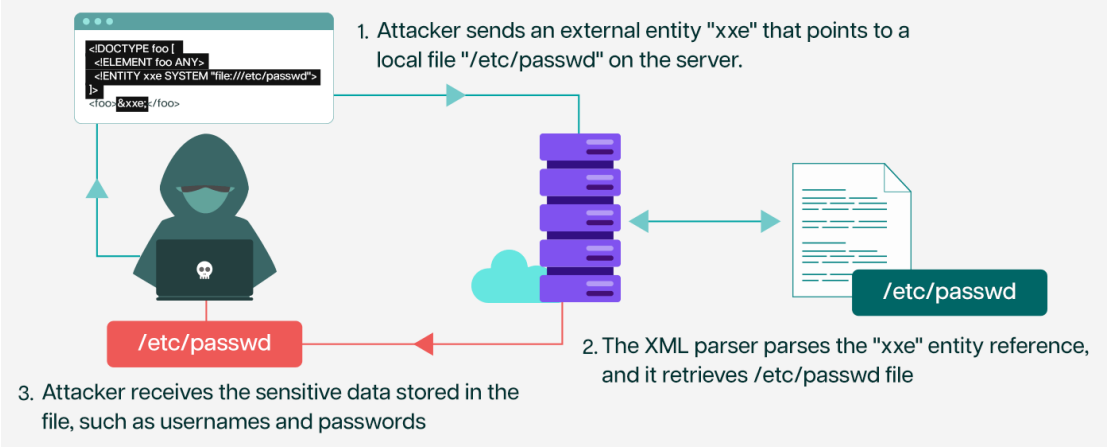
XML external entity (XXE) injection Attack

Karthik B

What is XXE Vulnerability?

XML External Entity Injection (XXE) is a security vulnerability that empowers a malicious actor to inject unsafe XML entities into a web application designed to process XML data. Successfully exploiting XXE vulnerabilities enables threat actors to interact with systems accessible to the application, view server files, and, in certain instances, execute remote code.

XXE vulnerabilities often arise from outdated or improperly configured XML parsers. Theoretically, prevention is straightforward by configuring XML parser settings to disallow custom document type definitions (DTD). However, in practice, web applications comprise numerous components, each potentially employing its own XML parser. Determining which parts of the application process XML can be challenging, and, in some cases, application owners lack access to configure the XML parser used by specific components. This complexity contributes to the persistence of XXE vulnerabilities in real-world scenarios.



How do XXE Injection occur?

Certain applications utilize the XML format for data transmission between the browser and the server. In nearly all cases, applications adopting this approach rely on a standard library or platform API to handle XML data processing on the server. The emergence of XXE vulnerabilities stems from the fact that the XML specification

incorporates potentially risky features, and standard parsers support these features even if they are not typically utilized by the application.

XML external entities represent a category of custom XML entities whose defined values are sourced from outside the DTD in which they are declared. From a security standpoint, external entities are of particular concern because they enable the definition of an entity based on the contents of a file path or URL.

What are the different types of attacks you can perform through XXE?

There are various types of XML external entity attacks:

