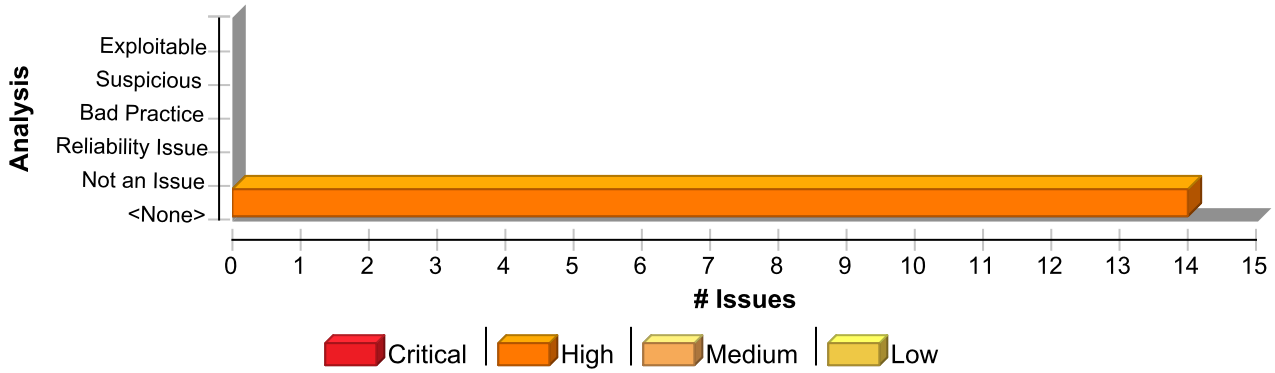
This code uses the `Random.nextInt()` function to generate "unique" identifiers for the receipt pages it generates. Since `Random.nextInt()` is a statistical PRNG, it is easy for an attacker to guess the strings it generates. Although the underlying design of the receipt system is also faulty, it would be more secure if it used a random number generator that did not produce predictable receipt identifiers, such as a cryptographic PRNG.

Recommendation

When unpredictability is critical, as is the case with most security-sensitive uses of randomness, use a cryptographic PRNG. Regardless of the PRNG you choose, always use a value with sufficient entropy to seed the algorithm. (Do not use values such as the current time because it offers only negligible entropy.) The Java language provides a cryptographic PRNG in `java.security.SecureRandom`. As is the case with other algorithm-based classes in `java.security`, `SecureRandom` provides an implementation-independent wrapper around a particular set of algorithms. When you request an instance of a `SecureRandom` object using `SecureRandom.getInstance()`, you can request a specific implementation of the algorithm. If the algorithm is available, then it is given as a `SecureRandom` object. If it is unavailable or if you do not specify a particular implementation, then you are given a `SecureRandom` implementation selected by the system. Sun provides a single `SecureRandom` implementation with the Java distribution named `SHA1PRNG`, which Sun describes as computing: "The SHA-1 hash over a true-random seed value concatenated with a 64-bit counter which is incremented by 1 for each operation. From the 160-bit SHA-1 output, only 64 bits are used [1]." However, the specifics of the Sun implementation of the `SHA1PRNG` algorithm are poorly documented, and it is unclear what sources of entropy the implementation uses and therefore what amount of true randomness exists in its output. Although there is speculation on the Web about the Sun implementation, there is no evidence to contradict the claim that the algorithm is cryptographically strong and can be used safely in security-sensitive contexts.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Insecure Randomness	14	0	0	14
Total	**14**	**0**	**0**	**14**

Insecure Randomness — High

Package: org.owasp.webgoat.lessons.challenges.challenge1

src/main/java/org/owasp/webgoat/lessons/challenges/challenge1/ImageServlet.java, line 17 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge1/ImageServlet.java:17
Taint Flags:

```
14 @RestController
15 public class ImageServlet {
16
17  public static final int PINCODE = new Random().nextInt(10000);
18
19  @RequestMapping(
20  method = {GET, POST},
```

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java, line 25 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness — High

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java, line 25 (Insecure Randomness) — High

Sink Details

Sink: nextInt()
Enclosing Method: scramble()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java:25
Taint Flags:

```
22  public static String scramble(Random random, String inputString) {
23  char[] a = inputString.toCharArray();
24  for (int i = 0; i < a.length; i++) {
25  int j = random.nextInt(a.length);
26  char temp = a[i];
27  a[i] = a[j];
28  a[j] = temp;
```

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/EncodingAssignment.java, line 52 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: getBasicAuth()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/EncodingAssignment.java:52
Taint Flags:

```
49  String username = request.getUserPrincipal().getName();
50  if (basicAuth == null) {
51  String password =
52  HashingAssignment.SECRETS[new Random().nextInt(HashingAssignment.SECRETS.length)];
53  basicAuth = getBasicAuth(username, password);
54  request.getSession().setAttribute("basicAuth", basicAuth);
55  }
```

src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java, line 53 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness — High

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java, line 53 (Insecure Randomness) — High

Sink: nextInt()
Enclosing Method: getMd5()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java:53
Taint Flags:

```
50  String md5Hash = (String) request.getSession().getAttribute("md5Hash");
51  if (md5Hash == null) {
52
53  String secret = SECRETS[new Random().nextInt(SECRETS.length)];
54
55  MessageDigest md = MessageDigest.getInstance("MD5");
56  md.update(secret.getBytes());
```

src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java, line 71 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: getSha256()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java:71
Taint Flags:

```
68
69  String sha256 = (String) request.getSession().getAttribute("sha256");
70  if (sha256 == null) {
71  String secret = SECRETS[new Random().nextInt(SECRETS.length)];
72  sha256 = getHash(secret, "SHA-256");
73  request.getSession().setAttribute("sha256Hash", sha256);
74  request.getSession().setAttribute("sha256Secret", secret);
```

Package: org.owasp.webgoat.lessons.csrf

src/main/java/org/owasp/webgoat/lessons/csrf/CSRFGetFlag.java, line 75 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness — High

Package: org.owasp.webgoat.lessons.csrf

src/main/java/org/owasp/webgoat/lessons/csrf/CSRFGetFlag.java, line 75 (Insecure Randomness) — High

Sink: nextInt()
Enclosing Method: invoke()
File: src/main/java/org/owasp/webgoat/lessons/csrf/CSRFGetFlag.java:75
Taint Flags:

```
72  response.put("flag", null);
73  } else {
74  Random random = new Random();
75  userSessionData.setValue("csrf-get-success", random.nextInt(65536));
76  response.put("success", true);
77  response.put("message", pluginMessages.getMessage("csrf-get-other-referer.success"));
78  response.put("flag", userSessionData.getValue("csrf-get-success"));
```

src/main/java/org/owasp/webgoat/lessons/csrf/CSRFGetFlag.java, line 64 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: invoke()
File: src/main/java/org/owasp/webgoat/lessons/csrf/CSRFGetFlag.java:64
Taint Flags:

```
61  response.put("flag", userSessionData.getValue("csrf-get-success"));
62  } else {
63  Random random = new Random();
64  userSessionData.setValue("csrf-get-success", random.nextInt(65536));
65  response.put("success", true);
66  response.put("message", pluginMessages.getMessage("csrf-get-other-referer.success"));
67  response.put("flag", userSessionData.getValue("csrf-get-success"));
```

src/main/java/org/owasp/webgoat/lessons/csrf/CSRFGetFlag.java, line 58 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: invoke()
File: src/main/java/org/owasp/webgoat/lessons/csrf/CSRFGetFlag.java:58
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness — High

Package: org.owasp.webgoat.lessons.csrf

src/main/java/org/owasp/webgoat/lessons/csrf/CSRFGetFlag.java, line 58 (Insecure Randomness) — High

```
55  if (referer.equals("NULL")) {
56  if ("true".equals(req.getParameter("csrf"))) {
57  Random random = new Random();
58  userSessionData.setValue("csrf-get-success", random.nextInt(65536));
59  response.put("success", true);
60  response.put("message", pluginMessages.getMessage("csrf-get-null-referer.success"));
61  response.put("flag", userSessionData.getValue("csrf-get-success"));
```

Package: org.owasp.webgoat.lessons.hijacksession.cas

src/main/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProvider.java, line 79 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextDouble()
Enclosing Method: authorizedUserAutoLogin()
File: src/main/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProvider.java:79
Taint Flags:

```
76  }
77
78  protected void authorizedUserAutoLogin() {
79  if (!PROBABILITY_DOUBLE_PREDICATE.test(ThreadLocalRandom.current().nextDouble())) {
80  Authentication authentication = AUTHENTICATION_SUPPLIER.get();
81  authentication.setAuthenticated(true);
82  addSession(authentication.getId());
```

src/main/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProvider.java, line 48 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextLong()
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProvider.java:48
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness — High

Package: org.owasp.webgoat.lessons.hijacksession.cas

src/main/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProvider.java, line 48 (Insecure Randomness) — High

```
45 public class HijackSessionAuthenticationProvider implements
AuthenticationProvider<Authentication> {
46
47  private Queue<String> sessions = new LinkedList<>();
48  private static long id = new Random().nextLong() & Long.MAX_VALUE;
49  protected static final int MAX_SESSIONS = 50;
50
51  private static final DoublePredicate PROBABILITY_DOUBLE_PREDICATE = pr -> pr < 0.75;
```

Package: org.owasp.webgoat.lessons.jwt

src/main/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpoint.java, line 53 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpoint.java:53
Taint Flags:

```
50  "victory", "business", "available", "shipping", "washington"
51  };
52  public static final String JWT_SECRET =
53  TextCodec.BASE64.encode(SECRETS[new Random().nextInt(SECRETS.length)]);
54  private static final String WEBGOAT_USER = "WebGoat";
55  private static final List<String> expectedClaims =
56  List.of("iss", "iat", "exp", "aud", "sub", "username", "Email", "Role");
```

Package: src.main.resources.lessons.chromedevtools.html

src/main/resources/lessons/chromedevtools/html/ChromeDevTools.html, line 53 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: random
Enclosing Method: lambda()
File: src/main/resources/lessons/chromedevtools/html/ChromeDevTools.html:53
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness — High

Package: src.main.resources.lessons.chromedevtools.html

src/main/resources/lessons/chromedevtools/html/ChromeDevTools.html, line 53 (Insecure Randomness) — High

```
50  // sample custom javascript in the recommended way ...
51  // a namespace has been assigned for it, but you can roll your own if you prefer
52  document.getElementById("btn").addEventListener("click", function() {
53  document.getElementById("networkNum").value = Math.random() * 100;
54  document.getElementById("networkNumCopy").value =
document.getElementById("networkNum").value;
55  });
56  </script>
```

Package: src.main.resources.lessons.httpbasics.html

src/main/resources/lessons/httpbasics/html/HttpBasics.html, line 59 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: random
Enclosing Method: assignRandomVal()
File: src/main/resources/lessons/httpbasics/html/HttpBasics.html:59
Taint Flags:

```
56  // sample custom javascript in the recommended way ...
57  // a namespace has been assigned for it, but you can roll your own if you prefer
58  webgoat.customjs.assignRandomVal = function () {
59  var x = Math.floor((Math.random() * 100) + 1);
60  document.getElementById("magic_num").value = x;
61  };
62  webgoat.customjs.assignRandomVal();
```

Package: src.main.resources.webgoat.static.js.libs

src/main/resources/webgoat/static/js/libs/underscore-min.js, line 6 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: random
Enclosing Method: ar()
File: src/main/resources/webgoat/static/js/libs/underscore-min.js:6
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness	High
Package: src.main.resources.webgoat.static.js.libs	
src/main/resources/webgoat/static/js/libs/underscore-min.js, line 6 (Insecure Randomness)	**High**

```
3 // https://underscorejs.org
4 // (c) 2009-2020 Jeremy Ashkenas, DocumentCloud and Investigative Reporters & Editors
5 // Underscore may be freely distributed under the MIT license.
6 [Too long 18064 chars line truncated to 3500 ones]var n="object"==typeof
self&&self.self===self&&self||"object"==typeof global&&global.global===global&&global||
Function("return this")()||{},e=Array.prototype,i=Object.prototype,p="undefined"!=typeof
Symbol?
Symbol.prototype:null,u=e.push,f=e.slice,s=i.toString,o=i.hasOwnProperty,r=Array.isArray,a=Objec
{};function h(n){return n instanceof h?n:this instanceof h?void(this._wrapped=n):new h(n)}var
g=h.VERSION="1.10.2";function y(u,o,n){if(void 0===o)return u;switch(null==n?3:n){case
1:return function(n){return u.call(o,n)};case 3:return function(n,r,t){return
u.call(o,n,r,t)};case 4:return function(n,r,t,e){return u.call(o,n,r,t,e)}}return function()
{return u.apply(o,arguments)}}function d(n,r,t){return null==n?ur:Cn(n)?y(n,r,t):Ln(n)&&!Kn(n)?
ir(n):or(n)}function m(n,r){return d(n,r,1/0)}function b(n,r,t){return h.iteratee!==m?
h.iteratee(n,r):d(n,r,t)}function j(u,o){return o=null==o?u.length-1:+o,function(){for(var
n=Math.max(arguments.length-o,0),r=Array(n),t=0;t<n;t++)r[t]=arguments[t+o];switch(o){case
0:return u.call(this,r);case 1:return u.call(this,arguments[0],r);case 2:return
u.call(this,arguments[0],arguments[1],r)}var e=Array(o+1);for(t=0;t<o;t+
+)e[t]=arguments[t];return e[o]=r,u.apply(this,e)}}function _(n){if(!
Ln(n))return{};if(t)return t(n);v.prototype=n;var r=new v;return v.prototype=null,r}function
w(r){return function(n){return null==n?void 0:n[r]}}function x(n,r){return null!
=n&&o.call(n,r)}function S(n,r){for(var t=r.length,e=0;e<t;e++){if(null==n)return;n=n[r[e]]}
return t?n:void 0}h.iteratee=m;var A=Math.pow(2,53)-1,O=w("length");function M(n){var
r=O(n);return"number"==typeof r&&0<=r&&r<=A}function E(n,r,t){var
e,u;if(r=y(r,t),M(n))for(e=0,u=n.length;e<u;e++)r(n[e],e,n);else{var
o=Sn(n);for(e=0,u=o.length;e<u;e++)r(n[o[e]],o[e],n)}return n}function N(n,r,t)
{r=b(r,t);for(var e=!M(n)&&Sn(n),u=(e||n).length,o=Array(u),i=0;i<u;i++){var a=e?
e[i]:i;o[i]=r(n[a],a,n)}return o}function k(f){return function(n,r,t,e){var
u=3<=arguments.length;return function(n,r,t,e){var u=!M(n)&&Sn(n),o=(u||n).length,i=0<f?
0:o-1;for(e||(t=n[u?u[i]:i],i+=f);0<=i&&i<o;i+=f){var a=u?u[i]:i;t=r(t,n[a],a,n)}return t}
(n,y(r,e,4),t,u)}}var I=k(1),T=k(-1);function B(n,r,t){var e=(M(n)?on:Tn)(n,r,t);if(void 0!
==e&&-1!==e)return n[e]}function R(n,e,r){var u=[];return e=b(e,r),E(n,function(n,r,t)
{e(n,r,t)&&u.push(n)}),u}function F(n,r,t){r=b(r,t);for(var e=!M(n)&&Sn(n),u=(e||
n).length,o=0;o<u;o++){var i=e?e[o]:o;if(!r(n[i],i,n))return!1}return!0}function q(n,r,t)
{r=b(r,t);for(var e=!M(n)&&Sn(n),u=(e||n).length,o=0;o<u;o++){var i=e?
e[o]:o;if(r(n[i],i,n))return!0}return!1}function D(n,r,t,e){return M(n)||(n=On(n)),("number"!
=typeof t||e)&&(t=0),0<=ln(n,r,t)}var W=j(function(n,t,e){var u,o;return Cn(t)?
o=t:Kn(t)&&(u=t.slice(0,-1),t=t[t.length-1]),N(n,function(n){var r=o;if(!r)
{if(u&&u.length&&(n=S(n,u)),null==n)return;r=n[t]}return null==r?r:r.apply(n,e)})});function
z(n,r){return N(n,or(r))}function P(n,e,r){var t,u,o=-1/0,i=-1/0;if(null==e||"number"==typeof
e&&"object"!=typeof n[0]&&null!=n)for(var a=0,f=(n=M(n)?n:On(n)).length;a<f;a++)null!
=(t=n[a])&&o<t&&(o=t);else e=b(e,r),E(n,function(n,r,t){u=e(n,r,t),(i<u||
u===-1/0&&o===-1/0)&&(o=n,i=u)});return o}
7
8 undefined
9 undefined
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness: User-Controlled Seed (1 issue)

Abstract

Functions that generate random or pseudorandom values, which are passed a seed, should not be called with a tainted integer argument.

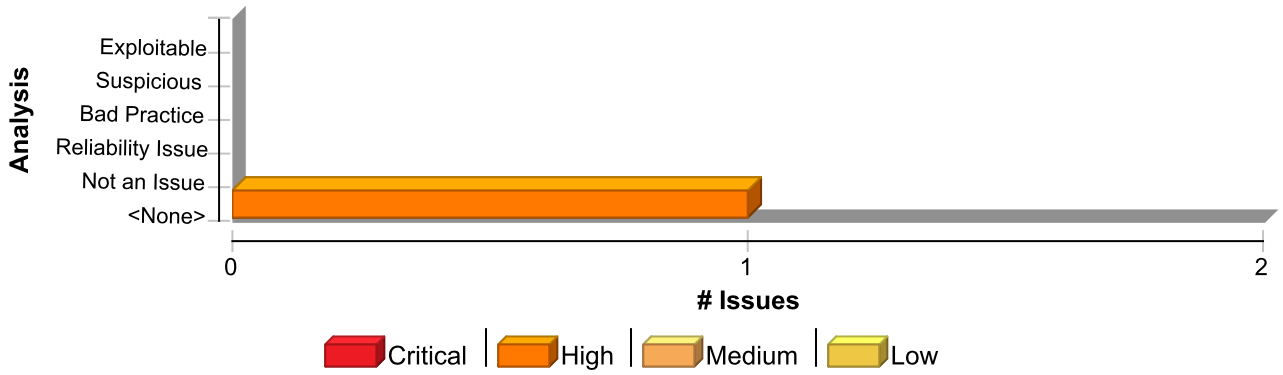
Explanation

`Random.setSeed()` should not be called with a tainted integer argument. Doing so allows an attacker to control the value used to seed the pseudorandom number generator, and therefore predict the sequence of values (usually integers) produced by calls to `Random.nextInt()`, `Random.nextShort()`, `Random.nextLong()`, or returned by `Random.nextBoolean()`, or set in `Random.nextBytes(byte[])`.

Recommendation

Use a cryptographic PRNG such as `java.security.SecureRandom` seeded with hardware-based sources of randomness, such as ring oscillators, disk drive timing, thermal noise, or radioactive decay.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Insecure Randomness: User-Controlled Seed	1	0	0	1
Total	**1**	**0**	**0**	**1**

Insecure Randomness: User-Controlled Seed	High
Package: org.owasp.webgoat.lessons.challenges.challenge7	
src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java, line 17 (Insecure Randomness: User-Controlled Seed)	High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Source Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Insecure Randomness: User-Controlled Seed	High
Package: org.owasp.webgoat.lessons.challenges.challenge7	
src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java, line 17 (Insecure Randomness: User-Controlled Seed)	High

Source: main(0)
From: org.owasp.webgoat.lessons.challenges.challenge7.PasswordResetLink.main
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java:33

```
30  return new String(a);
31  }
32
33  public static void main(String[] args) {
34  if (args == null || args.length != 2) {
35  System.out.println("Need a username and key");
36  System.exit(1);
```

Sink Details

Sink: java.util.Random.setSeed()
Enclosing Method: createPasswordReset()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java:17
Taint Flags: ARGS, NUMBER, PRIMARY_KEY

```
14  Random random = new Random();
15  if (username.equalsIgnoreCase("admin")) {
16  // Admin has a fix reset link
17  random.setSeed(key.length());
18  }
19  return scramble(random, scramble(random, scramble(random, MD5.getHashString(username))));
20  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

J2EE Bad Practices: JVM Termination (3 issues)

Abstract

A web application should not attempt to shut down its container.

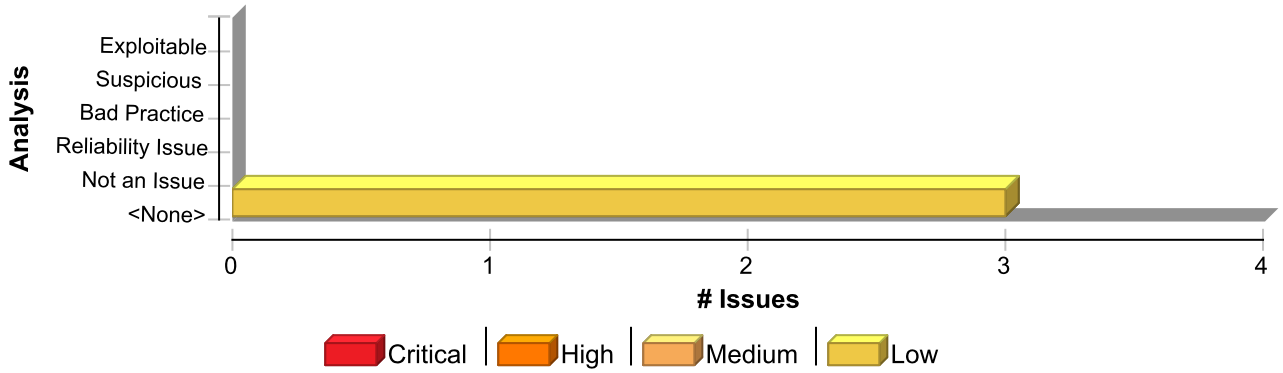
Explanation

It is never a good idea for a web application to attempt to shut down the application container. A call to a termination method is probably part of leftover debug code or code imported from a non-J2EE application.

Recommendation

Never call a termination method within a web application. Such method calls in a J2EE application indicates poor software hygiene and should be removed. Regardless of whether there is a perceived threat, it is unlikely that there is a legitimate reason for such code to remain in the application.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: JVM Termination	3	0	0	3
Total	**3**	**0**	**0**	**3**

J2EE Bad Practices: JVM Termination	Low
Package: .mvn.wrapper	
.mvn/wrapper/MavenWrapperDownloader.java, line 89 (J2EE Bad Practices: JVM Termination)	**Low**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: main()
File: .mvn/wrapper/MavenWrapperDownloader.java:89
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

J2EE Bad Practices: JVM Termination — Low

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 89 (J2EE Bad Practices: JVM Termination) — Low

```
86  try {
87  downloadFileFromURL(url, outputFile);
88  System.out.println("Done");
89  System.exit(0);
90  } catch (Throwable e) {
91  System.out.println("- Error downloading");
92  e.printStackTrace();
```

.mvn/wrapper/MavenWrapperDownloader.java, line 93 (J2EE Bad Practices: JVM Termination) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: main()
File: .mvn/wrapper/MavenWrapperDownloader.java:93
Taint Flags:

```
90  } catch (Throwable e) {
91  System.out.println("- Error downloading");
92  e.printStackTrace();
93  System.exit(1);
94  }
95  }
96
```

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java, line 36 (J2EE Bad Practices: JVM Termination) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: main()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java:36
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

J2EE Bad Practices: JVM Termination	Low

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java, line 36 (J2EE Bad Practices: JVM Termination)	Low

```
33  public static void main(String[] args) {
34  if (args == null || args.length != 2) {
35  System.out.println("Need a username and key");
36  System.exit(1);
37  }
38  String username = args[0];
39  String key = args[1];
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

J2EE Bad Practices: Leftover Debug Code (4 issues)

Abstract

Debug code can create unintended entry points in a deployed web application.

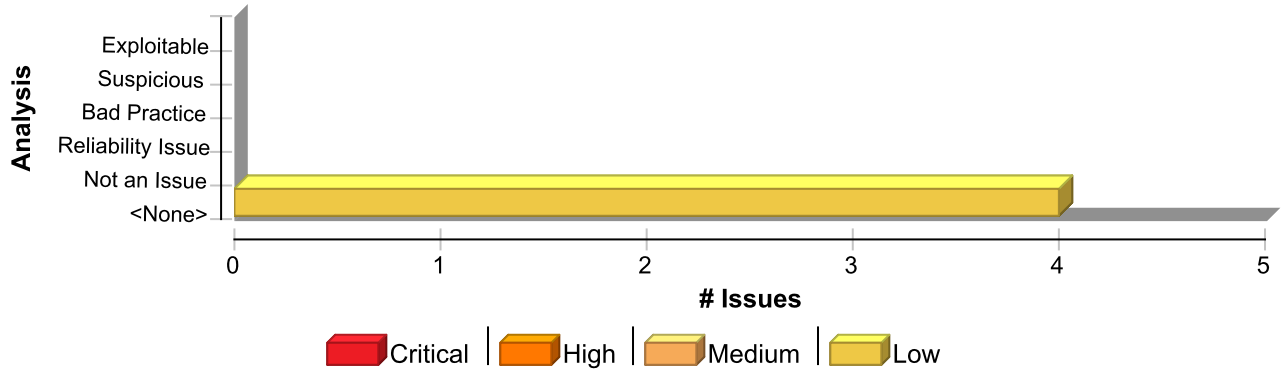
Explanation

A common development practice is to add "back door" code specifically designed for debugging or testing purposes that is not intended to be shipped or deployed with the application. When this sort of debug code is accidentally left in the application, the application is open to unintended modes of interaction. These back door entry points create security risks because they are not considered during design or testing and fall outside of the expected operating conditions of the application. The most common example of forgotten debug code is a `main()` method appearing in a web application. Although this is an acceptable practice during product development, classes that are part of a production J2EE application should not define a `main()`.

Recommendation

Remove debug code before deploying a production version of an application. Regardless of whether a direct security threat can be articulated, it is unlikely that there is a legitimate reason for such code to remain in the application after the early stages of development.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Leftover Debug Code	4	0	0	4
Total	**4**	**0**	**0**	**4**

J2EE Bad Practices: Leftover Debug Code	Low
Package: .mvn.wrapper	
.mvn/wrapper/MavenWrapperDownloader.java, line 48 (J2EE Bad Practices: Leftover Debug Code)	**Low**
Issue Details	

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

J2EE Bad Practices: Leftover Debug Code — Low

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 48 (J2EE Bad Practices: Leftover Debug Code) — Low

Sink Details

Sink: Function: main
Enclosing Method: main()
File: .mvn/wrapper/MavenWrapperDownloader.java:48
Taint Flags:

```
45  */
46  private static final String PROPERTY_NAME_WRAPPER_URL = "wrapperUrl";
47
48  public static void main(String args[]) {
49  System.out.println("- Downloader started");
50  File baseDirectory = new File(args[0]);
51  System.out.println("- Using base directory: " + baseDirectory.getAbsolutePath());
```

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java, line 43 (J2EE Bad Practices: Leftover Debug Code) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: main
Enclosing Method: main()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java:43
Taint Flags:

```
40  * @param args command line arguments
41  * @since ostermillerutils 1.00.00
42  */
43  public static void main(String[] args) {
44  if (args.length == 0) {
45  System.err.println("Please specify a file.");
46  } else {
```

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java, line 33 (J2EE Bad Practices: Leftover Debug Code) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

J2EE Bad Practices: Leftover Debug Code — Low

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java, line 33 (J2EE Bad Practices: Leftover Debug Code) — **Low**

Sink: Function: main
Enclosing Method: main()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/PasswordResetLink.java:33
Taint Flags:

```
30  return new String(a);
31  }
32
33  public static void main(String[] args) {
34  if (args == null || args.length != 2) {
35  System.out.println("Need a username and key");
36  System.exit(1);
```

Package: org.owasp.webgoat.server

src/main/java/org/owasp/webgoat/server/StartWebGoat.java, line 39 (J2EE Bad Practices: Leftover Debug Code) — **Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: main
Enclosing Method: main()
File: src/main/java/org/owasp/webgoat/server/StartWebGoat.java:39
Taint Flags:

```
36 @Slf4j
37 public class StartWebGoat {
38
39  public static void main(String[] args) {
40  var parentBuilder =
41  new SpringApplicationBuilder()
42  .parent(ParentConfig.class)
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Key Management: Hardcoded Encryption Key (6 issues)

Abstract

Hardcoded encryption keys can compromise security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode an encryption key because it allows all of the project's developers to view the encryption key, and makes fixing the problem extremely difficult. After the code is in production, a software patch is required to change the encryption key. If the account that is protected by the encryption key is compromised, the owners of the system must choose between security and availability. **Example 1:** The following code uses a hardcoded encryption key:

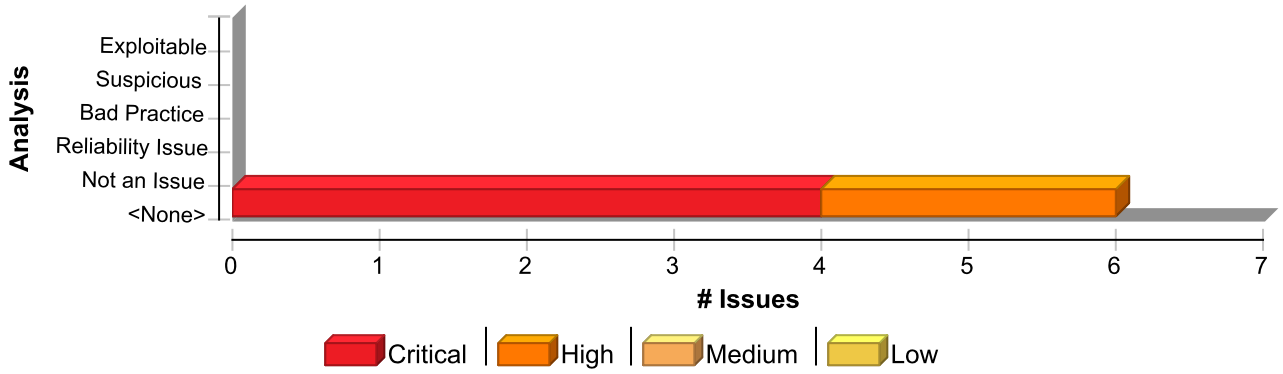
```
...
private static final String encryptionKey = "lakdsljkalkjlksdfkl";
byte[] keyBytes = encryptionKey.getBytes();
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher encryptCipher = Cipher.getInstance("AES");
encryptCipher.init(Cipher.ENCRYPT_MODE, key);
...
```

Anyone with access to the code has access to the encryption key. After the application has shipped, there is no way to change the encryption key unless the program is patched. An employee with access to this information can use it to break into the system. If attackers had access to the executable for the application, they could extract the encryption key value.

Recommendation

Encryption keys should never be hardcoded and should be obfuscated and managed in an external source. Storing encryption keys in plain text anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the encryption key.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Key Management: Hardcoded Encryption Key	6	0	0	6
Total	**6**	**0**	**0**	**6**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Key Management: Hardcoded Encryption Key — Critical

Package: .org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 45 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:45
Taint Flags:

```
42  }
43
44  public static String getPrivateKeyInPEM(KeyPair keyPair) {
45  String encodedString = "-----BEGIN PRIVATE KEY-----\n";
46  encodedString =
47  encodedString
48  + new String(
```

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 133 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:133
Taint Flags:

```
130
131  public static PrivateKey getPrivateKeyFromPEM(String privateKeyPem)
132  throws NoSuchAlgorithmException, InvalidKeySpecException {
133  privateKeyPem = privateKeyPem.replace("-----BEGIN PRIVATE KEY-----", "");
134  privateKeyPem = privateKeyPem.replace("-----END PRIVATE KEY-----", "");
135  privateKeyPem = privateKeyPem.replace("\n", "").replace("\r", "");
136
```

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/TokenTest.java, line 44 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Key Management: Hardcoded Encryption Key — Critical

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/TokenTest.java, line 44 (Key Management: Hardcoded Encryption Key) — Critical

Sink: VariableAccess: key
Enclosing Method: test()
File: src/test/java/org/owasp/webgoat/lessons/jwt/TokenTest.java:44
Taint Flags:

```
41
42  @Test
43  public void test() {
44  String key = "qwertyqwerty1234";
45  Map<String, Object> claims =
46  Map.of("username", "Jerry", "aud", "webgoat.org", "email", "jerry@webgoat.com");
47  String token =
```

Package: org.owasp.webgoat.lessons.jwt.claimmisuse

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java, line 34 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: key
Enclosing Method: solveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java:34
Taint Flags:

```
31
32  @Test
33  public void solveAssignment() throws Exception {
34  String key = "deletingTom";
35  Map<String, Object> claims = new HashMap<>();
36  claims.put("username", "Tom");
37  String token =
```

Key Management: Hardcoded Encryption Key — High

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 37 (Key Management: Hardcoded Encryption Key) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Key Management: Hardcoded Encryption Key — High

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 37 (Key Management: Hardcoded Encryption Key) — High

Sink Details

Sink: FunctionCall: RSAKeyGenParameterSpec
Enclosing Method: generateKeyPair()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:37
Taint Flags:

```
34  throws NoSuchAlgorithmException, InvalidAlgorithmParameterException {
35  KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
36  RSAKeyGenParameterSpec kpgSpec =
37  new RSAKeyGenParameterSpec(
38  2048, FERMAT_PRIMES[new SecureRandom().nextInt(FERMAT_PRIMES.length)]);
39  keyPairGenerator.initialize(kpgSpec);
40  // keyPairGenerator.initialize(2048);
```

Package: src.main.resources

src/main/resources/goatkeystore.pkcs12, line 0 (Key Management: Hardcoded Encryption Key) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Sink Details

File: src/main/resources/goatkeystore.pkcs12:0
Taint Flags:

```
1 0, ÿ^B^A^C0, ¸^F *†H†÷
2 ^A^G^A , ©^D, ¥0, ¡0,^Ee^F *†H†÷
3 ^A^G^A ,^EV^D,^ER0,^EN0,^EJ^F^K*†H†÷
4 ^A^L
5
6 undefined
7 undefined
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Log Forging (3 issues)

Abstract

Writing unvalidated user input to log files can allow an attacker to forge log entries or inject malicious content into the logs.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Explanation

Log forging vulnerabilities occur when: 1. Data enters an application from an untrusted source. 2. The data is written to an application or system log file. Applications typically use log files to store a history of events or transactions for later review, statistics gathering, or debugging. Depending on the nature of the application, the task of reviewing log files may be performed manually on an as-needed basis or automated with a tool that automatically culls logs for important events or trending information. Interpretation of the log files may be hindered or misdirected if an attacker can supply data to the application that is subsequently logged verbatim. In the most benign case, an attacker may be able to insert false entries into the log file by providing the application with input that includes appropriate characters. If the log file is processed automatically, the attacker may be able to render the file unusable by corrupting the format of the file or injecting unexpected characters. A more subtle attack might involve skewing the log file statistics. Forged or otherwise, corrupted log files can be used to cover an attacker's tracks or even to implicate another party in the commission of a malicious act [1]. In the worst case, an attacker may inject code or other commands into the log file and take advantage of a vulnerability in the log processing utility [2]. **Example 1:** The following web application code attempts to read an integer value from a request object. If the value fails to parse as an integer, then the input is logged with an error message indicating what happened.

```
...
        String val = request.getParameter("val");
        try {
                int value = Integer.parseInt(val);
        }
        catch (NumberFormatException nfe) {
                log.info("Failed to parse val = " + val);
        }
...
```

If a user submits the string "`twenty-one`" for `val`, the following entry is logged:

```
INFO: Failed to parse val=twenty-one
```

However, if an attacker submits the string "`twenty-one%0a%0aINFO:+User+logged+out%3dbadguy`", the following entry is logged:

```
INFO: Failed to parse val=twenty-one

INFO: User logged out=badguy
```

Clearly, attackers may use this same mechanism to insert arbitrary log entries. Some think that in the mobile world, classic web application vulnerabilities, such as log forging, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication. **Example 2:** The following code adapts `Example 1` to the Android platform.

```
...
        String val = this.getIntent().getExtras().getString("val");
        try {
                int value = Integer.parseInt();
        }
        catch (NumberFormatException nfe) {
                Log.e(TAG, "Failed to parse val = " + val);
        }
...
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

Prevent log forging attacks with indirection: create a set of legitimate log entries that correspond to different events that must be logged and only log entries from this set. To capture dynamic content, such as users logging out of the system, always use server-controlled values rather than user-supplied data. This ensures that the input provided by the user is never used directly in a log entry. Example 1 can be rewritten to use a pre-defined log entry that corresponds to a `NumberFormatException` as follows:

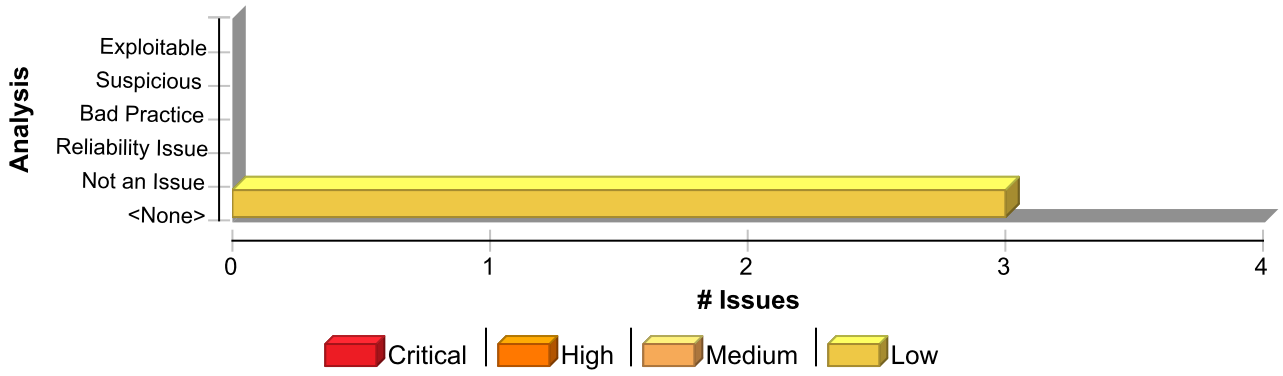
```
...
        public static final String NFE = "Failed to parse val. The input is
required to be an integer value."
...
        String val = request.getParameter("val");
        try {
                int value = Integer.parseInt(val);
        }
        catch (NumberFormatException nfe) {
                log.info(NFE);
        }
..
```

And here is an Android equivalent:

```
...
        public static final String NFE = "Failed to parse val. The input is
required to be an integer value."
...
        String val = this.getIntent().getExtras().getString("val");
        try {
                int value = Integer.parseInt();
        }
        catch (NumberFormatException nfe) {
                Log.e(TAG, NFE);
        }
...
```

In some situations this approach is impractical because the set of legitimate log entries is too large or complicated. In these situations, developers often fall back on implementing a deny list. A deny list is used to selectively reject or escape potentially dangerous characters before using the input. However, a list of unsafe characters can quickly become incomplete or outdated. A better approach is to create a list of characters that are permitted to appear in log entries and accept input composed exclusively of characters in the approved set. The most critical character in most log forging attacks is the '\n' (newline) character, which should never appear on a log entry allow list.

Issue Summary



© Copyright [2008-2024] Micro Focus or one of its affiliates.

Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Log Forging	3	0	0	3
Total	**3**	**0**	**0**	**3**

Log Forging — Low

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 49 (Log Forging) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read this.taskName
From: org.dummy.insecure.framework.VulnerableTaskHolder.readObject
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:49

```
46  stream.defaultReadObject();
47
48  // do something with the data
49  log.info("restoring task: {}", taskName);
50  log.info("restoring time: {}", requestedExecutionTime);
51
52  if (requestedExecutionTime != null
```

Sink Details

Sink: org.slf4j.Logger.info()
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:49
Taint Flags: SERIALIZED

```
46  stream.defaultReadObject();
47
48  // do something with the data
49  log.info("restoring task: {}", taskName);
50  log.info("restoring time: {}", requestedExecutionTime);
51
52  if (requestedExecutionTime != null
```

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 50 (Log Forging) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Log Forging — Low

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 50 (Log Forging) — Low

Source: Read this.requestedExecutionTime
From: org.dummy.insecure.framework.VulnerableTaskHolder.readObject
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:50

```
47
48  // do something with the data
49  log.info("restoring task: {}", taskName);
50  log.info("restoring time: {}", requestedExecutionTime);
51
52  if (requestedExecutionTime != null
53  && (requestedExecutionTime.isBefore(LocalDateTime.now().minusMinutes(10))
```

Sink Details

Sink: org.slf4j.Logger.info()
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:50
Taint Flags: SERIALIZED

```
47
48  // do something with the data
49  log.info("restoring task: {}", taskName);
50  log.info("restoring time: {}", requestedExecutionTime);
51
52  if (requestedExecutionTime != null
53  && (requestedExecutionTime.isBefore(LocalDateTime.now().minusMinutes(10))
```

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 63 (Log Forging) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read this.taskAction
From: org.dummy.insecure.framework.VulnerableTaskHolder.readObject
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:63

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Log Forging	Low
Package: org.dummy.insecure.framework	
src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 63 (Log Forging)	**Low**

```
60  // condition is here to prevent you from destroying the goat altogether
61  if ((taskAction.startsWith("sleep") || taskAction.startsWith("ping"))
62  && taskAction.length() < 22) {
63  log.info("about to execute: {}", taskAction);
64  try {
65  Process p = Runtime.getRuntime().exec(taskAction);
66  BufferedReader in = new BufferedReader(new
InputStreamReader(p.getInputStream()));
```

Sink Details

Sink: org.slf4j.Logger.info()
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:63
Taint Flags: SERIALIZED

```
60  // condition is here to prevent you from destroying the goat altogether
61  if ((taskAction.startsWith("sleep") || taskAction.startsWith("ping"))
62  && taskAction.length() < 22) {
63  log.info("about to execute: {}", taskAction);
64  try {
65  Process p = Runtime.getRuntime().exec(taskAction);
66  BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Missing Check against Null (3 issues)

Abstract

The program can dereference a null-pointer because it does not check the return value of a function that might return `null`.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break. Two dubious assumptions that are easy to spot in code are "this function call can never fail" and "it doesn't matter if this function call fails". When a programmer ignores the return value from a function, they implicitly state that they are operating under one of these assumptions. **Example 1:** The following code does not check to see if the string returned by `getParameter()` is `null` before calling the member function `compareTo()`, potentially causing a `null` dereference.

```
String itemName = request.getParameter(ITEM_NAME);
        if (itemName.compareTo(IMPORTANT_ITEM)) {
                ...
        }
        ...
```

Example 2:. The following code shows a system property that is set to `null` and later dereferenced by a programmer who mistakenly assumes it will always be defined.

```
System.clearProperty("os.name");
...
String os = System.getProperty("os.name");
if (os.equalsIgnoreCase("Windows 95") )
        System.out.println("Not supported");
```

The traditional defense of this coding error is: "I know the requested value will always exist because.... If it does not exist, the program cannot perform the desired behavior so it doesn't matter whether I handle the error or simply allow the program to die dereferencing a `null` value." But attackers are skilled at finding unexpected paths through programs, particularly when exceptions are involved.

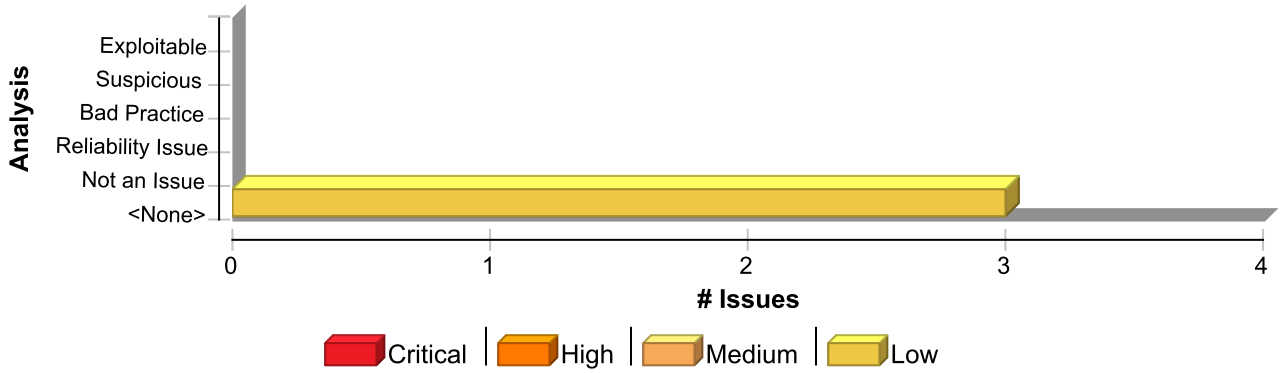
Recommendation

If a function can return an error code or any other evidence of its success or failure, always check for the error condition, even if there is no obvious way for it to occur. In addition to preventing security errors, many initially mysterious bugs have eventually led back to a failed method call with an unchecked return value. Create an easy to use and standard way for dealing with failure in your application. If error handling is straightforward, programmers will be less inclined to omit it. One approach to standardized error handling is to write wrappers around commonly-used functions that check and handle error conditions without additional programmer intervention. When wrappers are implemented and adopted, the use of non-wrapped equivalents can be prohibited and enforced by using custom rules. **Example 3:** The following code implements a wrapper around `getParameter()` that checks the return value of `getParameter()` against `null` and uses a default value if the requested parameter is not defined.

