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

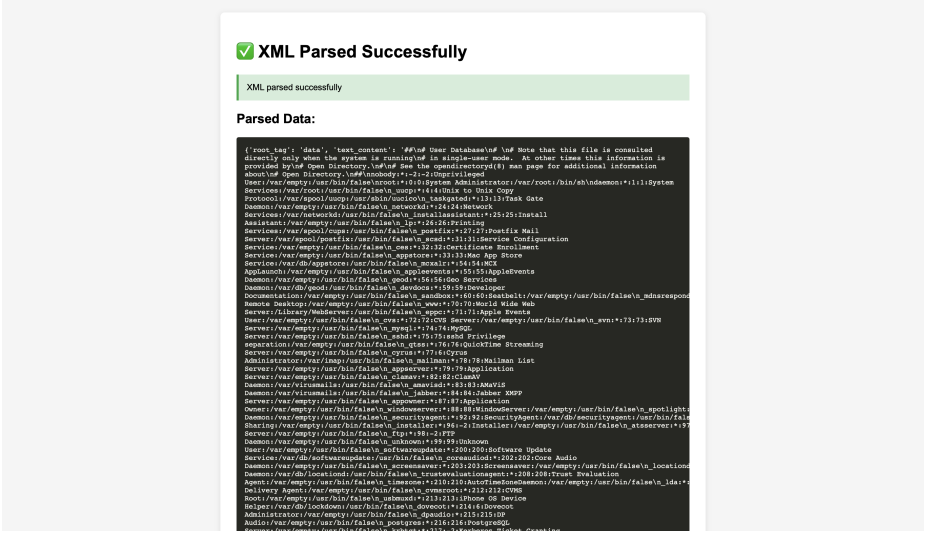
Example Payloads:

Normal XML:

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

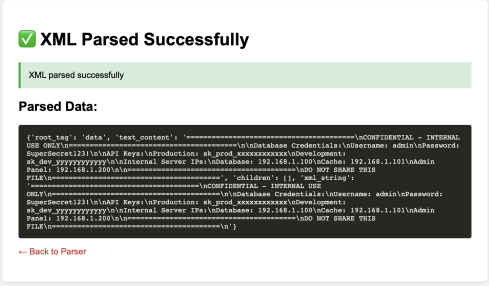
Attack 1: File Disclosure - System Files

Target: `/etc/passwd`



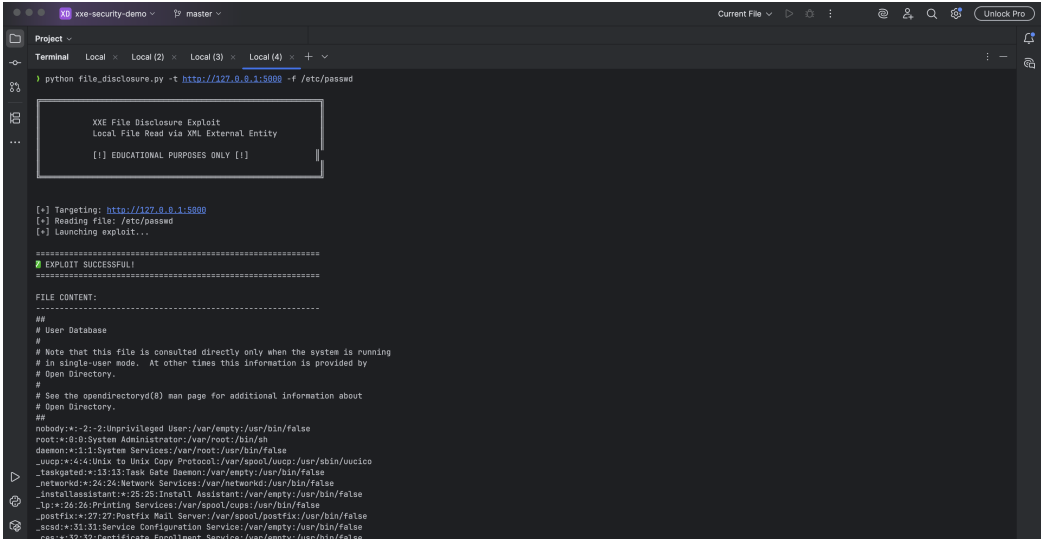
Attack 2: File Disclosure - Application Secrets

