

# NETREBELS

## CYBERSECURITY PENETRATION TEST REPORT

### Black Box 2



## EXECUTIVE SUMMARY

Il presente Report documenta l'attività di Penetration Testing sulla macchina vulnerabile "Empire Lupin One", focalizzata sull'analisi di configurazioni errate e l'esposizione di servizi critici. La superficie d'attacco iniziale è stata identificata attraverso scansioni di rete e servizi, rivelando un server web Apache e un servizio SSH attivi. La presenza di un file `robots.txt` con directory disabilitate indica una potenziale esposizione di informazioni sensibili, classificabile sotto la categoria **A01:2021-Broken Access Control** della OWASP Top 10. L'obiettivo dell'analisi è dimostrare come un'enumerazione metodica possa portare alla compromissione del sistema partendo da vettori web apparentemente innocui.

## REQUISITI E AMBIENTE DI LABORATORIO

- **Attaccante (Kali Linux):** 192.168.64.16/24 (Eseguito su Mac M1 tramite UTM).
  - **Vittima (Empire Lupin One):** 192.168.64.20/24.
- 

## FASE 1: DISCOVERY & ENUMERATION

In questa fase abbiamo "interrogato" la rete per identificare il target e mappare i servizi esposti, gettando le basi per l'analisi delle vulnerabilità.

```
(kali㉿kali)-[~]
$ sudo netdiscover -r 192.168.64.16/24
```

### Identificazione del Target (netdiscover)

**Spiegazione Tecnica:** Il comando `netdiscover` è un tool di ricognizione di rete che utilizza il protocollo **ARP (Address Resolution Protocol)**. L'ARP è il protocollo incaricato di convertire gli indirizzi IP (livello logico) in indirizzi fisici MAC. Il comando invia richieste ARP "Who has" all'interno del segmento di rete specificato e resta in ascolto delle risposte (ARP Replies) per mappare gli indirizzi IP attivi e i relativi **MAC Address**.

Il **MAC Address (Media Access Control)** è un identificativo univoco di 48 bit assegnato dal produttore a ogni scheda di rete (NIC); rappresenta l'indirizzo "fisico" e immutabile dell'hardware, a differenza dell'IP che è dinamico. Questo tool è particolarmente efficace in reti locali poiché non si affida all'instradamento IP ma opera al **Livello 2 (Data Link Layer)** del **modello OSI**. Questo

livello si occupa del trasferimento affidabile dei dati attraverso il supporto fisico, gestendo l'indirizzamento fisico (MAC) e la rilevazione degli errori di frame, permettendoci di vedere dispositivi che potrebbero ignorare i classici ping (ICMP).

**Spiegazione Semplice:** Immagina di entrare in una stanza buia piena di persone e di gridare: "C'è qualcuno?". Ogni persona che ti risponde ti dice dove si trova. `netdiscover` fa esattamente questo con i computer in una rete: invia un messaggio a tutti e segna chi risponde, permettendoci di trovare l'indirizzo "di casa" (l'IP) della nostra vittima, che nel nostro caso è **192.168.64.20**.

```
Currently scanning: Finished! | Screen View: Unique Hosts
2 Captured ARP Req/Rep packets, from 2 hosts. Total size: 102
IP          At MAC Address      Count      Len  MAC Vendor / Hostname
192.168.64.1 12:bd:3a:a6:00:64      1        42  Unknown vendor
192.168.64.20 72:e4:42:27:ff:05      1        60  Unknown vendor
```

## Scansione dei Servizi (nmap)

**Nmap (Network Mapper)** è il tool standard de facto per l'esplorazione di rete e l'auditing della sicurezza. Viene utilizzato per scoprire host e servizi su una rete informatica, creando una "mappa" della superficie d'attacco.

**Spiegazione Tecnica:** Abbiamo eseguito il comando `nmap -sV -sC -p- 192.168.64.20`.

- **-sV (Service Versioning):** Analizza i banner dei servizi per identificare le versioni esatte dei software (es. Apache 2.4.48).
- **-sC (Default Scripts):** Esegue una serie di script automatici **NSE (Nmap Scripting Engine)**. Gli NSE sono potenti estensioni scritte in linguaggio Lua che permettono a Nmap di andare oltre la semplice scansione, automatizzando compiti come il rilevamento di vulnerabilità note, brute forcing leggero e l'estrazione di metadati dai servizi.
- **-p-:** Scansiona tutte le 65.535 **porte TCP**. Le porte TCP sono "punti terminali" logici della comunicazione; immagina che l'IP sia l'indirizzo di un palazzo e le porte siano i singoli appartamenti numerati dove risiedono servizi specifici. Scansionarle tutte assicura che non ci siano servizi "nascosti" su porte non standard.