```
String safeGetParameter (HttpRequest request, String name)
{
    String value = request.getParameter(name);
    if (value == null) {
        return getDefaultValue(name)
    }
    return value;
}
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Missing Check against Null	3	0	0	3
Total	**3**	**0**	**0**	**3**

Missing Check against Null — Low

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 100 (Missing Check against Null) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Control Flow)

Sink Details

Sink: getenv(?) : System.getenv may return NULL
Enclosing Method: downloadFileFromURL()
File: .mvn/wrapper/MavenWrapperDownloader.java:100
Taint Flags:

```
97  private static void downloadFileFromURL(String urlString, File destination) throws
Exception {
98  if (System.getenv("MVNW_USERNAME") != null && System.getenv("MVNW_PASSWORD") != null) {
99  String username = System.getenv("MVNW_USERNAME");
100  char[] password = System.getenv("MVNW_PASSWORD").toCharArray();
101  Authenticator.setDefault(new Authenticator() {
102  @Override
103  protected PasswordAuthentication getPasswordAuthentication() {
```

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/DeserializationIntegrationTest.java, line 12 (Missing Check against Null) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Control Flow)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Missing Check against Null — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/DeserializationIntegrationTest.java, line 12 (Missing Check against Null) — Low

Sink Details

Sink: getProperty(?) : System.getProperty may return NULL
Enclosing Method: ()
File: src/it/java/org/owasp/webgoat/DeserializationIntegrationTest.java:12
Taint Flags:

```
9
10 public class DeserializationIntegrationTest extends IntegrationTest {
11
12  private static String OS = System.getProperty("os.name").toLowerCase();
13
14  @Test
15  public void runTests() throws IOException {
```

Package: org.owasp.webgoat.lessons.deserialization

src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java, line 23 (Missing Check against Null) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Control Flow)

Sink Details

Sink: getProperty(?) : System.getProperty may return NULL
Enclosing Method: ()
File: src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java:23
Taint Flags:

```
20
21  private MockMvc mockMvc;
22
23  private static String OS = System.getProperty("os.name").toLowerCase();
24
25  @BeforeEach
26  void setup() {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Often Misused: Authentication (1 issue)

Abstract

Attackers may spoof DNS entries. Do not rely on DNS names for security.

Explanation

Many DNS servers are susceptible to spoofing attacks, so you should assume that your software will someday run in an environment with a compromised DNS server. If attackers are allowed to make DNS updates (sometimes called DNS cache poisoning), they can route your network traffic through their machines or make it appear as if their IP addresses are part of your domain. Do not base the security of your system on DNS names. **Example:** The following code uses a DNS lookup to determine whether an inbound request is from a trusted host. If an attacker can poison the DNS cache, they can gain trusted status.

```
String ip = request.getRemoteAddr();
InetAddress addr = InetAddress.getByName(ip);
if (addr.getCanonicalHostName().endsWith("trustme.com")) {
trusted = true;
}
```

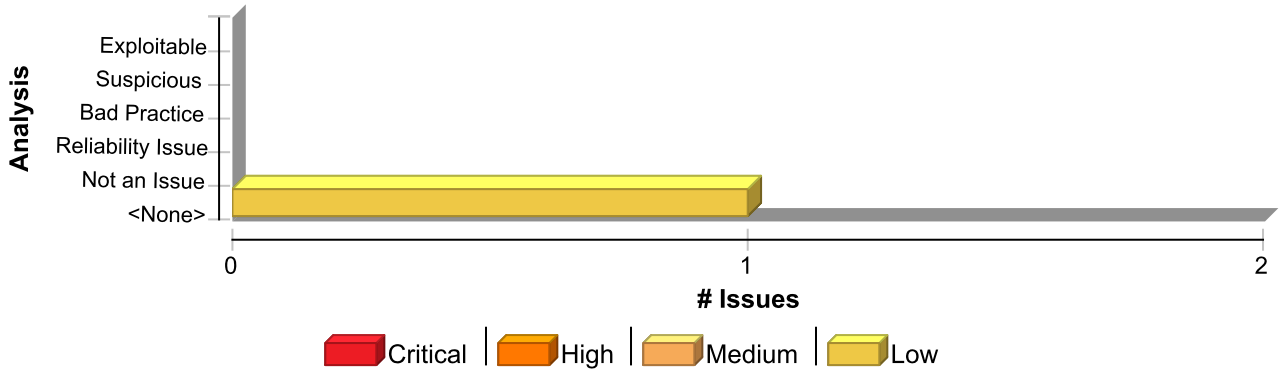
IP addresses are more reliable than DNS names, but they can also be spoofed. Attackers may easily forge the source IP address of the packets they send, but response packets will return to the forged IP address. To see the response packets, the attacker has to sniff the traffic between the victim machine and the forged IP address. In order to accomplish the required sniffing, attackers typically attempt to locate themselves on the same subnet as the victim machine. Attackers may be able to circumvent this requirement by using source routing, but source routing is disabled across much of the Internet today. In summary, IP address verification can be a useful part of an authentication scheme, but it should not be the single factor required for authentication.

Recommendation

You can increase confidence in a domain name lookup if you check to make sure that the host's forward and backward DNS entries match. Attackers will not be able to spoof both the forward and the reverse DNS entries without controlling the nameservers for the target domain. This is not a foolproof approach however: attackers may be able to convince the domain registrar to turn over the domain to a malicious nameserver. Basing authentication on DNS entries is simply a risky proposition. While no authentication mechanism is foolproof, there are better alternatives than host-based authentication. Password systems offer decent security, but are susceptible to bad password choices, insecure password transmission, and bad password management. A cryptographic scheme like SSL is worth considering, but such schemes are often so complex that they bring with them the risk of significant implementation errors, and key material can always be stolen. In many situations, multi-factor authentication including a physical token offers the most security available at a reasonable price.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Often Misused: Authentication	1	0	0	1
Total	**1**	**0**	**0**	**1**

Often Misused: Authentication — Low

Package: org.owasp.webgoat.lessons.challenges

src/test/java/org/owasp/webgoat/lessons/challenges/Assignment1Test.java, line 56 (Often Misused: Authentication) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getLocalHost()
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/challenges/Assignment1Test.java:56
Taint Flags:

```
53
54  @Test
55  void success() throws Exception {
56  InetAddress addr = InetAddress.getLocalHost();
57  String host = addr.getHostAddress();
58  mockMvc
59  .perform(
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Open Redirect (5 issues)

Abstract

Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.

Explanation

Redirects allow web applications to direct users to different pages within the same application or to external sites. Applications utilize redirects to aid in site navigation and, in some cases, to track how users exit the site. Open redirect vulnerabilities occur when a web application redirects clients to any arbitrary URL that can be controlled by an attacker. Attackers might utilize open redirects to trick users into visiting a URL to a trusted site, but then redirecting them to a malicious site. By encoding the URL, an attacker can make it difficult for end-users to notice the malicious destination of the redirect, even when it is passed as a URL parameter to the trusted site. Open redirects are often abused as part of phishing scams to harvest sensitive end-user data. **Example 1:** The following JavaScript code instructs the user's browser to open a URL read from the `dest` request parameter when a user clicks the link.

```
...
strDest = form.dest.value;
window.open(strDest,"myresults");
...
```

If a victim received an email instructing them to follow a link to "http://trusted.example.com/ecommerce/redirect.asp?dest=www.wilyhacker.com", the user would likely click on the link believing they would be transferred to the trusted site. However, when the victim clicks the link, the code in `Example 1` will redirect the browser to "http://www.wilyhacker.com". Many users have been educated to always inspect URLs they receive in emails to make sure the link specifies a trusted site they know. However, if the attacker Hex encoded the destination url as follows: "http://trusted.example.com/ecommerce/redirect.asp?dest=%77%69%6C%79%68%61%63%6B%65%72%2E%63%6F%6D" then even a savvy end-user may be fooled into following the link.

Recommendation

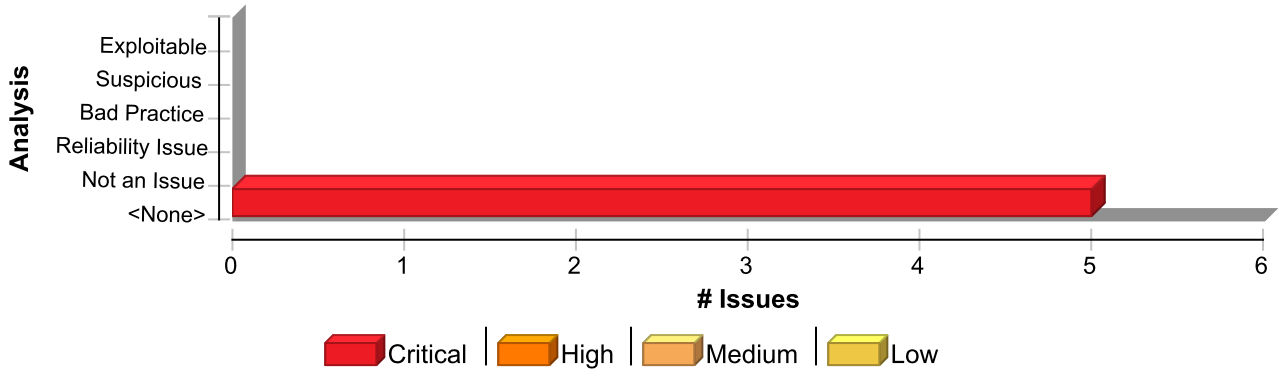
Unvalidated user input should not be allowed to control the destination URL in a redirect. Instead, use a level of indirection: create a list of legitimate URLs that users are allowed to specify, and only allow users to select from the list. With this approach, input provided by users is never used directly to specify a URL for redirects. **Example 2:** The following code references an array populated with valid URLs. The link the user clicks passes in the array index that corresponds to the desired URL.

```
...
        strDest = form.dest.value;
        if((strDest.value != null)||(strDest.value.length!=0))
        {
                if((strDest >= 0) && (strDest <= strURLArray.length -1 ))
                {
                        strFinalURL = strURLArray[strDest];
                        window.open(strFinalURL,"myresults");
                }
        }
...
```

In some situations this approach is impractical because the set of legitimate URLs is too large or too hard to keep track of. In such cases, use a similar approach to restrict the domains that users can be redirected to, which can at least prevent attackers from sending users to malicious external sites.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Open Redirect	5	0	0	5
Total	**5**	**0**	**0**	**5**

Open Redirect — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 30 (Open Redirect) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:29

```
26 }
27
28 webgoat.customjs.profileUploadCallbackFix = function () {
29  $.get("PathTraversal/profile-picture", function (result, status) {
30  document.getElementById("previewFix").src = "" +
result;
31  });
32 }
```

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:30
Taint Flags: WEB, XSS

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Open Redirect — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 30 (Open Redirect) — Critical

```
27
28 webgoat.customjs.profileUploadCallbackFix = function () {
29  $.get("PathTraversal/profile-picture", function (result, status) {
30  document.getElementById("previewFix").src = "" + result;
31  });
32 }
33
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 47 (Open Redirect) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:46

```
43 }
44
45 webgoat.customjs.profileUploadCallbackRemoveUserInput = function () {
46  $.get("PathTraversal/profile-picture", function (result, status) {
47  document.getElementById("previewRemoveUserInput").src = "" + result;
48  });
49 }
```

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:47
Taint Flags: WEB, XSS

```
44
45 webgoat.customjs.profileUploadCallbackRemoveUserInput = function () {
46  $.get("PathTraversal/profile-picture", function (result, status) {
47  document.getElementById("previewRemoveUserInput").src = "" + result;
48  });
49 }
50
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Open Redirect — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 14 (Open Redirect) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:13

```
10 }
11
12 webgoat.customjs.profileUploadCallback = function () {
13  $.get("PathTraversal/profile-picture", function (result, status) {
14  document.getElementById("preview").src = "" +
result;
15  });
16 }
```

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:14
Taint Flags: WEB, XSS

```
11
12 webgoat.customjs.profileUploadCallback = function () {
13  $.get("PathTraversal/profile-picture", function (result, status) {
14  document.getElementById("preview").src = "" + result;
15  });
16 }
17
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 60 (Open Redirect) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:59

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Open Redirect — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 60 (Open Redirect) — Critical

```
56 }
57
58 function newRandomPicture() {
59  $.get("PathTraversal/random-picture", function (result, status) {
60  document.getElementById("randomCatPicture").src = "" + result;
61  });
62 }
```

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:60
Taint Flags: WEB, XSS

```
57
58 function newRandomPicture() {
59  $.get("PathTraversal/random-picture", function (result, status) {
60  document.getElementById("randomCatPicture").src = "" + result;
61  });
62 }
63
```

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 54 (Open Redirect) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: lambda(0)
From: lambda
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:53

```
50
51
52 webgoat.customjs.profileUploadCallbackRetrieval = function () {
53  $.get("PathTraversal/profile-picture", function (result, status) {
54  document.getElementById("previewRetrieval").src = "" + result;
55  });
56 }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Open Redirect — Critical

Package: src.main.resources.lessons.pathtraversal.js

src/main/resources/lessons/pathtraversal/js/path_traversal.js, line 54 (Open Redirect) — Critical

Sink Details

Sink: Assignment to src
Enclosing Method: lambda()
File: src/main/resources/lessons/pathtraversal/js/path_traversal.js:54
Taint Flags: WEB, XSS

```
51
52 webgoat.customjs.profileUploadCallbackRetrieval = function () {
53  $.get("PathTraversal/profile-picture", function (result, status) {
54  document.getElementById("previewRetrieval").src = "" + result;
55  });
56 }
57
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Empty Password (1 issue)

Abstract

Empty passwords may compromise system security in a way that cannot be easily remedied.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Explanation

It is never a good idea to assign an empty string to a password variable. If the empty password is used to successfully authenticate against another system, then the corresponding account's security is likely compromised because it accepts an empty password. If the empty password is merely a placeholder until a legitimate value can be assigned to the variable, then it can confuse anyone unfamiliar with the code and potentially cause problems on unexpected control flow paths. **Example 1:** The following code attempts to connect to a database with an empty password.

```
...
DriverManager.getConnection(url, "scott", "");
...
```

If the code in `Example 1` succeeds, it indicates that the database user account "scott" is configured with an empty password, which an attacker can easily guess. After the program ships, updating the account to use a non-empty password will require a code change. **Example 2:** The following code initializes a password variable to an empty string, attempts to read a stored value for the password, and compares it against a user-supplied value.

```
...
String storedPassword = "";
String temp;

if ((temp = readPassword()) != null) {
    storedPassword = temp;
}

if(storedPassword.equals(userPassword))
    // Access protected resources
    ...
}
...
```

If `readPassword()` fails to retrieve the stored password due to a database error or another problem, then an attacker could trivially bypass the password check by providing an empty string for `userPassword`. In the mobile environment, password management is especially important given that there is such a high chance of device loss. **Example 3:** The following code initializes username and password variables to empty strings, reads credentials from an Android WebView store if they have not been previously rejected by the server for the current request, and uses them to setup authentication for viewing protected pages.

```
...
webview.setWebViewClient(new WebViewClient() {
  public void onReceivedHttpAuthRequest(WebView view,
        HttpAuthHandler handler, String host, String realm) {
    String username = "";
    String password = "";

    if (handler.useHttpAuthUsernamePassword()) {
      String[] credentials = view.getHttpAuthUsernamePassword(host, realm);
      username = credentials[0];
      password = credentials[1];
    }
    handler.proceed(username, password);
  }
});
...
```

Similar to `Example 2`, if `useHttpAuthUsernamePassword()` returns `false`, an attacker will be able to view protected pages by supplying an empty password.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

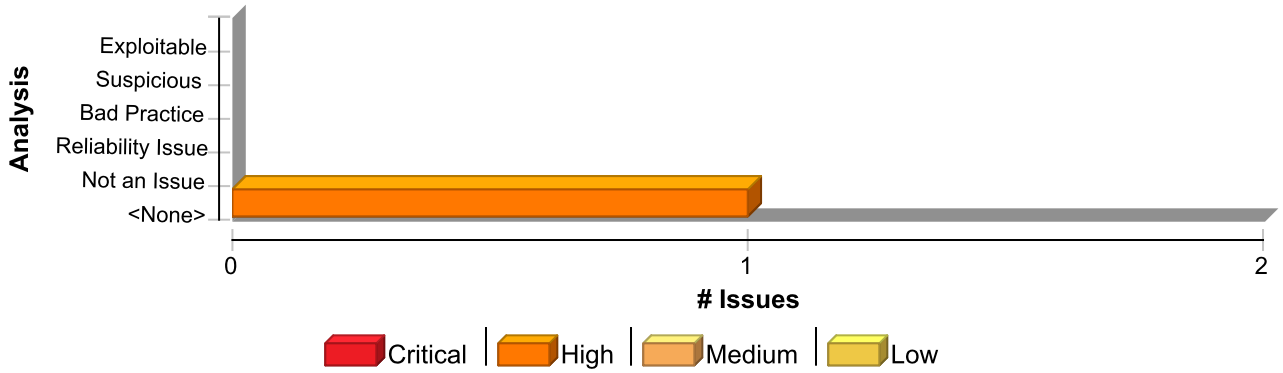
Recommendation

Always read stored password values from encrypted, external resources and assign password variables meaningful values. Ensure that sensitive resources are never protected with empty or `null` passwords. For Android, as well as any other platform that uses SQLite database, SQLCipher is a good alternative. SQLCipher is an extension to the SQLite database that provides transparent 256-bit AES encryption of database files. Thus, credentials can be stored in an encrypted database. **Example 4:** The following code demonstrates how to integrate SQLCipher into an Android application after downloading the necessary binaries, and store credentials into the database file.

```
import net.sqlcipher.database.SQLiteDatabase;
...
  SQLiteDatabase.loadLibs(this);
  File dbFile = getDatabasePath("credentials.db");
  dbFile.mkdirs();
  dbFile.delete();
  SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(dbFile,
"credentials", null);
  db.execSQL("create table credentials(u, p)");
  db.execSQL("insert into credentials(u, p) values(?, ?)", new Object[]
{username, password});
...
```

Note that references to `android.database.sqlite.SQLiteDatabase` are substituted with those of `net.sqlcipher.database.SQLiteDatabase`. To enable encryption on the WebView store, you must recompile WebKit with the `sqlcipher.so` library.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Empty Password	1	0	0	1
Total	**1**	**0**	**0**	**1**

Password Management: Empty Password	**High**
Package: org.owasp.webgoat.lessons.xxe	
src/main/java/org/owasp/webgoat/lessons/xxe/User.java, line 31 (Password Management: Empty Password)	**High**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Empty Password	High
Package: org.owasp.webgoat.lessons.xxe	
src/main/java/org/owasp/webgoat/lessons/xxe/User.java, line 31 (Password Management: Empty Password)	High

Sink Details

Sink: FieldAccess: password
Enclosing Method: User()
File: src/main/java/org/owasp/webgoat/lessons/xxe/User.java:31
Taint Flags:

```
28 public class User {
29
30  private String username = "";
31  private String password = "";
32
33  public String getPassword() {
34  return password;
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password (17 issues)

Abstract

Hardcoded passwords can compromise system security in a way that is not easy to remedy.

Explanation

It is never a good idea to hardcode a password. Not only does hardcoding a password allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. After the code is in production, the password cannot be changed without patching the software. If the account protected by the password is compromised, the owners of the system must choose between security and availability. **Example 1:** The following code uses a hardcoded password to connect to a database:

```
...
DriverManager.getConnection(url, "scott", "tiger");
...
```

This code will run successfully, but anyone who has access to it will have access to the password. After the program ships, there is likely no way to change the database user "scott" with a password of "tiger" unless the program is patched. An employee with access to this information can use it to break into the system. Even worse, if attackers have access to the bytecode for the application they can use the `javap -c` command to access the disassembled code, which will contain the values of the passwords used. The result of this operation might look something like the following for `Example 1`:

```
javap -c ConnMngr.class

22: ldc   #36; //String jdbc:mysql://ixne.com/rxsql
24: ldc   #38; //String scott
26: ldc   #17; //String tiger
```

In the mobile environment, password management is especially important given that there is such a high chance of device loss. **Example 2:** The following code uses hardcoded username and password to setup authentication for viewing protected pages with Android's WebView.

```
...
webview.setWebViewClient(new WebViewClient() {
  public void onReceivedHttpAuthRequest(WebView view,
        HttpAuthHandler handler, String host, String realm) {
    handler.proceed("guest", "allow");
  }
});
...
```

Similar to `Example 1`, this code will run successfully, but anyone who has access to it will have access to the password.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

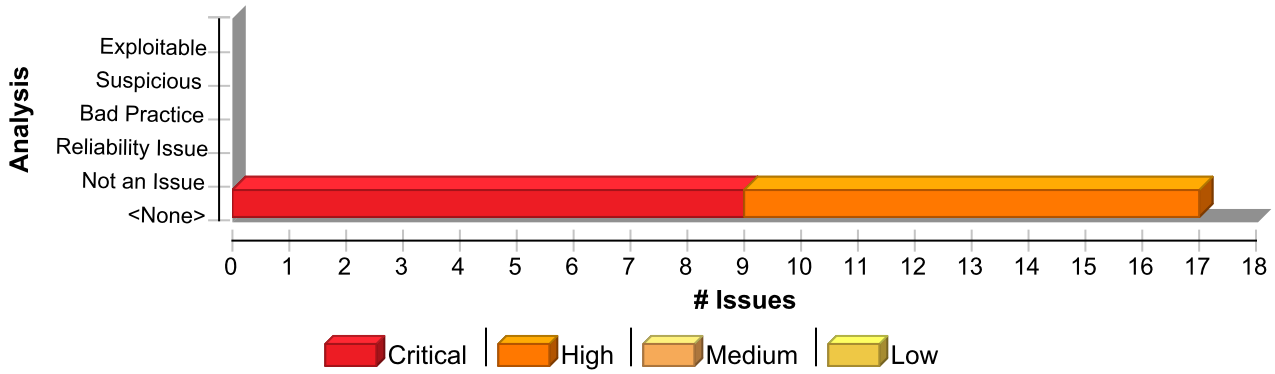
Recommendation

Passwords should never be hardcoded and should generally be obfuscated and managed in an external source. Storing passwords in plain text anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the password. At the very least, hash passwords before storing them. Some third-party products claim the ability to securely manage passwords. For example, WebSphere Application Server 4.x uses a simple XOR encryption algorithm for obfuscating values, but be skeptical about such facilities. WebSphere and other application servers offer outdated and relatively weak encryption mechanisms that are insufficient for security-sensitive environments. Today, the best option for a secure generic solution is to create a proprietary mechanism yourself. For Android, as well as any other platform that uses SQLite database, SQLCipher is a good alternative. SQLCipher is an extension to the SQLite database that provides transparent 256-bit AES encryption of database files. Thus, credentials can be stored in an encrypted database. **Example 3:** The following code demonstrates how to integrate SQLCipher into an Android application after downloading the necessary binaries, and store credentials into the database file.

```
import net.sqlcipher.database.SQLiteDatabase;
...
  SQLiteDatabase.loadLibs(this);
  File dbFile = getDatabasePath("credentials.db");
  dbFile.mkdirs();
  dbFile.delete();
  SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(dbFile,
"credentials", null);
  db.execSQL("create table credentials(u, p)");
  db.execSQL("insert into credentials(u, p) values(?, ?)", new Object[]
{username, password});
...
```

Note that references to `android.database.sqlite.SQLiteDatabase` are substituted with those of `net.sqlcipher.database.SQLiteDatabase`. To enable encryption on the WebView store, you must recompile WebKit with the `sqlcipher.so` library.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Hardcoded Password	17	0	0	17
Total	**17**	**0**	**0**	**17**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password — Critical

Package: org.owasp.webgoat.lessons.challenges

src/main/java/org/owasp/webgoat/lessons/challenges/SolutionConstants.java, line 34 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FieldAccess: PASSWORD
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/challenges/SolutionConstants.java:34
Taint Flags:

```
31 public interface SolutionConstants {
32
33  // TODO should be random generated when starting the server
34  String PASSWORD = "!!webgoat_admin_1234!!";
35 }
36
37 undefined
```

Package: org.owasp.webgoat.lessons.insecurelogin

src/main/java/org/owasp/webgoat/lessons/insecurelogin/InsecureLoginTask.java, line 36 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/insecurelogin/InsecureLoginTask.java:36
Taint Flags:

```
33  @PostMapping("/InsecureLogin/task")
34  @ResponseBody
35  public AttackResult completed(@RequestParam String username, @RequestParam String
password) {
36  if ("CaptainJack".equals(username) && "BlackPearl".equals(password)) {
37  return success(this).build();
38  }
39  return failed(this).build();
```

Package: org.owasp.webgoat.lessons.jwt

src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java, line 77 (Password Management: Hardcoded Password) — Critical

Issue Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password — Critical

Package: org.owasp.webgoat.lessons.jwt

src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java, line 77 (Password Management: Hardcoded Password) — Critical

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: follow()
File: src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java:77
Taint Flags:

```
74  String user = (String) json.get("user");
75  String password = (String) json.get("password");
76
77  if ("Jerry".equalsIgnoreCase(user) && PASSWORD.equals(password)) {
78  return ok(createNewTokens(user));
79  }
80  return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
```

src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java, line 61 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FieldAccess: PASSWORD
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java:61
Taint Flags:

```
58 })
59 public class JWTRefreshEndpoint extends AssignmentEndpoint {
60
61  public static final String PASSWORD = "bm5nhSkxCXZkKRy4";
62  private static final String JWT_PASSWORD = "bm5n3SkxCX4kKRy4";
63  private static final List<String> validRefreshTokens = new ArrayList<>();
64
```

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 91 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password — Critical

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 91 (Password Management: Hardcoded Password) — Critical

Sink Details

Sink: VariableAccess: password
Enclosing Method: validLoginWithoutCookieTest()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:91
Taint Flags:

```
88  @DisplayName("Valid credentials login without authentication cookie")
89  void validLoginWithoutCookieTest() throws Exception {
90  String username = "webgoat";
91  String password = "webgoat";
92
93  ResultActions result =
94  mockMvc.perform(
```

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java, line 58 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: password
Enclosing Method: getPassword()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java:58
Taint Flags:

```
55  }
56
57  protected String getPassword() {
58  String password = "dave";
59  try (Connection connection = dataSource.getConnection()) {
60  String query = "SELECT password FROM user_system_data WHERE user_name = 'dave'";
61  try {
```

Package: org.owasp.webgoat.webwolf.user

src/test/java/org/owasp/webgoat/webwolf/user/UserServiceTest.java, line 72 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password — Critical

Package: org.owasp.webgoat.webwolf.user

src/test/java/org/owasp/webgoat/webwolf/user/UserServiceTest.java, line 72 (Password Management: Hardcoded Password) — Critical

Sink Details

Sink: VariableAccess: password
Enclosing Method: testAddUser()
File: src/test/java/org/owasp/webgoat/webwolf/user/UserServiceTest.java:72
Taint Flags:

```
69  @Test
70  public void testAddUser() {
71  var username = "guest";
72  var password = "guest";
73
74  sut.addUser(username, password);
75
```

src/test/java/org/owasp/webgoat/webwolf/user/UserServiceTest.java, line 49 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: password
Enclosing Method: testLoadUserByUsername()
File: src/test/java/org/owasp/webgoat/webwolf/user/UserServiceTest.java:49
Taint Flags:

```
46  @Test
47  public void testLoadUserByUsername() {
48  var username = "guest";
49  var password = "123";
50  WebGoatUser user = new WebGoatUser(username, password);
51  when(mockUserRepository.findByUsername(username)).thenReturn(user);
52
```

Package: src.main.resources.lessons.jwt.js

src/main/resources/lessons/jwt/js/jwt-refresh.js, line 10 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password — Critical

Package: src.main.resources.lessons.jwt.js

src/main/resources/lessons/jwt/js/jwt-refresh.js, line 10 (Password Management: Hardcoded Password) — Critical

Sink: FieldAccess: password
Enclosing Method: login()
File: src/main/resources/lessons/jwt/js/jwt-refresh.js:10
Taint Flags:

```
7  type: 'POST',
8  url: 'JWT/refresh/login',
9  contentType: "application/json",
10  data: JSON.stringify({user: user, password: "bm5nhSkxCXZkKRy4"})
11  }).success(
12  function (response) {
13  localStorage.setItem('access_token', response['access_token']);
```

Password Management: Hardcoded Password — High

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7.java, line 34 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FieldAccess: ADMIN_PASSWORD_LINK
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7.java:34
Taint Flags:

```
31 @Slf4j
32 public class Assignment7 extends AssignmentEndpoint {
33
34  public static final String ADMIN_PASSWORD_LINK = "375afe1104f4a487a73823c50a9292a2";
35
36  private static final String TEMPLATE =
37  "Hi, you requested a password reset link, please use this <a target='_blank'"
```

src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java, line 53 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password — High

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java, line 53 (Password Management: Hardcoded Password) — High

Sink: FieldAccess: RESET_PASSWORD_PATH
Enclosing Method: ()
File: src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java:53
Taint Flags:

```
50  private MockMvc mockMvc;
51
52  private static final String CHALLENGE_PATH = "/challenge/7";
53  private static final String RESET_PASSWORD_PATH = CHALLENGE_PATH + "/reset-password";
54  private static final String GIT_PATH = CHALLENGE_PATH + "/.git";
55
56  @Mock private RestTemplate restTemplate;
```

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/XOREncodingAssignment.java, line 40 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/XOREncodingAssignment.java:40
Taint Flags:

```
37  @PostMapping("/crypto/encoding/xor")
38  @ResponseBody
39  public AttackResult completed(@RequestParam String answer_pwd1) {
40  if (answer_pwd1 != null && answer_pwd1.equals("databasepassword")) {
41  return success(this).feedback("crypto-encoding-xor.success").build();
42  }
43  return failed(this).feedback("crypto-encoding-xor.empty").build();
```

Package: org.owasp.webgoat.lessons.jwt

src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java, line 62 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password — High

Package: org.owasp.webgoat.lessons.jwt

src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java, line 62 (Password Management: Hardcoded Password) — High

Sink: FieldAccess: JWT_PASSWORD
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java:62
Taint Flags:

```
59 public class JWTRefreshEndpoint extends AssignmentEndpoint {
60
61  public static final String PASSWORD = "bm5nhSkxCXZkKRy4";
62  private static final String JWT_PASSWORD = "bm5n3SkxCX4kKRy4";
63  private static final List<String> validRefreshTokens = new ArrayList<>();
64
65  @PostMapping(
```

Package: org.owasp.webgoat.lessons.missingac

src/main/java/org/owasp/webgoat/lessons/missingac/MissingFunctionAC.java, line 32 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FieldAccess: PASSWORD_SALT_SIMPLE
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/missingac/MissingFunctionAC.java:32
Taint Flags:

```
29 @Component
30 public class MissingFunctionAC extends Lesson {
31
32  public static final String PASSWORD_SALT_SIMPLE = "DeliberatelyInsecure1234";
33  public static final String PASSWORD_SALT_ADMIN = "DeliberatelyInsecure1235";
34
35  @Override
```

src/main/java/org/owasp/webgoat/lessons/missingac/MissingFunctionAC.java, line 33 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password — High

Package: org.owasp.webgoat.lessons.missingac

src/main/java/org/owasp/webgoat/lessons/missingac/MissingFunctionAC.java, line 33 (Password Management: Hardcoded Password) — High

Sink: FieldAccess: PASSWORD_SALT_ADMIN
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/missingac/MissingFunctionAC.java:33
Taint Flags:

```
30 public class MissingFunctionAC extends Lesson {
31
32  public static final String PASSWORD_SALT_SIMPLE = "DeliberatelyInsecure1234";
33  public static final String PASSWORD_SALT_ADMIN = "DeliberatelyInsecure1235";
34
35  @Override
36  public Category getDefaultCategory() {
```

Package: org.owasp.webgoat.lessons.passwordreset

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java, line 61 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FieldAccess: PASSWORD_TOM_9
Enclosing Method: ()
File: src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java:61
Taint Flags:

```
58 public class ResetLinkAssignment extends AssignmentEndpoint {
59
60  private static final String VIEW_FORMATTER = "lessons/passwordreset/templates/%s.html";
61  static final String PASSWORD_TOM_9 =
62  "somethingVeryRandomWhichNoOneWillEverTypeInAsPasswordForTom";
63  static final String TOM_EMAIL = "tom@webgoat-cloud.org";
64  static Map<String, String> userToTomResetLink = new HashMap<>();
```

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java, line 88 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Hardcoded Password	High
Package: org.owasp.webgoat.lessons.passwordreset	
src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java, line 88 (Password Management: Hardcoded Password)	High

Sink: FunctionCall: equals
Enclosing Method: login()
File: src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java:88
Taint Flags:

```
85  if (TOM_EMAIL.equals(email)) {
86  String passwordTom =
87  usersToTomPassword.getOrDefault(getWebSession().getUserName(), PASSWORD_TOM_9);
88  if (passwordTom.equals(PASSWORD_TOM_9)) {
89  return failed(this).feedback("login_failed").build();
90  } else if (passwordTom.equals(password)) {
91  return success(this).build();
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Password in Comment (7 issues)

Abstract

Storing passwords or password details in plain text anywhere in the system or system code might compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode a password. Storing password details within comments is equivalent to hardcoding passwords. Not only is the password visible to the project's developers, it also makes fixing the problem extremely difficult. After the code is in production, the password is now leaked to the outside world and cannot be protected or changed without patching the software. If the account protected by the password is compromised, the owners of the system must choose between security and availability.
Example: The following comment specifies the default password to connect to a database:

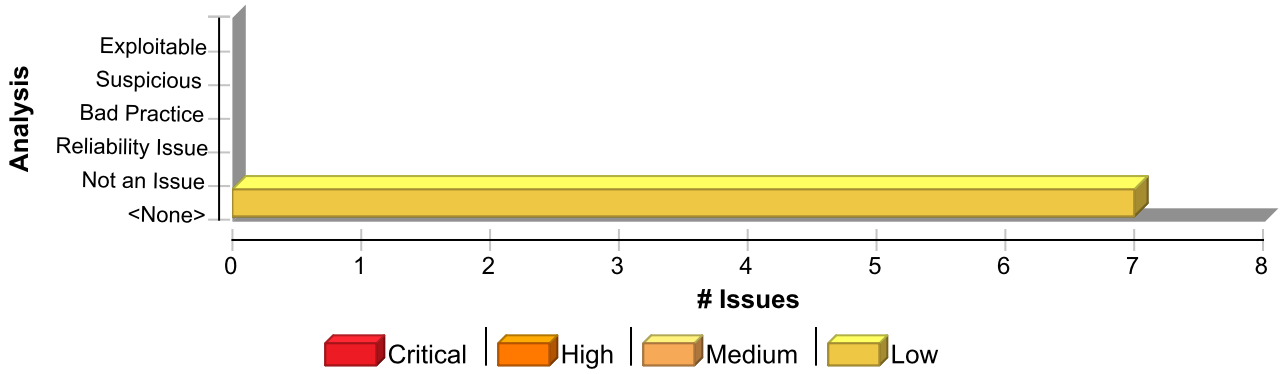
```
...
// Default username for database connection is "scott"
// Default password for database connection is "tiger"
...
```

This code will run successfully, but anyone who has access to it will have access to the password. An employee with access to this information can use it to break into the system.

Recommendation

Passwords should never be hardcoded and should generally be obfuscated and managed in an external source. Storing passwords in plain text anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the password.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Password in Comment	7	0	0	7
Total	**7**	**0**	**0**	**7**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Password in Comment — Low

Package: .org.owasp.webgoat.lessons.passwordreset

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java, line 71 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: Comment
File: src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java:71
Taint Flags:

```
68  static final String TEMPLATE =
69  """
70  Hi, you requested a password reset link, please use this <a target='_blank'
71  href='http://%s/WebGoat/PasswordReset/reset/reset-password/%s'>link</a> to reset your
72  password.
73
74  If you did not request this password change you can ignore this message.
```

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java, line 39 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: Comment
File: src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java:39
Taint Flags:

```
36 import org.springframework.web.bind.annotation.RestController;
37 import org.springframework.web.client.RestTemplate;
38
39 /**
40  * Part of the password reset assignment. Used to send the e-mail.
41  *
42  * @author nbaars
```

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java, line 106 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Password in Comment — Low

Package: .org.owasp.webgoat.lessons.passwordreset

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java, line 106 (Password Management: Password in Comment) — Low

Sink Details

Sink: Comment
File: src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignment.java:106
Taint Flags:

```
103  model.addAttribute("form", form);
104  modelAndView.addObject("form", form);
105  modelAndView.setViewName(
106  VIEW_FORMATTER.formatted("password_reset")); // Display html page for changing password
107  } else {
108  modelAndView.setViewName(VIEW_FORMATTER.formatted("password_link_not_found"));
109  }
```

Package: .org.owasp.webgoat.lessons.securepasswords

src/main/java/org/owasp/webgoat/lessons/securepasswords/SecurePasswordsAssignment.java, line 80 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: Comment
File: src/main/java/org/owasp/webgoat/lessons/securepasswords/SecurePasswordsAssignment.java:80
Taint Flags:

```
77  if (strength.getFeedback().getWarning().length() != 0)
78  output.append("<b>Warning: </b>" + strength.getFeedback().getWarning() + "</br>");
79  // possible feedback: https://github.com/dropbox/zxcvbn/blob/master/src/feedback.coffee
80  // maybe ask user to try also weak passwords to see and understand feedback?
81  if (strength.getFeedback().getSuggestions().size() != 0) {
82  output.append("<b>Suggestions:</b></br><ul>");
83  for (String sug : strength.getFeedback().getSuggestions())
```

Package: .org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java, line 58 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Password in Comment — Low

Package: .org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java, line 58 (Password Management: Password in Comment) — Low

Sink Details

Sink: Comment
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java:58
Taint Flags:

```
55  @ResponseBody
56  public AttackResult completed(@RequestParam(value = "userid_6a") String userId) {
57  return injectableQuery(userId);
58  // The answer: Smith' union select userid,user_name, password,cookie,cookie, cookie,userid
from
59  // user_system_data --
60  }
61
```

Package: src.main.resources.webgoat.static.js.jquery_form

src/main/resources/webgoat/static/js/jquery_form/jquery.form.js, line 931 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: Comment
File: src/main/resources/webgoat/static/js/jquery_form/jquery.form.js:931
Taint Flags:

```
928  return this.unbind('submit.form-plugin click.form-plugin');
929 };
930
931 /**
932  * formToArray() gathers form element data into an array of objects that can
933  * be passed to any of the following ajax functions: $.get, $.post, or load.
934  * Each object in the array has both a 'name' and 'value' property. An example of
```

Package: src.main.resources.webgoat.static.js.libs

src/main/resources/webgoat/static/js/libs/jquery.form.js, line 931 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Password in Comment	Low
Package: src.main.resources.webgoat.static.js.libs	
src/main/resources/webgoat/static/js/libs/jquery.form.js, line 931 (Password Management: Password in Comment)	Low

Sink: Comment
File: src/main/resources/webgoat/static/js/libs/jquery.form.js:931
Taint Flags:

```
928  return this.unbind('submit.form-plugin click.form-plugin');
929 };
930
931 /**
932  * formToArray() gathers form element data into an array of objects that can
933  * be passed to any of the following ajax functions: $.get, $.post, or load.
934  * Each object in the array has both a 'name' and 'value' property. An example of
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Password in Configuration File (2 issues)

Abstract

Storing a plain text password in a configuration file may result in a system compromise.

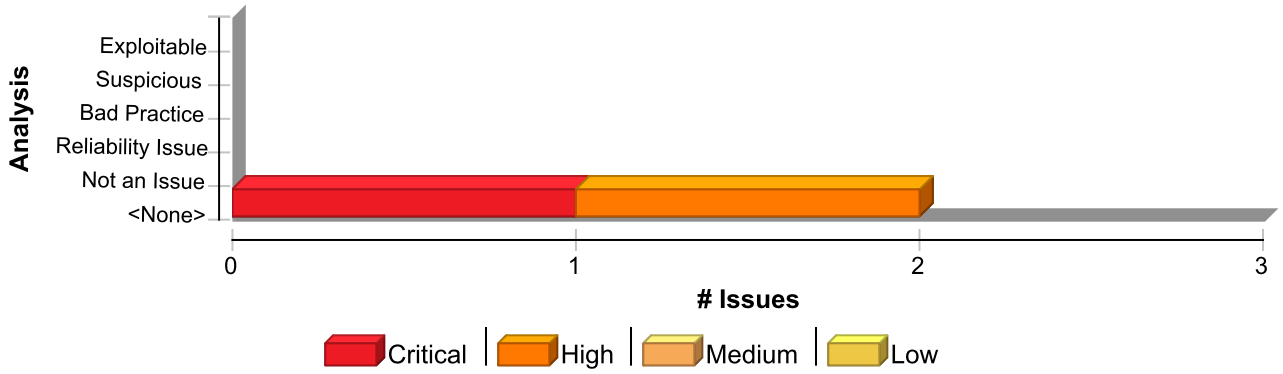
Explanation

Storing a plain text password in a configuration file allows anyone who can read the file access to the password-protected resource. Developers sometimes believe that they cannot defend the application from someone who has access to the configuration, but this attitude makes an attacker's job easier. Good password management guidelines require that a password never be stored in plain text.

Recommendation

A password should never be stored in plain text. An administrator should be required to enter the password when the system starts. If that approach is impractical, a less secure but often adequate solution is to obfuscate the password and scatter the de-obfuscation material around the system so that an attacker has to obtain and correctly combine multiple system resources to decipher the password. Some third-party products claim the ability to manage passwords in a more secure way. For example, WebSphere Application Server 4.x uses a simple XOR encryption algorithm for obfuscating values, but be skeptical about such facilities. WebSphere and other application servers offer outdated and relatively weak encryption mechanisms that are insufficient for security-sensitive environments. For a secure solution the only viable option is a proprietary one.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Password in Configuration File	2	0	0	2
Total	**2**	**0**	**0**	**2**

Password Management: Password in Configuration File	**Critical**
Package: src.main.resources.i18n	
src/main/resources/i18n/messages.properties, line 34 (Password Management: Password in Configuration File)	**Critical**
Issue Details	

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Password Management: Password in Configuration File — Critical

Package: src.main.resources.i18n

src/main/resources/i18n/messages.properties, line 34 (Password Management: Password in Configuration File) — Critical

Kingdom: Environment
Scan Engine: SCA (Configuration)

Sink Details

Sink: password
File: src/main/resources/i18n/messages.properties:34
Taint Flags:

```
31 ErrorGenerating=Error generating
32 InvalidData=Invalid Data
33 Go!=Go!
34 password=Password
35 password.confirm=Confirm password
36 username=Username
37 logged_out=You've been logged out successfully.
```

Password Management: Password in Configuration File — High

Package: src.main.resources.i18n

src/main/resources/i18n/messages.properties, line 43 (Password Management: Password in Configuration File) — High

Issue Details

Kingdom: Environment
Scan Engine: SCA (Configuration)

Sink Details

Sink: accounts.table.password
File: src/main/resources/i18n/messages.properties:43
Taint Flags:

```
40 accounts.build.in=The following accounts are built into WebGoat
41 accounts.table.account=Account
42 accounts.table.user=User
43 accounts.table.password=Password
44 logout=Logout
45 version=Version
46 build=Build
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Path Manipulation (3 issues)

Abstract

Allowing user input to control paths used in file system operations could enable an attacker to access or modify otherwise protected system resources.

Explanation

Path manipulation errors occur when the following two conditions are met: 1. An attacker can specify a path used in an operation on the file system. 2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted. For example, the program might give the attacker the ability to overwrite the specified file or run with a configuration controlled by the attacker. **Example 1:** The following code uses input from an HTTP request to create a file name. The programmer has not considered the possibility that an attacker could provide a file name such as "`../../tomcat/conf/server.xml`", which causes the application to delete one of its own configuration files.