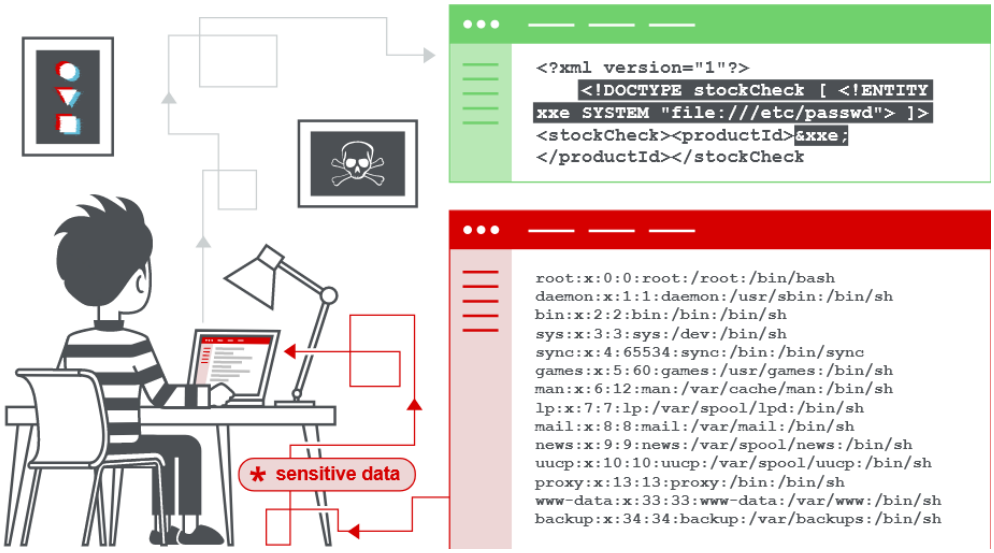
1. **XXE Exploit to Retrieve Files:** An attacker can use an XXE attack to fetch an arbitrary file from the target server's filesystem by manipulating the submitted XML. This involves introducing a DOCTYPE element that defines an external entity containing the file path. The attacker then alters the XML data value in the response to achieve file retrieval.
2. **XXE Exploit to Perform SSRF:** XXE attacks can be utilized for server-side request forgery (SSRF), prompting the application to make requests to malicious URLs. This attack includes defining an external entity with the target URL and incorporating the entity in the response's data value. It allows the attacker to observe responses from the specified URL in an application's response, facilitating interaction with the back end. In some cases, the attacker may perform a blind SSRF attack without directly viewing the response.
3. **Blind XXE Exploit to Exfiltrate Data:** XXE vulnerabilities are often blind, indicating that the application doesn't reveal external entity values. Attackers, however, can still exploit blind XXE vulnerabilities using advanced techniques such as out-of-band data exfiltration or triggering an XML parsing error to disclose sensitive information.
4. **Blind XXE Exploit to Generate Error Messages:** Another method to exploit blind XXE vulnerabilities is by triggering parsing errors to generate error messages containing sensitive data. This is effective for applications that include error messages

in their responses. Attackers can perform a blind XXE exploit by using malicious external Document Type Definitions (DTDs) to trigger an error message that discloses contents, such as the password file.

A malicious DTD can define a "file" XML parameter entity with the password file contents. It also defines an evaluation entity, allowing an evaluation using a decoy file. Triggering the evaluation entity attempts to load a nonexistent file, resulting in an error message that reveals the nonexistent decoy file's name.

How to test for XXE Injection Vulnerability?

The majority of XXE vulnerabilities can be promptly and reliably identified through the web vulnerability scanner in Burp Suite.



Manual testing for XXE vulnerabilities typically involves:

- Verifying file retrieval by creating an external entity based on a well-known operating system file and utilizing this entity in data returned within the application's response.
- Examining blind XXE vulnerabilities through the creation of an external entity based on a URL leading to a controlled system, while monitoring interactions

with that system. The use of Burp Collaborator is particularly suitable for this purpose.

- Assessing the potential inclusion of user-supplied non-XML data within a server-side XML document vulnerability by employing an XInclude attack to attempt retrieval of a well-known operating system file.

What are the impacts of XXE Injection Attack?

Here are typical outcomes of XXE attacks:

1. **Disclosing Local Files:** Threat actors can unveil files containing sensitive information, such as passwords, leveraging file: schemes or relative paths in the system identifier.
2. **Expanding the Attack:** XXE attacks capitalize on the application processing the XML document. Malicious actors can utilize this trusted application as a gateway to infiltrate different internal systems.
3. **Remote Code Execution:** In cases where the XML processor library is susceptible to client-side memory corruption, threat actors can dereference a malicious URI, enabling arbitrary code execution under the application account.
4. **Impact on Application Availability:** Certain XML attacks may grant actors access to local resources that continuously provide data. If numerous processes or threads are not released, this can adversely affect the availability of the application.

How to mitigate XXE Injection Vulnerability?

Almost universally, XXE vulnerabilities stem from the XML parsing library of the application supporting XML features that are potentially hazardous and unnecessary for the application's intended use. The most straightforward and impactful approach to thwarting XXE attacks is to deactivate these features.

Typically, disabling the resolution of external entities and turning off support for XInclude proves adequate for prevention. This can often be accomplished through configuration options or by programmatically overriding default behaviors.

References:

- https://portswigger.net/web-security/xxe#how-do-xxe-vulnerabilities-arise
- https://www.hackerone.com/knowledge-center/xxe-complete-guide-impact-examples-and-prevention
- https://www.imperva.com/learn/application-security/xxe-xml-external-entity/