Target: `sensitive_data.txt`



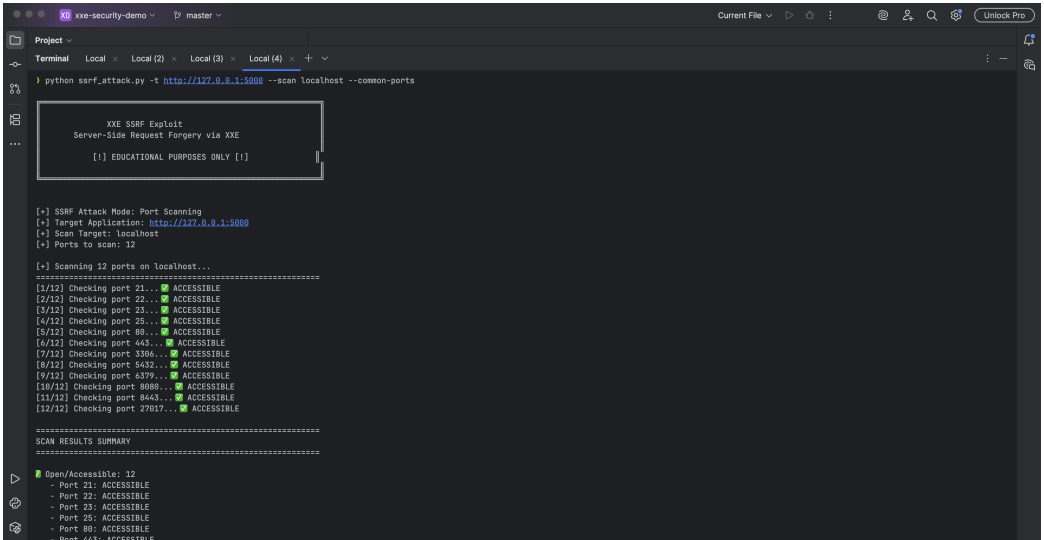
Automated Exploitation

Python Exploit Script: File Disclosure



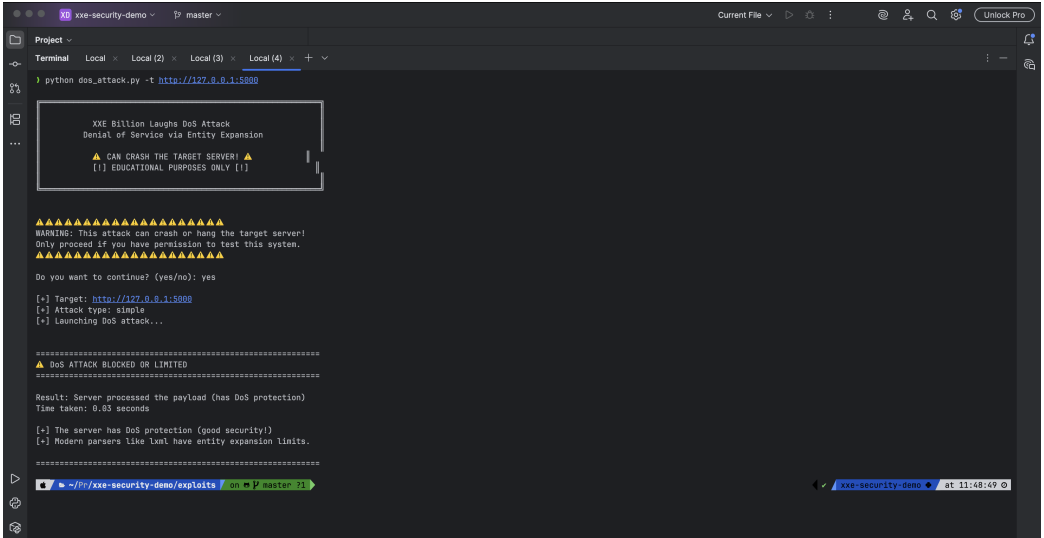
Attack 3: SSRF - Port Scanning

Technique: Server-Side Request Forgery via XXE



Attack 4: Denial of Service

Technique: Billion Laughs / XML Bomb



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

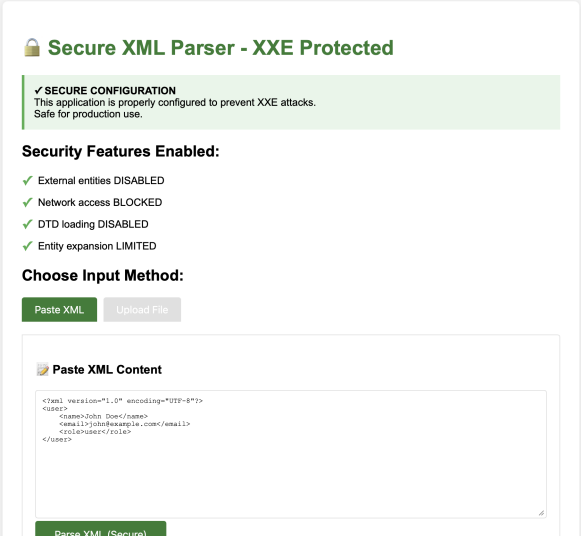
Company	Year	Impact
Facebook	2013	RCE, $33.5k bounty
Google	2012	Server file access
Android Tools	2017	Millions of developers
SharePoint	2019	APT exploitation

Table: Notable XXE vulnerabilities in major platforms

Key Finding

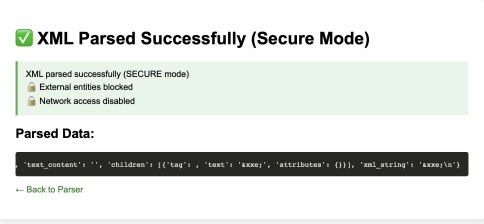
XXE vulnerabilities affect even the most security-conscious organizations

Secure Application



XXE Attack Blocked

Same payload, different result:



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

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