```
String rName = request.getParameter("reportName");
File rFile = new File("/usr/local/apfr/reports/" + rName);
...
rFile.delete();
```

Example 2: The following code uses input from a configuration file to determine which file to open and echo back to the user. If the program runs with adequate privileges and malicious users can change the configuration file, they can use the program to read any file on the system that ends with the extension `.txt`.

```
fis = new FileInputStream(cfg.getProperty("sub")+".txt");
amt = fis.read(arr);
out.println(arr);
```

Some think that in the mobile environment, classic vulnerabilities, such as path manipulation, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication. **Example 3:** The following code adapts `Example 1` to the Android platform.

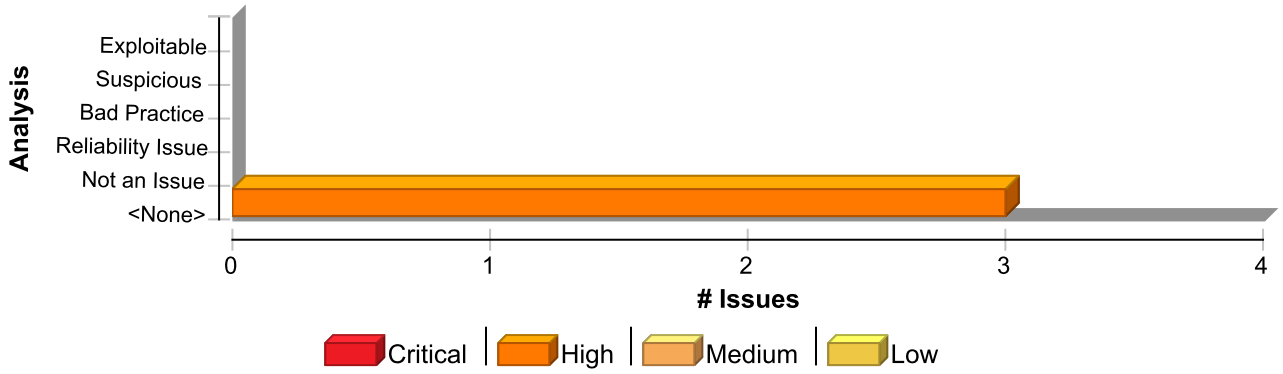
```
...
        String rName = this.getIntent().getExtras().getString("reportName");
        File rFile = getBaseContext().getFileStreamPath(rName);
...
        rFile.delete();
...
```

Recommendation

The best way to prevent path manipulation is with a level of indirection: create a list of legitimate values from which the user must select. With this approach, the user-provided input is never used directly to specify the resource name. In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to maintain. Programmers often resort to implementing a deny list in these situations. A deny list is used to selectively reject or escape potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a list of characters that are permitted to appear in the resource name and accept input composed exclusively of characters in the approved set.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Path Manipulation	3	0	0	3
Total	**3**	**0**	**0**	**3**

Path Manipulation — High

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 50 (Path Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: main(0)
From: MavenWrapperDownloader.main
File: .mvn/wrapper/MavenWrapperDownloader.java:48

```
45  */
46  private static final String PROPERTY_NAME_WRAPPER_URL = "wrapperUrl";
47
48  public static void main(String args[]) {
49  System.out.println("- Downloader started");
50  File baseDirectory = new File(args[0]);
51  System.out.println("- Using base directory: " +
baseDirectory.getAbsolutePath());
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: main()
File: .mvn/wrapper/MavenWrapperDownloader.java:50
Taint Flags: ARGS

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Path Manipulation — High

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 50 (Path Manipulation) — High

```
47
48  public static void main(String args[]) {
49  System.out.println("- Downloader started");
50  File baseDirectory = new File(args[0]);
51  System.out.println("- Using base directory: " + baseDirectory.getAbsolutePath());
52
53  // If the maven-wrapper.properties exists, read it and check if it contains a custom
```

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java, line 49 (Path Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: main(0)
From: org.owasp.webgoat.lessons.challenges.challenge7.MD5.main
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java:43

```
40   * @param args command line arguments
41   * @since ostermillerutils 1.00.00
42   */
43  public static void main(String[] args) {
44  if (args.length == 0) {
45  System.err.println("Please specify a file.");
46  } else {
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: main()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java:49
Taint Flags: ARGS

```
46  } else {
47  for (String element : args) {
48  try {
49  System.out.println(MD5.getHashString(new File(element)) + " " + element);
50  } catch (IOException x) {
51  System.err.println(x.getMessage());
52  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Path Manipulation — High

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java, line 73 (Path Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.util.zip.ZipFile.entries()
From: org.owasp.webgoat.lessons.pathtraversal.ProfileZipSlip.processZipUpload
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java:70

```
67  FileCopyUtils.copy(file.getBytes(), uploadedZipFile.toFile());
68
69  ZipFile zip = new ZipFile(uploadedZipFile.toFile());
70  Enumeration<? extends ZipEntry> entries = zip.entries();
71  while (entries.hasMoreElements()) {
72  ZipEntry e = entries.nextElement();
73  File f = new File(tmpZipDirectory.toFile(), e.getName());
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: processZipUpload()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java:73
Taint Flags: FILE_SYSTEM

```
70  Enumeration<? extends ZipEntry> entries = zip.entries();
71  while (entries.hasMoreElements()) {
72  ZipEntry e = entries.nextElement();
73  File f = new File(tmpZipDirectory.toFile(), e.getName());
74  InputStream is = zip.getInputStream(e);
75  Files.copy(is, f.toPath(), StandardCopyOption.REPLACE_EXISTING);
76  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Path Manipulation: Zip Entry Overwrite (1 issue)

Abstract

Allowing user input to control paths used in file system operations could enable an attacker to arbitrarily overwrite files on the system.

Explanation

Path Manipulation: ZIP Entry Overwrite errors occur when a ZIP file is opened and expanded without checking the file path of the ZIP entry. **Example 1:** The following example extracts files from a ZIP file and insecurely writes them to disk.

```
  private static final int BUFSIZE = 512;
  private static final int TOOBIG = 0x640000;
  ...
  public final void unzip(String filename) throws IOException {
    FileInputStream fis = new FileInputStream(filename);
    ZipInputStream zis = new ZipInputStream(new BufferedInputStream(fis));
    ZipEntry zipEntry = null;

    int numOfEntries = 0;
    long total = 0;

    try {
      while ((zipEntry = zis.getNextEntry()) != null) {
        byte data[] = new byte[BUFSIZE];
        int count = 0;
        String outFileName = zipEntry.getName();
        if (zipEntry.isDirectory()){
          new File(outFileName).mkdir(); //create the new directory
          continue;
        }
        FileOutputStream outFile = new FileOutputStream(outFileName);
        BufferedOutputStream dest = new BufferedOutputStream(outFile, BUFSIZE);
        //read data from ZIP, but do not read huge entries
        while (total + BUFSIZE <= TOOBIG && (count = zis.read(data, 0,
BUFSIZE)) != -1) {
          dest.write(data, 0, count);
          total += count;
        }
        ...
      }
    } finally{
      zis.close();
    }
  }
  ...
```

In `Example 1`, there is no validation of `zipEntry.getName()` prior to performing read/write functions on the data within this entry. If the ZIP file was originally placed in the directory "`/tmp/`" of a Unix-based machine, a ZIP entry was "`../etc/hosts`", and the application was run under the necessary permissions, it would overwrite the system `hosts` file. This in turn would allow traffic from the machine to go anywhere the attacker wants, such as back to the attacker's machine.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

© Copyright [2008-2024] Micro Focus or one of its affiliates.

The best way to prevent path manipulation via a ZIP file is with a level of indirection. Create a list of legitimate path names to which a ZIP entry can write, and write to the file that matches the ZIP entry location. With this approach, the user input in the ZIP file is never used directly to specify the file location. This may be impractical in this example. Programmers often resort to implementing a deny list in these situations. A deny list is used to selectively reject or escape potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out-of-date. A better approach is to create a list of characters that are permitted to appear in the resource name and accept input composed exclusively of characters in the approved set. In this case, APIs such as `java.io.File.getCanonicalPath()` can be used to retrieve the canonicalized form of the file path, which can be used for checking the directory in which the file will be written. **Example 2:** The following is a utility function used to check that a file name to be created is within a programmer-specified directory.

```
public class MyFileUtils{
...
  public static validateFilenameInDir(String filename, String
intendedDirectory) throws IOException{
    File checkFile = new File(filename);
    String canonicalPathToCheck = checkFile.getCanonicalPath();

    File intendedDir = new File(intendedDirectory);
    String canonicalPathToVerify = intendedDir.getCanonicalPath();

    if (canonicalPathToCheck.startsWith(canonicalPathToVerify)){
      return canonicalPathToCheck;
    } else{
      throw new IllegalStateException("This file is outside the intended
extraction directory.");
    }
  }
...
}
```

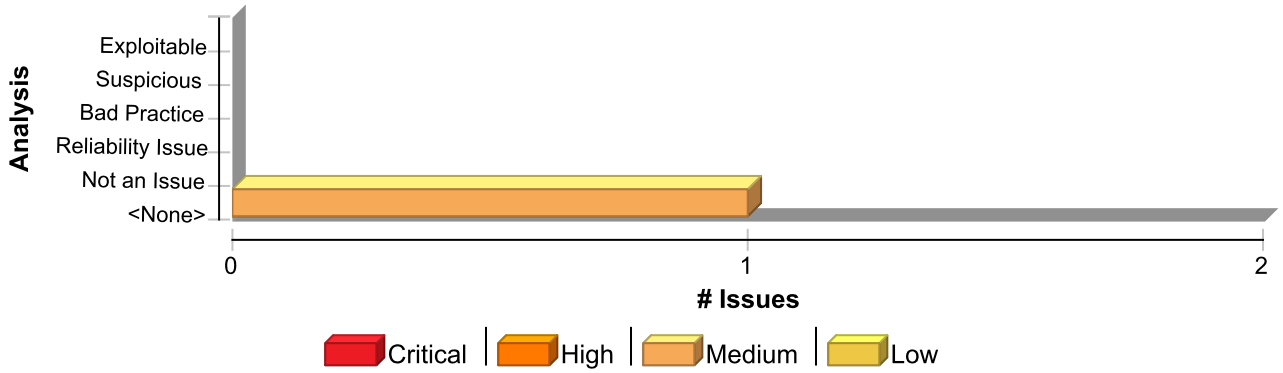
Example 3: The following uses `Example 2` to fix `Example 1`.

```
  private static final int BUFSIZE = 512;
  private static final int TOOBIG = 0x640000;
  ...
  public final void unzip(String filename) throws IOException {
    FileInputStream fis = new FileInputStream(filename);
    ZipInputStream zis = new ZipInputStream(new BufferedInputStream(fis));
    ZipEntry zipEntry = null;

    int numOfEntries = 0;
    long total = 0;

    try {
      while ((zipEntry = zis.getNextEntry()) != null) {
        byte data[] = new byte[BUFSIZE];
        int count = 0;

        // verify that the file to be written to is located in the current
directory
        String outFileName =
MyFileUtils.validateFilenameInDir(zipEntry.getName(), ".");

        if (zipEntry.isDirectory()) {
          new File(outFileName).mkdir(); //create the new directory
          continue;
        }
        FileOutputStream outFile = new FileOutputStream(outFileName);
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

```
        BufferedOutputStream dest = new BufferedOutputStream(outFile, BUFSIZE);
        //read data from ZIP, but do not read huge entries
        while (total + BUFSIZE <= TOOBIG && (count = zis.read(data, 0,
BUFSIZE)) != -1) {
          dest.write(data, 0, count);
          total += count;
        }
        ...
      }
    } finally {
      zis.close();
    }
  }
  ...
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Path Manipulation: Zip Entry Overwrite	1	0	0	1
Total	**1**	**0**	**0**	**1**

Path Manipulation: Zip Entry Overwrite	**Medium**
Package: org.owasp.webgoat.lessons.pathtraversal	
src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java, line 75 (Path Manipulation: Zip Entry Overwrite)	**Medium**

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.util.zip.ZipEntry.getName()
From: org.owasp.webgoat.lessons.pathtraversal.ProfileZipSlip.processZipUpload
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java:73

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Path Manipulation: Zip Entry Overwrite	Medium
Package: org.owasp.webgoat.lessons.pathtraversal	
src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java, line 75 (Path Manipulation: Zip Entry Overwrite)	Medium

```
70  Enumeration<? extends ZipEntry> entries = zip.entries();
71  while (entries.hasMoreElements()) {
72  ZipEntry e = entries.nextElement();
73  File f = new File(tmpZipDirectory.toFile(), e.getName());
74  InputStream is = zip.getInputStream(e);
75  Files.copy(is, f.toPath(), StandardCopyOption.REPLACE_EXISTING);
76  }
```

Sink Details

Sink: java.nio.file.Files.copy()
Enclosing Method: processZipUpload()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java:75
Taint Flags: TAINTED_PATH, ZIP_ENTRY_NAME

```
72  ZipEntry e = entries.nextElement();
73  File f = new File(tmpZipDirectory.toFile(), e.getName());
74  InputStream is = zip.getInputStream(e);
75  Files.copy(is, f.toPath(), StandardCopyOption.REPLACE_EXISTING);
76  }
77
78  return isSolved(currentImage, getProfilePictureAsBase64());
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Empty Catch Block (4 issues)

Abstract

Ignoring an exception can cause the program to overlook unexpected states and conditions.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break. Two dubious assumptions that are easy to spot in code are "this method call can never fail" and "it doesn't matter if this call fails". When a programmer ignores an exception, they implicitly state that they are operating under one of these assumptions. **Example 1:** The following code excerpt ignores a rarely-thrown exception from `doExchange()`.

```
try {
  doExchange();
}
catch (RareException e) {
  // this can never happen
}
```

If a `RareException` were to ever be thrown, the program would continue to execute as though nothing unusual had occurred. The program records no evidence indicating the special situation, potentially frustrating any later attempt to explain the program's behavior.

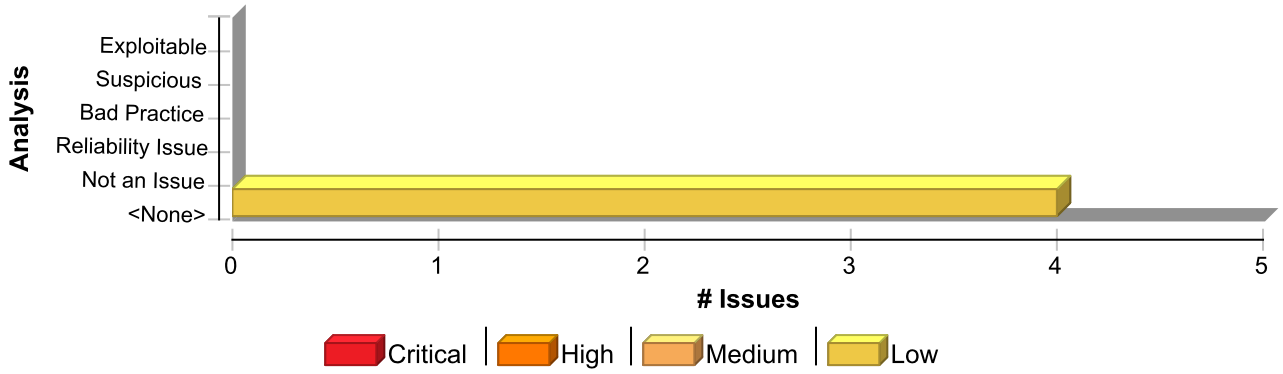
Recommendation

At a minimum, log the fact that the exception was thrown so that it will be possible to come back later and make sense of the resulting program behavior. Better yet, abort the current operation. If the exception is being ignored because the caller cannot properly handle it but the context makes it inconvenient or impossible for the caller to declare that it throws the exception itself, consider throwing a `RuntimeException` or an `Error`, both of which are unchecked exceptions. As of JDK 1.4, `RuntimeException` has a constructor that makes it easy to wrap another exception. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
try {
  doExchange();
}
catch (RareException e) {
  throw new RuntimeException("This can never happen", e);
}
```

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Empty Catch Block	4	0	0	4
Total	**4**	**0**	**0**	**4**

Poor Error Handling: Empty Catch Block — Low

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 71 (Poor Error Handling: Empty Catch Block) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: main()
File: .mvn/wrapper/MavenWrapperDownloader.java:71
Taint Flags:

```
68  if(mavenWrapperPropertyFileInputStream != null) {
69  mavenWrapperPropertyFileInputStream.close();
70  }
71  } catch (IOException e) {
72  // Ignore ...
73  }
74  }
```

Package: org.owasp.webgoat.lessons.passwordreset

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java, line 113 (Poor Error Handling: Empty Catch Block) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Empty Catch Block — Low

Package: org.owasp.webgoat.lessons.passwordreset

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java, line 113 (Poor Error Handling: Empty Catch Block) — Low

Sink: CatchBlock
Enclosing Method: fakeClickingLinkEmail()
File: src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java:113
Taint Flags:

```
110  HttpMethod.GET,
111  httpEntity,
112  Void.class);
113  } catch (Exception e) {
114  // don't care
115  }
116  }
```

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java, line 63 (Poor Error Handling: Empty Catch Block) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createUser()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java:63
Taint Flags:

```
60  connection.prepareStatement("CREATE USER unauthorized_user PASSWORD test")) {
61  statement.execute();
62  }
63  } catch (Exception e) {
64  // user already exists continue
65  }
66  }
```

Package: org.owasp.webgoat.webwolf.jwt

src/main/java/org/owasp/webgoat/webwolf/jwt/JWTToken.java, line 94 (Poor Error Handling: Empty Catch Block) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Empty Catch Block	Low
Package: org.owasp.webgoat.webwolf.jwt	
src/main/java/org/owasp/webgoat/webwolf/jwt/JWTToken.java, line 94 (Poor Error Handling: Empty Catch Block)	**Low**

Sink: CatchBlock
Enclosing Method: encode()
File: src/main/java/org/owasp/webgoat/webwolf/jwt/JWTToken.java:94
Taint Flags:

```
91  try {
92  builder.encoded(jws.getCompactSerialization());
93  builder.signatureValid(true);
94  } catch (JoseException e) {
95  // Do nothing
96  }
97  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch (35 issues)

Abstract

The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

Explanation

Multiple catch blocks can get repetitive, but "condensing" catch blocks by catching a high-level class such as `Exception` can obscure exceptions that deserve special treatment or that should not be caught at this point in the program. Catching an overly broad exception essentially defeats the purpose of Java's typed exceptions, and can become particularly dangerous if the program grows and begins to throw new types of exceptions. The new exception types will not receive any attention. **Example:** The following code excerpt handles three types of exceptions in an identical fashion.

```
try {
  doExchange();
}
catch (IOException e) {
  logger.error("doExchange failed", e);
}
catch (InvocationTargetException e) {
  logger.error("doExchange failed", e);
}
catch (SQLException e) {
  logger.error("doExchange failed", e);
}
```

At first blush, it may seem preferable to deal with these exceptions in a single catch block, as follows:

```
try {
  doExchange();
}
catch (Exception e) {
  logger.error("doExchange failed", e);
}
```

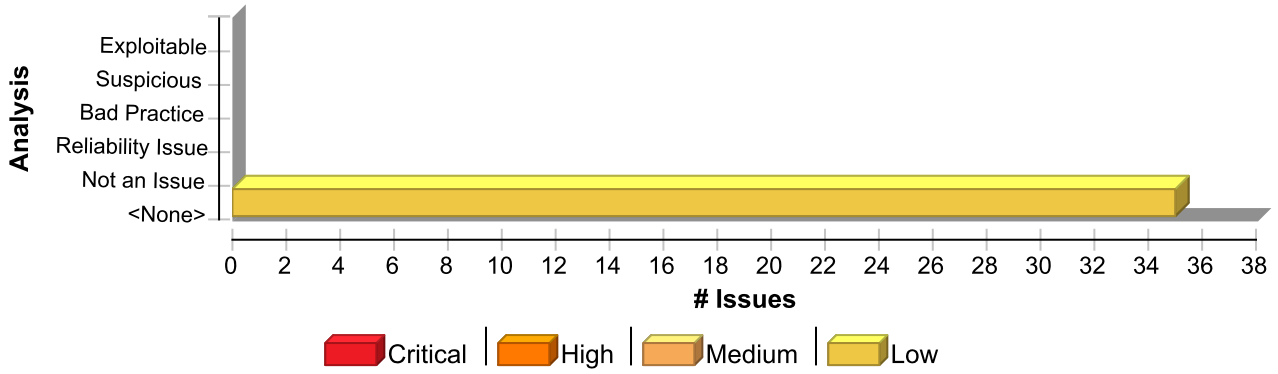
However, if `doExchange()` is modified to throw a new type of exception that should be handled in some different kind of way, the broad catch block will prevent the compiler from pointing out the situation. Further, the new catch block will now also handle exceptions derived from `RuntimeException` such as `ClassCastException`, and `NullPointerException`, which is not the programmer's intent.

Recommendation

Do not catch broad exception classes such as `Exception`, `Throwable`, `Error`, or `RuntimeException` except at the very top level of the program or thread.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Overly Broad Catch	35	0	0	35
Total	**35**	**0**	**0**	**35**

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 170 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getProperties()
File: src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java:170
Taint Flags:

```
167  prop = new Properties();
168  // load a properties file
169  prop.load(input);
170  } catch (Exception e) {
171  e.printStackTrace();
172  }
173  return prop;
```

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 38 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 38 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: runTests()
File: src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java:38
Taint Flags:

```
35
36  try {
37  checkAssignmentSigning();
38  } catch (Exception e) {
39  e.printStackTrace();
40  fail();
41  }
```

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 100 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: verifyMessage()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:100
Taint Flags:

```
97  result = instance.verify(decodedSignature);
98
99  log.info("Verified the signature with result: {}", result);
100  } catch (Exception e) {
101  log.error("Signature verification failed", e);
102  }
103
```

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 71 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 71 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: signMessage()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:71
Taint Flags:

```
68  signature = new String(Base64.getEncoder().encode(instance.sign()),
Charset.forName("UTF-8"));
69
70  log.info("signe the signature with result: {}", signature);
71  } catch (Exception e) {
72  log.error("Signature signing failed", e);
73  }
74
```

src/test/java/org/owasp/webgoat/lessons/cryptography/CryptoUtilTest.java, line 27 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: testSigningAssignment()
File: src/test/java/org/owasp/webgoat/lessons/cryptography/CryptoUtilTest.java:27
Taint Flags:

```
24  String signature = CryptoUtil.signMessage(modulus, privateKey);
25  log.debug("public exponent {}", rsaPubKey.getPublicExponent());
26  assertThat(CryptoUtil.verifyAssignment(modulus, signature, keyPair.getPublic())).isTrue();
27  } catch (Exception e) {
28  fail("Signing failed");
29  }
30  }
```

Package: org.owasp.webgoat.lessons.deserialization

src/main/java/org/owasp/webgoat/lessons/deserialization/InsecureDeserializationTask.java, line 72 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.deserialization

src/main/java/org/owasp/webgoat/lessons/deserialization/InsecureDeserializationTask.java, line 72 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/deserialization/InsecureDeserializationTask.java:72
Taint Flags:

```
69  return failed(this).feedback("insecure-deserialization.invalidversion").build();
70  } catch (IllegalArgumentException e) {
71  return failed(this).feedback("insecure-deserialization.expired").build();
72  } catch (Exception e) {
73  return failed(this).feedback("insecure-deserialization.invalidversion").build();
74  }
75
```

Package: org.owasp.webgoat.lessons.idor

src/main/java/org/owasp/webgoat/lessons/idor/IDORViewOwnProfile.java, line 62 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: invoke()
File: src/main/java/org/owasp/webgoat/lessons/idor/IDORViewOwnProfile.java:62
Taint Flags:

```
59  "error",
60  "You do not have privileges to view the profile. Authenticate as tom first please.");
61  }
62  } catch (Exception ex) {
63  log.error("something went wrong", ex.getMessage());
64  }
65  return details;
```

src/main/java/org/owasp/webgoat/lessons/idor/IDORViewOwnProfileAltUrl.java, line 71 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.idor

src/main/java/org/owasp/webgoat/lessons/idor/IDORViewOwnProfileAltUrl.java, line 71 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/idor/IDORViewOwnProfileAltUrl.java:71
Taint Flags:

```
68  } else {
69  return failed(this).feedback("idor.view.own.profile.failure2").build();
70  }
71  } catch (Exception ex) {
72  return failed(this).output("an error occurred with your request").build();
73  }
74  }
```

Package: org.owasp.webgoat.lessons.jwt

src/main/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpoint.java, line 91 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: login()
File: src/main/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpoint.java:91
Taint Flags:

```
88  return failed(this).feedback("jwt-secret-incorrect-user").feedbackArgs(user).build();
89  }
90  }
91  } catch (Exception e) {
92  return failed(this).feedback("jwt-invalid-token").output(e.getMessage()).build();
93  }
94  }
```

Package: org.owasp.webgoat.lessons.missingac

src/main/java/org/owasp/webgoat/lessons/missingac/DisplayUser.java, line 48 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.missingac

src/main/java/org/owasp/webgoat/lessons/missingac/DisplayUser.java, line 48 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: DisplayUser()
File: src/main/java/org/owasp/webgoat/lessons/missingac/DisplayUser.java:48
Taint Flags:

```
45
46  try {
47  this.userHash = genUserHash(user.getUsername(), user.getPassword(), passwordSalt);
48  } catch (Exception ex) {
49  this.userHash = "Error generating user hash";
50  }
51  }
```

src/main/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACUsers.java, line 104 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: addUser()
File: src/main/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACUsers.java:104
Taint Flags:

```
101  try {
102  userRepository.save(newUser);
103  return newUser;
104  } catch (Exception ex) {
105  log.error("Error creating new User", ex);
106  return null;
107  }
```

Package: org.owasp.webgoat.lessons.passwordreset

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java, line 82 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.passwordreset

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java, line 82 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: sendPasswordResetLink()
File: src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java:82
Taint Flags:

```
79  } else {
80  try {
81  sendMailToUser(email, host, resetLink);
82  } catch (Exception e) {
83  return failed(this).output("E-mail can't be send. please try again.").build();
84  }
85  }
```

src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java, line 113 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: fakeClickingLinkEmail()
File: src/main/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentForgotPassword.java:113
Taint Flags:

```
110  HttpMethod.GET,
111  httpEntity,
112  Void.class);
113  } catch (Exception e) {
114  // don't care
115  }
116  }
```

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 58 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 58 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: initAssignment()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java:58
Taint Flags:

```
55  new ClassPathResource("lessons/pathtraversal/images/cats/" + i + ".jpg")
56  .getInputStream()) {
57  FileCopyUtils.copy(is, new FileOutputStream(new File(catPicturesDirectory, i + ".jpg")));
58  } catch (Exception e) {
59  log.error("Unable to copy pictures" + e.getMessage());
60  }
61  }
```

Package: org.owasp.webgoat.lessons.spoofcookie

src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java, line 112 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: cookieLoginFlow()
File: src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java:112
Taint Flags:

```
109  String cookieUsername;
110  try {
111  cookieUsername = EncDec.decode(cookieValue).toLowerCase();
112  } catch (Exception e) {
113  // for providing some instructive guidance, we won't return 4xx error here
114  return failed(this).output(e.getMessage()).build();
115  }
```

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java, line 110 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java, line 110 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java:110
Taint Flags:

```
107  } catch (SQLException sqle) {
108  return failed(this).output(sqle.getMessage() + YOUR_QUERY_WAS + query).build();
109  }
110  } catch (Exception e) {
111  return failed(this)
112  .output(this.getClass().getName() + " : " + e.getMessage() + YOUR_QUERY_WAS + query)
113  .build();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java, line 74 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getPassword()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java:74
Taint Flags:

```
71  sqle.printStackTrace();
72  // do nothing
73  }
74  } catch (Exception e) {
75  e.printStackTrace();
76  // do nothing
77  }
```

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5b.java, line 124 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5b.java, line 124 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5b.java:124
Taint Flags:

```
121  sqle.getMessage() + "<br> Your query was: " + queryString.replace("?", login_count))
122  .build();
123  }
124  } catch (Exception e) {
125  return failed(this)
126  .output(
127  this.getClass().getName()
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5a.java, line 95 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5a.java:95
Taint Flags:

```
92  } catch (SQLException sqle) {
93  return failed(this).output(sqle.getMessage() + "<br> Your query was: " + query).build();
94  }
95  } catch (Exception e) {
96  return failed(this)
97  .output(
98  this.getClass().getName() + " : " + e.getMessage() + "<br> Your query was: " + query)
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java, line 109 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: injectableQueryConfidentiality()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java:109
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java, line 109 (Poor Error Handling: Overly Broad Catch) — Low

```
106  .build();
107  }
108
109  } catch (Exception e) {
110  return failed(this)
111  .output("<br><span class='feedback-negative'>" + e.getMessage() + "</span>")
112  .build();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java, line 63 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createUser()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java:63
Taint Flags:

```
60  connection.prepareStatement("CREATE USER unauthorized_user PASSWORD test")) {
61  statement.execute();
62  }
63  } catch (Exception e) {
64  // user already exists continue
65  }
66  }
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson4.java, line 76 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson4.java:76
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson4.java, line 76 (Poor Error Handling: Overly Broad Catch) — Low

```
73  } catch (SQLException sqle) {
74  return failed(this).output(sqle.getMessage()).build();
75  }
76  } catch (Exception e) {
77  return failed(this).output(this.getClass().getName() + " : " + e.getMessage()).build();
78  }
79  }
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java, line 104 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: injectableQueryAvailability()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java:104
Taint Flags:

```
101  }
102  }
103
104  } catch (Exception e) {
105  return failed(this)
106  .output("<span class='feedback-negative'>" + e.getMessage() + "</span>")
107  .build();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java, line 86 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java:86
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java, line 86 (Poor Error Handling: Overly Broad Catch) — Low

```
83  }
84  return failed(this).output("Your query was: " + query).build();
85  }
86  } catch (Exception e) {
87  return failed(this)
88  .output(
89  this.getClass().getName() + " : " + e.getMessage() + "<br> Your query was: " + query)
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5b.java, line 71 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5b.java:71
Taint Flags:

```
68  int count = 0;
69  try {
70  count = Integer.parseInt(login_count);
71  } catch (Exception e) {
72  return failed(this)
73  .output(
74  "Could not parse: "
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson3.java, line 80 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson3.java:80
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson3.java, line 80 (Poor Error Handling: Overly Broad Catch) — Low

```
77  } catch (SQLException sqle) {
78  return failed(this).output(sqle.getMessage()).build();
79  }
80  } catch (Exception e) {
81  return failed(this).output(this.getClass().getName() + " : " + e.getMessage()).build();
82  }
83  }
```

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java, line 129 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getJavaFileContentsAsString()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java:129
Taint Flags:

```
126  JavaObjectFromString javaFileObject = null;
127  try {
128  javaFileObject = new JavaObjectFromString("TestClass.java", javaFileContents.toString());
129  } catch (Exception exception) {
130  exception.printStackTrace();
131  }
132  return javaFileObject;
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java, line 100 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java:100
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java, line 100 (Poor Error Handling: Overly Broad Catch) — Low

```
97  } else {
98  return failed(this).feedback("sql-injection.10b.failed").build();
99  }
100  } catch (Exception e) {
101  return failed(this).output(e.getMessage()).build();
102  }
103  }
```

Package: org.owasp.webgoat.lessons.ssrf

src/main/java/org/owasp/webgoat/lessons/ssrf/SSRFTask1.java, line 61 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: stealTheCheese()
File: src/main/java/org/owasp/webgoat/lessons/ssrf/SSRFTask1.java:61
Taint Flags:

```
58  html.append("<img class=\"image\" alt=\"Silly Cat\" src=\"images/cat.jpg\">");
59  return failed(this).feedback("ssrf.failure").output(html.toString()).build();
60  }
61  } catch (Exception e) {
62  e.printStackTrace();
63  return failed(this).output(e.getMessage()).build();
64  }
```

Package: org.owasp.webgoat.lessons.vulnerablecomponents

src/main/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLesson.java, line 58 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLesson.java:58
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.vulnerablecomponents

src/main/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLesson.java, line 58 (Poor Error Handling: Overly Broad Catch) — Low

```
55  .replace(" <", "<");
56  }
57  contact = (Contact) xstream.fromXML(payload);
58  } catch (Exception ex) {
59  return failed(this).feedback("vulnerable-components.close").output(ex.getMessage()).build();
60  }
61
```

src/main/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLesson.java, line 70 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLesson.java:70
Taint Flags:

```
67  if (!(contact instanceof ContactImpl)) {
68  return success(this).feedback("vulnerable-components.success").build();
69  }
70  } catch (Exception e) {
71  return success(this).feedback("vulnerable-components.success").output(e.getMessage()).build();
72  }
73  return failed(this).feedback("vulnerable-components.fromXML").feedbackArgs(contact).build();
```

Package: org.owasp.webgoat.lessons.xss

src/main/java/org/owasp/webgoat/lessons/xss/CrossSiteScriptingLesson3.java, line 83 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.xss

src/main/java/org/owasp/webgoat/lessons/xss/CrossSiteScriptingLesson3.java, line 83 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/xss/CrossSiteScriptingLesson3.java:83
Taint Flags:

```
80  } else {
81  return failed(this).feedback("xss-mitigation-3-failure").build();
82  }
83  } catch (Exception e) {
84  return failed(this).output(e.getMessage()).build();
85  }
86  }
```

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignment.java, line 101 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: addComment()
File: src/main/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignment.java:101
Taint Flags:

```
98  comment.setText("Nice try, you need to send the file to WebWolf");
99  }
100  comments.addComment(comment, false);
101  } catch (Exception e) {
102  return failed(this).output(e.toString()).build();
103  }
104  return failed(this).build();
```

src/main/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignment.java, line 79 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Catch — Low

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignment.java, line 79 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: createNewUser()
File: src/main/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignment.java:79
Taint Flags:

```
76  if (checkSolution(comment)) {
77  attackResult = success(this).build();
78  }
79  } catch (Exception e) {
80  error = ExceptionUtils.getStackTrace(e);
81  attackResult =
failed(this).feedback("xxe.content.type.feedback.xml").output(error).build();
82  }
```

src/main/java/org/owasp/webgoat/lessons/xxe/SimpleXXE.java, line 81 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createNewComment()
File: src/main/java/org/owasp/webgoat/lessons/xxe/SimpleXXE.java:81
Taint Flags:

```
78  if (checkSolution(comment)) {
79  return success(this).build();
80  }
81  } catch (Exception e) {
82  error = ExceptionUtils.getStackTrace(e);
83  }
84  return failed(this).output(error).build();
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws (217 issues)

Abstract

The method throws a generic exception making it harder for callers to do a good job of error handling and recovery.

Explanation

Declaring a method to throw `Exception` or `Throwable` makes it difficult for callers to do good error handling and error recovery. Java's exception mechanism is set up to make it easy for callers to anticipate what can go wrong and write code to handle each specific exceptional circumstance. Declaring that a method throws a generic form of exception defeats this system. **Example:** The following method throws three types of exceptions.

```
public void doExchange()
  throws IOException, InvocationTargetException,
         SQLException {
  ...
}
```

While it might seem tidier to write

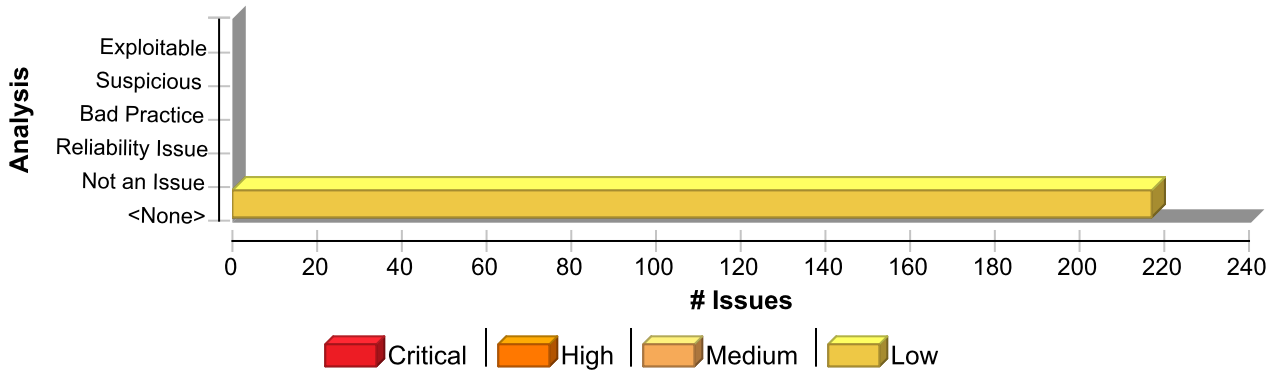
```
public void doExchange()
  throws Exception {
  ...
}
```

doing so hampers the caller's ability to understand and handle the exceptions that occur. Further, if a later revision of `doExchange()` introduces a new type of exception that should be treated differently than previous exceptions, there is no easy way to enforce this requirement.

Recommendation

Do not declare methods to throw `Exception` or `Throwable`. If the exceptions thrown by a method are not recoverable or should not generally be caught by the caller, consider throwing unchecked exceptions rather than checked exceptions. This can be accomplished by implementing exception classes that extend `RuntimeException` or `Error` instead of `Exception`, or add a try/catch wrapper in your method to convert checked exceptions to unchecked exceptions.

Issue Summary



© Copyright [2008-2024] Micro Focus or one of its affiliates.

Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Overly Broad Throws	217	0	0	217
Total	**217**	**0**	**0**	**217**

Poor Error Handling: Overly Broad Throws — Low

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 97 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: downloadFileFromURL
Enclosing Method: downloadFileFromURL()
File: .mvn/wrapper/MavenWrapperDownloader.java:97
Taint Flags:

```
94   }
95   }
96
97   private static void downloadFileFromURL(String urlString, File destination) throws
Exception {
98   if (System.getenv("MVNW_USERNAME") != null && System.getenv("MVNW_PASSWORD") != null) {
99   String username = System.getenv("MVNW_USERNAME");
100   char[] password = System.getenv("MVNW_PASSWORD").toCharArray();
```

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 44 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: readObject
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:44
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 44 (Poor Error Handling: Overly Broad Throws) — Low

```
41  *
42  * @author stupid develop
43  */
44  private void readObject(ObjectInputStream stream) throws Exception {
45  // unserialize data so taskName and taskAction are available
46  stream.defaultReadObject();
47
```

Package: org.owasp.webgoat.container

src/main/java/org/owasp/webgoat/container/WebSecurityConfig.java, line 56 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: filterChain
Enclosing Method: filterChain()
File: src/main/java/org/owasp/webgoat/container/WebSecurityConfig.java:56
Taint Flags:

```
53  private final UserService userDetailsService;
54
55  @Bean
56  public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
57  return http.authorizeHttpRequests(
58  auth ->
59  auth.requestMatchers(
```

src/main/java/org/owasp/webgoat/container/UserInterceptor.java, line 48 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: afterCompletion
Enclosing Method: afterCompletion()
File: src/main/java/org/owasp/webgoat/container/UserInterceptor.java:48
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.container

src/main/java/org/owasp/webgoat/container/UserInterceptor.java, line 48 (Poor Error Handling: Overly Broad Throws) — Low

```
45  }
46
47  @Override
48  public void afterCompletion(
49  HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)
50  throws Exception {
51  // Do nothing
```

src/main/java/org/owasp/webgoat/container/WebSecurityConfig.java, line 95 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: configureGlobal
Enclosing Method: configureGlobal()
File: src/main/java/org/owasp/webgoat/container/WebSecurityConfig.java:95
Taint Flags:

```
92  }
93
94  @Autowired
95  public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
96  auth.userDetailsService(userDetailsService);
97  }
98
```

src/main/java/org/owasp/webgoat/container/WebSecurityConfig.java, line 105 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: authenticationManager
Enclosing Method: authenticationManager()
File: src/main/java/org/owasp/webgoat/container/WebSecurityConfig.java:105
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.container

src/main/java/org/owasp/webgoat/container/WebSecurityConfig.java, line 105 (Poor Error Handling: Overly Broad Throws) — Low

```
102  }
103
104  @Bean
105  public AuthenticationManager authenticationManager(
106  AuthenticationConfiguration authenticationConfiguration) throws Exception {
107  return authenticationConfiguration.getAuthenticationManager();
108  }
```

src/main/java/org/owasp/webgoat/container/UserInterceptor.java, line 24 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postHandle
Enclosing Method: postHandle()
File: src/main/java/org/owasp/webgoat/container/UserInterceptor.java:24
Taint Flags:

```
21  }
22
23  @Override
24  public void postHandle(
25  HttpServletRequest request,
26  HttpServletResponse response,
27  Object handler,
```

src/main/java/org/owasp/webgoat/container/UserInterceptor.java, line 17 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: preHandle
Enclosing Method: preHandle()
File: src/main/java/org/owasp/webgoat/container/UserInterceptor.java:17
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.container

src/main/java/org/owasp/webgoat/container/UserInterceptor.java, line 17 (Poor Error Handling: Overly Broad Throws) — Low

```
14  private Environment env = EnvironmentExposure.getEnv();
15
16  @Override
17  public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object
handler)
18  throws Exception {
19  // Do nothing
20  return true;
```

Package: org.owasp.webgoat.container.report

src/test/java/org/owasp/webgoat/container/report/ReportCardControllerTest.java, line 52 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: withLessons
Enclosing Method: withLessons()
File: src/test/java/org/owasp/webgoat/container/report/ReportCardControllerTest.java:52
Taint Flags:

```
49
50  @Test
51  @WithMockUser(username = "guest", password = "guest")
52  void withLessons() throws Exception {
53  when(lesson.getTitle()).thenReturn("Test");
54  when(course.getTotalOfLessons()).thenReturn(1);
55  when(course.getTotalOfAssignments()).thenReturn(10);
```

Package: org.owasp.webgoat.container.service

src/test/java/org/owasp/webgoat/container/service/HintServiceTest.java, line 37 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: hintsPerAssignment
Enclosing Method: hintsPerAssignment()
File: src/test/java/org/owasp/webgoat/container/service/HintServiceTest.java:37
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.container.service

src/test/java/org/owasp/webgoat/container/service/HintServiceTest.java, line 37 (Poor Error Handling: Overly Broad Throws) — Low

```
34  }
35
36  @Test
37  void hintsPerAssignment() throws Exception {
38  Assignment assignment = Mockito.mock(Assignment.class);
39  when(assignment.getPath()).thenReturn("/HttpBasics/attack1");
40  when(assignment.getHints()).thenReturn(Lists.newArrayList("hint 1", "hint 2"));
```

src/test/java/org/owasp/webgoat/container/service/LessonMenuServiceTest.java, line 76 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: lessonsShouldBeOrdered
Enclosing Method: lessonsShouldBeOrdered()
File: src/test/java/org/owasp/webgoat/container/service/LessonMenuServiceTest.java:76
Taint Flags:

```
73  }
74
75  @Test
76  void lessonsShouldBeOrdered() throws Exception {
77  Lesson l1 = Mockito.mock(Lesson.class);
78  Lesson l2 = Mockito.mock(Lesson.class);
79  when(l1.getTitle()).thenReturn("ZA");
```

src/test/java/org/owasp/webgoat/container/service/LessonMenuServiceTest.java, line 95 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: lessonCompleted
Enclosing Method: lessonCompleted()
File: src/test/java/org/owasp/webgoat/container/service/LessonMenuServiceTest.java:95
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.container.service

src/test/java/org/owasp/webgoat/container/service/LessonMenuServiceTest.java, line 95 (Poor Error Handling: Overly Broad Throws) — Low

```
92  }
93
94  @Test
95  void lessonCompleted() throws Exception {
96  Lesson l1 = Mockito.mock(Lesson.class);
97  when(l1.getTitle()).thenReturn("ZA");
98  when(lessonTracker.isLessonSolved()).thenReturn(true);
```

src/test/java/org/owasp/webgoat/container/service/LessonProgressServiceTest.java, line 82 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: jsonLessonOverview
Enclosing Method: jsonLessonOverview()
File: src/test/java/org/owasp/webgoat/container/service/LessonProgressServiceTest.java:82
Taint Flags:

```
79  }
80
81  @Test
82  void jsonLessonOverview() throws Exception {
83  this.mockMvc
84  .perform(
85  MockMvcRequestBuilders.get("/service/lessonoverview.mvc")
```

Package: org.owasp.webgoat.lessons.bypassrestrictions

src/test/java/org/owasp/webgoat/lessons/bypassrestrictions/BypassRestrictionsFrontendValidationTest.java, line 27 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: noChangesShouldNotPassTheLesson
Enclosing Method: noChangesShouldNotPassTheLesson()
File: src/test/java/org/owasp/webgoat/lessons/bypassrestrictions/BypassRestrictionsFrontendValidationTest.java:27
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.bypassrestrictions

src/test/java/org/owasp/webgoat/lessons/bypassrestrictions/BypassRestrictionsFrontendValidationTest.java, line 27 (Poor Error Handling: Overly Broad Throws) — Low

```
24  }
25
26  @Test
27  void noChangesShouldNotPassTheLesson() throws Exception {
28  mockMvc
29  .perform(
30  MockMvcRequestBuilders.post("/BypassRestrictions/frontendValidation")
```

src/test/java/org/owasp/webgoat/lessons/bypassrestrictions/BypassRestrictionsFrontendValidationTest.java, line 44 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: bypassAllFieldShouldPass
Enclosing Method: bypassAllFieldShouldPass()
File: src/test/java/org/owasp/webgoat/lessons/bypassrestrictions/BypassRestrictionsFrontendValidationTest.java:44
Taint Flags:

```
41  }
42
43  @Test
44  void bypassAllFieldShouldPass() throws Exception {
45  mockMvc
46  .perform(
47  MockMvcRequestBuilders.post("/BypassRestrictions/frontendValidation")
```

src/test/java/org/owasp/webgoat/lessons/bypassrestrictions/BypassRestrictionsFrontendValidationTest.java, line 61 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: notBypassingAllFieldShouldNotPass
Enclosing Method: notBypassingAllFieldShouldNotPass()
File: src/test/java/org/owasp/webgoat/lessons/bypassrestrictions/BypassRestrictionsFrontendValidationTest.java:61
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.bypassrestrictions

src/test/java/org/owasp/webgoat/lessons/bypassrestrictions/BypassRestrictionsFrontendValidationTest.java, line 61 (Poor Error Handling: Overly Broad Throws) — Low

```
58  }
59
60  @Test
61  void notBypassingAllFieldShouldNotPass() throws Exception {
62  mockMvc
63  .perform(
64  MockMvcRequestBuilders.post("/BypassRestrictions/frontendValidation")
```

Package: org.owasp.webgoat.lessons.challenges

src/test/java/org/owasp/webgoat/lessons/challenges/Assignment1Test.java, line 55 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/challenges/Assignment1Test.java:55
Taint Flags:

```
52  }
53
54  @Test
55  void success() throws Exception {
56  InetAddress addr = InetAddress.getLocalHost();
57  String host = addr.getHostAddress();
58  mockMvc
```

src/test/java/org/owasp/webgoat/lessons/challenges/Assignment1Test.java, line 72 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongPassword
Enclosing Method: wrongPassword()
File: src/test/java/org/owasp/webgoat/lessons/challenges/Assignment1Test.java:72
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.challenges

src/test/java/org/owasp/webgoat/lessons/challenges/Assignment1Test.java, line 72 (Poor Error Handling: Overly Broad Throws) — Low

```
69  }
70
71  @Test
72  void wrongPassword() throws Exception {
73  mockMvc
74  .perform(
75  MockMvcRequestBuilders.post("/challenge/1")
```

Package: org.owasp.webgoat.lessons.challenges.challenge5

src/main/java/org/owasp/webgoat/lessons/challenges/challenge5/Assignment5.java, line 49 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: login
Enclosing Method: login()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge5/Assignment5.java:49
Taint Flags:

```
46
47  @PostMapping("/challenge/5")
48  @ResponseBody
49  public AttackResult login(
50  @RequestParam String username_login, @RequestParam String password_login) throws Exception
{
51  if (!StringUtils.hasText(username_login) || !StringUtils.hasText(password_login)) {
52  return failed(this).feedback("required4").build();
```

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java, line 70 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: resetPasswordTest
Enclosing Method: resetPasswordTest()
File: src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java:70
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java, line 70 (Poor Error Handling: Overly Broad Throws) — Low

```
67
68  @Test
69  @DisplayName("Reset password test")
70  void resetPasswordTest() throws Exception {
71  ResultActions result =
72  mockMvc.perform(MockMvcRequestBuilders.get(RESET_PASSWORD_PATH + "/any"));
73  result.andExpect(status().is(equalTo(HttpStatus.I_AM_A_TEAPOT.value())));
```

src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java, line 84 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: sendPasswordResetLinkTest
Enclosing Method: sendPasswordResetLinkTest()
File: src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java:84
Taint Flags:

```
81
82  @Test
83  @DisplayName("Send password reset link test")
84  void sendPasswordResetLinkTest() throws Exception {
85  ResultActions result =
86  mockMvc.perform(
87  MockMvcRequestBuilders.post(CHALLENGE_PATH)
```

src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java, line 95 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: gitTest
Enclosing Method: gitTest()
File: src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java:95
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/test/java/org/owasp/webgoat/lessons/challenges/challenge7/Assignment7Test.java, line 95 (Poor Error Handling: Overly Broad Throws) — Low

```
92
93  @Test
94  @DisplayName("git test")
95  void gitTest() throws Exception {
96  ResultActions result = mockMvc.perform(MockMvcRequestBuilders.get(GIT_PATH));
97  result.andExpect(content().contentType("application/zip"));
98  }
```

Package: org.owasp.webgoat.lessons.chromedevtools

src/test/java/org/owasp/webgoat/lessons/chromedevtools/ChromeDevToolsTest.java, line 30 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: NetworkAssignmentTest_Success
Enclosing Method: NetworkAssignmentTest_Success()
File: src/test/java/org/owasp/webgoat/lessons/chromedevtools/ChromeDevToolsTest.java:30
Taint Flags:

```
27  }
28
29  @Test
30  public void NetworkAssignmentTest_Success() throws Exception {
31  mockMvc
32  .perform(
33  MockMvcRequestBuilders.post("/ChromeDevTools/network")
```

src/test/java/org/owasp/webgoat/lessons/chromedevtools/ChromeDevToolsTest.java, line 41 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: NetworkAssignmentTest_Fail
Enclosing Method: NetworkAssignmentTest_Fail()
File: src/test/java/org/owasp/webgoat/lessons/chromedevtools/ChromeDevToolsTest.java:41
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.chromedevtools

src/test/java/org/owasp/webgoat/lessons/chromedevtools/ChromeDevToolsTest.java, line 41 (Poor Error Handling: Overly Broad Throws) — Low

```
38  }
39
40  @Test
41  public void NetworkAssignmentTest_Fail() throws Exception {
42  mockMvc
43  .perform(
44  MockMvcRequestBuilders.post("/ChromeDevTools/network")
```

Package: org.owasp.webgoat.lessons.cia

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 191 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: allAnswersFalseGetResultsReturnsFalseFalseFalseFalse
Enclosing Method: allAnswersFalseGetResultsReturnsFalseFalseFalseFalse()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:191
Taint Flags:

```
188  }
189
190  @Test
191  public void allAnswersFalseGetResultsReturnsFalseFalseFalseFalse() throws Exception {
192  String[] solution0 = {"Solution 1"};
193  String[] solution1 = {"Solution 2"};
194  String[] solution2 = {"Solution 1"};
```

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 101 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: allAnswersWrongIsFailure
Enclosing Method: allAnswersWrongIsFailure()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:101
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.cia

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 101 (Poor Error Handling: Overly Broad Throws) — Low

```
98  }
99
100 @Test
101 public void allAnswersWrongIsFailure() throws Exception {
102 String[] solution0 = {"Solution 2"};
103 String[] solution1 = {"Solution 1"};
104 String[] solution2 = {"Solution 3"};
```

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 47 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: oneAnswerWrongIsFailure
Enclosing Method: oneAnswerWrongIsFailure()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:47
Taint Flags:

```
44  }
45
46  @Test
47  public void oneAnswerWrongIsFailure() throws Exception {
48  String[] solution0 = {"Solution 1"};
49  String[] solution1 = {"Solution 1"};
50  String[] solution2 = {"Solution 4"};
```

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 29 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: allAnswersCorrectIsSuccess
Enclosing Method: allAnswersCorrectIsSuccess()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:29
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.cia

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 29 (Poor Error Handling: Overly Broad Throws) — Low

```
26  }
27
28  @Test
29  public void allAnswersCorrectIsSuccess() throws Exception {
30  String[] solution0 = {"Solution 3"};
31  String[] solution1 = {"Solution 1"};
32  String[] solution2 = {"Solution 4"};
```

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 65 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: twoAnswersWrongIsFailure
Enclosing Method: twoAnswersWrongIsFailure()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:65
Taint Flags:

```
62  }
63
64  @Test
65  public void twoAnswersWrongIsFailure() throws Exception {
66  String[] solution0 = {"Solution 1"};
67  String[] solution1 = {"Solution 1"};
68  String[] solution2 = {"Solution 4"};
```

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 167 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: secondAnswerFalseGetResultsReturnsTrueFalseTrueTrue
Enclosing Method: secondAnswerFalseGetResultsReturnsTrueFalseTrueTrue()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:167
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.cia

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 167 (Poor Error Handling: Overly Broad Throws) — Low

```
164  }
165
166  @Test
167  public void secondAnswerFalseGetResultsReturnsTrueFalseTrueTrue() throws Exception {
168  String[] solution0 = {"Solution 3"};
169  String[] solution1 = {"Solution 2"};
170  String[] solution2 = {"Solution 4"};
```

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 119 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: allAnswersCorrectGetResultsReturnsTrueTrueTrueTrue
Enclosing Method: allAnswersCorrectGetResultsReturnsTrueTrueTrueTrue()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:119
Taint Flags:

```
116  }
117
118  @Test
119  public void allAnswersCorrectGetResultsReturnsTrueTrueTrueTrue() throws Exception {
120  String[] solution0 = {"Solution 3"};
121  String[] solution1 = {"Solution 1"};
122  String[] solution2 = {"Solution 4"};
```

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 143 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: firstAnswerFalseGetResultsReturnsFalseTrueTrueTrue
Enclosing Method: firstAnswerFalseGetResultsReturnsFalseTrueTrueTrue()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:143
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.cia

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 143 (Poor Error Handling: Overly Broad Throws) — Low

```
140  }
141
142  @Test
143  public void firstAnswerFalseGetResultsReturnsFalseTrueTrueTrue() throws Exception {
144  String[] solution0 = {"Solution 2"};
145  String[] solution1 = {"Solution 1"};
146  String[] solution2 = {"Solution 4"};
```

src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java, line 83 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: threeAnswersWrongIsFailure
Enclosing Method: threeAnswersWrongIsFailure()
File: src/test/java/org/owasp/webgoat/lessons/cia/CIAQuizTest.java:83
Taint Flags:

```
80  }
81
82  @Test
83  public void threeAnswersWrongIsFailure() throws Exception {
84  String[] solution0 = {"Solution 1"};
85  String[] solution1 = {"Solution 1"};
86  String[] solution2 = {"Solution 1"};
```

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java, line 72 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: askForUnknownCouponCode
Enclosing Method: askForUnknownCouponCode()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java:72
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java, line 72 (Poor Error Handling: Overly Broad Throws) — Low

```
69  }
70
71  @Test
72  public void askForUnknownCouponCode() throws Exception {
73  mockMvc
74  .perform(
75  MockMvcRequestBuilders.get(
```

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringFreeAssignmentTest.java, line 31 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongSalary
Enclosing Method: wrongSalary()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringFreeAssignmentTest.java:31
Taint Flags:

```
28  }
29
30  @Test
31  public void wrongSalary() throws Exception {
32  mockMvc
33  .perform(
34  MockMvcRequestBuilders.post("/clientSideFiltering/attack1").param("answer", "10000"))
```

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringAssignmentTest.java, line 36 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongCouponCode
Enclosing Method: wrongCouponCode()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringAssignmentTest.java:36
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringAssignmentTest.java, line 36 (Poor Error Handling: Overly Broad Throws) — Low

```
33  }
34
35  @Test
36  public void wrongCouponCode() throws Exception {
37  mockMvc
38  .perform(
39  MockMvcRequestBuilders.post("/clientSideFiltering/getItForFree")
```

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringFreeAssignmentTest.java, line 23 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringFreeAssignmentTest.java:23
Taint Flags:

```
20  }
21
22  @Test
23  public void success() throws Exception {
24  mockMvc
25  .perform(
26  MockMvcRequestBuilders.post("/clientSideFiltering/attack1").param("answer", "450000"))
```

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java, line 64 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getCoupon
Enclosing Method: getCoupon()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java:64
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java, line 64 (Poor Error Handling: Overly Broad Throws) — Low

```
61  }
62
63  @Test
64  public void getCoupon() throws Exception {
65  mockMvc
66  .perform(MockMvcRequestBuilders.get("/clientSideFiltering/challenge-store/coupons/
webgoat"))
67  .andExpect(jsonPath("$.code", CoreMatchers.is("webgoat")))
```

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringFreeAssignmentTest.java, line 43 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getSalaries
Enclosing Method: getSalaries()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringFreeAssignmentTest.java:43
Taint Flags:

```
40  }
41
42  @Test
43  public void getSalaries() throws Exception {
44  mockMvc
45  .perform(MockMvcRequestBuilders.get("/clientSideFiltering/salaries"))
46  .andExpect(jsonPath("$[0]", Matchers.hasKey("UserID")))
```

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java, line 54 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getSuperCoupon
Enclosing Method: getSuperCoupon()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java:54
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java, line 54 (Poor Error Handling: Overly Broad Throws) — Low

```
51  }
52
53  @Test
54  public void getSuperCoupon() throws Exception {
55  mockMvc
56  .perform(
57  MockMvcRequestBuilders.get(
```

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringAssignmentTest.java, line 27 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ClientSideFilteringAssignmentTest.java:27
Taint Flags:

```
24  }
25
26  @Test
27  public void success() throws Exception {
28  mockMvc
29  .perform(
30  MockMvcRequestBuilders.post("/clientSideFiltering/getItForFree")
```

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java, line 82 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: fetchAllTheCouponsShouldContainGetItForFree
Enclosing Method: fetchAllTheCouponsShouldContainGetItForFree()
File: src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java:82
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/test/java/org/owasp/webgoat/lessons/clientsidefiltering/ShopEndpointTest.java, line 82 (Poor Error Handling: Overly Broad Throws) — Low

```
79  }
80
81  @Test
82  public void fetchAllTheCouponsShouldContainGetItForFree() throws Exception {
83  mockMvc
84  .perform(MockMvcRequestBuilders.get("/clientSideFiltering/challenge-store/coupons/"))
85  .andExpect(jsonPath("$.codes[3].code", is("get_it_for_free")));
```

Package: org.owasp.webgoat.lessons.csrf

src/test/java/org/owasp/webgoat/lessons/csrf/CSRFFeedbackTest.java, line 52 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postingJsonMessageThroughWebGoatShouldWork
Enclosing Method: postingJsonMessageThroughWebGoatShouldWork()
File: src/test/java/org/owasp/webgoat/lessons/csrf/CSRFFeedbackTest.java:52
Taint Flags:

```
49  }
50
51  @Test
52  public void postingJsonMessageThroughWebGoatShouldWork() throws Exception {
53  mockMvc
54  .perform(
55  post("/csrf/feedback/message")
```

src/test/java/org/owasp/webgoat/lessons/csrf/CSRFFeedbackTest.java, line 64 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: csrfAttack
Enclosing Method: csrfAttack()
File: src/test/java/org/owasp/webgoat/lessons/csrf/CSRFFeedbackTest.java:64
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.csrf

src/test/java/org/owasp/webgoat/lessons/csrf/CSRFFeedbackTest.java, line 64 (Poor Error Handling: Overly Broad Throws) — Low

```
61  }
62
63  @Test
64  public void csrfAttack() throws Exception {
65  mockMvc
66  .perform(
67  post("/csrf/feedback/message")
```

Package: org.owasp.webgoat.lessons.deserialization

src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java, line 33 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java:33
Taint Flags:

```
30  }
31
32  @Test
33  void success() throws Exception {
34  if (OS.indexOf("win") > -1) {
35  mockMvc
36  .perform(
```

src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java, line 98 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: checkOtherObject
Enclosing Method: checkOtherObject()
File: src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java:98
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.deserialization

src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java, line 98 (Poor Error Handling: Overly Broad Throws) — Low

```
95  }
96
97  @Test
98  void checkOtherObject() throws Exception {
99  String token =
100 "rO0ABXQAVklmIHlvdSBkZXNlcmlhbGl6ZSBtZSBkb3duLCBJIHNoYWxsIGJlY29tZSBtb3JlIHBvd2VyZnVsIHRoYW4geW9
101  mockMvc
```

src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java, line 84 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: expiredTask
Enclosing Method: expiredTask()
File: src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java:84
Taint Flags:

```
81  }
82
83  @Test
84  void expiredTask() throws Exception {
85  String token =
86 "rO0ABXNyADFvcmcuZHVtbXkuaW5zZWN1cmUuZnJhbWV3b3JrLlZ1bG5lcmFibGVUYXNrSG9sZGVyAAAAAAAAAAICAANMABZ
87  mockMvc
```

src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java, line 69 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongVersion
Enclosing Method: wrongVersion()
File: src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java:69
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.deserialization

src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java, line 69 (Poor Error Handling: Overly Broad Throws) — Low

```
66  }
67
68  @Test
69  void wrongVersion() throws Exception {
70  String token =
71
"rO0ABXNyADFvcmcuZHVtbXkuaW5zZWN1cmUuZnJhbWV3b3JrLlZ1bG5lcmFibGVUYXNrSG9sZGVyAAAAAAAAAAECAANMABZ
72  mockMvc
```

src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java, line 57 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: fail
Enclosing Method: fail()
File: src/test/java/org/owasp/webgoat/lessons/deserialization/DeserializeTest.java:57
Taint Flags:

```
54  }
55
56  @Test
57  void fail() throws Exception {
58  mockMvc
59  .perform(
60  MockMvcRequestBuilders.post("/InsecureDeserialization/task")
```

Package: org.owasp.webgoat.lessons.hijacksession

src/test/java/org/owasp/webgoat/lessons/hijacksession/HijackSessionAssignmentTest.java, line 96 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: testBlankCookie
Enclosing Method: testBlankCookie()
File: src/test/java/org/owasp/webgoat/lessons/hijacksession/HijackSessionAssignmentTest.java:96
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.hijacksession

src/test/java/org/owasp/webgoat/lessons/hijacksession/HijackSessionAssignmentTest.java, line 96 (Poor Error Handling: Overly Broad Throws) — Low

```
93  }
94
95  @Test
96  void testBlankCookie() throws Exception {
97  ResultActions result =
98  mockMvc.perform(
99  MockMvcRequestBuilders.post(LOGIN_CONTEXT_PATH)
```

src/test/java/org/owasp/webgoat/lessons/hijacksession/HijackSessionAssignmentTest.java, line 76 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: testValidCookie
Enclosing Method: testValidCookie()
File: src/test/java/org/owasp/webgoat/lessons/hijacksession/HijackSessionAssignmentTest.java:76
Taint Flags:

```
73  }
74
75  @Test
76  void testValidCookie() throws Exception {
77  lenient().when(authenticationMock.isAuthenticated()).thenReturn(true);
78  lenient()
79  .when(providerMock.authenticate(any(Authentication.class)))
```

Package: org.owasp.webgoat.lessons.httpproxies

src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java, line 81 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: missingParam
Enclosing Method: missingParam()
File: src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java:81
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.httpproxies

src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java, line 81 (Poor Error Handling: Overly Broad Throws) — Low

```
78  }
79
80  @Test
81  public void missingParam() throws Exception {
82  mockMvc
83  .perform(
84  MockMvcRequestBuilders.get("/HttpProxies/intercept-request")
```

src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java, line 109 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: whenPostAssignmentShouldNotPass
Enclosing Method: whenPostAssignmentShouldNotPass()
File: src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java:109
Taint Flags:

```
106  }
107
108  @Test
109  public void whenPostAssignmentShouldNotPass() throws Exception {
110  mockMvc
111  .perform(
112  MockMvcRequestBuilders.post("/HttpProxies/intercept-request")
```

src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java, line 66 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: failure
Enclosing Method: failure()
File: src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java:66
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.httpproxies

src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java, line 66 (Poor Error Handling: Overly Broad Throws) — Low

```
63  }
64
65  @Test
66  public void failure() throws Exception {
67  mockMvc
68  .perform(
69  MockMvcRequestBuilders.get("/HttpProxies/intercept-request")
```

src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java, line 95 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: missingHeader
Enclosing Method: missingHeader()
File: src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java:95
Taint Flags:

```
92  }
93
94  @Test
95  public void missingHeader() throws Exception {
96  mockMvc
97  .perform(
98  MockMvcRequestBuilders.get("/HttpProxies/intercept-request")
```

src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java, line 51 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java:51
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.httpproxies

src/test/java/org/owasp/webgoat/lessons/httpproxies/HttpBasicsInterceptRequestTest.java, line 51 (Poor Error Handling: Overly Broad Throws) — Low

```
48  }
49
50  @Test
51  public void success() throws Exception {
52  mockMvc
53  .perform(
54  MockMvcRequestBuilders.get("/HttpProxies/intercept-request")
```

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 76 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solveAssignmentWithBoolean
Enclosing Method: solveAssignmentWithBoolean()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:76
Taint Flags:

```
73  }
74
75  @Test
76  public void solveAssignmentWithBoolean() throws Exception {
77  // Create new token and set alg to none and do not sign it
78  Claims claims = Jwts.claims();
79  claims.put("admin", true);
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 58 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solveAssignment
Enclosing Method: solveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:58
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 58 (Poor Error Handling: Overly Broad Throws) — Low

```
55  }
56
57  @Test
58  public void solveAssignment() throws Exception {
59  // Create new token and set alg to none and do not sign it
60  Claims claims = Jwts.claims();
61  claims.put("admin", "true");
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 241 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: unknownUserShouldSeeGuestView
Enclosing Method: unknownUserShouldSeeGuestView()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:241
Taint Flags:

```
238  }
239
240  @Test
241  public void unknownUserShouldSeeGuestView() throws Exception {
242  Claims claims = Jwts.claims();
243  claims.put("admin", "true");
244  claims.put("user", "Intruder");
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 91 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: oneOfClaimIsMissingShouldNotSolveAssignment
Enclosing Method: oneOfClaimIsMissingShouldNotSolveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java:91
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 91 (Poor Error Handling: Overly Broad Throws) — Low

```
88  }
89
90  @Test
91  public void oneOfClaimIsMissingShouldNotSolveAssignment() throws Exception {
92  Claims claims = createClaims("WebGoat");
93  claims.remove("aud");
94  String token = Jwts.builder().setClaims(claims).signWith(HS512, JWT_SECRET).compact();
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 115 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: tomShouldGetAToken
Enclosing Method: tomShouldGetAToken()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:115
Taint Flags:

```
112  }
113
114  @Test
115  public void tomShouldGetAToken() throws Exception {
116  mockMvc
117  .perform(
118  MockMvcRequestBuilders.get("/JWT/votings/login")
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 266 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: noTokenWhileLoginShouldReturn401
Enclosing Method: noTokenWhileLoginShouldReturn401()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:266
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 266 (Poor Error Handling: Overly Broad Throws) — Low

```
263  }
264
265  @Test
266  void noTokenWhileLoginShouldReturn401() throws Exception {
267  mockMvc
268  .perform(MockMvcRequestBuilders.post("/JWT/refresh/login"))
269  .andExpect(status().isUnauthorized());
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 131 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: checkoutWitRandomTokenShouldFail
Enclosing Method: checkoutWitRandomTokenShouldFail()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:131
Taint Flags:

```
128  }
129
130  @Test
131  void checkoutWitRandomTokenShouldFail() throws Exception {
132  String accessTokenTom =
133
"eyJhbGciOiJIUzUxMiJ9.eyJpLXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsInVzZX
bjm7Q";
134  mockMvc
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTDecodeEndpointTest.java, line 23 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solveAssignment
Enclosing Method: solveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTDecodeEndpointTest.java:23
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTDecodeEndpointTest.java, line 23 (Poor Error Handling: Overly Broad Throws) — Low

```
20  }
21
22  @Test
23  public void solveAssignment() throws Exception {
24  mockMvc
25  .perform(
26  MockMvcRequestBuilders.post("/JWT/decode").param("jwt-encode-user", "user").content(""))
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 136 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: tomShouldSeeNumberOfVotes
Enclosing Method: tomShouldSeeNumberOfVotes()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:136
Taint Flags:

```
133  }
134
135  @Test
136  public void tomShouldSeeNumberOfVotes() throws Exception {
137  MvcResult result =
138  mockMvc
139  .perform(
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 126 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: guestShouldNotSeeNumberOfVotes
Enclosing Method: guestShouldNotSeeNumberOfVotes()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:126
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 126 (Poor Error Handling: Overly Broad Throws) — Low

```
123  }
124
125  @Test
126  public void guestShouldNotSeeNumberOfVotes() throws Exception {
127  mockMvc
128  .perform(MockMvcRequestBuilders.get("/JWT/votings").cookie(new Cookie("access_token",
"")))
129  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 168 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: tomShouldBeAbleToVote
Enclosing Method: tomShouldBeAbleToVote()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:168
Taint Flags:

```
165  }
166
167  @Test
168  public void tomShouldBeAbleToVote() throws Exception {
169  MvcResult result =
170  mockMvc
171  .perform(
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 214 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: guestShouldNotBeAbleToVote
Enclosing Method: guestShouldNotBeAbleToVote()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:214
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 214 (Poor Error Handling: Overly Broad Throws) — Low

```
211  }
212
213  @Test
214  public void guestShouldNotBeAbleToVote() throws Exception {
215  mockMvc
216  .perform(
217  MockMvcRequestBuilders.post("/JWT/votings/Admin lost password")
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 169 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: loginShouldNotWorkForJerryWithWrongPassword
Enclosing Method: loginShouldNotWorkForJerryWithWrongPassword()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:169
Taint Flags:

```
166  }
167
168  @Test
169  void loginShouldNotWorkForJerryWithWrongPassword() throws Exception {
170  ObjectMapper objectMapper = new ObjectMapper();
171
172  var loginJson = Map.of("user", "Jerry", "password", PASSWORD + "wrong");
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 104 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: guestShouldNotGetAToken
Enclosing Method: guestShouldNotGetAToken()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:104
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 104 (Poor Error Handling: Overly Broad Throws) — Low

```
101  }
102
103  @Test
104  public void guestShouldNotGetAToken() throws Exception {
105  mockMvc
106  .perform(
107  MockMvcRequestBuilders.get("/JWT/votings/login")
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 259 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: noTokenWhileRequestingNewTokenShouldReturn401
Enclosing Method: noTokenWhileRequestingNewTokenShouldReturn401()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:259
Taint Flags:

```
256  }
257
258  @Test
259  void noTokenWhileRequestingNewTokenShouldReturn401() throws Exception {
260  mockMvc
261  .perform(MockMvcRequestBuilders.post("/JWT/refresh/newToken"))
262  .andExpect(status().isUnauthorized());
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 120 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: incorrectToken
Enclosing Method: incorrectToken()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java:120
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 120 (Poor Error Handling: Overly Broad Throws) — Low

```
117  }
118
119  @Test
120  public void incorrectToken() throws Exception {
121  Claims claims = createClaims("Tom");
122  String token = Jwts.builder().setClaims(claims).signWith(HS512,
"wrong_password").compact();
123
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 80 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solveAssignmentWithLowercase
Enclosing Method: solveAssignmentWithLowercase()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java:80
Taint Flags:

```
77  }
78
79  @Test
80  public void solveAssignmentWithLowercase() throws Exception {
81  Claims claims = createClaims("webgoat");
82  String token = Jwts.builder().setClaims(claims).signWith(HS512, JWT_SECRET).compact();
83
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 223 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: unknownUserWithValidTokenShouldNotBeAbleToVote
Enclosing Method: unknownUserWithValidTokenShouldNotBeAbleToVote()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:223
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 223 (Poor Error Handling: Overly Broad Throws) — Low

```
220  }
221
222  @Test
223  public void unknownUserWithValidTokenShouldNotBeAbleToVote() throws Exception {
224  Claims claims = Jwts.claims();
225  claims.put("admin", "true");
226  claims.put("user", "Intruder");
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTDecodeEndpointTest.java, line 32 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongUserShouldNotSolveAssignment
Enclosing Method: wrongUserShouldNotSolveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTDecodeEndpointTest.java:32
Taint Flags:

```
29  }
30
31  @Test
32  public void wrongUserShouldNotSolveAssignment() throws Exception {
33  mockMvc
34  .perform(
35  MockMvcRequestBuilders.post("/JWT/decode")
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 94 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: resetWithoutTokenShouldNotWork
Enclosing Method: resetWithoutTokenShouldNotWork()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:94
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 94 (Poor Error Handling: Overly Broad Throws) — Low

```
91  }
92
93  @Test
94  public void resetWithoutTokenShouldNotWork() throws Exception {
95  mockMvc
96  .perform(
97  MockMvcRequestBuilders.post("/JWT/votings").contentType(MediaType.APPLICATION_JSON))
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 100 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solutionWithAlgNone
Enclosing Method: solutionWithAlgNone()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:100
Taint Flags:

```
97  }
98
99  @Test
100  void solutionWithAlgNone() throws Exception {
101  String tokenWithNoneAlgorithm =
102  Jwts.builder()
103  .setHeaderParam("alg", "none")
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 182 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: loginShouldNotWorkForTom
Enclosing Method: loginShouldNotWorkForTom()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:182
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 182 (Poor Error Handling: Overly Broad Throws) — Low

```
179  }
180
181  @Test
182  void loginShouldNotWorkForTom() throws Exception {
183  ObjectMapper objectMapper = new ObjectMapper();
184
185  var loginJson = Map.of("user", "Tom", "password", PASSWORD);
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 119 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: checkoutWithTomsTokenFromAccessLogShouldFail
Enclosing Method: checkoutWithTomsTokenFromAccessLogShouldFail()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:119
Taint Flags:

```
116  }
117
118  @Test
119  void checkoutWithTomsTokenFromAccessLogShouldFail() throws Exception {
120  String accessTokenTom =
121
"eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsInVzZX
bjm7Q";
122  mockMvc
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 252 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: noTokenWhileCheckoutShouldReturn401
Enclosing Method: noTokenWhileCheckoutShouldReturn401()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:252
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 252 (Poor Error Handling: Overly Broad Throws) — Low

```
249  }
250
251  @Test
252  void noTokenWhileCheckoutShouldReturn401() throws Exception {
253  mockMvc
254  .perform(MockMvcRequestBuilders.post("/JWT/refresh/checkout"))
255  .andExpect(status().isUnauthorized());
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 69 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solveAssignment
Enclosing Method: solveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java:69
Taint Flags:

```
66  }
67
68  @Test
69  public void solveAssignment() throws Exception {
70  Claims claims = createClaims("WebGoat");
71  String token = Jwts.builder().setClaims(claims).signWith(HS512, JWT_SECRET).compact();
72
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 54 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solveAssignment
Enclosing Method: solveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:54
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 54 (Poor Error Handling: Overly Broad Throws) — Low

```
51  }
52
53  @Test
54  void solveAssignment() throws Exception {
55  ObjectMapper objectMapper = new ObjectMapper();
56
57  // First login to obtain tokens for Jerry
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 195 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: newTokenShouldWorkForJerry
Enclosing Method: newTokenShouldWorkForJerry()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:195
Taint Flags:

```
192  }
193
194  @Test
195  void newTokenShouldWorkForJerry() throws Exception {
196  ObjectMapper objectMapper = new ObjectMapper();
197  Map<String, Object> loginJson = new HashMap<>();
198  loginJson.put("user", "Jerry");
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 156 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: invalidTokenShouldSeeGuestView
Enclosing Method: invalidTokenShouldSeeGuestView()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:156
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 156 (Poor Error Handling: Overly Broad Throws) — Low

```
153  }
154
155  @Test
156  public void invalidTokenShouldSeeGuestView() throws Exception {
157  mockMvc
158  .perform(
159  MockMvcRequestBuilders.get("/JWT/votings")
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 132 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: unsignedToken
Enclosing Method: unsignedToken()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java:132
Taint Flags:

```
129  }
130
131  @Test
132  void unsignedToken() throws Exception {
133  Claims claims = createClaims("WebGoat");
134  String token = Jwts.builder().setClaims(claims).compact();
135
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 224 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: unknownRefreshTokenShouldGiveUnauthorized
Enclosing Method: unknownRefreshTokenShouldGiveUnauthorized()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:224
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 224 (Poor Error Handling: Overly Broad Throws) — Low

```
221  }
222
223  @Test
224  void unknownRefreshTokenShouldGiveUnauthorized() throws Exception {
225  ObjectMapper objectMapper = new ObjectMapper();
226  Map<String, Object> loginJson = new HashMap<>();
227  loginJson.put("user", "Jerry");
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 144 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: flowForJerryAlwaysWorks
Enclosing Method: flowForJerryAlwaysWorks()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:144
Taint Flags:

```
141  }
142
143  @Test
144  void flowForJerryAlwaysWorks() throws Exception {
145  ObjectMapper objectMapper = new ObjectMapper();
146
147  var loginJson = Map.of("user", "Jerry", "password", PASSWORD);
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 105 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: incorrectUser
Enclosing Method: incorrectUser()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java:105
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpointTest.java, line 105 (Poor Error Handling: Overly Broad Throws) — Low

```
102  }
103
104  @Test
105  public void incorrectUser() throws Exception {
106  Claims claims = createClaims("Tom");
107  String token = Jwts.builder().setClaims(claims).signWith(HS512, JWT_SECRET).compact();
108
```

Package: org.owasp.webgoat.lessons.jwt.claimmisuse

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java, line 52 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: withJerrysKeyShouldNotSolveAssignment
Enclosing Method: withJerrysKeyShouldNotSolveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java:52
Taint Flags:

```
49  }
50
51  @Test
52  public void withJerrysKeyShouldNotSolveAssignment() throws Exception {
53  mockMvc
54  .perform(
55  MockMvcRequestBuilders.post("/JWT/kid/delete").param("token", TOKEN_JERRY).content(""))
```

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java, line 56 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java:56
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt.claimmisuse

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java, line 56 (Poor Error Handling: Overly Broad Throws) — Low

```
53  }
54
55  @Test
56  void solve() throws Exception {
57  setupJsonWebKeySetInWebWolf();
58  var token = createTokenAndSignIt();
59
```

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java, line 35 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: setup
Enclosing Method: setup()
File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java:35
Taint Flags:

```
32  private int port;
33
34  @BeforeEach
35  public void setup() throws Exception {
36  when(webSession.getCurrentLesson()).thenReturn(new JWT());
37  this.mockMvc = MockMvcBuilders.webAppContextSetup(this.wac).build();
38
```

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java, line 49 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: generateRsaKey
Enclosing Method: generateRsaKey()
File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java:49
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt.claimmisuse

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java, line 49 (Poor Error Handling: Overly Broad Throws) — Low

```
46  this.port = webwolfServer.port();
47  }
48
49  private KeyPair generateRsaKey() throws Exception {
50  KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
51  keyPairGenerator.initialize(2048);
52  return keyPairGenerator.generateKeyPair();
```

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java, line 33 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solveAssignment
Enclosing Method: solveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java:33
Taint Flags:

```
30  }
31
32  @Test
33  public void solveAssignment() throws Exception {
34  String key = "deletingTom";
35  Map<String, Object> claims = new HashMap<>();
36  claims.put("username", "Tom");
```

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java, line 63 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: shouldNotBeAbleToBypassWithSimpleToken
Enclosing Method: shouldNotBeAbleToBypassWithSimpleToken()
File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java:63
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.jwt.claimmisuse

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpointTest.java, line 63 (Poor Error Handling: Overly Broad Throws) — Low

```
60  }
61
62  @Test
63  public void shouldNotBeAbleToBypassWithSimpleToken() throws Exception {
64  mockMvc
65  .perform(
66  MockMvcRequestBuilders.post("/JWT/kid/delete")
```

src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java, line 68 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: shouldFailNotPresent
Enclosing Method: shouldFailNotPresent()
File: src/test/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpointTest.java:68
Taint Flags:

```
65
66  @Test
67  @DisplayName("When JWKS is not present in WebWolf then the call should fail")
68  void shouldFailNotPresent() throws Exception {
69  var token = createTokenAndSignIt();
70
71  mockMvc
```

Package: org.owasp.webgoat.lessons.missingac

src/main/java/org/owasp/webgoat/lessons/missingac/DisplayUser.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: genUserHash
Enclosing Method: genUserHash()
File: src/main/java/org/owasp/webgoat/lessons/missingac/DisplayUser.java:53
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.missingac

src/main/java/org/owasp/webgoat/lessons/missingac/DisplayUser.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

```
50  }
51  }
52
53  protected String genUserHash(String username, String password, String passwordSalt)
54  throws Exception {
55  MessageDigest md = MessageDigest.getInstance("SHA-256");
56  // salting is good, but static & too predictable ... short too for a salt
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACYourHashAdminTest.java, line 24 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACYourHashAdminTest.java:24
Taint Flags:

```
21  }
22
23  @Test
24  void solve() throws Exception {
25  var userHash =
26  new DisplayUser(new User("Jerry", "doesnotreallymatter", true), PASSWORD_SALT_ADMIN)
27  .getUserHash();
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACHiddenMenusTest.java, line 50 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: HiddenMenusSuccess
Enclosing Method: HiddenMenusSuccess()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACHiddenMenusTest.java:50
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.missingac

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACHiddenMenusTest.java, line 50 (Poor Error Handling: Overly Broad Throws) — Low

```
47  }
48
49  @Test
50  public void HiddenMenusSuccess() throws Exception {
51  mockMvc
52  .perform(
53  MockMvcRequestBuilders.post("/access-control/hidden-menu")
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACHiddenMenusTest.java, line 78 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: HiddenMenusFailure
Enclosing Method: HiddenMenusFailure()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACHiddenMenusTest.java:78
Taint Flags:

```
75  }
76
77  @Test
78  public void HiddenMenusFailure() throws Exception {
79  mockMvc
80  .perform(
81  MockMvcRequestBuilders.post("/access-control/hidden-menu")
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACUsersTest.java, line 46 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getUsers
Enclosing Method: getUsers()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACUsersTest.java:46
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.missingac

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACUsersTest.java, line 46 (Poor Error Handling: Overly Broad Throws) — Low

```
43  }
44
45  @Test
46  void getUsers() throws Exception {
47  mockMvc
48  .perform(
49  MockMvcRequestBuilders.get("/access-control/users")
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionYourHashTest.java, line 45 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: hashDoesNotMatch
Enclosing Method: hashDoesNotMatch()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionYourHashTest.java:45
Taint Flags:

```
42  }
43
44  @Test
45  void hashDoesNotMatch() throws Exception {
46  mockMvc
47  .perform(MockMvcRequestBuilders.post("/access-control/user-hash").param("userHash", "42"))
48  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionYourHashTest.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: hashMatches
Enclosing Method: hashMatches()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionYourHashTest.java:53
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.missingac

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionYourHashTest.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

```
50  }
51
52  @Test
53  void hashMatches() throws Exception {
54  mockMvc
55  .perform(
56  MockMvcRequestBuilders.post("/access-control/user-hash")
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACYourHashAdminTest.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongUserHash
Enclosing Method: wrongUserHash()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACYourHashAdminTest.java:42
Taint Flags:

```
39  }
40
41  @Test
42  void wrongUserHash() throws Exception {
43  mockMvc
44  .perform(
45  MockMvcRequestBuilders.post("/access-control/user-hash-fix").param("userHash", "wrong"))
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACHiddenMenusTest.java, line 64 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: HiddenMenusClose
Enclosing Method: HiddenMenusClose()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACHiddenMenusTest.java:64
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.missingac

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACHiddenMenusTest.java, line 64 (Poor Error Handling: Overly Broad Throws) — Low

```
61  }
62
63  @Test
64  public void HiddenMenusClose() throws Exception {
65  mockMvc
66  .perform(
67  MockMvcRequestBuilders.post("/access-control/hidden-menu")
```

src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACUsersTest.java, line 60 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: addUser
Enclosing Method: addUser()
File: src/test/java/org/owasp/webgoat/lessons/missingac/MissingFunctionACUsersTest.java:60
Taint Flags:

```
57  }
58
59  @Test
60  void addUser() throws Exception {
61  var user =
62  """
63  {"username":"newUser","password":"newUser12","admin": "true"}
```

Package: org.owasp.webgoat.lessons.passwordreset

src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java, line 45 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: twoQuestionsShouldSolveTheAssignment
Enclosing Method: twoQuestionsShouldSolveTheAssignment()
File: src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java:45
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.passwordreset

src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java, line 45 (Poor Error Handling: Overly Broad Throws) — Low

```
42  }
43
44  @Test
45  public void twoQuestionsShouldSolveTheAssignment() throws Exception {
46  MockHttpSession mocksession = new MockHttpSession();
47  mockMvc
48  .perform(
```

src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java, line 79 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: knownLinkShouldReturnPasswordResetPage
Enclosing Method: knownLinkShouldReturnPasswordResetPage()
File: src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java:79
Taint Flags:

```
76  }
77
78  @Test
79  void knownLinkShouldReturnPasswordResetPage() throws Exception {
80  // Create a reset link
81  mockMvc
82  .perform(
```

src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: changePasswordWithoutPasswordShouldReturnPasswordForm
Enclosing Method: changePasswordWithoutPasswordShouldReturnPasswordForm()
File: src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java:53
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.passwordreset

src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

```
50  }
51
52  @Test
53  void changePasswordWithoutPasswordShouldReturnPasswordForm() throws Exception {
54  MvcResult mvcResult =
55  mockMvc
56  .perform(MockMvcRequestBuilders.post("/PasswordReset/reset/change-password"))
```

src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java, line 65 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: changePasswordWithoutLinkShouldReturnPasswordLinkNotFound
Enclosing Method: changePasswordWithoutLinkShouldReturnPasswordLinkNotFound()
File: src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java:65
Taint Flags:

```
62  }
63
64  @Test
65  void changePasswordWithoutLinkShouldReturnPasswordLinkNotFound() throws Exception {
66  MvcResult mvcResult =
67  mockMvc
68  .perform(
```

src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java, line 30 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: oneQuestionShouldNotSolveTheAssignment
Enclosing Method: oneQuestionShouldNotSolveTheAssignment()
File: src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java:30
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.passwordreset

src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java, line 30 (Poor Error Handling: Overly Broad Throws) — Low

```
27  }
28
29  @Test
30  public void oneQuestionShouldNotSolveTheAssignment() throws Exception {
31  mockMvc
32  .perform(
33  MockMvcRequestBuilders.post("/PasswordReset/SecurityQuestions")
```

src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java, line 68 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: answeringSameQuestionTwiceShouldNotSolveAssignment
Enclosing Method: answeringSameQuestionTwiceShouldNotSolveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java:68
Taint Flags:

```
65  }
66
67  @Test
68  public void answeringSameQuestionTwiceShouldNotSolveAssignment() throws Exception {
69  MockHttpSession mocksession = new MockHttpSession();
70  mockMvc
71  .perform(
```

src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java, line 40 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongResetLink
Enclosing Method: wrongResetLink()
File: src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java:40
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.passwordreset

src/test/java/org/owasp/webgoat/lessons/passwordreset/ResetLinkAssignmentTest.java, line 40 (Poor Error Handling: Overly Broad Throws) — Low

```
37  }
38
39  @Test
40  void wrongResetLink() throws Exception {
41  MvcResult mvcResult =
42  mockMvc
43  .perform(
```

src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java, line 91 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solvingForOneUserDoesNotSolveForOtherUser
Enclosing Method: solvingForOneUserDoesNotSolveForOtherUser()
File: src/test/java/org/owasp/webgoat/lessons/passwordreset/SecurityQuestionAssignmentTest.java:91
Taint Flags:

```
88  }
89
90  @Test
91  public void solvingForOneUserDoesNotSolveForOtherUser() throws Exception {
92  MockHttpSession mocksession = new MockHttpSession();
93  mockMvc.perform(
94  MockMvcRequestBuilders.post("/PasswordReset/SecurityQuestions")
```

Package: org.owasp.webgoat.lessons.pathtraversal

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadFixTest.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: normalUpdate
Enclosing Method: normalUpdate()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadFixTest.java:42
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadFixTest.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

```
39  }
40
41  @Test
42  public void normalUpdate() throws Exception {
43  var profilePicture =
44  new MockMultipartFile(
45  "uploadedFileFix", "picture.jpg", "text/plain", "an image".getBytes());
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java, line 26 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java:26
Taint Flags:

```
23  }
24
25  @Test
26  public void solve() throws Exception {
27  var profilePicture =
28  new MockMultipartFile(
29  "uploadedFile", "../picture.jpg", "text/plain", "an image".getBytes());
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java, line 33 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java:33
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java, line 33 (Poor Error Handling: Overly Broad Throws) — Low

```
30  }
31
32  @Test
33  public void solve() throws Exception {
34  // Look at the response
35  mockMvc
36  .perform(get("/PathTraversal/random-picture"))
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRemoveUserInputTest.java, line 26 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRemoveUserInputTest.java:26
Taint Flags:

```
23  }
24
25  @Test
26  public void solve() throws Exception {
27  var profilePicture =
28  new MockMultipartFile(
29  "uploadedFileRemoveUserInput", "../picture.jpg", "text/plain", "an image".getBytes());
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadFixTest.java, line 26 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadFixTest.java:26
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadFixTest.java, line 26 (Poor Error Handling: Overly Broad Throws) — Low

```
23  }
24
25  @Test
26  public void solve() throws Exception {
27  var profilePicture =
28  new MockMultipartFile(
29  "uploadedFileFix", "../picture.jpg", "text/plain", "an image".getBytes());
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: attemptWithWrongDirectory
Enclosing Method: attemptWithWrongDirectory()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java:42
Taint Flags:

```
39  }
40
41  @Test
42  public void attemptWithWrongDirectory() throws Exception {
43  var profilePicture =
44  new MockMultipartFile(
45  "uploadedFile", "../picture.jpg", "text/plain", "an image".getBytes());
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java, line 69 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: shouldReceiveRandomPicture
Enclosing Method: shouldReceiveRandomPicture()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java:69
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java, line 69 (Poor Error Handling: Overly Broad Throws) — Low

```
66  }
67
68  @Test
69  public void shouldReceiveRandomPicture() throws Exception {
70  mockMvc
71  .perform(get("/PathTraversal/random-picture"))
72  .andExpect(status().is(200))
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRemoveUserInputTest.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: normalUpdate
Enclosing Method: normalUpdate()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRemoveUserInputTest.java:42
Taint Flags:

```
39  }
40
41  @Test
42  public void normalUpdate() throws Exception {
43  var profilePicture =
44  new MockMultipartFile(
45  "uploadedFileRemoveUserInput", "picture.jpg", "text/plain", "an image".getBytes());
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java, line 78 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: unknownFileShouldGiveDirectoryContents
Enclosing Method: unknownFileShouldGiveDirectoryContents()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java:78
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrievalTest.java, line 78 (Poor Error Handling: Overly Broad Throws) — Low

```
75  }
76
77  @Test
78  public void unknownFileShouldGiveDirectoryContents() throws Exception {
79  mockMvc
80  .perform(get("/PathTraversal/random-picture?id=test"))
81  .andExpect(status().is(404))
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java, line 77 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: normalUpdate
Enclosing Method: normalUpdate()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java:77
Taint Flags:

```
74  }
75
76  @Test
77  public void normalUpdate() throws Exception {
78  var profilePicture =
79  new MockMultipartFile("uploadedFile", "picture.jpg", "text/plain", "an image".getBytes());
80
```

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java, line 59 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: shouldNotOverrideExistingFile
Enclosing Method: shouldNotOverrideExistingFile()
File: src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java:59
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/test/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadTest.java, line 59 (Poor Error Handling: Overly Broad Throws) — Low

```
56  }
57
58  @Test
59  public void shouldNotOverrideExistingFile() throws Exception {
60  var profilePicture =
61  new MockMultipartFile("uploadedFile", "picture.jpg", "text/plain", "an image".getBytes());
62  mockMvc
```

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 191 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: eraseAuthenticationCookie
Enclosing Method: eraseAuthenticationCookie()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:191
Taint Flags:

```
188
189  @Test
190  @DisplayName("Erase authentication cookie")
191  void eraseAuthenticationCookie() throws Exception {
192  mockMvc
193  .perform(MockMvcRequestBuilders.get(ERASE_COOKIE_CONTEXT_PATH))
194  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 125 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: invalidLoginWithUnsatisfiedServletRequestParameterExceptionOnUsernameMissing
Enclosing Method: invalidLoginWithUnsatisfiedServletRequestParameterExceptionOnUsernameMissing()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:125
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 125 (Poor Error Handling: Overly Broad Throws) — Low

```
122
123  @Test
124  @DisplayName("UnsatisfiedServletRequestParameterException test for missing username")
125  void invalidLoginWithUnsatisfiedServletRequestParameterExceptionOnUsernameMissing()
126  throws Exception {
127  mockMvc
128  .perform(MockMvcRequestBuilders.post(LOGIN_CONTEXT_PATH).param("password", "anypassword"))
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 89 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: validLoginWithoutCookieTest
Enclosing Method: validLoginWithoutCookieTest()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:89
Taint Flags:

```
86
87  @Test
88  @DisplayName("Valid credentials login without authentication cookie")
89  void validLoginWithoutCookieTest() throws Exception {
90  String username = "webgoat";
91  String password = "webgoat";
92
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 112 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: cookieLoginNotSolvedFlow
Enclosing Method: cookieLoginNotSolvedFlow()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:112
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 112 (Poor Error Handling: Overly Broad Throws) — Low

```
109  + "1.- Invalid encoded cookie sent. "
110  + "2.- Valid cookie login (not tom) sent. "
111  + "3.- Valid cookie with not known username sent ")
112  void cookieLoginNotSolvedFlow(String cookieValue) throws Exception {
113  Cookie cookie = new Cookie(COOKIE_NAME, cookieValue);
114  mockMvc
115  .perform(
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 179 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: invalidTomLogin
Enclosing Method: invalidTomLogin()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:179
Taint Flags:

```
176
177  @Test
178  @DisplayName("Invalid login with tom username")
179  void invalidTomLogin() throws Exception {
180  ResultActions result =
181  mockMvc.perform(
182  MockMvcRequestBuilders.post(LOGIN_CONTEXT_PATH)
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 134 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: invalidLoginWithUnsatisfiedServletRequestParameterExceptionOnPasswordMissing
Enclosing Method: invalidLoginWithUnsatisfiedServletRequestParameterExceptionOnPasswordMissing()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:134
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 134 (Poor Error Handling: Overly Broad Throws) — Low

```
131
132  @Test
133  @DisplayName("UnsatisfiedServletRequestParameterException test for missing password")
134  void invalidLoginWithUnsatisfiedServletRequestParameterExceptionOnPasswordMissing()
135  throws Exception {
136  mockMvc
137  .perform(MockMvcRequestBuilders.post(LOGIN_CONTEXT_PATH).param("username", "webgoat"))
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 167 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: cheat
Enclosing Method: cheat()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:167
Taint Flags:

```
164
165  @Test
166  @DisplayName("cheater test")
167  void cheat() throws Exception {
168  ResultActions result =
169  mockMvc.perform(
170  MockMvcRequestBuilders.post(LOGIN_CONTEXT_PATH)
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 155 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: invalidLoginWithBlankPassword
Enclosing Method: invalidLoginWithBlankPassword()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:155
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 155 (Poor Error Handling: Overly Broad Throws) — Low

```
152
153  @Test
154  @DisplayName("Invalid blank password login")
155  void invalidLoginWithBlankPassword() throws Exception {
156  ResultActions result =
157  mockMvc.perform(
158  MockMvcRequestBuilders.post(LOGIN_CONTEXT_PATH)
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 73 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:73
Taint Flags:

```
70
71  @Test
72  @DisplayName("Lesson completed")
73  void success() throws Exception {
74  Cookie cookie = new Cookie(COOKIE_NAME, "NjI2MTcwNGI3YTQxNGE1OTU2NzQ2ZDZmNzQ=");
75
76  ResultActions result =
```

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 143 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: invalidLoginWithBlankCredentials
Enclosing Method: invalidLoginWithBlankCredentials()
File: src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java:143
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.spoofcookie

src/test/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignmentTest.java, line 143 (Poor Error Handling: Overly Broad Throws) — Low

```
140
141  @Test
142  @DisplayName("Invalid blank credentials login")
143  void invalidLoginWithBlankCredentials() throws Exception {
144  ResultActions result =
145  mockMvc.perform(
146  MockMvcRequestBuilders.post(LOGIN_CONTEXT_PATH)
```

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallenge.java, line 56 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: registerNewUser
Enclosing Method: registerNewUser()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallenge.java:56
Taint Flags:

```
53  @PutMapping("/SqlInjectionAdvanced/challenge")
54  // assignment path is bounded to class so we use different http method :-)
55  @ResponseBody
56  public AttackResult registerNewUser(
57  @RequestParam String username_reg,
58  @RequestParam String email_reg,
59  @RequestParam String password_reg)
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallengeLogin.java, line 52 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: login
Enclosing Method: login()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallengeLogin.java:52
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallengeLogin.java, line 52 (Poor Error Handling: Overly Broad Throws) — Low

```
49
50  @PostMapping("/SqlInjectionAdvanced/challenge_Login")
51  @ResponseBody
52  public AttackResult login(
53  @RequestParam String username_login, @RequestParam String password_login) throws Exception
{
54  try (var connection = dataSource.getConnection()) {
55  var statement =
```

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6bTest.java, line 36 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: submitCorrectPassword
Enclosing Method: submitCorrectPassword()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6bTest.java:36
Taint Flags:

```
33 public class SqlInjectionLesson6bTest extends SqlLessonTest {
34
35  @Test
36  public void submitCorrectPassword() throws Exception {
37  mockMvc
38  .perform(
39  MockMvcRequestBuilders.post("/SqlInjectionAdvanced/attack6b")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson2Test.java, line 36 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solution
Enclosing Method: solution()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson2Test.java:36
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson2Test.java, line 36 (Poor Error Handling: Overly Broad Throws) — Low

```
33 public class SqlInjectionLesson2Test extends SqlLessonTest {
34
35  @Test
36  public void solution() throws Exception {
37  mockMvc
38  .perform(
39  MockMvcRequestBuilders.post("/SqlInjection/attack2")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java, line 95 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: OnlySmithMustMostEarning
Enclosing Method: OnlySmithMustMostEarning()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java:95
Taint Flags:

```
92  }
93
94  @Test
95  public void OnlySmithMustMostEarning() throws Exception {
96  mockMvc
97  .perform(
98  MockMvcRequestBuilders.post("/SqlInjection/attack9")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java, line 70 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: noGrantShouldNotSolveIt
Enclosing Method: noGrantShouldNotSolveIt()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java:70
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java, line 70 (Poor Error Handling: Overly Broad Throws) — Low

```
67  }
68
69  @Test
70  public void noGrantShouldNotSolveIt() throws Exception {
71  mockMvc
72  .perform(
73  MockMvcRequestBuilders.post("/SqlInjection/attack5")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java, line 41 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: oneAccount
Enclosing Method: oneAccount()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java:41
Taint Flags:

```
38 public class SqlInjectionLesson8Test extends SqlLessonTest {
39
40  @Test
41  public void oneAccount() throws Exception {
42  mockMvc
43  .perform(
44  MockMvcRequestBuilders.post("/SqlInjection/attack8")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java, line 84 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongTANReturnsNoAccounts
Enclosing Method: wrongTANReturnsNoAccounts()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java:84
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java, line 84 (Poor Error Handling: Overly Broad Throws) — Low

```
81  }
82
83  @Test
84  public void wrongTANReturnsNoAccounts() throws Exception {
85  mockMvc
86  .perform(
87  MockMvcRequestBuilders.post("/SqlInjection/attack8")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 37 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongSolution
Enclosing Method: wrongSolution()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java:37
Taint Flags:

```
34 public class SqlInjectionLesson6aTest extends SqlLessonTest {
35
36  @Test
37  public void wrongSolution() throws Exception {
38  mockMvc
39  .perform(
40  MockMvcRequestBuilders.post("/SqlInjectionAdvanced/attack6a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java, line 67 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: sqlInjection
Enclosing Method: sqlInjection()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java:67
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java, line 67 (Poor Error Handling: Overly Broad Throws) — Low

```
64  }
65
66  @Test
67  public void sqlInjection() throws Exception {
68  mockMvc
69  .perform(
70  MockMvcRequestBuilders.post("/SqlInjection/assignment5a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java, line 43 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: malformedQueryReturnsError
Enclosing Method: malformedQueryReturnsError()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java:43
Taint Flags:

```
40  private final String completedError = "JSON path \"lessonCompleted\"";
41
42  @Test
43  public void malformedQueryReturnsError() throws Exception {
44  try {
45  mockMvc
46  .perform(
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 66 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongDataTypeOfColumns
Enclosing Method: wrongDataTypeOfColumns()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java:66
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 66 (Poor Error Handling: Overly Broad Throws) — Low

```
63  }
64
65  @Test
66  public void wrongDataTypeOfColumns() throws Exception {
67  mockMvc
68  .perform(
69  MockMvcRequestBuilders.post("/SqlInjectionAdvanced/attack6a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java, line 71 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongNameReturnsNoAccounts
Enclosing Method: wrongNameReturnsNoAccounts()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java:71
Taint Flags:

```
68  }
69
70  @Test
71  public void wrongNameReturnsNoAccounts() throws Exception {
72  mockMvc
73  .perform(
74  MockMvcRequestBuilders.post("/SqlInjection/attack8")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10Test.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: tableExistsIsFailure
Enclosing Method: tableExistsIsFailure()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10Test.java:42
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10Test.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

```
39  private String completedError = "JSON path \"lessonCompleted\"";
40
41  @Test
42  public void tableExistsIsFailure() throws Exception {
43  try {
44  mockMvc
45  .perform(MockMvcRequestBuilders.post("/SqlInjection/attack10").param("action_string", ""))
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 79 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: correctSolution
Enclosing Method: correctSolution()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java:79
Taint Flags:

```
76  }
77
78  @Test
79  public void correctSolution() throws Exception {
80  mockMvc
81  .perform(
82  MockMvcRequestBuilders.post("/SqlInjectionAdvanced/attack6a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java, line 69 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: SmithIsNotMostEarning
Enclosing Method: SmithIsNotMostEarning()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java:69
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java, line 69 (Poor Error Handling: Overly Broad Throws) — Low

```
66  }
67
68  @Test
69  public void SmithIsNotMostEarning() throws Exception {
70  mockMvc
71  .perform(
72  MockMvcRequestBuilders.post("/SqlInjection/attack9")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 47 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongNumberOfColumns
Enclosing Method: wrongNumberOfColumns()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java:47
Taint Flags:

```
44  }
45
46  @Test
47  public void wrongNumberOfColumns() throws Exception {
48  mockMvc
49  .perform(
50  MockMvcRequestBuilders.post("/SqlInjectionAdvanced/attack6a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: unknownAccount
Enclosing Method: unknownAccount()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java:53
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

```
50
51  @Disabled
52  @Test
53  public void unknownAccount() throws Exception {
54  mockMvc
55  .perform(
56  MockMvcRequestBuilders.post("/SqlInjection/assignment5a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java, line 83 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: OnlySmithSalaryMustBeUpdated
Enclosing Method: OnlySmithSalaryMustBeUpdated()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java:83
Taint Flags:

```
80  }
81
82  @Test
83  public void OnlySmithSalaryMustBeUpdated() throws Exception {
84  mockMvc
85  .perform(
86  MockMvcRequestBuilders.post("/SqlInjection/attack9")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 90 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: noResultsReturned
Enclosing Method: noResultsReturned()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java:90
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 90 (Poor Error Handling: Overly Broad Throws) — Low

```
87  }
88
89  @Test
90  public void noResultsReturned() throws Exception {
91  mockMvc
92  .perform(
93  MockMvcRequestBuilders.post("/SqlInjectionAdvanced/attack6a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java, line 54 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: multipleAccounts
Enclosing Method: multipleAccounts()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java:54
Taint Flags:

```
51  }
52
53  @Test
54  public void multipleAccounts() throws Exception {
55  mockMvc
56  .perform(
57  MockMvcRequestBuilders.post("/SqlInjection/attack8")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6bTest.java, line 46 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: submitWrongPassword
Enclosing Method: submitWrongPassword()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6bTest.java:46
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6bTest.java, line 46 (Poor Error Handling: Overly Broad Throws) — Low

```
43  }
44
45  @Test
46  public void submitWrongPassword() throws Exception {
47  mockMvc
48  .perform(
49  MockMvcRequestBuilders.post("/SqlInjectionAdvanced/attack6b")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java, line 60 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: differentTableShouldNotSolveIt
Enclosing Method: differentTableShouldNotSolveIt()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java:60
Taint Flags:

```
57  }
58
59  @Test
60  public void differentTableShouldNotSolveIt() throws Exception {
61  mockMvc
62  .perform(
63  MockMvcRequestBuilders.post("/SqlInjection/attack5")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java, line 97 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: malformedQueryReturnsError
Enclosing Method: malformedQueryReturnsError()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java:97
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8Test.java, line 97 (Poor Error Handling: Overly Broad Throws) — Low

```
94  }
95
96  @Test
97  public void malformedQueryReturnsError() throws Exception {
98  mockMvc
99  .perform(
100  MockMvcRequestBuilders.post("/SqlInjection/attack8")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java, line 50 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: grantSolution
Enclosing Method: grantSolution()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java:50
Taint Flags:

```
47  }
48
49  @Test
50  public void grantSolution() throws Exception {
51  mockMvc
52  .perform(
53  MockMvcRequestBuilders.post("/SqlInjection/attack5")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java, line 38 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: knownAccountShouldDisplayData
Enclosing Method: knownAccountShouldDisplayData()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java:38
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java, line 38 (Poor Error Handling: Overly Broad Throws) — Low

```
35 public class SqlInjectionLesson5aTest extends SqlLessonTest {
36
37  @Test
38  public void knownAccountShouldDisplayData() throws Exception {
39  mockMvc
40  .perform(
41  MockMvcRequestBuilders.post("/SqlInjection/assignment5a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java, line 81 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: sqlInjectionWrongShouldDisplayError
Enclosing Method: sqlInjectionWrongShouldDisplayError()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5aTest.java:81
Taint Flags:

```
78  }
79
80  @Test
81  public void sqlInjectionWrongShouldDisplayError() throws Exception {
82  mockMvc
83  .perform(
84  MockMvcRequestBuilders.post("/SqlInjection/assignment5a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10Test.java, line 61 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: tableMissingIsSuccess
Enclosing Method: tableMissingIsSuccess()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10Test.java:61
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10Test.java, line 61 (Poor Error Handling: Overly Broad Throws) — Low

```
58  }
59
60  @Test
61  public void tableMissingIsSuccess() throws Exception {
62  mockMvc
63  .perform(
64  MockMvcRequestBuilders.post("/SqlInjection/attack10")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java, line 107 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: SmithIsMostEarningCompletesAssignment
Enclosing Method: SmithIsMostEarningCompletesAssignment()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9Test.java:107
Taint Flags:

```
104  }
105
106  @Test
107  public void SmithIsMostEarningCompletesAssignment() throws Exception {
108  mockMvc
109  .perform(
110  MockMvcRequestBuilders.post("/SqlInjection/attack9")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 101 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: noUnionUsed
Enclosing Method: noUnionUsed()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java:101
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson6aTest.java, line 101 (Poor Error Handling: Overly Broad Throws) — Low

```
98  }
99
100 @Test
101 public void noUnionUsed() throws Exception {
102 mockMvc
103  .perform(
104 MockMvcRequestBuilders.post("/SqlInjectionAdvanced/attack6a")
```

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 96 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: falseShouldSortById
Enclosing Method: falseShouldSortById()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:96
Taint Flags:

```
93  }
94
95  @Test
96  public void falseShouldSortById() throws Exception {
97  mockMvc
98  .perform(
99  MockMvcRequestBuilders.get("/SqlInjectionMitigations/servers")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationTest.java, line 28 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: containsSpace
Enclosing Method: containsSpace()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationTest.java:28
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationTest.java, line 28 (Poor Error Handling: Overly Broad Throws) — Low

```
25  }
26
27  @Test
28  public void containsSpace() throws Exception {
29  mockMvc
30  .perform(
31  MockMvcRequestBuilders.post("/SqlOnlyInputValidation/attack")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 39 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: addressCorrectShouldOrderByHostnameUsingSubstr
Enclosing Method: addressCorrectShouldOrderByHostnameUsingSubstr()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:39
Taint Flags:

```
36  }
37
38  @Test
39  public void addressCorrectShouldOrderByHostnameUsingSubstr() throws Exception {
40  mockMvc
41  .perform(
42  MockMvcRequestBuilders.get("/SqlInjectionMitigations/servers")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationTest.java, line 15 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationTest.java:15
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationTest.java, line 15 (Poor Error Handling: Overly Broad Throws) — Low

```
12 public class SqlOnlyInputValidationTest extends SqlLessonTest {
13
14  @Test
15  public void solve() throws Exception {
16  mockMvc
17  .perform(
18  MockMvcRequestBuilders.post("/SqlOnlyInputValidation/attack")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationOnKeywordsTest.java, line 28 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: containsForbiddenSqlKeyword
Enclosing Method: containsForbiddenSqlKeyword()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationOnKeywordsTest.java:28
Taint Flags:

```
25  }
26
27  @Test
28  public void containsForbiddenSqlKeyword() throws Exception {
29  mockMvc
30  .perform(
31  MockMvcRequestBuilders.post("/SqlOnlyInputValidationOnKeywords/attack")
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java, line 138 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: JavaObjectFromString
Enclosing Method: JavaObjectFromString()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java:138
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java, line 138 (Poor Error Handling: Overly Broad Throws) — Low

```
135  class JavaObjectFromString extends SimpleJavaFileObject {
136  private String contents = null;
137
138  public JavaObjectFromString(String className, String contents) throws Exception {
139  super(new URI(className), Kind.SOURCE);
140  this.contents = contents;
141  }
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/Servers.java, line 67 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: sort
Enclosing Method: sort()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/Servers.java:67
Taint Flags:

```
64
65  @GetMapping(produces = MediaType.APPLICATION_JSON_VALUE)
66  @ResponseBody
67  public List<Server> sort(@RequestParam String column) throws Exception {
68  List<Server> servers = new ArrayList<>();
69
70  try (var connection = dataSource.getConnection()) {
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 107 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: addressIncorrectShouldOrderByHostname
Enclosing Method: addressIncorrectShouldOrderByHostname()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:107
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 107 (Poor Error Handling: Overly Broad Throws) — Low

```
104  }
105
106  @Test
107  public void addressIncorrectShouldOrderByHostname() throws Exception {
108  mockMvc
109  .perform(
110  MockMvcRequestBuilders.get("/SqlInjectionMitigations/servers")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationOnKeywordsTest.java, line 15 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlOnlyInputValidationOnKeywordsTest.java:15
Taint Flags:

```
12 public class SqlOnlyInputValidationOnKeywordsTest extends SqlLessonTest {
13
14  @Test
15  public void solve() throws Exception {
16  mockMvc
17  .perform(
18  MockMvcRequestBuilders.post("/SqlOnlyInputValidationOnKeywords/attack")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 130 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postingWrongAnswerShouldNotPassTheLesson
Enclosing Method: postingWrongAnswerShouldNotPassTheLesson()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:130
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 130 (Poor Error Handling: Overly Broad Throws) — Low

```
127  }
128
129  @Test
130  public void postingWrongAnswerShouldNotPassTheLesson() throws Exception {
131  mockMvc
132  .perform(
133  MockMvcRequestBuilders.post("/SqlInjectionMitigations/attack12a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 72 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: addressIncorrectShouldOrderByIdUsingSubstr
Enclosing Method: addressIncorrectShouldOrderByIdUsingSubstr()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:72
Taint Flags:

```
69  }
70
71  @Test
72  public void addressIncorrectShouldOrderByIdUsingSubstr() throws Exception {
73  mockMvc
74  .perform(
75  MockMvcRequestBuilders.get("/SqlInjectionMitigations/servers")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 18 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: knownAccountShouldDisplayData
Enclosing Method: knownAccountShouldDisplayData()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:18
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 18 (Poor Error Handling: Overly Broad Throws) — Low

```
15 public class SqlInjectionLesson13Test extends SqlLessonTest {
16
17  @Test
18  public void knownAccountShouldDisplayData() throws Exception {
19  mockMvc
20  .perform(
21  MockMvcRequestBuilders.get("/SqlInjectionMitigations/servers").param("column", "id"))
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 120 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postingCorrectAnswerShouldPassTheLesson
Enclosing Method: postingCorrectAnswerShouldPassTheLesson()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:120
Taint Flags:

```
117  }
118
119  @Test
120  public void postingCorrectAnswerShouldPassTheLesson() throws Exception {
121  mockMvc
122  .perform(
123  MockMvcRequestBuilders.post("/SqlInjectionMitigations/attack12a")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 85 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: trueShouldSortByHostname
Enclosing Method: trueShouldSortByHostname()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:85
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 85 (Poor Error Handling: Overly Broad Throws) — Low

```
82  }
83
84  @Test
85  public void trueShouldSortByHostname() throws Exception {
86  mockMvc
87  .perform(
88  MockMvcRequestBuilders.get("/SqlInjectionMitigations/servers")
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java, line 26 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: addressCorrectShouldOrderByHostname
Enclosing Method: addressCorrectShouldOrderByHostname()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13Test.java:26
Taint Flags:

```
23  }
24
25  @Test
26  public void addressCorrectShouldOrderByHostname() throws Exception {
27  mockMvc
28  .perform(
29  MockMvcRequestBuilders.get("/SqlInjectionMitigations/servers")
```

Package: org.owasp.webgoat.lessons.ssrf

src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest1.java, line 30 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: modifyUrlTom
Enclosing Method: modifyUrlTom()
File: src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest1.java:30
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.ssrf

src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest1.java, line 30 (Poor Error Handling: Overly Broad Throws) — Low

```
27  }
28
29  @Test
30  public void modifyUrlTom() throws Exception {
31  mockMvc
32  .perform(MockMvcRequestBuilders.post("/SSRF/task1").param("url", "images/tom.png"))
33  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest2.java, line 60 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: modifyUrlCat
Enclosing Method: modifyUrlCat()
File: src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest2.java:60
Taint Flags:

```
57  }
58
59  @Test
60  public void modifyUrlCat() throws Exception {
61  mockMvc
62  .perform(MockMvcRequestBuilders.post("/SSRF/task2").param("url", "images/cat.jpg"))
63  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest2.java, line 52 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: modifyUrlIfconfigPro
Enclosing Method: modifyUrlIfconfigPro()
File: src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest2.java:52
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.ssrf

src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest2.java, line 52 (Poor Error Handling: Overly Broad Throws) — Low

```
49  }
50
51  @Test
52  public void modifyUrlIfconfigPro() throws Exception {
53  mockMvc
54  .perform(MockMvcRequestBuilders.post("/SSRF/task2").param("url", "http://ifconfig.pro"))
55  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest1.java, line 46 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: modifyUrlCat
Enclosing Method: modifyUrlCat()
File: src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest1.java:46
Taint Flags:

```
43  }
44
45  @Test
46  public void modifyUrlCat() throws Exception {
47  mockMvc
48  .perform(MockMvcRequestBuilders.post("/SSRF/task1").param("url", "images/cat.jpg"))
49  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest1.java, line 38 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: modifyUrlJerry
Enclosing Method: modifyUrlJerry()
File: src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest1.java:38
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.ssrf

src/test/java/org/owasp/webgoat/lessons/ssrf/SSRFTest1.java, line 38 (Poor Error Handling: Overly Broad Throws) — Low

```
35  }
36
37  @Test
38  public void modifyUrlJerry() throws Exception {
39  mockMvc
40  .perform(MockMvcRequestBuilders.post("/SSRF/task1").param("url", "images/jerry.png"))
41  .andExpect(status().isOk())
```

Package: org.owasp.webgoat.lessons.vulnerablecomponents

src/test/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLessonTest.java, line 50 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: testTransformation
Enclosing Method: testTransformation()
File: src/test/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLessonTest.java:50
Taint Flags:

```
47  String contact = "<contact>\n" + "</contact>";
48
49  @Test
50  public void testTransformation() throws Exception {
51  XStream xstream = new XStream();
52  xstream.setClassLoader(Contact.class.getClassLoader());
53  xstream.alias("contact", ContactImpl.class);
```

src/test/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLessonTest.java, line 73 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: testIllegalPayload
Enclosing Method: testIllegalPayload()
File: src/test/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLessonTest.java:73
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.vulnerablecomponents

src/test/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLessonTest.java, line 73 (Poor Error Handling: Overly Broad Throws) — Low

```
70  }
71
72  @Test
73  public void testIllegalPayload() throws Exception {
74  XStream xstream = new XStream();
75  xstream.setClassLoader(Contact.class.getClassLoader());
76  xstream.alias("contact", ContactImpl.class);
```

src/test/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLessonTest.java, line 60 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: testIllegalTransformation
Enclosing Method: testIllegalTransformation()
File: src/test/java/org/owasp/webgoat/lessons/vulnerablecomponents/VulnerableComponentsLessonTest.java:60
Taint Flags:

```
57
58  @Test
59  @Disabled
60  public void testIllegalTransformation() throws Exception {
61  XStream xstream = new XStream();
62  xstream.setClassLoader(Contact.class.getClassLoader());
63  xstream.alias("contact", ContactImpl.class);
```

Package: org.owasp.webgoat.lessons.xss

src/test/java/org/owasp/webgoat/lessons/xss/DOMCrossSiteScriptingTest.java, line 54 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/xss/DOMCrossSiteScriptingTest.java:54
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xss

src/test/java/org/owasp/webgoat/lessons/xss/DOMCrossSiteScriptingTest.java, line 54 (Poor Error Handling: Overly Broad Throws) — Low

```
51  }
52
53  @Test
54  public void success() throws Exception {
55  mockMvc
56  .perform(
57  MockMvcRequestBuilders.post("/CrossSiteScripting/phone-home-xss")
```

src/test/java/org/owasp/webgoat/lessons/xss/StoredXssCommentsTest.java, line 86 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: isNotEncoded
Enclosing Method: isNotEncoded()
File: src/test/java/org/owasp/webgoat/lessons/xss/StoredXssCommentsTest.java:86
Taint Flags:

```
83
84  // Ensures it is vulnerable
85  @Test
86  public void isNotEncoded() throws Exception {
87  // do get to get comments after posting xss payload
88  ResultActions taintedResults =
89  mockMvc.perform(MockMvcRequestBuilders.get("/CrossSiteScriptingStored/stored-xss"));
```

src/test/java/org/owasp/webgoat/lessons/xss/StoredXssCommentsTest.java, line 68 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: failure
Enclosing Method: failure()
File: src/test/java/org/owasp/webgoat/lessons/xss/StoredXssCommentsTest.java:68
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xss

src/test/java/org/owasp/webgoat/lessons/xss/StoredXssCommentsTest.java, line 68 (Poor Error Handling: Overly Broad Throws) — Low

```
65  }
66
67  @Test
68  public void failure() throws Exception {
69  ResultActions results =
70  mockMvc.perform(
71  MockMvcRequestBuilders.post("/CrossSiteScriptingStored/stored-xss")
```

src/test/java/org/owasp/webgoat/lessons/xss/StoredXssCommentsTest.java, line 55 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/xss/StoredXssCommentsTest.java:55
Taint Flags:

```
52  }
53
54  @Test
55  public void success() throws Exception {
56  ResultActions results =
57  mockMvc.perform(
58  MockMvcRequestBuilders.post("/CrossSiteScriptingStored/stored-xss")
```

src/test/java/org/owasp/webgoat/lessons/xss/CrossSiteScriptingLesson1Test.java, line 66 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: failure
Enclosing Method: failure()
File: src/test/java/org/owasp/webgoat/lessons/xss/CrossSiteScriptingLesson1Test.java:66
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xss

src/test/java/org/owasp/webgoat/lessons/xss/CrossSiteScriptingLesson1Test.java, line 66 (Poor Error Handling: Overly Broad Throws) — Low

```
63  }
64
65  @Test
66  void failure() throws Exception {
67  mockMvc
68  .perform(MockMvcRequestBuilders.post(CONTEXT_PATH))
69  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/xss/CrossSiteScriptingLesson1Test.java, line 58 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: success
Enclosing Method: success()
File: src/test/java/org/owasp/webgoat/lessons/xss/CrossSiteScriptingLesson1Test.java:58
Taint Flags:

```
55  }
56
57  @Test
58  void success() throws Exception {
59  mockMvc
60  .perform(MockMvcRequestBuilders.post(CONTEXT_PATH).param("checkboxAttack1", "value"))
61  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/xss/DOMCrossSiteScriptingTest.java, line 68 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: failure
Enclosing Method: failure()
File: src/test/java/org/owasp/webgoat/lessons/xss/DOMCrossSiteScriptingTest.java:68
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xss

src/test/java/org/owasp/webgoat/lessons/xss/DOMCrossSiteScriptingTest.java, line 68 (Poor Error Handling: Overly Broad Throws) — Low

```
65  }
66
67  @Test
68  public void failure() throws Exception {
69  mockMvc
70  .perform(
71  MockMvcRequestBuilders.post("/CrossSiteScripting/phone-home-xss")
```

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java, line 69 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: workingAttack
Enclosing Method: workingAttack()
File: src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java:69
Taint Flags:

```
66  }
67
68  @Test
69  public void workingAttack() throws Exception {
70  mockMvc
71  .perform(
72  MockMvcRequestBuilders.post("/xxe/content-type")
```

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 85 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: simpleXXEShouldNotWork
Enclosing Method: simpleXXEShouldNotWork()
File: src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java:85
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 85 (Poor Error Handling: Overly Broad Throws) — Low

```
82  }
83
84  @Test
85  public void simpleXXEShouldNotWork() throws Exception {
86  File targetFile =
87  new File(webGoatHomeDirectory, "/XXE/" + webSession.getUserName() + "/secret.txt");
88  String content =
```

src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java, line 101 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: countComments
Enclosing Method: countComments()
File: src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java:101
Taint Flags:

```
98  .andExpect(jsonPath("$.[*].text").value(Matchers.hasItem("Hello World")));
99  }
100
101  private int countComments() throws Exception {
102  var response =
103  mockMvc
104  .perform(get("/xxe/comments").contentType(MediaType.APPLICATION_JSON))
```

src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java, line 77 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postingXmlCommentWithoutXXEShouldNotSolveAssignment
Enclosing Method: postingXmlCommentWithoutXXEShouldNotSolveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java:77
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java, line 77 (Poor Error Handling: Overly Broad Throws) — Low

```
74  }
75
76  @Test
77  public void postingXmlCommentWithoutXXEShouldNotSolveAssignment() throws Exception {
78  mockMvc
79  .perform(
80  MockMvcRequestBuilders.post("/xxe/simple")
```

src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java, line 52 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: workingAttack
Enclosing Method: workingAttack()
File: src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java:52
Taint Flags:

```
49  }
50
51  @Test
52  public void workingAttack() throws Exception {
53  // Call with XXE injection
54  mockMvc
55  .perform(
```

src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java, line 90 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postingPlainTextShouldThrowException
Enclosing Method: postingPlainTextShouldThrowException()
File: src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java:90
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java, line 90 (Poor Error Handling: Overly Broad Throws) — Low

```
87  }
88
89  @Test
90  public void postingPlainTextShouldThrowException() throws Exception {
91  mockMvc
92  .perform(MockMvcRequestBuilders.post("/xxe/simple").content("test"))
93  .andExpect(status().isOk())
```

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 100 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solve
Enclosing Method: solve()
File: src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java:100
Taint Flags:

```
97  }
98
99  @Test
100  public void solve() throws Exception {
101  File targetFile =
102  new File(webGoatHomeDirectory, "/XXE/" + webSession.getUserName() + "/secret.txt");
103  // Host DTD on WebWolf site
```

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: countComments
Enclosing Method: countComments()
File: src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java:42
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 42 (Poor Error Handling: Overly Broad Throws) — Low

```
39  this.mockMvc = MockMvcBuilders.webAppContextSetup(this.wac).build();
40  }
41
42  private int countComments() throws Exception {
43  var response =
44  mockMvc
45  .perform(get("/xxe/comments").contentType(MediaType.APPLICATION_JSON))
```

src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java, line 111 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postingInvalidJsonShouldNotAddComment
Enclosing Method: postingInvalidJsonShouldNotAddComment()
File: src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java:111
Taint Flags:

```
108  }
109
110  @Test
111  public void postingInvalidJsonShouldNotAddComment() throws Exception {
112  var numberOfComments = countComments();
113  mockMvc
114  .perform(
```

src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: sendingXmlButContentTypeIsJson
Enclosing Method: sendingXmlButContentTypeIsJson()
File: src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java:53
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java, line 53 (Poor Error Handling: Overly Broad Throws) — Low

```
50  }
51
52  @Test
53  public void sendingXmlButContentTypeIsJson() throws Exception {
54  mockMvc
55  .perform(
56  MockMvcRequestBuilders.post("/xxe/content-type")
```

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 72 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: wrongXmlShouldGiveErrorBack
Enclosing Method: wrongXmlShouldGiveErrorBack()
File: src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java:72
Taint Flags:

```
69  }
70
71  @Test
72  public void wrongXmlShouldGiveErrorBack() throws Exception {
73  mockMvc
74  .perform(
75  MockMvcRequestBuilders.post("/xxe/blind")
```

src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java, line 66 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postingJsonCommentShouldNotSolveAssignment
Enclosing Method: postingJsonCommentShouldNotSolveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java:66
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/SimpleXXETest.java, line 66 (Poor Error Handling: Overly Broad Throws) — Low

```
63  }
64
65  @Test
66  public void postingJsonCommentShouldNotSolveAssignment() throws Exception {
67  mockMvc
68  .perform(
69  MockMvcRequestBuilders.post("/xxe/simple")
```

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 165 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: performXXE
Enclosing Method: performXXE()
File: src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java:165
Taint Flags:

```
162  performXXE(xml);
163  }
164
165  private void performXXE(String xml) throws Exception {
166  // Call with XXE injection
167  mockMvc
168  .perform(MockMvcRequestBuilders.post("/xxe/blind").content(xml))
```

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 133 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: solveOnlyParamReferenceEntityInExternalDTD
Enclosing Method: solveOnlyParamReferenceEntityInExternalDTD()
File: src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java:133
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 133 (Poor Error Handling: Overly Broad Throws) — Low

```
130  }
131
132  @Test
133  public void solveOnlyParamReferenceEntityInExternalDTD() throws Exception {
134  File targetFile =
135  new File(webGoatHomeDirectory, "/XXE/" + webSession.getUserName() + "/secret.txt");
136  // Host DTD on WebWolf site
```

src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java, line 83 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: postingJsonShouldAddComment
Enclosing Method: postingJsonShouldAddComment()
File: src/test/java/org/owasp/webgoat/lessons/xxe/ContentTypeAssignmentTest.java:83
Taint Flags:

```
80  }
81
82  @Test
83  public void postingJsonShouldAddComment() throws Exception {
84  mockMvc
85  .perform(
86  MockMvcRequestBuilders.post("/xxe/content-type")
```

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 59 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: validCommentMustBeAdded
Enclosing Method: validCommentMustBeAdded()
File: src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java:59
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.lessons.xxe

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 59 (Poor Error Handling: Overly Broad Throws) — Low

```
56  }
57
58  @Test
59  public void validCommentMustBeAdded() throws Exception {
60  int nrOfComments = countComments();
61  mockMvc
62  .perform(
```

src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java, line 51 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: containsComment
Enclosing Method: containsComment()
File: src/test/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignmentTest.java:51
Taint Flags:

```
48  return new
ObjectMapper().reader().readTree(response.getResponse().getContentAsString()).size();
49  }
50
51  private void containsComment(String expected) throws Exception {
52  mockMvc
53  .perform(get("/xxe/comments").contentType(MediaType.APPLICATION_JSON))
54  .andExpect(status().isOk())
```

Package: org.owasp.webgoat.webwolf

src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java, line 49 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: filterChain
Enclosing Method: filterChain()
File: src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java:49
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.webwolf

src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java, line 49 (Poor Error Handling: Overly Broad Throws) — Low

```
46  private final UserService userDetailsService;
47
48  @Bean
49  public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
50  return http.authorizeHttpRequests(
51  auth -> {
52  auth.requestMatchers("/css/**", "/webjars/**", "/favicon.ico", "/js/**", "/images/**")
```

src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java, line 86 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: configureGlobal
Enclosing Method: configureGlobal()
File: src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java:86
Taint Flags:

```
83  }
84
85  @Autowired
86  public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
87  auth.userDetailsService(userDetailsService);
88  }
89
```

src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java, line 96 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: authenticationManager
Enclosing Method: authenticationManager()
File: src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java:96
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.webwolf

src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java, line 96 (Poor Error Handling: Overly Broad Throws) — Low

```
93  }
94
95  @Bean
96  public AuthenticationManager authenticationManager(
97  AuthenticationConfiguration authenticationConfiguration) throws Exception {
98  return authenticationConfiguration.getAuthenticationManager();
99  }
```

Package: org.owasp.webgoat.webwolf.mailbox

src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java, line 93 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: userShouldBeAbleToReadOwnEmail
Enclosing Method: userShouldBeAbleToReadOwnEmail()
File: src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java:93
Taint Flags:

```
90
91  @Test
92  @WithMockUser(username = "test1234")
93  public void userShouldBeAbleToReadOwnEmail() throws Exception {
94  Email email =
95  Email.builder()
96  .contents("This is a test mail")
```

src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java, line 119 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: differentUserShouldNotBeAbleToReadOwnEmail
Enclosing Method: differentUserShouldNotBeAbleToReadOwnEmail()
File: src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java:119
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Error Handling: Overly Broad Throws — Low

Package: org.owasp.webgoat.webwolf.mailbox

src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java, line 119 (Poor Error Handling: Overly Broad Throws) — Low

```
116
117  @Test
118  @WithMockUser(username = "test1233")
119  public void differentUserShouldNotBeAbleToReadOwnEmail() throws Exception {
120  Email email =
121  Email.builder()
122  .contents("This is a test mail")
```

src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java, line 73 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: sendingMailShouldStoreIt
Enclosing Method: sendingMailShouldStoreIt()
File: src/test/java/org/owasp/webgoat/webwolf/mailbox/MailboxControllerTest.java:73
Taint Flags:

```
70  }
71
72  @Test
73  public void sendingMailShouldStoreIt() throws Exception {
74  Email email =
75  Email.builder()
76  .contents("This is a test mail")
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Logging Practice: Use of a System Output Stream (9 issues)

Abstract

Using `System.out` or `System.err` rather than a dedicated logging facility makes it difficult to monitor the program behavior.

Explanation

Example 1: The first Java program that a developer learns to write is the following:

```
public class MyClass
  ...
    System.out.println("hello world");
  ...
}
```

While most programmers go on to learn many nuances and subtleties about Java, a surprising number hang on to this first lesson and never give up on writing messages to standard output using `System.out.println()`. The problem is that writing directly to standard output or standard error is often used as an unstructured form of logging. Structured logging facilities provide features like logging levels, uniform formatting, a logger identifier, timestamps, and, perhaps most critically, the ability to direct the log messages to the right place. When the use of system output streams is jumbled together with the code that uses loggers properly, the result is often a well-kept log that is missing critical information. Developers widely accept the need for structured logging, but many continue to use system output streams in their "pre-production" development. If the code you are reviewing is past the initial phases of development, use of `System.out` or `System.err` may indicate an oversight in the move to a structured logging system.

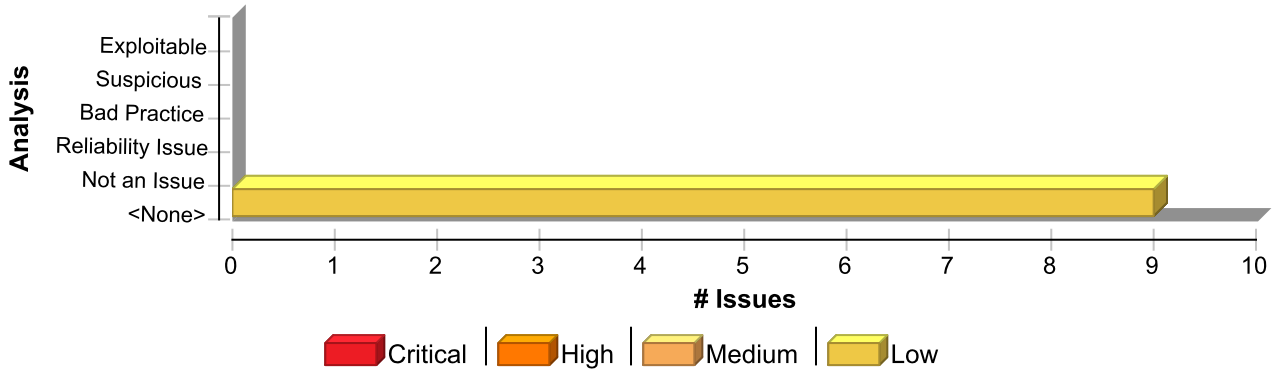
Recommendation

Use a Java logging facility rather than `System.out` or `System.err`. **Example 2:** For example, you can rewrite the "hello world" program in `Example 1` using log4j as follows:

```
import org.apache.log4j.Logger;
import org.apache.log4j.BasicConfigurator;

public class MyClass {
  private final static Logger logger =
            Logger.getLogger(MyClass.class);

  ...
    BasicConfigurator.configure();
    logger.info("hello world");
  ...
}
```

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Logging Practice: Use of a System Output Stream	9	0	0	9
Total	**9**	**0**	**0**	**9**

Poor Logging Practice: Use of a System Output Stream — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 182 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: checkLang()
File: src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java:182
Taint Flags:

```
179
180   for (String key : propsLang.stringPropertyNames()) {
181   if (!propsDefault.containsKey(key)) {
182   System.err.println("key: " + key + " in (" + lang + ") is missing from default
properties");
183   Assertions.fail();
184   }
185   if (!jsonPath
```

src/it/java/org/owasp/webgoat/ProgressRaceConditionIntegrationTest.java, line 53 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Logging Practice: Use of a System Output Stream — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/ProgressRaceConditionIntegrationTest.java, line 53 (Poor Logging Practice: Use of a System Output Stream) — Low

Sink: FunctionCall: println
Enclosing Method: runTests()
File: src/it/java/org/owasp/webgoat/ProgressRaceConditionIntegrationTest.java:53
Taint Flags:

```
50  }
51  })
52  .count();
53  System.err.println("counted status 500: " + countStatusCode500);
54  Assertions.assertThat(countStatusCode500)
55  .isLessThanOrEqualTo((NUMBER_OF_CALLS - (NUMBER_OF_CALLS / NUMBER_OF_PARALLEL_THREADS)));
56  }
```

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 192 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: checkLang()
File: src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java:192
Taint Flags:

```
189  "key: " + key + " in (" + lang + ") has incorrect translation in label service");
190  System.out.println(
191  "actual:" + jsonPath.getString(ESCAPE_JSON_PATH_CHAR + key + ESCAPE_JSON_PATH_CHAR));
192  System.out.println("expected: " + propsLang.getProperty(key));
193  System.out.println();
194  Assertions.fail();
195  }
```

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 190 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: checkLang()
File: src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java:190
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Logging Practice: Use of a System Output Stream — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 190 (Poor Logging Practice: Use of a System Output Stream) — Low

```
187  .equals(propsLang.get(key))) {
188  System.out.println(
189  "key: " + key + " in (" + lang + ") has incorrect translation in label service");
190  System.out.println(
191  "actual:" + jsonPath.getString(ESCAPE_JSON_PATH_CHAR + key + ESCAPE_JSON_PATH_CHAR));
192  System.out.println("expected: " + propsLang.getProperty(key));
193  System.out.println();
```

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 193 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: checkLang()
File: src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java:193
Taint Flags:

```
190  System.out.println(
191  "actual:" + jsonPath.getString(ESCAPE_JSON_PATH_CHAR + key + ESCAPE_JSON_PATH_CHAR));
192  System.out.println("expected: " + propsLang.getProperty(key));
193  System.out.println();
194  Assertions.fail();
195  }
196  }
```

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 188 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: checkLang()
File: src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java:188
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Logging Practice: Use of a System Output Stream — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 188 (Poor Logging Practice: Use of a System Output Stream) — Low

```
185  if (!jsonPath
186  .getString(ESCAPE_JSON_PATH_CHAR + key + ESCAPE_JSON_PATH_CHAR)
187  .equals(propsLang.get(key))) {
188  System.out.println(
189  "key: " + key + " in (" + lang + ") has incorrect translation in label service");
190  System.out.println(
191  "actual:" + jsonPath.getString(ESCAPE_JSON_PATH_CHAR + key + ESCAPE_JSON_PATH_CHAR));
```

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java, line 160 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: log()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java:160
Taint Flags:

```
157  Statement statement = connection.createStatement(TYPE_SCROLL_SENSITIVE, CONCUR_UPDATABLE);
158  statement.executeUpdate(logQuery);
159  } catch (SQLException e) {
160  System.err.println(e.getMessage());
161  }
162  }
163 }
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java, line 102 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: injectableQueryIntegrity()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java:102
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Logging Practice: Use of a System Output Stream — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java, line 102 (Poor Logging Practice: Use of a System Output Stream) — Low

```
99  SqlInjectionLesson8.generateTable(this.getEmployeesDataOrderBySalaryDesc(connection)))
100  .build();
101  } catch (SQLException e) {
102  System.err.println(e.getMessage());
103  return failed(this)
104  .output("<br><span class='feedback-negative'>" + e.getMessage() + "</span>")
105  .build();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java, line 123 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: tableExists()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java:123
Taint Flags:

```
120  if (errorMsg.contains("object not found: ACCESS_LOG")) {
121  return false;
122  } else {
123  System.err.println(e.getMessage());
124  return false;
125  }
126  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Confusing Naming (5 issues)

Abstract

The class contains a field and a method with the same name.

Explanation

It is confusing to have a member field and a method with the same name. It makes it easy for a programmer to accidentally call the method when attempting to access the field or vice versa. **Example 1:**

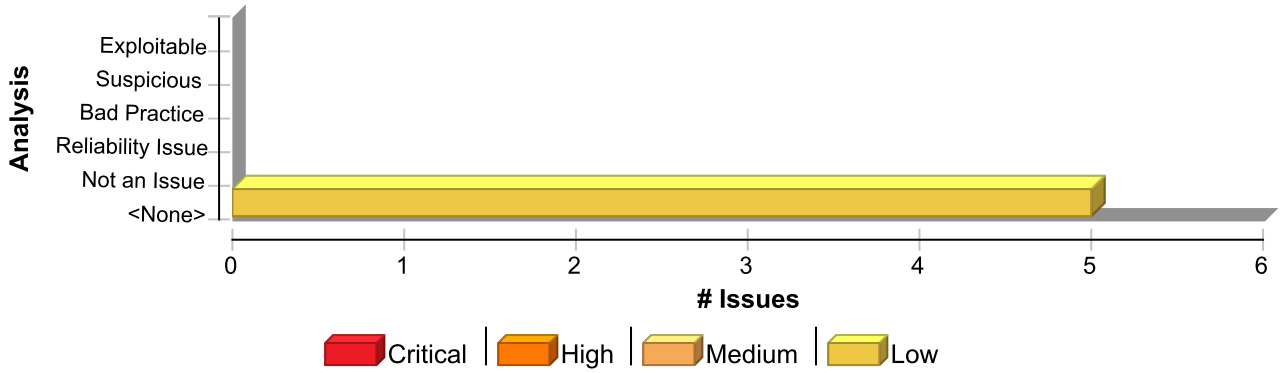
```
public class Totaller {
  private int total;
  public int total() {
    ...
  }
}
```

Recommendation

Rename either the method or the field. If the method returns the field, consider following the standard getter/setter naming convention. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
public class Totaller {
  private int total;
  public int getTotal() {
    ...
  }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Style: Confusing Naming	5	0	0	5
Total	**5**	**0**	**0**	**5**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Confusing Naming — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/IntegrationTest.java, line 36 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: webWolfUrl
File: src/it/java/org/owasp/webgoat/IntegrationTest.java:36
Taint Flags:

```
33  Boolean.valueOf(System.getenv().getOrDefault("WEBGOAT_SSLENABLED", "false"));
34  private static String webgoatUrl =
35  (useSSL ? "https://" : "http://") + webGoatHost + ":" + webGoatPort + webGoatContext;
36  private static String webWolfUrl = "http://" + webWolfHost + ":" + webWolfPort +
webWolfContext;
37  @Getter private String webGoatCookie;
38  @Getter private String webWolfCookie;
39  @Getter private final String user = "webgoat";
```

Package: org.owasp.webgoat.container.assignments

src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java, line 41 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: output
File: src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java:41
Taint Flags:

```
38  private PluginMessages messages;
39  private Object[] feedbackArgs;
40  private String feedbackResourceBundleKey;
41  private String output;
42  private Object[] outputArgs;
43  private AssignmentEndpoint assignment;
44  private boolean attemptWasMade = false;
```

src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java, line 39 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Confusing Naming — Low

Package: org.owasp.webgoat.container.assignments

src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java, line 39 (Poor Style: Confusing Naming) — Low

Sink Details

Sink: Field: feedbackArgs
File: src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java:39
Taint Flags:

```
36
37  private boolean lessonCompleted;
38  private PluginMessages messages;
39  private Object[] feedbackArgs;
40  private String feedbackResourceBundleKey;
41  private String output;
42  private Object[] outputArgs;
```

src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java, line 43 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: assignment
File: src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java:43
Taint Flags:

```
40  private String feedbackResourceBundleKey;
41  private String output;
42  private Object[] outputArgs;
43  private AssignmentEndpoint assignment;
44  private boolean attemptWasMade = false;
45
46  public AttackResultBuilder(PluginMessages messages) {
```

src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java, line 42 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: outputArgs
File: src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java:42
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Confusing Naming	Low
Package: org.owasp.webgoat.container.assignments	
src/main/java/org/owasp/webgoat/container/assignments/AttackResult.java, line 42 (Poor Style: Confusing Naming)	Low

```
39  private Object[] feedbackArgs;
40  private String feedbackResourceBundleKey;
41  private String output;
42  private Object[] outputArgs;
43  private AssignmentEndpoint assignment;
44  private boolean attemptWasMade = false;
45
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Identifier Contains Dollar Symbol ($) (1 issue)

Abstract

Using a dollar sign ($) as part of an identifier is not recommended.

Explanation

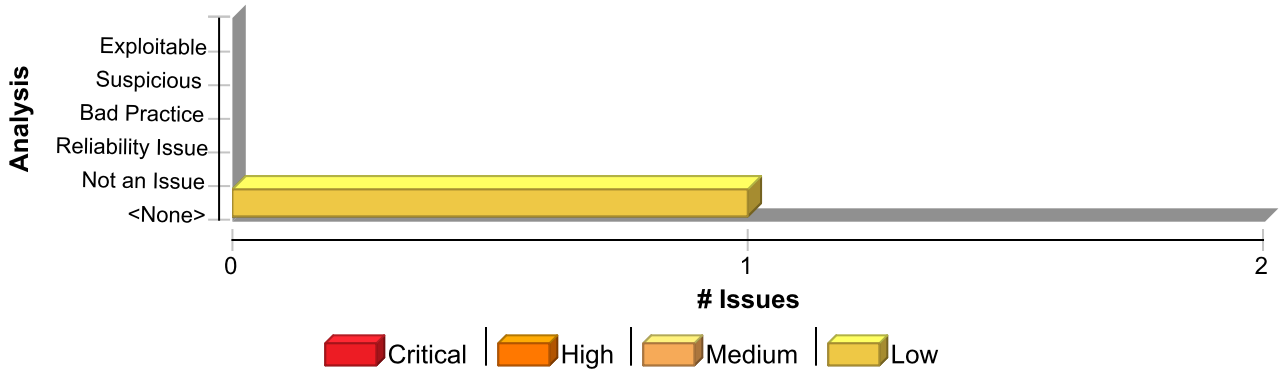
Section 3.8 of the Java Language Specification reserves the dollar sign ($) for identifiers that are used only in mechanically generated source code. **Example:**

```
int un$afe;
```

Recommendation

Rename identifiers that use the dollar sign ($) to names that do not carry an overloaded meaning.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Style: Identifier Contains Dollar Symbol ($)	1	0	0	1
Total	**1**	**0**	**0**	**1**

Poor Style: Identifier Contains Dollar Symbol ($)	Low
Package: .org.owasp.webgoat.lessons.xxe	
src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java, line 54 (Poor Style: Identifier Contains Dollar Symbol ($))	**Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: lambda [java.util.function.Function]
File: src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java:54
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Identifier Contains Dollar Symbol ($)	Low
Package: .org.owasp.webgoat.lessons.xxe	
src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java, line 54 (Poor Style: Identifier Contains Dollar Symbol ($))	Low

```
51
52  static class Comments extends ArrayList<Comment> {
53  void sort() {
54  sort(Comparator.comparing(Comment::getDateTime).reversed());
55  }
56  }
57
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read (21 issues)

Abstract

The variable's value is assigned but never used, making it a dead store.

Explanation

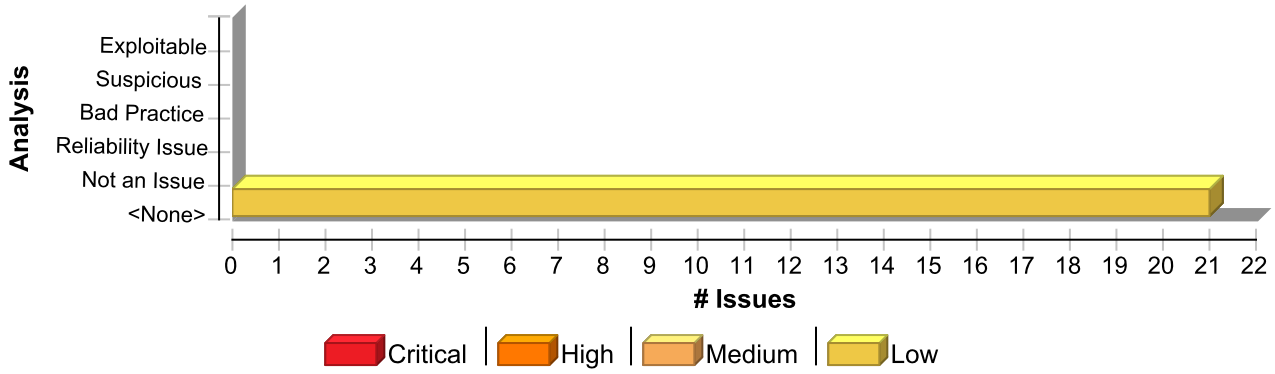
This variable's value is not used. After the assignment, the variable is either assigned another value or goes out of scope. **Example:** The following code excerpt assigns to the variable `r` and then overwrites the value without using it.

```
r = getName();
r = getNewBuffer(buf);
```

Recommendation

Remove unnecessary assignments in order to make the code easier to understand and maintain.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Style: Value Never Read	21	0	0	21
Total	**21**	**0**	**0**	**21**

Poor Style: Value Never Read	**Low**
Package: org.owasp.webgoat	
src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 75 (Poor Style: Value Never Read)	**Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java, line 75 (Poor Style: Value Never Read) — Low

Sink: VariableAccess: jwt
Enclosing Method: getSecretToken()
File: src/it/java/org/owasp/webgoat/JWTLessonIntegrationTest.java:75
Taint Flags:

```
72  private String getSecretToken(String token) {
73  for (String key : JWTSecretKeyEndpoint.SECRETS) {
74  try {
75  Jwt jwt = Jwts.parser().setSigningKey(TextCodec.BASE64.encode(key)).parse(token);
76  } catch (JwtException e) {
77  continue;
78  }
```

src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java, line 253 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: assignments
Enclosing Method: checkAssignment8()
File: src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java:253
Taint Flags:

```
250  login();
251  startLesson("CSRF", false);
252
253  Overview[] assignments =
254  RestAssured.given()
255  .cookie("JSESSIONID", getWebGoatCookie())
256  .relaxedHTTPSValidation()
```

Package: org.owasp.webgoat.container

src/main/java/org/owasp/webgoat/container/WebWolfRedirect.java, line 17 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.container

src/main/java/org/owasp/webgoat/container/WebWolfRedirect.java, line 17 (Poor Style: Value Never Read) — Low

Sink: VariableAccess: url
Enclosing Method: openWebWolf()
File: src/main/java/org/owasp/webgoat/container/WebWolfRedirect.java:17
Taint Flags:

```
14
15  @GetMapping("/WebWolf")
16  public ModelAndView openWebWolf() {
17  var url = applicationContext.getEnvironment().getProperty("webwolf.url");
18
19  return new ModelAndView("redirect:" + url + "/home");
20  }
```

Package: org.owasp.webgoat.container.i18n

src/main/java/org/owasp/webgoat/container/i18n/PluginMessages.java, line 56 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: lastModified
Enclosing Method: refreshProperties()
File: src/main/java/org/owasp/webgoat/container/i18n/PluginMessages.java:56
Taint Flags:

```
53  @Override
54  protected PropertiesHolder refreshProperties(String filename, PropertiesHolder propHolder)
{
55  Properties properties = new Properties();
56  long lastModified = System.currentTimeMillis();
57
58  try {
59  var resources =
```

Package: org.owasp.webgoat.container.service

src/main/java/org/owasp/webgoat/container/service/LabelDebugService.java, line 67 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.container.service

src/main/java/org/owasp/webgoat/container/service/LabelDebugService.java, line 67 (Poor Style: Value Never Read) — Low

Sink: VariableAccess: result
Enclosing Method: checkDebuggingStatus()
File: src/main/java/org/owasp/webgoat/container/service/LabelDebugService.java:67
Taint Flags:

```
64  @RequestMapping(path = URL_DEBUG_LABELS_MVC, produces = MediaType.APPLICATION_JSON_VALUE)
65  public @ResponseBody ResponseEntity<Map<String, Object>> checkDebuggingStatus() {
66  log.debug("Checking label debugging, it is {}", labelDebugger.isEnabled());
67  Map<String, Object> result = createResponse(labelDebugger.isEnabled());
68  return new ResponseEntity<>(result, HttpStatus.OK);
69  }
70
```

src/main/java/org/owasp/webgoat/container/service/LabelDebugService.java, line 84 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: result
Enclosing Method: setDebuggingStatus()
File: src/main/java/org/owasp/webgoat/container/service/LabelDebugService.java:84
Taint Flags:

```
81  public @ResponseBody ResponseEntity<Map<String, Object>> setDebuggingStatus(
82  @RequestParam("enabled") Boolean enabled) {
83  log.debug("Setting label debugging to {} ", labelDebugger.isEnabled());
84  Map<String, Object> result = createResponse(enabled);
85  labelDebugger.setEnabled(enabled);
86  return new ResponseEntity<>(result, HttpStatus.OK);
87  }
```

Package: org.owasp.webgoat.container.users

src/test/java/org/owasp/webgoat/container/users/UserServiceTest.java, line 28 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.container.users

src/test/java/org/owasp/webgoat/container/users/UserServiceTest.java, line 28 (Poor Style: Value Never Read) — Low

Sink: VariableAccess: userService
Enclosing Method: shouldThrowExceptionWhenUserIsNotFound()
File: src/test/java/org/owasp/webgoat/container/users/UserServiceTest.java:28
Taint Flags:

```
25  @Test
26  void shouldThrowExceptionWhenUserIsNotFound() {
27  when(userRepository.findByUsername(any())).thenReturn(null);
28  UserService userService =
29  new UserService(
30  userRepository, userTrackerRepository, jdbcTemplate, flywayLessons, List.of());
31  Assertions.assertThatThrownBy(() -> userService.loadUserByUsername("unknown"))
```

Package: org.owasp.webgoat.lessons.authbypass

src/main/java/org/owasp/webgoat/lessons/authbypass/AccountVerificationHelper.java, line 54 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: likely
Enclosing Method: didUserLikelylCheat()
File: src/main/java/org/owasp/webgoat/lessons/authbypass/AccountVerificationHelper.java:54
Taint Flags:

```
51  boolean likely = false;
52
53  if (submittedAnswers.size() == secQuestionStore.get(verifyUserId).size()) {
54  likely = true;
55  }
56
57  if ((submittedAnswers.containsKey("secQuestion0")
```

Package: org.owasp.webgoat.lessons.challenges.challenge8

src/main/java/org/owasp/webgoat/lessons/challenges/challenge8/Assignment8.java, line 41 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.lessons.challenges.challenge8

src/main/java/org/owasp/webgoat/lessons/challenges/challenge8/Assignment8.java, line 41 (Poor Style: Value Never Read) — Low

Sink: VariableAccess: msg
Enclosing Method: vote()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge8/Assignment8.java:41
Taint Flags:

```
38  public ResponseEntity<?> vote(
39  @PathVariable(value = "stars") int nrOfStars, HttpServletRequest request) {
40  // Simple implementation of VERB Based Authentication
41  String msg = "";
42  if (request.getMethod().equals("GET")) {
43  var json =
44  Map.of("error", true, "message", "Sorry but you need to login first in order to vote");
```

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 245 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: token
Enclosing Method: unknownUserShouldSeeGuestView()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:245
Taint Flags:

```
242  Claims claims = Jwts.claims();
243  claims.put("admin", "true");
244  claims.put("user", "Intruder");
245  String token =
246  Jwts.builder()
247  .signWith(io.jsonwebtoken.SignatureAlgorithm.HS512, JWT_PASSWORD)
248  .setClaims(claims)
```

src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java, line 125 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.lessons.jwt

src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java, line 125 (Poor Style: Value Never Read) — Low

Sink: VariableAccess: token
Enclosing Method: login()
File: src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java:125
Taint Flags:

```
122  Claims claims =
Jwts.claims().setIssuedAt(Date.from(Instant.now().plus(Duration.ofDays(10))));
123  claims.put("admin", "false");
124  claims.put("user", user);
125  String token =
126  Jwts.builder()
127  .setClaims(claims)
128  .signWith(io.jsonwebtoken.SignatureAlgorithm.HS512, JWT_PASSWORD)
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 227 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: token
Enclosing Method: unknownUserWithValidTokenShouldNotBeAbleToVote()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:227
Taint Flags:

```
224  Claims claims = Jwts.claims();
225  claims.put("admin", "true");
226  claims.put("user", "Intruder");
227  String token =
228  Jwts.builder()
229  .signWith(io.jsonwebtoken.SignatureAlgorithm.HS512, JWT_PASSWORD)
230  .setClaims(claims)
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 81 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: token
Enclosing Method: solveAssignmentWithBoolean()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:81
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 81 (Poor Style: Value Never Read) — Low

```
78  Claims claims = Jwts.claims();
79  claims.put("admin", true);
80  claims.put("user", "Tom");
81  String token = Jwts.builder().setClaims(claims).setHeaderParam("alg", "none").compact();
82
83  // Call the reset endpoint
84  mockMvc
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java, line 63 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: token
Enclosing Method: solveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpointTest.java:63
Taint Flags:

```
60  Claims claims = Jwts.claims();
61  claims.put("admin", "true");
62  claims.put("user", "Tom");
63  String token = Jwts.builder().setClaims(claims).setHeaderParam("alg", "none").compact();
64
65  // Call the reset endpoint
66  mockMvc
```

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 69 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: accessToken
Enclosing Method: solveAssignment()
File: src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java:69
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.lessons.jwt

src/test/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpointTest.java, line 69 (Poor Style: Value Never Read) — Low

```
66  .andReturn();
67  Map<String, String> tokens =
68  objectMapper.readValue(result.getResponse().getContentAsString(), Map.class);
69  String accessToken = tokens.get("access_token");
70  String refreshToken = tokens.get("refresh_token");
71
72  // Now create a new refresh token for Tom based on Toms old access token and send the
refresh
```

src/test/java/org/owasp/webgoat/lessons/jwt/TokenTest.java, line 56 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: jwt
Enclosing Method: test()
File: src/test/java/org/owasp/webgoat/lessons/jwt/TokenTest.java:56
Taint Flags:

```
53  .compact();
54  log.debug(token);
55  Jwt jwt = Jwts.parser().setSigningKey("qwertyqwerty1234").parse(token);
56  jwt =
57  Jwts.parser()
58  .setSigningKeyResolver(
59  new SigningKeyResolverAdapter() {
```

Package: org.owasp.webgoat.lessons.spoofcookie

src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java, line 94 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: newCookieValue
Enclosing Method: credentialsLoginFlow()
File: src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java:94
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.lessons.spoofcookie

src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java, line 94 (Poor Style: Value Never Read) — Low

```
91
92  String authPassword = users.getOrDefault(lowerCasedUsername, "");
93  if (!authPassword.isBlank() && authPassword.equals(password)) {
94  String newCookieValue = EncDec.encode(lowerCasedUsername);
95  Cookie newCookie = new Cookie(COOKIE_NAME, newCookieValue);
96  newCookie.setPath("/WebGoat");
97  newCookie.setSecure(true);
```

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java, line 114 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: result
Enclosing Method: compileFromString()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java:114
Taint Flags:

```
111  Iterable fileObjects = Arrays.asList(javaObjectFromString);
112  JavaCompiler.CompilationTask task =
113  compiler.getTask(null, fileManager, diagnosticsCollector, null, null, fileObjects);
114  Boolean result = task.call();
115  List<Diagnostic> diagnostics = diagnosticsCollector.getDiagnostics();
116  return diagnostics;
117  }
```

Package: org.owasp.webgoat.lessons.webwolfintroduction

src/main/java/org/owasp/webgoat/lessons/webwolfintroduction/LandingAssignment.java, line 59 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: uri
Enclosing Method: openPasswordReset()
File: src/main/java/org/owasp/webgoat/lessons/webwolfintroduction/LandingAssignment.java:59
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read — Low

Package: org.owasp.webgoat.lessons.webwolfintroduction

src/main/java/org/owasp/webgoat/lessons/webwolfintroduction/LandingAssignment.java, line 59 (Poor Style: Value Never Read) — Low

```
56
57  @GetMapping("/WebWolf/landing/password-reset")
58  public ModelAndView openPasswordReset(HttpServletRequest request) throws
URISyntaxException {
59  URI uri = new URI(request.getRequestURL().toString());
60  ModelAndView modelAndView = new ModelAndView();
61  modelAndView.addObject(
62  "webwolfLandingPageUrl", landingPageUrl.replace("//landing", "/landing"));
```

Package: org.owasp.webgoat.lessons.xss

src/test/java/org/owasp/webgoat/lessons/xss/DOMCrossSiteScriptingTest.java, line 49 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: xss
Enclosing Method: setup()
File: src/test/java/org/owasp/webgoat/lessons/xss/DOMCrossSiteScriptingTest.java:49
Taint Flags:

```
46  DOMCrossSiteScripting domXss = new DOMCrossSiteScripting();
47  init(domXss);
48  this.mockMvc = standaloneSetup(domXss).build();
49  CrossSiteScripting xss = new CrossSiteScripting();
50  lenient().when(userSessionData.getValue("randValue")).thenReturn(randVal);
51  }
52
```

Package: org.owasp.webgoat.webwolf.requests

src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java, line 66 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: username
Enclosing Method: get()
File: src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java:66
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Poor Style: Value Never Read	Low
Package: org.owasp.webgoat.webwolf.requests	
src/main/java/org/owasp/webgoat/webwolf/requests/Requests.java, line 66 (Poor Style: Value Never Read)	Low

```
63  @GetMapping
64  public ModelAndView get(Authentication authentication) {
65  var model = new ModelAndView("requests");
66  String username = (null != authentication) ? authentication.getName() : "anonymous";
67  var traces =
68  traceRepository.findAll().stream()
69  .filter(t -> allowedTrace(t, username))
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Portability Flaw: Locale Dependent Comparison (6 issues)

Abstract

Unexpected portability problems can be found when the locale is not specified.

Explanation

When comparing data that may be locale-dependent, an appropriate locale should be specified. **Example 1:** The following example tries to perform validation to determine if user input includes a tag.

```
  ...
  public String tagProcessor(String tag){
    if (tag.toUpperCase().equals("SCRIPT")){
      return null;
    }
    //does not contain SCRIPT tag, keep processing input
    ...
  }
  ...
```

The problem with `Example 1` is that `java.lang.String.toUpperCase()` when used without a locale uses the rules of the default locale. Using the Turkish locale `"title".toUpperCase()` returns "T\u0130TLE", where "\u0130" is the "LATIN CAPITAL LETTER I WITH DOT ABOVE" character. This can lead to unexpected results, such as in `Example 1` where this will prevent the word "script" from being caught by this validation, potentially leading to a Cross-Site Scripting vulnerability.

Recommendation

To prevent this from occurring, always make sure to either specify the default locale, or specify the locale with APIs that accept them such as `toUpperCase()`. **Example 2:** The following specifies the locale manually as an argument to `toUpperCase()`.

```
import java.util.Locale;
  ...
  public String tagProcessor(String tag){
    if (tag.toUpperCase(Locale.ENGLISH).equals("SCRIPT")){
      return null;
    }
    //does not contain SCRIPT tag, keep processing input
    ...
  }
  ...
```

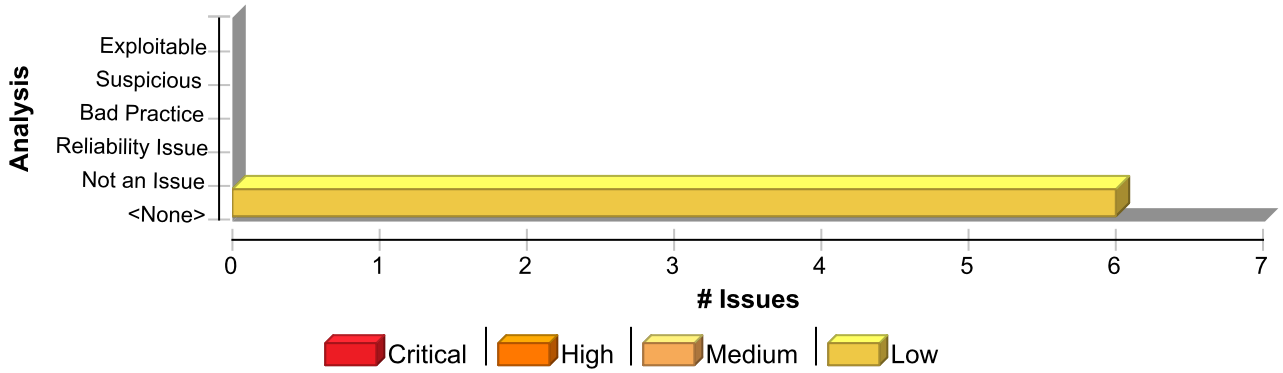
Example 3: The following uses the function `java.lang.String.equalsIgnoreCase()` API to prevent this issue.

```
  ...
  public String tagProcessor(String tag){
    if (tag.equalsIgnoreCase("SCRIPT")){
      return null;
    }
    //does not contain SCRIPT tag, keep processing input
    ...
  }
  ...
```

This prevents the problem because `equalsIgnoreCase()` changes case similar to `Character.toLowerCase()` and `Character.toUpperCase()`. This involves creating temporary canonical forms of both strings using information from the `UnicodeData` file that is part of the Unicode Character Database maintained by the Unicode Consortium, and even though this may render them unreadable if they were to be read out, it makes comparison possible without being dependent upon locale.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Portability Flaw: Locale Dependent Comparison	6	0	0	6
Total	**6**	**0**	**0**	**6**

Portability Flaw: Locale Dependent Comparison — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 106 (Portability Flaw: Locale Dependent Comparison) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: equals(getHash(secret, "SHA-256")) : Comparison without checking locale
Enclosing Method: checkAssignment4()
File: src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java:106
Taint Flags:

```
103  if (md5Hash.equals(HashingAssignment.getHash(secret, "MD5"))) {
104  answer_1 = secret;
105  }
106  if (sha256Hash.equals(HashingAssignment.getHash(secret, "SHA-256"))) {
107  answer_2 = secret;
108  }
109  }
```

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 103 (Portability Flaw: Locale Dependent Comparison) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Portability Flaw: Locale Dependent Comparison — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 103 (Portability Flaw: Locale Dependent Comparison) — Low

Sink: equals(getHash(secret, "MD5")) : Comparison without checking locale
Enclosing Method: checkAssignment4()
File: src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java:103
Taint Flags:

```
100  String answer_1 = "unknown";
101  String answer_2 = "unknown";
102  for (String secret : HashingAssignment.SECRETS) {
103  if (md5Hash.equals(HashingAssignment.getHash(secret, "MD5"))) {
104  answer_1 = secret;
105  }
106  if (sha256Hash.equals(HashingAssignment.getHash(secret, "SHA-256"))) {
```

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 126 (Portability Flaw: Locale Dependent Comparison) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: equals(modulus.toUpperCase()) : Comparison without checking locale
Enclosing Method: verifyAssignment()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:126
Taint Flags:

```
123  result =
124  result
125  && (DatatypeConverter.printHexBinary(rsaPubKey.getModulus().toByteArray())
126  .equals(modulus.toUpperCase()));
127  }
128  return result;
129  }
```

src/main/java/org/owasp/webgoat/lessons/cryptography/SigningAssignment.java, line 80 (Portability Flaw: Locale Dependent Comparison) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Portability Flaw: Locale Dependent Comparison — Low

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/SigningAssignment.java, line 80 (Portability Flaw: Locale Dependent Comparison) — Low

Sink: equals(tempModulus.toUpperCase()) : Comparison without checking locale
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/SigningAssignment.java:80
Taint Flags:

```
77  tempModulus = "00".concat(tempModulus);
78  }
79  if (!DatatypeConverter.printHexBinary(rsaPubKey.getModulus().toByteArray())
80  .equals(tempModulus.toUpperCase())) {
81  log.warn("modulus {} incorrect", modulus);
82  return failed(this).feedback("crypto-signing.modulusnotok").build();
83  }
```

Package: org.owasp.webgoat.lessons.spoofcookie

src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java, line 87 (Portability Flaw: Locale Dependent Comparison) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: equals(lowerCasedUsername) : Comparison without checking locale
Enclosing Method: credentialsLoginFlow()
File: src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java:87
Taint Flags:

```
84  private AttackResult credentialsLoginFlow(
85  String username, String password, HttpServletResponse response) {
86  String lowerCasedUsername = username.toLowerCase();
87  if (ATTACK_USERNAME.equals(lowerCasedUsername)
88  && users.get(lowerCasedUsername).equals(password)) {
89  return informationMessage(this).feedback("spoofcookie.cheating").build();
90  }
```

src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java, line 117 (Portability Flaw: Locale Dependent Comparison) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Portability Flaw: Locale Dependent Comparison	Low
Package: org.owasp.webgoat.lessons.spoofcookie	
src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java, line 117 (Portability Flaw: Locale Dependent Comparison)	**Low**

Sink: cookieUsername.equals(...) : Comparison without checking locale
Enclosing Method: cookieLoginFlow()
File: src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java:117
Taint Flags:

```
114  return failed(this).output(e.getMessage()).build();
115  }
116  if (users.containsKey(cookieUsername)) {
117  if (cookieUsername.equals(ATTACK_USERNAME)) {
118  return success(this).build();
119  }
120  return failed(this)
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Privacy Violation (2 issues)

Abstract

Mishandling private information, such as customer passwords or social security numbers, can compromise user privacy and is often illegal.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Explanation

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Privacy violations occur when: 1. Private user information enters the program. 2. The data is written to an external location, such as the console, file system, or network. **Example 1:** The following code contains a logging statement that tracks the records added to a database by storing the contents in a log file.

```
pass = getPassword();
...
dbmsLog.println(id+":"+pass+":"+type+":"+tstamp);
```

The code in `Example 1` logs a plain text password to the file system. Although many developers trust the file system as a safe storage location for data, it should not be trusted implicitly, particularly when privacy is a concern. Privacy is one of the biggest concerns in the mobile world for a couple of reasons. One of them is a much higher chance of device loss. The other has to do with inter-process communication between mobile applications. With mobile platforms, applications are downloaded from various sources and are run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which is why application authors need to be careful about what information they include in messages addressed to other applications running on the device. Sensitive information should never be part of inter-process communication between mobile applications. **Example 2:** The following code reads username and password for a given site from an Android WebView store and broadcasts them to all the registered receivers.

```
...
webview.setWebViewClient(new WebViewClient() {
  public void onReceivedHttpAuthRequest(WebView view,
        HttpAuthHandler handler, String host, String realm) {
    String[] credentials = view.getHttpAuthUsernamePassword(host, realm);
    String username = credentials[0];
    String password = credentials[1];
    Intent i = new Intent();
    i.setAction("SEND_CREDENTIALS");
    i.putExtra("username", username);
    i.putExtra("password", password);
    view.getContext().sendBroadcast(i);
  }
});
...
```

This example demonstrates several problems. First of all, by default, WebView credentials are stored in plain text and are not hashed. If a user has a rooted device (or uses an emulator), they can read stored passwords for given sites. Second, plain text credentials are broadcast to all the registered receivers, which means that any receiver registered to listen to intents with the `SEND_CREDENTIALS` action will receive the message. The broadcast is not even protected with a permission to limit the number of recipients, although in this case we do not recommend using permissions as a fix. Private data can enter a program in a variety of ways: - Directly from the user in the form of a password or personal information - Accessed from a database or other data store by the application - Indirectly from a partner or other third party Typically, in the context of the mobile environment, this private information includes (along with passwords, SSNs, and other general personal information): - Location - Cell phone number - Serial numbers and device IDs - Network Operator information - Voicemail information Sometimes data that is not labeled as private can have a privacy implication in a different context. For example, student identification numbers are usually not considered private because there is no explicit and publicly-available mapping to an individual student's personal information. However, if a school generates identification numbers based on student social security numbers, then the identification numbers should be considered private. Security and privacy concerns often seem to compete with each other. From a security perspective, you should record all important operations so that any anomalous activity can later be identified. However, when private data is involved, this practice can create risk. Although there are many ways in which private data can be handled unsafely, a common risk stems from misplaced trust. Programmers often trust the operating environment in which a program runs, and therefore believe that it is acceptable to store private information on the file system, in the registry, or in other locally-controlled resources. However, even if access to certain resources is restricted, this does not guarantee that the individuals who do have access can be trusted. For example, in 2004, an unscrupulous employee at AOL sold approximately 92 million private customer email addresses to a spammer marketing an offshore gambling web site [1]. In response to such high-profile

© Copyright [2008-2024] Micro Focus or one of its affiliates.

exploits, the collection and management of private data is becoming increasingly regulated. Depending on its location, the type of business it conducts, and the nature of any private data it handles, an organization may be required to comply with one or more of the following federal and state regulations: - Safe Harbor Privacy Framework [3] - Gramm-Leach Bliley Act (GLBA) [4] - Health Insurance Portability and Accountability Act (HIPAA) [5] - California SB-1386 [6] Despite these regulations, privacy violations continue to occur with alarming frequency.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

When security and privacy demands clash, privacy should usually be given the higher priority. To accomplish this and still maintain required security information, cleanse any private information before it exits the program. To enforce good privacy management, develop and strictly adhere to internal privacy guidelines. The guidelines should specifically describe how an application should handle private data. If your organization is regulated by federal or state law, ensure that your privacy guidelines are sufficiently strenuous to meet the legal requirements. Even if your organization is not regulated, you must protect private information or risk losing customer confidence. The best policy with respect to private data is to minimize its exposure. Applications, processes, and employees should not be granted access to any private data unless the access is required for the tasks that they are to perform. Just as the principle of least privilege dictates that no operation should be performed with more than the necessary privileges, access to private data should be restricted to the smallest possible group. For mobile applications, make sure they never communicate any sensitive data to other applications running on the device. When private data needs to be stored, it should always be encrypted. For Android, as well as any other platform that uses SQLite database, SQLCipher is a good alternative. SQLCipher is an extension to the SQLite database that provides transparent 256-bit AES encryption of database files. Thus, credentials can be stored in an encrypted database. **Example 3:** The following code demonstrates how to integrate SQLCipher into an Android application after downloading the necessary binaries, and store credentials into the database file.

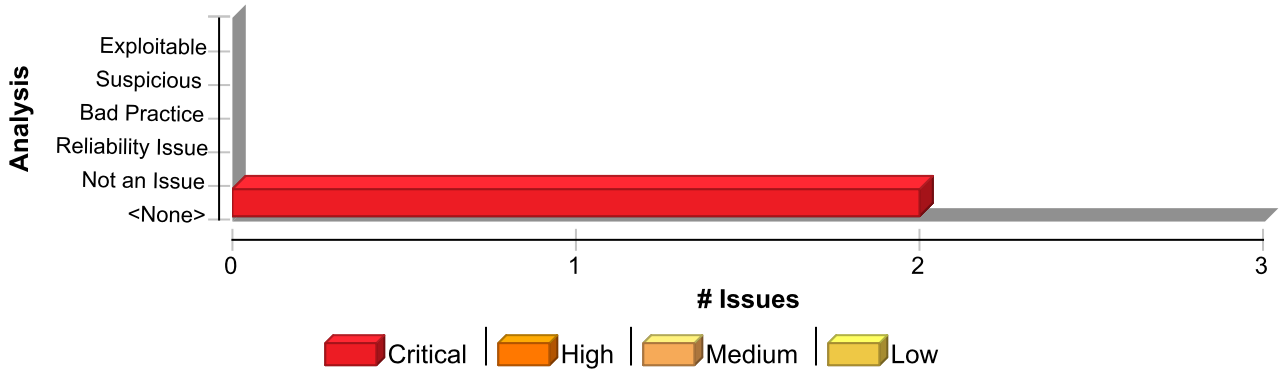
```
import net.sqlcipher.database.SQLiteDatabase;
...
  SQLiteDatabase.loadLibs(this);
  File dbFile = getDatabasePath("credentials.db");
  dbFile.mkdirs();
  dbFile.delete();
  SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(dbFile,
"credentials", null);
  db.execSQL("create table credentials(u, p)");
  db.execSQL("insert into credentials(u, p) values(?, ?)", new Object[]
{username, password});
...
```

Note that references to `android.database.sqlite.SQLiteDatabase` are substituted with those of `net.sqlcipher.database.SQLiteDatabase`. To enable encryption on the WebView store, you must recompile WebKit with the `sqlcipher.so` library. **Example 4:** The following code reads username and password for a given site from an Android WebView store and instead of broadcasting them to all the registered receivers, it only broadcasts internally so that the broadcast is only seen by other parts of the same application.

```
...
webview.setWebViewClient(new WebViewClient() {
  public void onReceivedHttpAuthRequest(WebView view,
        HttpAuthHandler handler, String host, String realm) {
    String[] credentials = view.getHttpAuthUsernamePassword(host, realm);
    String username = credentials[0];
    String password = credentials[1];
    Intent i = new Intent();
    i.setAction("SEND_CREDENTIALS");
    i.putExtra("username", username);
    i.putExtra("password", password);
    LocalBroadcastManager.getInstance(view.getContext()).sendBroadcast(i);
  }
});
...
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Privacy Violation	2	0	0	2
Total	**2**	**0**	**0**	**2**

Privacy Violation — Critical

Package: org.owasp.webgoat.lessons.logging

src/main/java/org/owasp/webgoat/lessons/logging/LogBleedingTask.java, line 50 (Privacy Violation) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Source Details

Source: Read this.password
From: org.owasp.webgoat.lessons.logging.LogBleedingTask.generatePassword
File: src/main/java/org/owasp/webgoat/lessons/logging/LogBleedingTask.java:50

```
47  password = UUID.randomUUID().toString();
48  log.info(
49  "Password for admin: {}",
50
Base64.getEncoder().encodeToString(password.getBytes(StandardCharsets.UTF_8)));
51  }
52
53  @PostMapping("/LogSpoofing/log-bleeding")
```

Sink Details

Sink: org.slf4j.Logger.info()
Enclosing Method: generatePassword()
File: src/main/java/org/owasp/webgoat/lessons/logging/LogBleedingTask.java:50
Taint Flags: BASE64_ENCODED, NO_NEW_LINE, PRIVATE

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Privacy Violation — Critical

Package: org.owasp.webgoat.lessons.logging

src/main/java/org/owasp/webgoat/lessons/logging/LogBleedingTask.java, line 50 (Privacy Violation) — Critical

```
47  password = UUID.randomUUID().toString();
48  log.info(
49  "Password for admin: {}",
50  Base64.getEncoder().encodeToString(password.getBytes(StandardCharsets.UTF_8)));
51  }
52
53  @PostMapping("/LogSpoofing/log-bleeding")
```

Package: src.main.resources.lessons.clientsidefiltering.js

src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js, line 38 (Privacy Violation) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Source Details

Source: Read SSN
From: lambda
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js:31

```
28   html = html + '<td>' + result[i].UserID + '</td>';
29   html = html + '<td>' + result[i].FirstName + '</td>';
30   html = html + '<td>' + result[i].LastName + '</td>';
31   html = html + '<td>' + result[i].SSN + '</td>';
32   html = html + '<td>' + result[i].Salary + '</td>';
33   html = html + '</tr>';
34   }
```

Sink Details

Sink: Assignment to newdiv.innerHTML
Enclosing Method: lambda()
File: src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js:38
Taint Flags: PRIVATE

```
35  html = html + '</tr></table>';
36
37  var newdiv = document.createElement("div");
38  newdiv.innerHTML = html;
39  var container = document.getElementById("hiddenEmployeeRecords");
40  container.appendChild(newdiv);
41  });
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Privacy Violation: Autocomplete (2 issues)

Abstract

Autocompletion of forms allows some browsers to retain sensitive information in their history.

Explanation

With autocompletion enabled, some browsers retain user input across sessions, which could allow someone using the computer after the initial user to see information previously submitted.

Recommendation

Explicitly disable autocompletion on forms or sensitive inputs. By disabling autocompletion, information previously entered will not be presented back to the user as they type. It will also disable the "remember my password" functionality of most major browsers. **Example 1:** In an HTML form, disable autocompletion for all input fields by explicitly setting the value of the `autocomplete` attribute to `off` on the `form` tag.

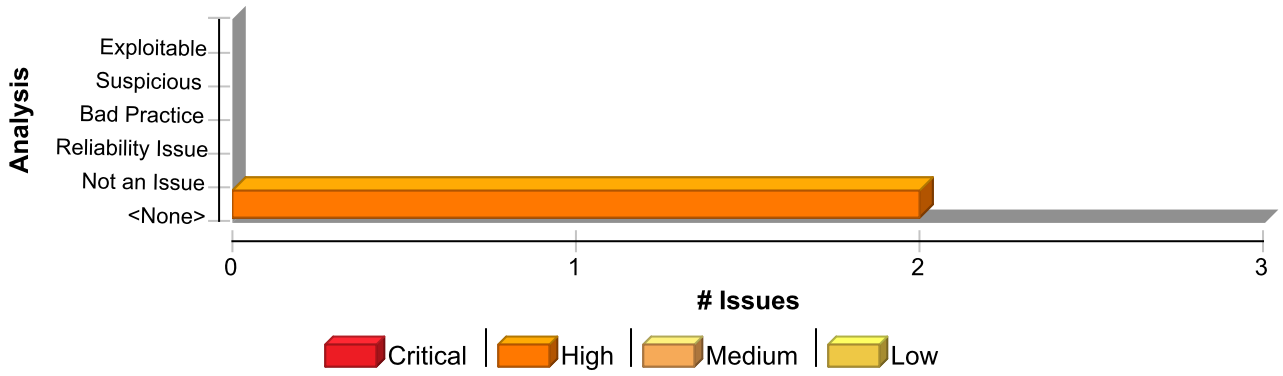
```
<form method="post" autocomplete="off">
      Address: <input name="address" />
      Password: <input name="password" type="password" />
</form>
```

Example 2: Alternatively, disable autocompletion for specific input fields by explicitly setting the value of the `autocomplete` attribute to `off` on the corresponding tags.

```
<form method="post">
      Address: <input name="address" />
      Password: <input name="password" type="password" autocomplete="off"/>
</form>
```

Note that the default value of the `autocomplete` attributed is on. Therefore do not omit the attribute when dealing with sensitive inputs.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Privacy Violation: Autocomplete	2	0	0	2
Total	**2**	**0**	**0**	**2**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Privacy Violation: Autocomplete — High

Package: src.main.resources.lessons.hijacksession.templates

src/main/resources/lessons/hijacksession/templates/hijackform.html, line 16 (Privacy Violation: Autocomplete) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Content)

Sink Details

File: src/main/resources/lessons/hijacksession/templates/hijackform.html:16
Taint Flags:

```
13  </div>
14  <div class="form-group input-group">
15  <span class="input-group-addon"><i
16  class="glyphicon glyphicon-lock"></i></span> <input class="form-control"
17  placeholder="Password" name="password" type="password" />
18  </div>
19  <button type="submit" class="btn btn-primary btn-block">Access</button>
```

Package: src.main.resources.lessons.spoofcookie.templates

src/main/resources/lessons/spoofcookie/templates/spoofcookieform.html, line 16 (Privacy Violation: Autocomplete) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Content)

Sink Details

File: src/main/resources/lessons/spoofcookie/templates/spoofcookieform.html:16
Taint Flags:

```
13  </div>
14  <div class="form-group input-group">
15  <span class="input-group-addon"><i
16  class="glyphicon glyphicon-lock"></i></span> <input class="form-control"
17  placeholder="Password" name="password" type="password"
18  id="spoof_password" />
19  </div>
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Race Condition (2 issues)

Abstract

The set callback could lead to a race condition.

Explanation

Node.js allows developers to assign callbacks to IO-blocked events. This allows better performance as the callbacks run asynchronously such that the main application isn't blocked by IO. However, this in turn may lead to race conditions when something outside the callback relies upon code within the callback to be run first. **Example 1:** The following code checks a user against a database for authentication.

```
...
var authenticated = true;
...
database_connect.query('SELECT * FROM users WHERE name == ? AND password = ?
LIMIT 1', userNameFromUser, passwordFromUser, function(err, results){
  if (!err && results.length > 0){
    authenticated = true;
  }else{
    authenticated = false;
  }
});

if (authenticated){
  //do something privileged stuff
  authenticatedActions();
}else{
  sendUnathenticatedMessage();
}
```

In this example we're supposed to be calling to a backend database to confirm a user's credentials for login, and if confirmed we set a variable to `true`, otherwise `false`. Unfortunately, since the callback is blocked by IO, it will run asynchronously and may be run after the check to `if (authenticated)`, and since the default was true, it will go into the if-statement whether the user is actually authenticated or not.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

When creating Node.js applications you must be careful of IO-blocked events and what functionality the related callbacks perform. There may be a series of callbacks that need to be called in a certain order, or code that can only be reached once a certain callback is run. **Example 2:** The following code fixes the race condition in `Example 1`.

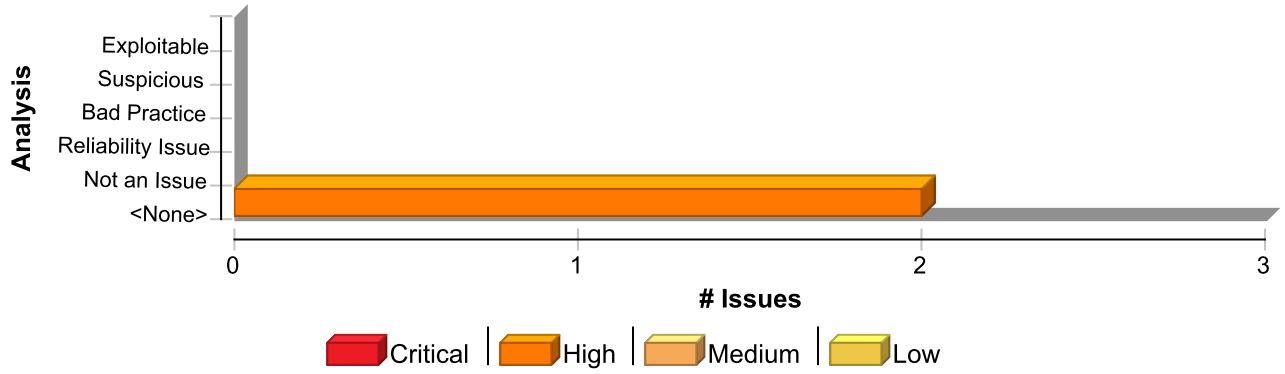
```
...
database_connect.query('SELECT * FROM users WHERE name == ? AND password = ?
LIMIT 1', userNameFromUser, passwordFromUser, function(err, results){
  if (!err && results.length > 0){
    // do privileged stuff
    authenticatedActions();
  }else{
    sendUnauthenticatedMessage();
  }
});
...
```

This is a simple example and real life scenarios may be far more complex and fixing them may require a larger refactoring of the codebase. A simple way to try and avoid these problems is by using APIs that utilize `promises`, as they represent the eventual outcome of asynchronous operations, and allow you to specify a callback for success and a callback for failure. If this piece of code is to be used often, it's best to create an API that returns a `promise` for the authentication, so the code the developer needs to write could be simplified to:

```
promiseAuthentication()
.then(authenticatedActions, sendUnauthenticatedMessage);
```

This in turn makes it easier to follow the code and prevent race conditions, as the code will always run in a clearly defined order.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Race Condition	2	0	0	2
Total	**2**	**0**	**0**	**2**

Race Condition	**High**
Package: src.main.resources.webgoat.static.js.libs	
src/main/resources/webgoat/static/js/libs/ace.js, line 2776 (Race Condition)	**High**
Issue Details	

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Race Condition — High

Package: src.main.resources.webgoat.static.js.libs

src/main/resources/webgoat/static/js/libs/ace.js, line 2776 (Race Condition) — High

Kingdom: Time and State
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: on
Enclosing Method: TextInput()
File: src/main/resources/webgoat/static/js/libs/ace.js:2776
Taint Flags:

```
2773  return isFocused;
2774  };
2775
2776  host.on("beforeEndOperation", function() {
2777  if (host.curOp && host.curOp.command.name == "insertstring")
2778  return;
2779  if (inComposition) {
```

src/main/resources/webgoat/static/js/libs/ace.js, line 2776 (Race Condition) — High

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionPointerCall: on
Enclosing Method: TextInput()
File: src/main/resources/webgoat/static/js/libs/ace.js:2776
Taint Flags:

```
2773  return isFocused;
2774  };
2775
2776  host.on("beforeEndOperation", function() {
2777  if (host.curOp && host.curOp.command.name == "insertstring")
2778  return;
2779  if (inComposition) {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Resource Injection (1 issue)

Abstract

Allowing user input to control resource identifiers could enable an attacker to access or modify otherwise protected system resources.

Explanation

A resource injection issue occurs when the following two conditions are met: 1. An attacker is able to specify the identifier used to access a system resource. For example, an attacker may be able to specify a port number to be used to connect to a network resource. 2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted. For example, the program may give the attacker the ability to transmit sensitive information to a third-party server. Note: Resource injections involving resources stored on the file system are reported in a separate category named path manipulation. See the path manipulation description for further details of this vulnerability. **Example 1:** The following code uses a port number read from an HTTP request to create a socket.

```
String remotePort = request.getParameter("remotePort");
...
ServerSocket srvr = new ServerSocket(remotePort);
Socket skt = srvr.accept();
...
```

Some think that in the mobile world, classic web application vulnerabilities, such as resource injection, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication. **Example 2:** The following code uses a URL read from an Android intent to load the page in `WebView`.

```
...
        WebView webview = new WebView(this);
        setContentView(webview);
        String url = this.getIntent().getExtras().getString("url");
        webview.loadUrl(url);
...
```

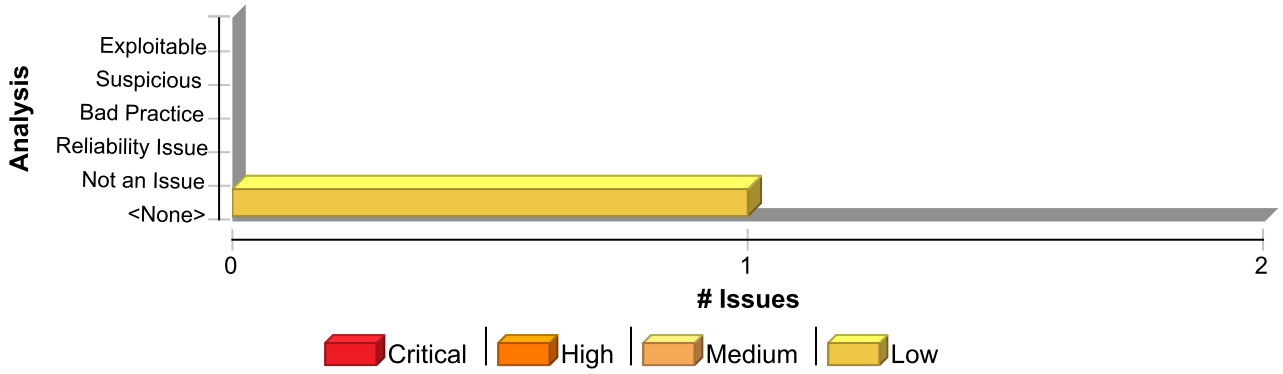
The kind of resource affected by user input indicates the kind of content that may be dangerous. For example, data containing special characters like period, slash, and backslash are risky when used in methods that interact with the file system. Similarly, data that contains URLs and URIs is risky for functions that create remote connections.

Recommendation

The best way to prevent resource injection is with a level of indirection: create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name. In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to maintain. Programmers often resort to implementing a deny list in these situations. A deny list is used to selectively reject or escape potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a list of characters that are permitted to appear in the resource name and accept input composed exclusively of characters in the approved set.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Resource Injection	1	0	0	1
Total	**1**	**0**	**0**	**1**

Resource Injection — Low

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 108 (Resource Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.util.Properties.load()
From: MavenWrapperDownloader.main
File: .mvn/wrapper/MavenWrapperDownloader.java:62

```
59  try {
60  mavenWrapperPropertyFileInputStream = new
FileInputStream(mavenWrapperPropertyFile);
61  Properties mavenWrapperProperties = new Properties();
62  mavenWrapperProperties.load(mavenWrapperPropertyFileInputStream);
63  url = mavenWrapperProperties.getProperty(PROPERTY_NAME_WRAPPER_URL, url);
64  } catch (IOException e) {
65  System.out.println("- ERROR loading '" + MAVEN_WRAPPER_PROPERTIES_PATH +
"'");
```

Sink Details

Sink: java.net.URL.URL()
Enclosing Method: downloadFileFromURL()
File: .mvn/wrapper/MavenWrapperDownloader.java:108
Taint Flags: PROPERTY

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Resource Injection	Low
Package: .mvn.wrapper	
.mvn/wrapper/MavenWrapperDownloader.java, line 108 (Resource Injection)	**Low**

```
105   }
106   });
107   }
108   URL website = new URL(urlString);
109   ReadableByteChannel rbc;
110   rbc = Channels.newChannel(website.openStream());
111   FileOutputStream fos = new FileOutputStream(destination);
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL Injection (16 issues)

Abstract

Constructing a dynamic SQL statement with input that comes from an untrusted source could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Explanation

© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL injection errors occur when: 1. Data enters a program from an untrusted source. In this case, Fortify Static Code Analyzer could not determine that the source of the data is trusted. 2. The data is used to dynamically construct a SQL query. **Example 1:** The following code dynamically constructs and executes a SQL query that searches for items matching a specified name. The query restricts the items displayed to those where the owner matches the user name of the currently-authenticated user.

```
...
        String userName = ctx.getAuthenticatedUserName();
        String itemName = request.getParameter("itemName");
        String query = "SELECT * FROM items WHERE owner = '"
                               + userName + "' AND itemname = '"
                               + itemName + "'";
        ResultSet rs = stmt.execute(query);
...
```

The query intends to execute the following code:

```
        SELECT * FROM items
        WHERE owner = <userName>
        AND itemname = <itemName>;
```

However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if `itemName` does not contain a single-quote character. If an attacker with the user name `wiley` enters the string "`name' OR 'a'='a`" for `itemName`, then the query becomes the following:

```
        SELECT * FROM items
        WHERE owner = 'wiley'
        AND itemname = 'name' OR 'a'='a';
```

The addition of the `OR 'a'='a'` condition causes the where clause to always evaluate to true, so the query becomes logically equivalent to the much simpler query:

```
        SELECT * FROM items;
```

This simplification of the query allows the attacker to bypass the requirement that the query must only return items owned by the authenticated user. The query now returns all entries stored in the `items` table, regardless of their specified owner. **Example 2:** This example examines the effects of a different malicious value passed to the query constructed and executed in `Example 1`. If an attacker with the user name `wiley` enters the string "`name'; DELETE FROM items; --`" for `itemName`, then the query becomes the following two queries:

```
        SELECT * FROM items
        WHERE owner = 'wiley'
        AND itemname = 'name';

        DELETE FROM items;

        --'
```

Many database servers, including Microsoft(R) SQL Server 2000, allow multiple SQL statements separated by semicolons to be executed at once. While this attack string results in an error on Oracle and other database servers that do not allow the batch-execution of statements separated by semicolons, on databases that do allow batch execution, this type of attack allows the attacker to execute arbitrary commands against the database. Notice the trailing pair of hyphens (--), which specifies to most database servers that the remainder of the statement is to be treated as a comment and not executed [4]. In this case the comment character serves to remove the trailing single-quote left over from the modified query. On a database where comments are not allowed to be used in this way, the general attack could still be made effective using a trick similar to the one used in `Example 1`. If an attacker enters the string "`name'); DELETE FROM items; SELECT * FROM items WHERE 'a'='a`", the following three valid statements will be created:

```
        SELECT * FROM items
        WHERE owner = 'wiley'
        AND itemname = 'name';

        DELETE FROM items;
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

```
        SELECT * FROM items WHERE 'a'='a';
```
Some think that in the mobile world, classic web application vulnerabilities, such as SQL injection, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.
Example 3: The following code adapts `Example 1` to the Android platform.
```
...
        PasswordAuthentication pa = authenticator.getPasswordAuthentication();
        String userName = pa.getUserName();
        String itemName = this.getIntent().getExtras().getString("itemName");
        String query = "SELECT * FROM items WHERE owner = '"
                               + userName + "' AND itemname = '"
                               + itemName + "'";
        SQLiteDatabase db = this.openOrCreateDatabase("DB", MODE_PRIVATE,
null);
        Cursor c = db.rawQuery(query, null);
...
```
One traditional approach to preventing SQL injection attacks is to handle them as an input validation problem and either accept only characters from an allow list of safe values or identify and escape a list of potentially malicious values (deny list). Checking an allow list can be a very effective means of enforcing strict input validation rules, but parameterized SQL statements require less maintenance and can offer more guarantees with respect to security. As is almost always the case, implementing a deny list is riddled with loopholes that make it ineffective at preventing SQL injection attacks. For example, attackers may: - Target fields that are not quoted - Find ways to bypass the need for certain escaped metacharacters - Use stored procedures to hide the injected metacharacters Manually escaping characters in input to SQL queries can help, but it will not make your application secure from SQL injection attacks. Another solution commonly proposed for dealing with SQL injection attacks is to use stored procedures. Although stored procedures prevent some types of SQL injection attacks, they fail to protect against many others. Stored procedures typically help prevent SQL injection attacks by limiting the types of statements that can be passed to their parameters. However, there are many ways around the limitations and many interesting statements that can still be passed to stored procedures. Again, stored procedures can prevent some exploits, but they will not make your application secure against SQL injection attacks.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

The root cause of a SQL injection vulnerability is the ability of an attacker to change context in the SQL query, causing a value that the programmer intended to be interpreted as data to be interpreted as a command instead. When a SQL query is constructed, the programmer knows what should be interpreted as part of the command and what should be interpreted as data. Parameterized SQL statements can enforce this behavior by disallowing data-directed context changes and preventing nearly all SQL injection attacks. Parameterized SQL statements are constructed using strings of regular SQL, but when user-supplied data needs to be included, they create bind parameters, which are placeholders for data that is subsequently inserted. Bind parameters allow the program to explicitly specify to the database what should be treated as a command and what should be treated as data. When the program is ready to execute a statement, it specifies to the database the runtime values to use for the value of each of the bind parameters, without the risk of the data being interpreted as commands. Example 1 can be rewritten to use parameterized SQL statements (instead of concatenating user supplied strings) as follows:

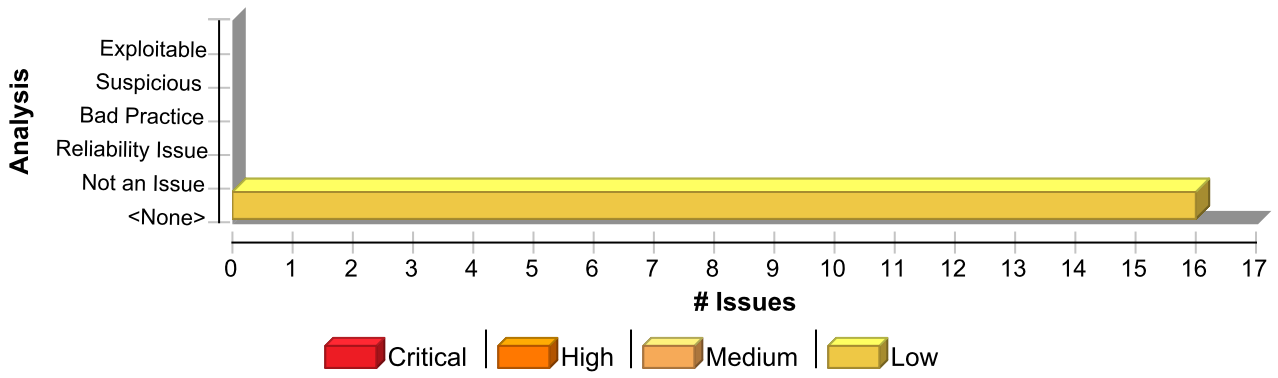
```
...
        String userName = ctx.getAuthenticatedUserName();
        String itemName = request.getParameter("itemName");
        String query =
                "SELECT * FROM items WHERE itemname=? AND owner=?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setString(1, itemName);
        stmt.setString(2, userName);
        ResultSet results = stmt.execute();
...
```

And here is an Android equivalent:

```
...
        PasswordAuthentication pa = authenticator.getPasswordAuthentication();
        String userName = pa.getUserName();
        String itemName = this.getIntent().getExtras().getString("itemName");
        String query = "SELECT * FROM items WHERE itemname=? AND owner=?";
        SQLiteDatabase db = this.openOrCreateDatabase("DB", MODE_PRIVATE,
null);
        Cursor c = db.rawQuery(query, new Object[]{itemName, userName});
...
```

More complicated scenarios, often found in report generation code, require that user input affect the command structure of the SQL statement, such as the addition of dynamic constraints in the `WHERE` clause. Do not use this requirement to justify concatenating user input into query strings. Prevent SQL injection attacks where user input must affect statement command structure with a level of indirection: create a set of legitimate strings that correspond to different elements you might include in a SQL statement. When constructing a statement, use input from the user to select from this set of application-controlled values.

Issue Summary



© Copyright [2008-2024] Micro Focus or one of its affiliates.

Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
SQL Injection	16	0	0	16
Total	**16**	**0**	**0**	**16**

SQL Injection — Low

Package: org.owasp.webgoat.container.lessons

src/main/java/org/owasp/webgoat/container/lessons/LessonConnectionInvocationHandler.java, line 27 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: execute()
Enclosing Method: invoke()
File: src/main/java/org/owasp/webgoat/container/lessons/LessonConnectionInvocationHandler.java:27
Taint Flags:

```
24  var authentication = SecurityContextHolder.getContext().getAuthentication();
25  if (authentication != null && authentication.getPrincipal() instanceof WebGoatUser user) {
26  try (var statement = targetConnection.createStatement()) {
27  statement.execute("SET SCHEMA \"" + user.getUsername() + "\"");
28  }
29  }
30  try {
```

Package: org.owasp.webgoat.lessons.challenges.challenge5

src/main/java/org/owasp/webgoat/lessons/challenges/challenge5/Assignment5.java, line 59 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: prepareStatement()
Enclosing Method: login()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge5/Assignment5.java:59
Taint Flags:

```
56  }
57  try (var connection = dataSource.getConnection()) {
58  PreparedStatement statement =
59  connection.prepareStatement(
60  "select password from challenge_users where userid = '"
61  + username_login
62  + "' and password = '"
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL Injection — Low

Package: org.owasp.webgoat.lessons.jwt.claimmisuse

src/main/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpoint.java, line 91 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeQuery()
Enclosing Method: resolveSigningKeyBytes()
File: src/main/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderKIDEndpoint.java:91
Taint Flags:

```
88  ResultSet rs =
89  connection
90  .createStatement()
91  .executeQuery(
92  "SELECT key FROM jwt_keys WHERE id = '" + kid + "'");
93  while (rs.next()) {
94  return TextCodec.BASE64.decode(rs.getString(1));
```

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java, line 74 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeQuery()
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java:74
Taint Flags:

```
71  try (Statement statement =
72  connection.createStatement(
73  ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY)) {
74  ResultSet results = statement.executeQuery(query);
75
76  if ((results != null) && results.first()) {
77  ResultSetMetaData resultsMetaData = results.getMetaData();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallenge.java, line 69 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL Injection — Low

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallenge.java, line 69 (SQL Injection) — Low

Sink Details

Sink: executeQuery()
Enclosing Method: registerNewUser()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallenge.java:69
Taint Flags:

```
66  String checkUserQuery =
67  "select userid from sql_challenge_users where userid = '" + username_reg + "'";
68  Statement statement = connection.createStatement();
69  ResultSet resultSet = statement.executeQuery(checkUserQuery);
70
71  if (resultSet.next()) {
72  if (username_reg.contains("tom'")) {
```

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java, line 71 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeQuery()
Enclosing Method: injectableQueryAvailability()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java:71
Taint Flags:

```
68  Statement statement =
69  connection.createStatement(
70  ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
71  ResultSet results = statement.executeQuery(query);
72
73  if (results.getStatement() != null) {
74  results.first();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java, line 80 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL Injection — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java, line 80 (SQL Injection) — Low

Sink: executeQuery()
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5.java:80
Taint Flags:

```
77  try (Statement statement =
78  connection.createStatement(
79  ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE)) {
80  statement.executeQuery(query);
81  if (checkSolution(connection)) {
82  return success(this).build();
83  }
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java, line 78 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeQuery()
Enclosing Method: injectableQueryConfidentiality()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java:78
Taint Flags:

```
75  connection.createStatement(
76  ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);
77  log(connection, query);
78  ResultSet results = statement.executeQuery(query);
79
80  if (results.getStatement() != null) {
81  if (results.first()) {
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson2.java, line 65 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeQuery()
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson2.java:65
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL Injection — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson2.java, line 65 (SQL Injection) — Low

```
62  protected AttackResult injectableQuery(String query) {
63  try (var connection = dataSource.getConnection()) {
64  Statement statement = connection.createStatement(TYPE_SCROLL_INSENSITIVE,
CONCUR_READ_ONLY);
65  ResultSet results = statement.executeQuery(query);
66  StringBuilder output = new StringBuilder();
67
68  results.first();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5b.java, line 65 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: prepareStatement()
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5b.java:65
Taint Flags:

```
62  String queryString = "SELECT * From user_data WHERE Login_Count = ? and userid= " +
accountName;
63  try (Connection connection = dataSource.getConnection()) {
64  PreparedStatement query =
65  connection.prepareStatement(
66  queryString, ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
67
68  int count = 0;
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson3.java, line 63 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeUpdate()
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson3.java:63
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL Injection — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson3.java, line 63 (SQL Injection) — Low

```
60  connection.createStatement(TYPE_SCROLL_INSENSITIVE, CONCUR_READ_ONLY)) {
61  Statement checkStatement =
62  connection.createStatement(TYPE_SCROLL_INSENSITIVE, CONCUR_READ_ONLY);
63  statement.executeUpdate(query);
64  ResultSet results =
65  checkStatement.executeQuery("SELECT * FROM employees WHERE last_name='Barnett';");
66  StringBuilder output = new StringBuilder();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java, line 158 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeUpdate()
Enclosing Method: log()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java:158
Taint Flags:

```
155
156  try {
157  Statement statement = connection.createStatement(TYPE_SCROLL_SENSITIVE, CONCUR_UPDATABLE);
158  statement.executeUpdate(logQuery);
159  } catch (SQLException e) {
160  System.err.println(e.getMessage());
161  }
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5a.java, line 67 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeQuery()
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5a.java:67
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL Injection — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5a.java, line 67 (SQL Injection) — Low

```
64  try (Statement statement =
65  connection.createStatement(
66  ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE)) {
67  ResultSet results = statement.executeQuery(query);
68
69  if ((results != null) && (results.first())) {
70  ResultSetMetaData resultsMetaData = results.getMetaData();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java, line 81 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: execute()
Enclosing Method: injectableQueryIntegrity()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java:81
Taint Flags:

```
78  // do injectable query
79  Statement statement = connection.createStatement(TYPE_SCROLL_SENSITIVE, CONCUR_UPDATABLE);
80  SqlInjectionLesson8.log(connection, queryInjection);
81  statement.execute(queryInjection);
82  // check new sum of salaries other employees and new salaries of John
83  int newJohnSalary = this.getJohnSalary(connection);
84  int newSumSalariesOfOtherEmployees = this.getSumSalariesOfOtherEmployees(connection);
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson4.java, line 62 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: executeUpdate()
Enclosing Method: injectableQuery()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson4.java:62
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

SQL Injection — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson4.java, line 62 (SQL Injection) — Low

```
59  try (Connection connection = dataSource.getConnection()) {
60  try (Statement statement =
61  connection.createStatement(TYPE_SCROLL_INSENSITIVE, CONCUR_READ_ONLY)) {
62  statement.executeUpdate(query);
63  connection.commit();
64  ResultSet results = statement.executeQuery("SELECT phone from employees;");
65  StringBuilder output = new StringBuilder();
```

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/Servers.java, line 72 (SQL Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: prepareStatement()
Enclosing Method: sort()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/Servers.java:72
Taint Flags:

```
69
70  try (var connection = dataSource.getConnection()) {
71  try (var statement =
72  connection.prepareStatement(
73  "select id, hostname, ip, mac, status, description from SERVERS where status <> 'out"
74  + " of order' order by "
75  + column)) {
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak (8 issues)

Abstract

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An information leak occurs when system data or debug information leaves the program through an output stream or logging function. **Example 1:** The following code writes an exception to the standard error stream:

```
try {
    ...
} catch (Exception e) {
    e.printStackTrace();
}
```

Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a remote user. For example, with scripting mechanisms it is trivial to redirect output information from "Standard error" or "Standard output" into a file or another program. Alternatively, the system that the program runs on could have a remote logging mechanism such as a "syslog" server that sends the logs to a remote device. During development, you have no way of knowing where this information might end up being displayed. In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In `Example 1`, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program. Here is another scenario, specific to the mobile world. Most mobile devices now implement a Near-Field Communication (NFC) protocol for quickly sharing information between devices using radio communication. It works by bringing devices to close proximity or simply having them touch each other. Even though the communication range of NFC is limited to just a few centimeters, eavesdropping, data modification and various other types of attacks are possible, since NFC alone does not ensure secure communication. **Example 2:** The Android platform provides support for NFC. The following code creates a message that gets pushed to the other device within the range.

```
...
public static final String TAG = "NfcActivity";
private static final String DATA_SPLITTER = "__:DATA:__";
private static final String MIME_TYPE = "application/my.applications.mimetype";
...
public NdefMessage createNdefMessage(NfcEvent event) {
    TelephonyManager tm =
(TelephonyManager)Context.getSystemService(Context.TELEPHONY_SERVICE);
    String VERSION = tm.getDeviceSoftwareVersion();
    String text = TAG + DATA_SPLITTER + VERSION;
    NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
            MIME_TYPE.getBytes(), new byte[0], text.getBytes());
    NdefRecord[] records = { record };
    NdefMessage msg = new NdefMessage(records);
    return msg;
}
...
```

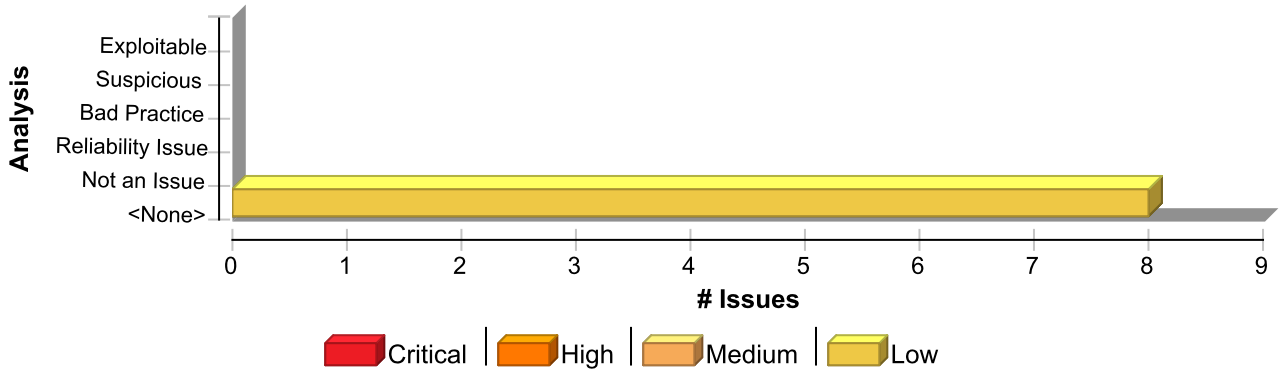
NFC Data Exchange Format (NDEF) message contains typed data, a URI, or a custom application payload. If the message contains information about the application, such as its name, MIME type, or device software version, this information could be leaked to an eavesdropper. In `Example 2`, Fortify Static Code Analyzer reports a System Information Leak vulnerability on the return statement.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Debug traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example). Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system. If you are concerned about leaking system data via NFC on an Android device, you could do one of the following three things. Do not include system data in the messages pushed to other devices in range, encrypt the payload of the message, or establish a secure communication channel at a higher layer.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak	8	0	0	8
Total	**8**	**0**	**0**	**8**

System Information Leak	Low
Package: .mvn.wrapper	
.mvn/wrapper/MavenWrapperDownloader.java, line 92 (System Information Leak)	**Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: main()
File: .mvn/wrapper/MavenWrapperDownloader.java:92
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak — Low

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 92 (System Information Leak) — Low

```
89  System.exit(0);
90  } catch (Throwable e) {
91  System.out.println("- Error downloading");
92  e.printStackTrace();
93  System.exit(1);
94  }
95  }
```

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 39 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: runTests()
File: src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java:39
Taint Flags:

```
36  try {
37  checkAssignmentSigning();
38  } catch (Exception e) {
39  e.printStackTrace();
40  fail();
41  }
42
```

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 32 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: runTests()
File: src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java:32
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak — Low

Package: org.owasp.webgoat

src/it/java/org/owasp/webgoat/CryptoIntegrationTest.java, line 32 (System Information Leak) — Low

```
29  try {
30  checkAssignment4();
31  } catch (NoSuchAlgorithmException e) {
32  e.printStackTrace();
33  fail();
34  }
35
```

src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java, line 171 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: getProperties()
File: src/it/java/org/owasp/webgoat/LabelAndHintIntegrationTest.java:171
Taint Flags:

```
168  // load a properties file
169  prop.load(input);
170  } catch (Exception e) {
171  e.printStackTrace();
172  }
173  return prop;
174  }
```

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java, line 75 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: getPassword()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java:75
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak — Low

Package: org.owasp.webgoat.lessons.sqlinjection.advanced

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java, line 75 (System Information Leak) — Low

```
72  // do nothing
73  }
74  } catch (Exception e) {
75  e.printStackTrace();
76  // do nothing
77  }
78  return (password);
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java, line 71 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: getPassword()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6b.java:71
Taint Flags:

```
68  password = results.getString("password");
69  }
70  } catch (SQLException sqle) {
71  sqle.printStackTrace();
72  // do nothing
73  }
74  } catch (Exception e) {
```

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java, line 130 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: getJavaFileContentsAsString()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java:130
Taint Flags:

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak — Low

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson10b.java, line 130 (System Information Leak) — Low

```
127  try {
128  javaFileObject = new JavaObjectFromString("TestClass.java", javaFileContents.toString());
129  } catch (Exception exception) {
130  exception.printStackTrace();
131  }
132  return javaFileObject;
133  }
```

Package: org.owasp.webgoat.lessons.ssrf

src/main/java/org/owasp/webgoat/lessons/ssrf/SSRFTask1.java, line 62 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: stealTheCheese()
File: src/main/java/org/owasp/webgoat/lessons/ssrf/SSRFTask1.java:62
Taint Flags:

```
59  return failed(this).feedback("ssrf.failure").output(html.toString()).build();
60  }
61  } catch (Exception e) {
62  e.printStackTrace();
63  return failed(this).output(e.getMessage()).build();
64  }
65  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: External (2 issues)

Abstract

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An external information leak occurs when system data or debugging information leaves the program to a remote machine via a socket or network connection. External leaks can reveal specific data about operating systems, full pathnames, the existence of usernames, or locations of configuration files. These are mores serious than internal information leaks, which are more difficult for an attacker to access.
Example 1: The following Spring Boot configuration property leaks stacktrace information in the HTTP response:

```
server.error.include-stacktrace=always
```

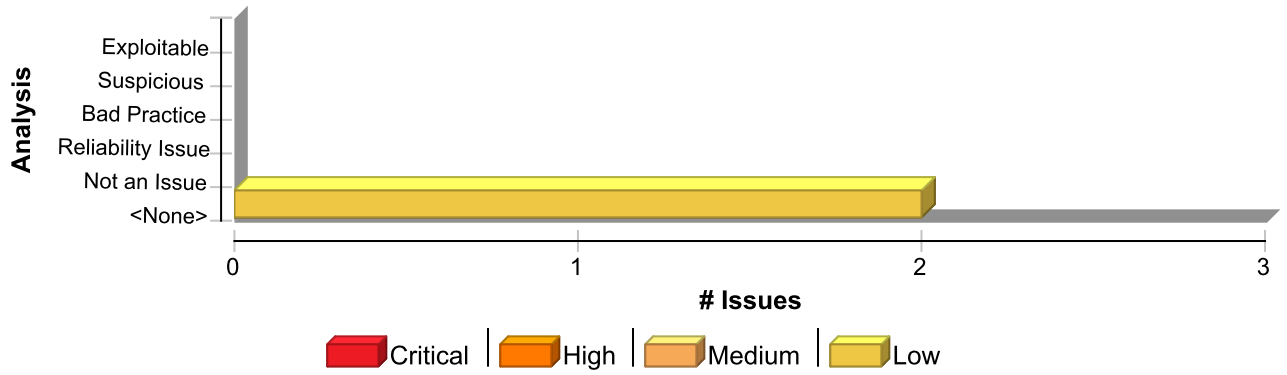
This information can be exposed to a remote user. In some cases, the error message reveals exactly the types of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more indirect clues about the system. In `Example 1`, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program.

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Be careful. Debug traces can sometimes appear in non-obvious places (for exaple, embedded in comments in the HTML for an error page). Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system. **Example 1:** The following Spring Boot configuration property prevents stacktrace information to be leaked in the HTTP response:

```
server.error.include-stacktrace=never
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak: External	2	0	0	2
Total	**2**	**0**	**0**	**2**

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: External — Low

Package: src.main.resources

src/main/resources/application-webgoat.properties, line 1 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Configuration)

Sink Details

Sink: server.error.include-stacktrace
File: src/main/resources/application-webgoat.properties:1
Taint Flags:

```
1 server.error.include-stacktrace=always
2 server.error.path=/error.html
3 server.servlet.context-path=${WEBGOAT_CONTEXT:/WebGoat}
4 server.servlet.session.persistent=false
5
6 undefined
7 undefined
```

src/main/resources/application-webwolf.properties, line 1 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Configuration)

Sink Details

Sink: server.error.include-stacktrace
File: src/main/resources/application-webwolf.properties:1
Taint Flags:

```
1 server.error.include-stacktrace=always
2 server.error.path=/error.html
3 server.servlet.context-path=${webwolf.context}
4 server.port=${webwolf.port}
5
6 undefined
7 undefined
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal (10 issues)

Abstract

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An internal information leak occurs when system data or debug information is sent to a local file, console, or screen via printing or logging. **Example 1:** The following code writes an exception to the standard error stream:

```
try {
    ...
} catch (Exception e) {
    e.printStackTrace();
}
```

Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a user. In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In `Example 1`, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program. Information leaks are also a concern in a mobile computing environment. **Example 2:** The following code logs the stack trace of a caught exception on the Android platform.

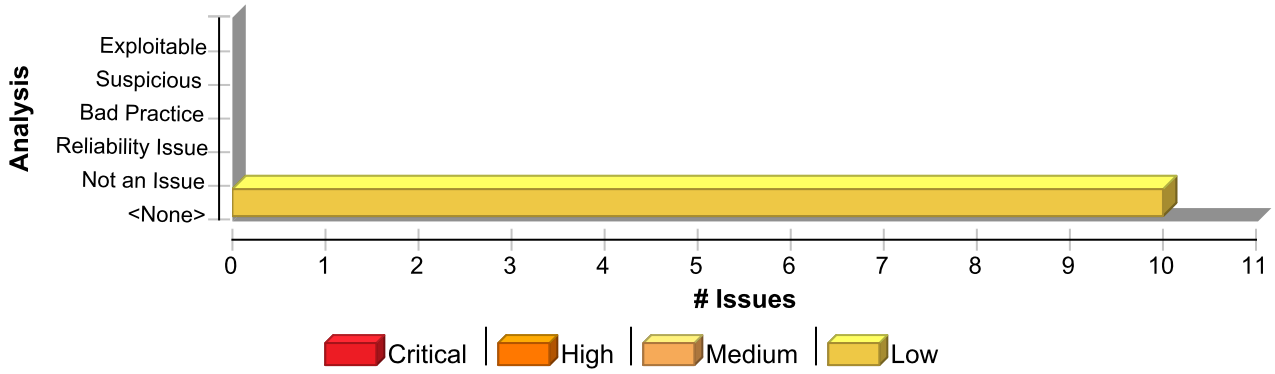
```
...
try {
  ...
} catch (Exception e) {
    Log.e(TAG, Log.getStackTraceString(e));
}
...
```

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Debug traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example). Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak: Internal	10	0	0	10
Total	**10**	**0**	**0**	**10**

System Information Leak: Internal — Low

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 72 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read e
From: org.dummy.insecure.framework.VulnerableTaskHolder.readObject
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:72

```
69  log.info(line);
70  }
71  } catch (IOException e) {
72  log.error("IO Exception", e);
73  }
74  }
75  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: readObject()
File: src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java:72
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal — Low

Package: org.dummy.insecure.framework

src/main/java/org/dummy/insecure/framework/VulnerableTaskHolder.java, line 72 (System Information Leak: Internal) — Low

```
69  log.info(line);
70  }
71  } catch (IOException e) {
72  log.error("IO Exception", e);
73  }
74  }
75  }
```

Package: org.owasp.webgoat.lessons.challenges.challenge7

src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java, line 51 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: org.owasp.webgoat.lessons.challenges.challenge7.MD5.main
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java:51

```
48  try {
49  System.out.println(MD5.getHashString(new File(element)) + " " + element);
50  } catch (IOException x) {
51  System.err.println(x.getMessage());
52  }
53  }
54  }
```

Sink Details

Sink: java.io.PrintStream.println()
Enclosing Method: main()
File: src/main/java/org/owasp/webgoat/lessons/challenges/challenge7/MD5.java:51
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
48  try {
49  System.out.println(MD5.getHashString(new File(element)) + " " + element);
50  } catch (IOException x) {
51  System.err.println(x.getMessage());
52  }
53  }
54  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal — Low

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 101 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read e
From: org.owasp.webgoat.lessons.cryptography.CryptoUtil.verifyMessage
File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:101

```
98
99   log.info("Verified the signature with result: {}", result);
100   } catch (Exception e) {
101   log.error("Signature verification failed", e);
102   }
103
104   log.debug("end verifyMessage");
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: verifyMessage()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:101
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
98
99   log.info("Verified the signature with result: {}", result);
100   } catch (Exception e) {
101   log.error("Signature verification failed", e);
102   }
103
104   log.debug("end verifyMessage");
```

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 72 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read e
From: org.owasp.webgoat.lessons.cryptography.CryptoUtil.signMessage
File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:72

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal — Low

Package: org.owasp.webgoat.lessons.cryptography

src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java, line 72 (System Information Leak: Internal) — Low

```
69
70  log.info("signe the signature with result: {}", signature);
71  } catch (Exception e) {
72  log.error("Signature signing failed", e);
73  }
74
75  log.debug("end signMessage");
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: signMessage()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/CryptoUtil.java:72
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
69
70  log.info("signe the signature with result: {}", signature);
71  } catch (Exception e) {
72  log.error("Signature signing failed", e);
73  }
74
75  log.debug("end signMessage");
```

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 111 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read e
From: org.owasp.webgoat.lessons.pathtraversal.ProfileUploadRetrieval.getProfilePicture
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java:111

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 111 (System Information Leak: Internal) — Low

```
108 StringUtils.arrayToCommaDelimitedString(catPicture.getParentFile().listFiles())
109   .getBytes());
110   } catch (IOException | URISyntaxException e) {
111   log.error("Image not found", e);
112   }
113
114   return ResponseEntity.badRequest().build();
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: getProfilePicture()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java:111
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
108 StringUtils.arrayToCommaDelimitedString(catPicture.getParentFile().listFiles())
109 .getBytes());
110 } catch (IOException | URISyntaxException e) {
111 log.error("Image not found", e);
112 }
113
114 return ResponseEntity.badRequest().build();
```

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 59 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: org.owasp.webgoat.lessons.pathtraversal.ProfileUploadRetrieval.initAssignment
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.
java:59

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal — Low

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 59 (System Information Leak: Internal) — Low

```
56  .getInputStream()) {
57  FileCopyUtils.copy(is, new FileOutputStream(new
File(catPicturesDirectory, i + ".jpg")));
58  } catch (Exception e) {
59  log.error("Unable to copy pictures" + e.getMessage());
60  }
61  }
62  var secretDirectory =
this.catPicturesDirectory.getParentFile().getParentFile();
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: initAssignment()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java:59
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
56  .getInputStream()) {
57  FileCopyUtils.copy(is, new FileOutputStream(new File(catPicturesDirectory, i + ".jpg")));
58  } catch (Exception e) {
59  log.error("Unable to copy pictures" + e.getMessage());
60  }
61  }
62  var secretDirectory = this.catPicturesDirectory.getParentFile().getParentFile();
```

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java, line 123 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: org.owasp.webgoat.lessons.sqlinjection.introduction.SqlInjectionLesson10.tabl
eExists
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectio
nLesson10.java:123

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java, line 123 (System Information Leak: Internal) — Low

```
120  if (errorMsg.contains("object not found: ACCESS_LOG")) {
121  return false;
122  } else {
123  System.err.println(e.getMessage());
124  return false;
125  }
126  }
```

Sink Details

Sink: java.io.PrintStream.println()
Enclosing Method: tableExists()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java:123
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
120  if (errorMsg.contains("object not found: ACCESS_LOG")) {
121  return false;
122  } else {
123  System.err.println(e.getMessage());
124  return false;
125  }
126  }
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java, line 102 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: org.owasp.webgoat.lessons.sqlinjection.introduction.SqlInjectionLesson9.injectableQueryIntegrity
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java:102

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java, line 102 (System Information Leak: Internal) — Low

```
99  SqlInjectionLesson8.generateTable(this.getEmployeesDataOrderBySalaryDesc(connec
100   .build();
101   } catch (SQLException e) {
102   System.err.println(e.getMessage());
103   return failed(this)
104   .output("<br><span class='feedback-negative'>" + e.getMessage() + "</
span>")
105   .build();
```

Sink Details

Sink: java.io.PrintStream.println()
Enclosing Method: injectableQueryIntegrity()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java:102
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
99  SqlInjectionLesson8.generateTable(this.getEmployeesDataOrderBySalaryDesc(connection)))
100  .build();
101  } catch (SQLException e) {
102  System.err.println(e.getMessage());
103  return failed(this)
104  .output("<br><span class='feedback-negative'>" + e.getMessage() + "</span>")
105  .build();
```

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java, line 160 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: org.owasp.webgoat.lessons.sqlinjection.introduction.SqlInjectionLesson8.log
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java:160

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal — Low

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java, line 160 (System Information Leak: Internal) — Low

```
157  Statement statement = connection.createStatement(TYPE_SCROLL_SENSITIVE,
CONCUR_UPDATABLE);
158  statement.executeUpdate(logQuery);
159  } catch (SQLException e) {
160  System.err.println(e.getMessage());
161  }
162  }
163 }
```

Sink Details

Sink: java.io.PrintStream.println()
Enclosing Method: log()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java:160
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
157  Statement statement = connection.createStatement(TYPE_SCROLL_SENSITIVE, CONCUR_UPDATABLE);
158  statement.executeUpdate(logQuery);
159  } catch (SQLException e) {
160  System.err.println(e.getMessage());
161  }
162  }
163 }
```

Package: org.owasp.webgoat.lessons.sqlinjection.mitigation

src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13.java, line 70 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read e
From: org.owasp.webgoat.lessons.sqlinjection.mitigation.SqlInjectionLesson13.comple
ted
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionL
esson13.java:70

© Copyright [2008-2024] Micro Focus or one of its affiliates.

System Information Leak: Internal	Low
Package: org.owasp.webgoat.lessons.sqlinjection.mitigation	
src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13.java, line 70 (System Information Leak: Internal)	Low

```
67  }
68  return failed(this).build();
69  } catch (SQLException e) {
70  log.error("Failed", e);
71  return (failed(this).build());
72  }
73  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: completed()
File: src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/SqlInjectionLesson13.java:70
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
67  }
68  return failed(this).build();
69  } catch (SQLException e) {
70  log.error("Failed", e);
71  return (failed(this).build());
72  }
73  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unchecked Return Value (4 issues)

Abstract

Ignoring a method's return value can cause the program to overlook unexpected states and conditions.

Explanation

It is not uncommon for Java programmers to misunderstand `read()` and related methods that are part of many `java.io` classes. Most errors and unusual events in Java result in an exception being thrown. (This is one of the advantages that Java has over languages like C: Exceptions make it easier for programmers to think about what can go wrong.) But the stream and reader classes do not consider it unusual or exceptional if only a small amount of data becomes available. These classes simply add the small amount of data to the return buffer, and set the return value to the number of bytes or characters read. There is no guarantee that the amount of data returned is equal to the amount of data requested. This behavior makes it important for programmers to examine the return value from `read()` and other IO methods to ensure that they receive the amount of data they expect. **Example:** The following code loops through a set of users, reading a private data file for each user. The programmer assumes that the files are always exactly 1 kilobyte in size and therefore ignores the return value from `read()`. If an attacker can create a smaller file, the program will recycle the remainder of the data from the previous user and handle it as though it belongs to the attacker.

```
FileInputStream fis;
byte[] byteArray = new byte[1024];
for (Iterator i=users.iterator(); i.hasNext();) {
    String userName = (String) i.next();
    String pFileName = PFILE_ROOT + "/" + userName;
    FileInputStream fis = new FileInputStream(pFileName);
    fis.read(byteArray); // the file is always 1k bytes
    fis.close();
    processPFile(userName, byteArray);
}
```

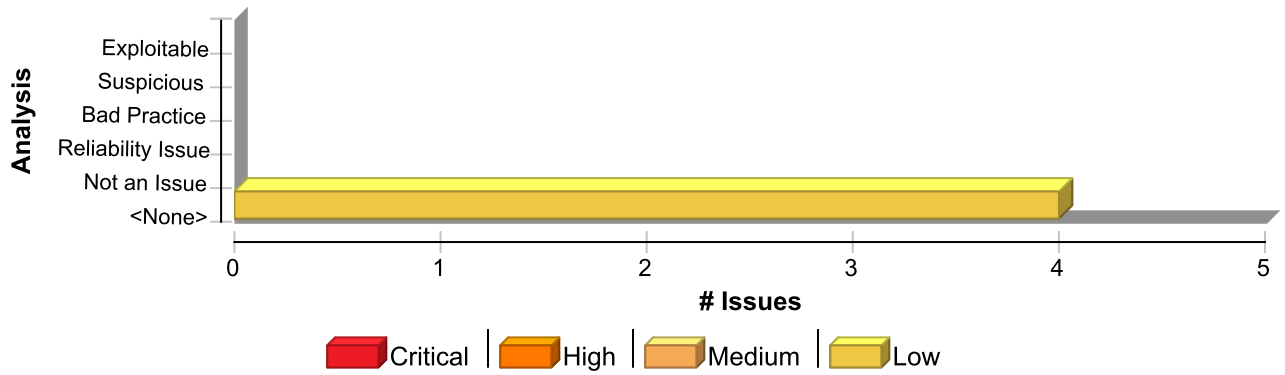

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

```
FileInputStream fis;
byte[] byteArray = new byte[1024];
for (Iterator i=users.iterator(); i.hasNext();) {
  String userName = (String) i.next();
  String pFileName = PFILE_ROOT + "/" + userName;
  fis = new FileInputStream(pFileName);
  int bRead = 0;
  while (bRead < 1024) {
      int rd = fis.read(byteArray, bRead, 1024 - bRead);
      if (rd == -1) {
        throw new IOException("file is unusually small");
      }
      bRead += rd;
  }
  // could add check to see if file is too large here
  fis.close();
  processPFile(userName, byteArray);
}
```

Note: Because the fix for this problem is relatively complicated, you might be tempted to use a simpler approach, such as checking the size of the file before you begin reading. Such an approach would render the application vulnerable to a file system race condition, whereby an attacker could replace a well-formed file with a malicious file between the file size check and the call to read data from the file.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unchecked Return Value	4	0	0	4
Total	**4**	**0**	**0**	**4**

Unchecked Return Value	**Low**
Package: org.owasp.webgoat.lessons.clientsidefiltering	
src/main/java/org/owasp/webgoat/lessons/clientsidefiltering/Salaries.java, line 62 (Unchecked Return Value)	**Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unchecked Return Value — Low

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/main/java/org/owasp/webgoat/lessons/clientsidefiltering/Salaries.java, line 62 (Unchecked Return Value) — Low

Sink Details

Sink: mkdir()
Enclosing Method: copyFiles()
File: src/main/java/org/owasp/webgoat/lessons/clientsidefiltering/Salaries.java:62
Taint Flags:

```
59  ClassPathResource classPathResource = new ClassPathResource("lessons/employees.xml");
60  File targetDirectory = new File(webGoatHomeDirectory, "/ClientSideFiltering");
61  if (!targetDirectory.exists()) {
62  targetDirectory.mkdir();
63  }
64  try {
65  FileCopyUtils.copy(
```

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 48 (Unchecked Return Value) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: mkdirs()
Enclosing Method: ProfileUploadRetrieval()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java:48
Taint Flags:

```
45
46  public ProfileUploadRetrieval(@Value("${webgoat.server.directory}") String
webGoatHomeDirectory) {
47  this.catPicturesDirectory = new File(webGoatHomeDirectory, "/PathTraversal/" + "/cats");
48  this.catPicturesDirectory.mkdirs();
49  }
50
51  @PostConstruct
```

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignment.java, line 76 (Unchecked Return Value) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unchecked Return Value — Low

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignment.java, line 76 (Unchecked Return Value) — Low

Sink: mkdirs()
Enclosing Method: createSecretFileWithRandomContents()
File: src/main/java/org/owasp/webgoat/lessons/xxe/BlindSendFileAssignment.java:76
Taint Flags:

```
73  userToFileContents.put(user, fileContents);
74  File targetDirectory = new File(webGoatHomeDirectory, "/XXE/" + user.getUsername());
75  if (!targetDirectory.exists()) {
76  targetDirectory.mkdirs();
77  }
78  try {
79  Files.writeString(new File(targetDirectory, "secret.txt").toPath(), fileContents, UTF_8);
```

Package: org.owasp.webgoat.webwolf

src/main/java/org/owasp/webgoat/webwolf/MvcConfiguration.java, line 72 (Unchecked Return Value) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: mkdirs()
Enclosing Method: createDirectory()
File: src/main/java/org/owasp/webgoat/webwolf/MvcConfiguration.java:72
Taint Flags:

```
69  public void createDirectory() {
70  File file = new File(fileLocation);
71  if (!file.exists()) {
72  file.mkdirs();
73  }
74  }
75 }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unreleased Resource: Database (2 issues)

Abstract

The program can potentially fail to release a database resource.

Explanation

Resource leaks have at least two common causes: - Error conditions and other exceptional circumstances. - Confusion over which part of the program is responsible for releasing the resource. Most unreleased resource issues result in general software reliability problems. However, if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool. **Example:** Under normal conditions, the following code executes a database query, processes the results returned by the database, and closes the allocated statement object. But if an exception occurs while executing the SQL or processing the results, the statement object will not be closed. If this happens often enough, the database will run out of available cursors and not be able to execute any more SQL queries.

```
  Statement stmt = conn.createStatement();
  ResultSet rs = stmt.executeQuery(CXN_SQL);
  harvestResults(rs);
  stmt.close();
```

In this case, there are program paths on which a Statement is not released.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

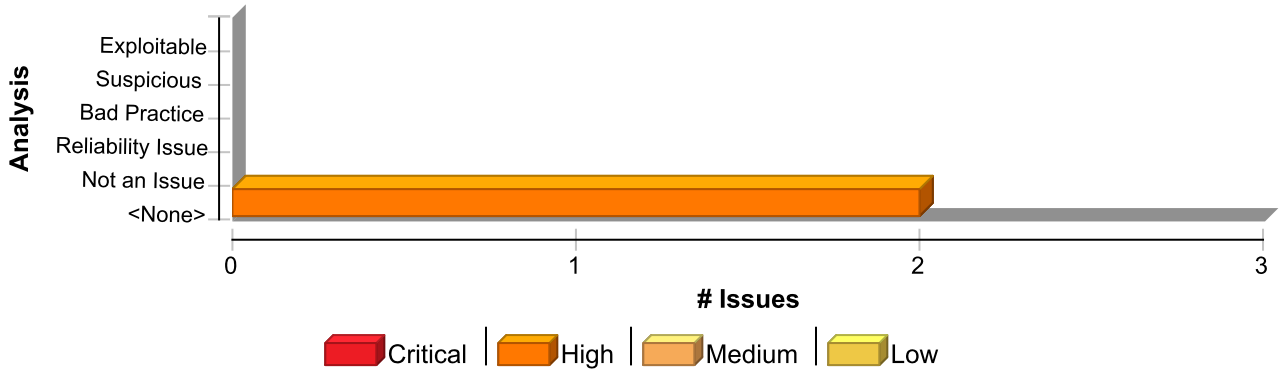
1. Never rely on `finalize()` to reclaim resources. In order for an object's `finalize()` method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the JVM is low on memory, there is no guarantee that an object's `finalize()` method will be invoked in an expedient fashion. When the garbage collector finally does run, it may cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. This effect becomes more pronounced as the load on the system increases. Finally, if it is possible for a resource reclamation operation to hang (if it requires communicating over a network to a database, for example), then the thread that is executing the `finalize()` method will hang. 2. Release resources in a `finally` block. The code for the Example should be rewritten as follows:

```
  public void execCxnSql(Connection conn) {
    Statement stmt;
    try {
      stmt = conn.createStatement();
      ResultSet rs = stmt.executeQuery(CXN_SQL);
      ...
    }
    finally {
      if (stmt != null) {
        safeClose(stmt);
      }
    }
}

public static void safeClose(Statement stmt) {
  if (stmt != null) {
    try {
      stmt.close();
    } catch (SQLException e) {
      log(e);
    }
  }
}
```

This solution uses a helper function to log the exceptions that might occur when trying to close the statement. Presumably this helper function will be reused whenever a statement needs to be closed. Also, the `execCxnSql` method does not initialize the `stmt` object to `null`. Instead, it checks to ensure that `stmt` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `stmt` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `stmt` is initialized to `null` in a more complex method, cases in which `stmt` is used without being initialized will not be detected by the compiler.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Database	2	0	0	2
Total	**2**	**0**	**0**	**2**

Unreleased Resource: Database — High

Package: org.owasp.webgoat.lessons.sqlinjection.introduction

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java, line 44 (Unreleased Resource: Database) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: getConnection()
Enclosing Method: removeGrant()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java:44
Taint Flags:

```
41  @AfterEach
42  public void removeGrant() throws SQLException {
43  dataSource
44  .getConnection()
45  .prepareStatement("revoke select on grant_rights from unauthorized_user cascade")
46  .execute();
47  }
```

src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java, line 45 (Unreleased Resource: Database) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unreleased Resource: Database	High
Package: org.owasp.webgoat.lessons.sqlinjection.introduction	
src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java, line 45 (Unreleased Resource: Database)	High

Sink: this.dataSource.getConnection().prepareStatement(...)
Enclosing Method: removeGrant()
File: src/test/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5Test.java:45
Taint Flags:

```
42  public void removeGrant() throws SQLException {
43  dataSource
44  .getConnection()
45  .prepareStatement("revoke select on grant_rights from unauthorized_user cascade")
46  .execute();
47  }
48
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unreleased Resource: Files (1 issue)

Abstract

The program can potentially fail to release a file handle.

Explanation

The program can potentially fail to release a file handle. Resource leaks have at least two common causes: - Error conditions and other exceptional circumstances. - Confusion over which part of the program is responsible for releasing the resource. Most unreleased resource issues result in general software reliability problems. However, if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool. **Example 1:** The following method never closes the file handle it opens. The `finalize()` method for `ZipFile` eventually calls `close()`, but there is no guarantee as to how long it will take before the `finalize()` method will be invoked. In a busy environment, this can result in the JVM using up all of its file handles.

```
public void printZipContents(String fName)
    throws ZipException, IOException, SecurityException,
IllegalStateException, NoSuchElementException
{
  ZipFile zf = new ZipFile(fName);
  Enumeration<ZipEntry> e = zf.entries();

  while (e.hasMoreElements()) {
    printFileInfo(e.nextElement());
  }
}
```

Example 2: Under normal conditions, the following fix properly closes the file handle after printing out all the zip file entries. But if an exception occurs while iterating through the entries, the zip file handle will not be closed. If this happens often enough, the JVM can still run out of available file handles.

```
public void printZipContents(String fName)
    throws ZipException, IOException, SecurityException,
IllegalStateException, NoSuchElementException
{
  ZipFile zf = new ZipFile(fName);
  Enumeration<ZipEntry> e = zf.entries();

  while (e.hasMoreElements()) {
    printFileInfo(e.nextElement());
  }
}
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

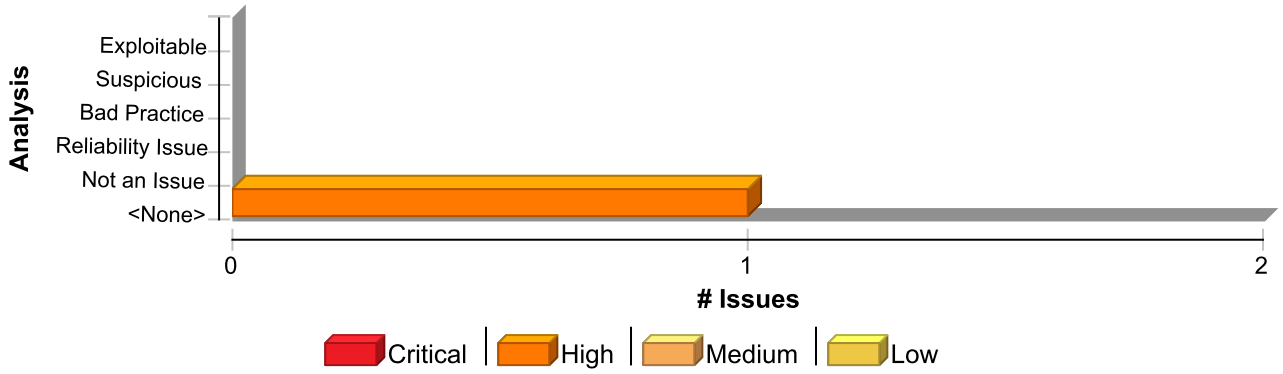
1. Never rely on `finalize()` to reclaim resources. In order for an object's `finalize()` method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the JVM is low on memory, there is no guarantee that an object's `finalize()` method will be invoked in an expedient fashion. When the garbage collector finally does run, it may cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. This effect becomes more pronounced as the load on the system increases. Finally, if it is possible for a resource reclamation operation to hang (if it requires communicating over a network, for example), then the thread that is executing the `finalize()` method will hang. 2. Release resources in a `finally` block. The code for `Example 2` should be rewritten as follows:

```
public void printZipContents(String fName)
    throws ZipException, IOException, SecurityException,
IllegalStateException, NoSuchElementException
{
  ZipFile zf;
  try {
    zf = new ZipFile(fName);
    Enumeration<ZipEntry> e = zf.entries();
    ...
  }
  finally {
    if (zf != null) {
      safeClose(zf);
    }
  }
}

public static void safeClose(ZipFile zf) {
  if (zf != null) {
    try {
      zf.close();
    } catch (IOException e) {
      log(e);
    }
  }
}
```

This solution uses a helper function to log the exceptions that might occur when trying to close the file. Presumably this helper function will be reused whenever a file needs to be closed. Also, the `printZipContents` method does not initialize the `zf` object to `null`. Instead, it checks to ensure that `zf` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `zf` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `zf` is initialized to `null` in a more complex method, cases in which `zf` is used without being initialized will not be detected by the compiler.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Files	1	0	0	1
Total	**1**	**0**	**0**	**1**

Unreleased Resource: Files — High

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java, line 69 (Unreleased Resource: Files) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: zip = new ZipFile(...)
Enclosing Method: processZipUpload()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileZipSlip.java:69
Taint Flags:

```
66  var uploadedZipFile = tmpZipDirectory.resolve(file.getOriginalFilename());
67  FileCopyUtils.copy(file.getBytes(), uploadedZipFile.toFile());
68
69  ZipFile zip = new ZipFile(uploadedZipFile.toFile());
70  Enumeration<? extends ZipEntry> entries = zip.entries();
71  while (entries.hasMoreElements()) {
72  ZipEntry e = entries.nextElement();
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unreleased Resource: Streams (4 issues)

Abstract

The program can potentially fail to release a system resource.

Explanation

The program can potentially fail to release a system resource. Resource leaks have at least two common causes: - Error conditions and other exceptional circumstances. - Confusion over which part of the program is responsible for releasing the resource. Most unreleased resource issues result in general software reliability problems. However, if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool. **Example:** The following method never closes the file handle it opens. The `finalize()` method for `FileInputStream` eventually calls `close()`, but there is no guarantee as to how long it will take before the `finalize()` method will be invoked. In a busy environment, this can result in the JVM using up all of its file handles.

```
private void processFile(String fName) throws FileNotFoundException,
IOException {
  FileInputStream fis = new FileInputStream(fName);
  int sz;
  byte[] byteArray = new byte[BLOCK_SIZE];
  while ((sz = fis.read(byteArray)) != -1) {
    processBytes(byteArray, sz);
  }
}
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Recommendation

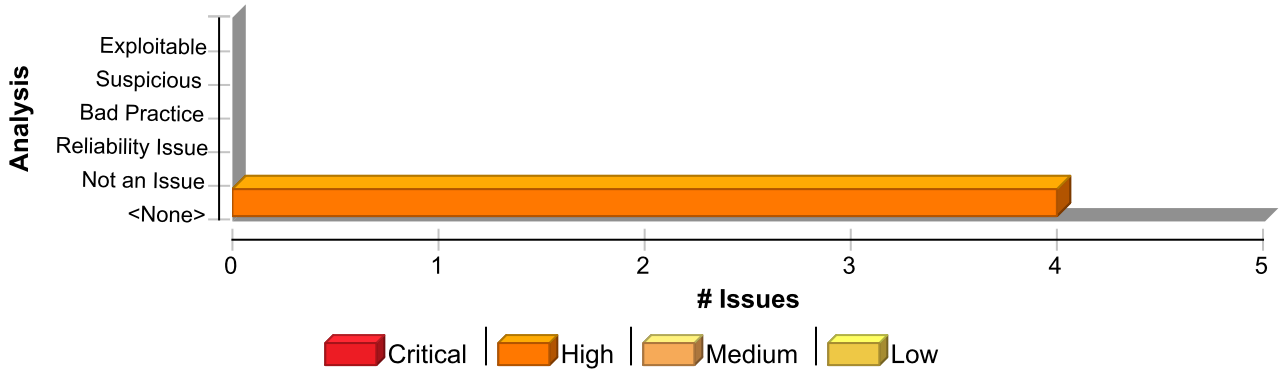
1. Never rely on `finalize()` to reclaim resources. In order for an object's `finalize()` method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the JVM is low on memory, there is no guarantee that an object's `finalize()` method will be invoked in an expedient fashion. When the garbage collector finally does run, it may cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. This effect becomes more pronounced as the load on the system increases. Finally, if it is possible for a resource reclamation operation to hang (if it requires communicating over a network to a database, for example), then the thread that is executing the `finalize()` method will hang. 2. Release resources in a `finally` block. The code for the Example should be rewritten as follows:

```
public void processFile(String fName) throws FileNotFoundException,
IOException {
  FileInputStream fis;
  try {
    fis = new FileInputStream(fName);
    int sz;
    byte[] byteArray = new byte[BLOCK_SIZE];
    while ((sz = fis.read(byteArray)) != -1) {
      processBytes(byteArray, sz);
    }
  }
  finally {
    if (fis != null) {
      safeClose(fis);
    }
  }
}

public static void safeClose(FileInputStream fis) {
  if (fis != null) {
    try {
      fis.close();
    } catch (IOException e) {
      log(e);
    }
  }
}
```

This solution uses a helper function to log the exceptions that might occur when trying to close the stream. Presumably this helper function will be reused whenever a stream needs to be closed. Also, the `processFile` method does not initialize the `fis` object to `null`. Instead, it checks to ensure that `fis` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `fis` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `fis` is initialized to `null` in a more complex method, cases in which `fis` is used without being initialized will not be detected by the compiler.

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Streams	4	0	0	4
Total	**4**	**0**	**0**	**4**

Unreleased Resource: Streams — High

Package: .mvn.wrapper

.mvn/wrapper/MavenWrapperDownloader.java, line 112 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: fos.getChannel()
Enclosing Method: downloadFileFromURL()
File: .mvn/wrapper/MavenWrapperDownloader.java:112
Taint Flags:

```
109  ReadableByteChannel rbc;
110  rbc = Channels.newChannel(website.openStream());
111  FileOutputStream fos = new FileOutputStream(destination);
112  fos.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);
113  fos.close();
114  rbc.close();
115  }
```

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/main/java/org/owasp/webgoat/lessons/clientsidefiltering/Salaries.java, line 67 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unreleased Resource: Streams — High

Package: org.owasp.webgoat.lessons.clientsidefiltering

src/main/java/org/owasp/webgoat/lessons/clientsidefiltering/Salaries.java, line 67 (Unreleased Resource: Streams) — High

Sink: new FileOutputStream(...)
Enclosing Method: copyFiles()
File: src/main/java/org/owasp/webgoat/lessons/clientsidefiltering/Salaries.java:67
Taint Flags:

```
64  try {
65  FileCopyUtils.copy(
66  classPathResource.getInputStream(),
67  new FileOutputStream(new File(targetDirectory, "employees.xml")));
68  } catch (IOException e) {
69  throw new RuntimeException(e);
70  }
```

Package: org.owasp.webgoat.lessons.pathtraversal

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadBase.java, line 119 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: inputStream = getResourceAsStream(...)
Enclosing Method: defaultImage()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadBase.java:119
Taint Flags:

```
116
117  @SneakyThrows
118  protected byte[] defaultImage() {
119  var inputStream = getClass().getResourceAsStream("/images/account.png");
120  return Base64.getEncoder().encode(FileCopyUtils.copyToByteArray(inputStream));
121  }
122 }
```

src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 57 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Unreleased Resource: Streams	High
Package: org.owasp.webgoat.lessons.pathtraversal	
src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java, line 57 (Unreleased Resource: Streams)	High

Sink: new FileOutputStream(...)
Enclosing Method: initAssignment()
File: src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java:57
Taint Flags:

```
54  try (InputStream is =
55  new ClassPathResource("lessons/pathtraversal/images/cats/" + i + ".jpg")
56  .getInputStream()) {
57  FileCopyUtils.copy(is, new FileOutputStream(new File(catPicturesDirectory, i + ".jpg")));
58  } catch (Exception e) {
59  log.error("Unable to copy pictures" + e.getMessage());
60  }
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Weak Cryptographic Hash (1 issue)

Abstract

Weak cryptographic hashes cannot guarantee data integrity and should not be used in security-critical contexts.

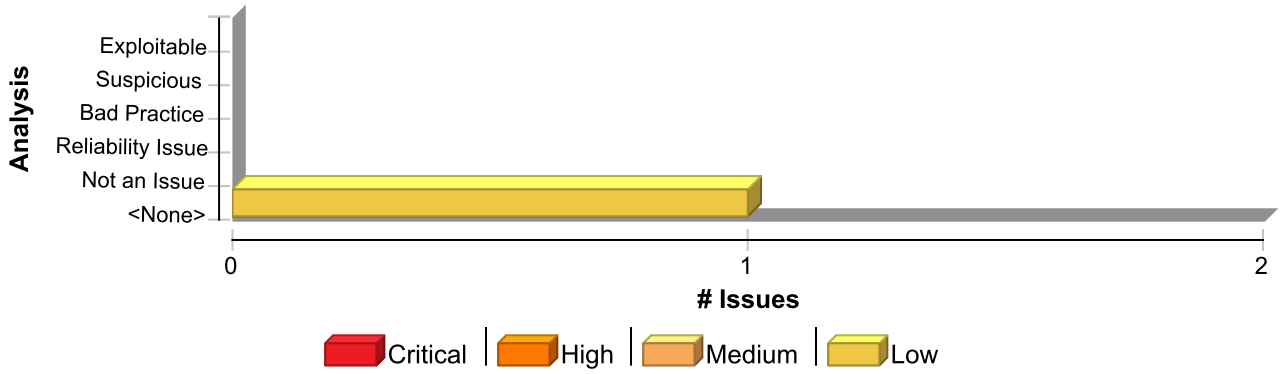
Explanation

MD2, MD4, MD5, RIPEMD-160, and SHA-1 are popular cryptographic hash algorithms often used to verify the integrity of messages and other data. However, as recent cryptanalysis research has revealed fundamental weaknesses in these algorithms, they should no longer be used within security-critical contexts. Effective techniques for breaking MD and RIPEMD hashes are widely available, so those algorithms should not be relied upon for security. In the case of SHA-1, current techniques still require a significant amount of computational power and are more difficult to implement. However, attackers have found the Achilles' heel for the algorithm, and techniques for breaking it will likely lead to the discovery of even faster attacks.

Recommendation

Discontinue the use of MD2, MD4, MD5, RIPEMD-160, and SHA-1 for data-verification in security-critical contexts. Currently, SHA-224, SHA-256, SHA-384, SHA-512, and SHA-3 are good alternatives. However, these variants of the Secure Hash Algorithm have not been scrutinized as closely as SHA-1, so be mindful of future research that might impact the security of these algorithms.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Weak Cryptographic Hash	1	0	0	1
Total	**1**	**0**	**0**	**1**

Weak Cryptographic Hash	**Low**
Package: org.owasp.webgoat.lessons.cryptography	
src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java, line 55 (Weak Cryptographic Hash)	**Low**

Issue Details

© Copyright [2008-2024] Micro Focus or one of its affiliates.

Weak Cryptographic Hash	Low
Package: org.owasp.webgoat.lessons.cryptography	
src/main/java/org/owasp/webgoat/lessons/cryptography/ HashingAssignment.java, line 55 (Weak Cryptographic Hash)	**Low**

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: getInstance()
Enclosing Method: getMd5()
File: src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java:55
Taint Flags:

```
52
53  String secret = SECRETS[new Random().nextInt(SECRETS.length)];
54
55  MessageDigest md = MessageDigest.getInstance("MD5");
56  md.update(secret.getBytes());
57  byte[] digest = md.digest();
58  md5Hash = DatatypeConverter.printHexBinary(digest).toUpperCase();
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

XML Entity Expansion Injection (1 issue)

Abstract

Using XML parsers configured to not prevent nor limit Document Type Definition (DTD) entity resolution can expose the parser to an XML Entity Expansion injection

Explanation

XML Entity Expansion injection also known as XML Bombs are Denial Of Service (DoS) attacks that benefit from valid and well-formed XML blocks that expand exponentially until they exhaust the server allocated resources. XML allows to define custom entities which act as string substitution macros. By nesting recurrent entity resolutions, an attacker may easily crash the server resources. The following XML document shows an example of an XML Bomb.

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

This test could crash the server by expanding the small XML document into more than 3GB in memory.

Recommendation

An XML parser should be configured securely so that it does not allow document type definition (DTD) custom entities as part of an incoming XML document. To avoid XML Entity Expansion injection the "secure-processing" property should be set for an XML factory, parser or reader:

```
factory.setFeature("http://javax.xml.XMLConstants/feature/secure-processing",
true);
```

In JAXP 1.3 and earlier versions, when the secure processing feature is on, default limitations are set for DOM and SAX parsers. These limits are:

```
entityExpansionLimit = 64,000
elementAttributeLimit = 10,000
```

Since JAXP 1.4, the secure processing feature is turned on by default. In addition to the preceding limits, a new maxOccur limit is added to the validating parser. The limit is:

```
maxOccur = 5,000
```

If inline DOCTYPE declaration is not needed, it can be completely disabled with the following property:

```
factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl",
true);
```

For StAX parsers the following configurations are recommended: - Disable entity reference resolution:

```
xmlInputFactory.setProperty("javax.xml.stream.isReplacingEntityReferences",
false);
```

- If inline DOCTYPE declaration is not needed, it can be completely disabled with the following property:

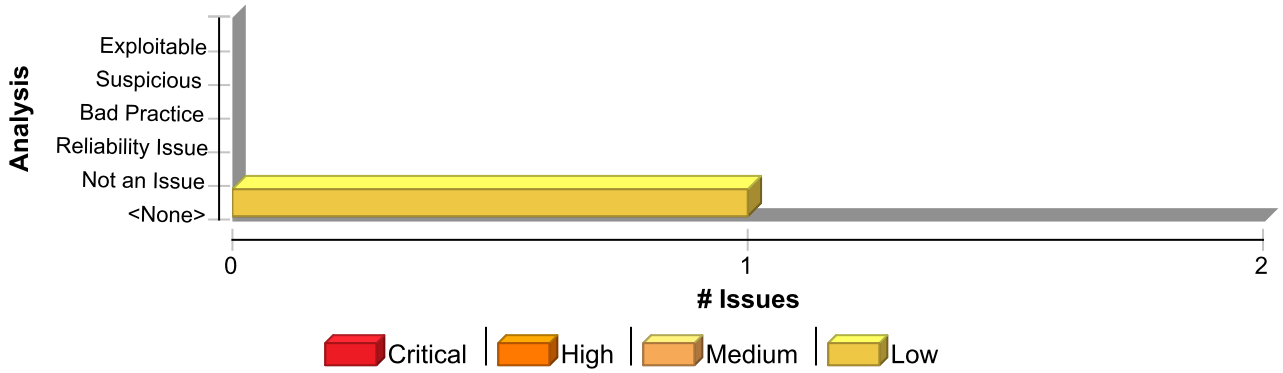
```
xmlInputFactory.setProperty("javax.xml.stream.supportDTD", false);
```

- If external or local entity resolution is required, build a secure resolver and set it up before parsing the XML document:

```
xmlInputFactory.setXMLResolver(mySafeResolver);
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
XML Entity Expansion Injection	1	0	0	1
Total	**1**	**0**	**0**	**1**

XML Entity Expansion Injection — Low

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java, line 105 (XML Entity Expansion Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Control Flow)

Sink Details

Sink: xif.createXMLStreamReader(...) : XML document parsed allowing external entity resolution
Enclosing Method: parseXml()
File: src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java:105
Taint Flags:

```
102  xif.setProperty(XMLConstants.ACCESS_EXTERNAL_SCHEMA, ""); // compliant
103  }
104
105  var xsr = xif.createXMLStreamReader(new StringReader(xml));
106
107  var unmarshaller = jc.createUnmarshaller();
108  return (Comment) unmarshaller.unmarshal(xsr);
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

XML External Entity Injection (1 issue)

Abstract

Using XML parsers configured to not prevent nor limit external entities resolution can expose the parser to an XML External Entities attack

Explanation

XML External Entities attacks benefit from an XML feature to build documents dynamically at the time of processing. An XML entity allows inclusion of data dynamically from a given resource. External entities allow an XML document to include data from an external URI. Unless configured to do otherwise, external entities force the XML parser to access the resource specified by the URI, e.g., a file on the local machine or on a remote system. This behavior exposes the application to XML External Entity (XXE) attacks, which can be used to perform denial of service of the local system, gain unauthorized access to files on the local machine, scan remote machines, and perform denial of service of remote systems. The following XML document shows an example of an XXE attack.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///dev/random" >]><foo>&xxe;</foo>
```

This example could crash the server (on a UNIX system), if the XML parser attempts to substitute the entity with the contents of the /dev/random file.

Recommendation

An XML parser should be configured securely so that it does not allow external entities as part of an incoming XML document. To avoid XXE injections the following properties should be set for an XML factory, parser or reader:

```
factory.setFeature("http://xml.org/sax/features/external-general-entities",
false);
factory.setFeature("http://xml.org/sax/features/external-parameter-entities",
false);
```

If inline DOCTYPE declaration is not needed, it can be completely disabled with the following property:

```
factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl",
true);
```

For StAX parsers the following configurations are recommended: - Disable external entity resolution:

```
xmlInputFactory.setProperty("javax.xml.stream.isSupportingExternalEntities",
false);
```

- If inline DOCTYPE declaration is not needed, it can be completely disabled with the following property:

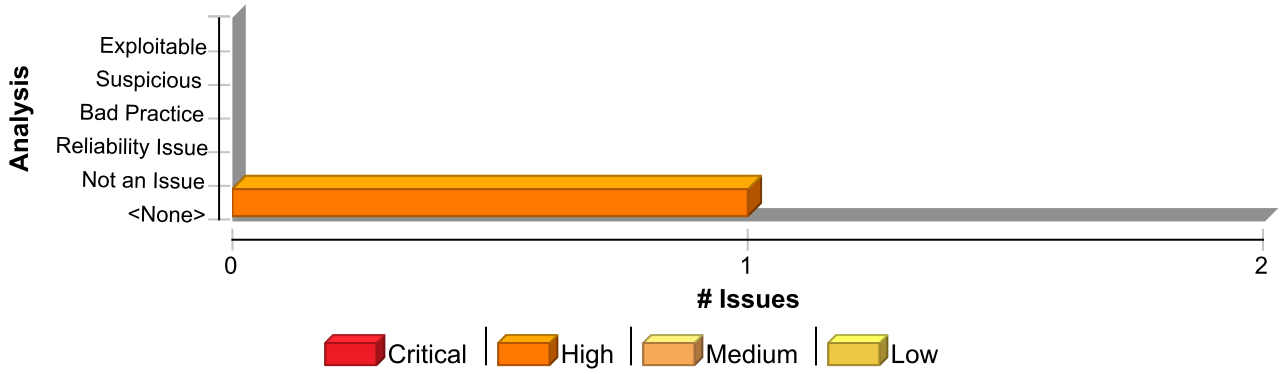
```
xmlInputFactory.setProperty("javax.xml.stream.supportDTD", false);
```

- If external or local entity resolution is required, build a secure resolver and set it up before parsing the XML document:

```
xmlInputFactory.setXMLResolver(mySafeResolver);
```

Issue Summary

© Copyright [2008-2024] Micro Focus or one of its affiliates.



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
XML External Entity Injection	1	0	0	1
Total	**1**	**0**	**0**	**1**

XML External Entity Injection — High

Package: org.owasp.webgoat.lessons.xxe

src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java, line 105 (XML External Entity Injection) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Control Flow)

Sink Details

Sink: xif.createXMLStreamReader(...) : XML document parsed allowing external entity resolution
Enclosing Method: parseXml()
File: src/main/java/org/owasp/webgoat/lessons/xxe/CommentsCache.java:105
Taint Flags:

```
102  xif.setProperty(XMLConstants.ACCESS_EXTERNAL_SCHEMA, ""); // compliant
103  }
104
105  var xsr = xif.createXMLStreamReader(new StringReader(xml));
106
107  var unmarshaller = jc.createUnmarshaller();
108  return (Comment) unmarshaller.unmarshal(xsr);
```


© Copyright [2008-2024] Micro Focus or one of its affiliates.

Description of Key Terminology

Likelihood and Impact

Likelihood
Likelihood is the probability that a vulnerability will be accurately identified and successfully exploited.

Impact
Impact is the potential damage an attacker could do to assets by successfully exploiting a vulnerability. This damage can be in the form of, but not limited to, financial loss, compliance violation, loss of brand reputation, and negative publicity.

Fortify Priority Order

Critical
Critical-priority issues have high impact and high likelihood. Critical-priority issues are easy to detect and exploit and result in large asset damage. These issues represent the highest security risk to the application. As such, they should be remediated immediately.

SQL Injection is an example of a critical issue.

High
High-priority issues have high impact and low likelihood. High-priority issues are often difficult to detect and exploit, but can result in large asset damage. These issues represent a high security risk to the application. High-priority issues should be remediated in the next scheduled patch release.

Password Management: Hardcoded Password is an example of a high issue.

Medium
Medium-priority issues have low impact and high likelihood. Medium-priority issues are easy to detect and exploit, but typically result in small asset damage. These issues represent a moderate security risk to the application. Medium-priority issues should be remediated in the next scheduled product update.

Path Manipulation is an example of a medium issue.

Low
Low-priority issues have low impact and low likelihood. Low-priority issues can be difficult to detect and exploit and typically result in small asset damage. These issues represent a minor security risk to the application. Low-priority issues should be remediated as time allows.

Dead Code is an example of a low issue.

© Copyright [2008-2024] Micro Focus or one of its affiliates.

About Fortify Solutions

Fortify is the leader in end-to-end application security solutions with the flexibility of testing on-premise and on-demand to cover the entire software development lifecycle. Learn more at www.microfocus.com/solutions/application-security.

© Copyright [2008-2024] Micro Focus or one of its affiliates.