Dallo scan emerge la **porta 80 (HTTP)** aperta. La porta 80 è il canale standard utilizzato dai server web per trasmettere contenuti non criptati (HyperText Transfer Protocol). È stata rilevata anche la presenza di una **directory** (ovvero una cartella nel file system del server web) denominata `/~myfiles`, segnalata nel file `robots.txt`.

**Spiegazione Semplice:** Se `netdiscover` ci ha indicato dove si trova la casa, `nmap` è come un sopralluogo per vedere quante porte e finestre ci sono e se sono chiuse a chiave. Abbiamo scoperto che la "porta principale" (Porta 80 - Sito Web) è aperta. Inoltre, abbiamo sbirciato sotto lo zerbino e trovato un biglietto (`robots.txt`) che ci dice che c'è una stanza segreta (una cartella) chiamata `~myfiles`.

```
(kali㉿kali)-[~]
$ sudo nmap -sV -sC -p- 192.168.64.20
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-30 04:01 -0500
Nmap scan report for 192.168.64.20
Host is up (0.00060s 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)
80/tcp    open  http    Apache httpd 2.4.48 ((Debian))
|_http-server-header: Apache/2.4.48 (Debian)
| http-robots.txt: 1 disallowed entry
|_/_/~myfiles
|_http-title: Site doesn't have a title (text/html).
MAC Address: 72:E4:42:27:FF:05 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.90 seconds
```

## FASE 2 WEB ENUMERATION

In questa fase, l'indagine si è spostata sull'analisi del server web (porta 80) per identificare percorsi nascosti o file di configurazione lasciati esposti.

### Analisi del file Robots.txt e Identificazione Directory

**Spiegazione Tecnica:** Il primo passo è stato visitare l'URL `http://192.168.64.20/robots.txt`. Il file `robots.txt` è uno standard utilizzato dai siti web per istruire i crawler (come Googlebot) su quali aree del sito non debbano essere scansionate o indicizzate. Nel nostro caso, il file ha rivelato una direttiva `Disallow: /~myfiles`, confermando l'esistenza di una directory denominata `~myfiles`. L'uso della tilde (~) suggerisce che si tratti di una cartella associata a uno specifico utente del sistema, una configurazione comune nei web server Apache (modulo `mod_userdir`).

**Spiegazione Semplice:** Siamo andati a leggere il file "istruzioni per i robot" del sito. È come trovare una mappa che dice esplicitamente: *"Per favore, non guardate dietro questa porta chiamata ~myfiles"*. Proprio perché ci è stato detto di non guardare, quella è diventata la nostra prima destinazione.

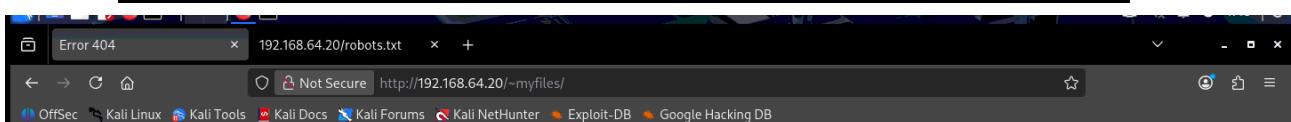
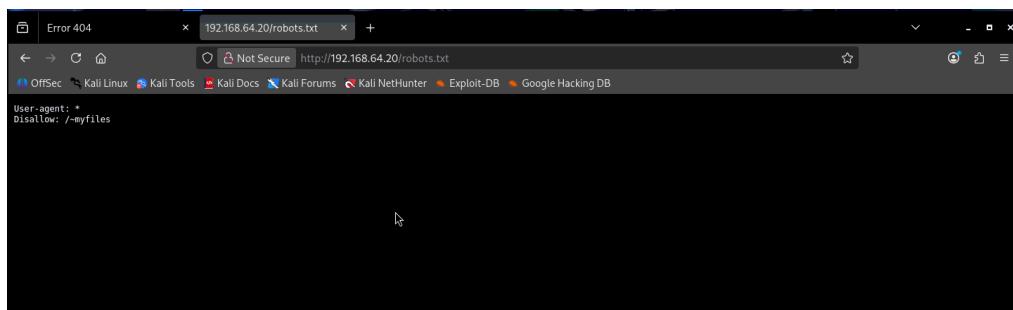
### Il fenomeno del "Falso 404" e Analisi del Codice Sorgente

**Spiegazione Tecnica:** Tentando di accedere direttamente a `http://192.168.64.20/~myfiles`, il server ha risposto con un errore 404 Not Found. Tuttavia, sospettando una configurazione ingannevole, abbiamo proceduto all'analisi del Codice Sorgente della pagina di errore.

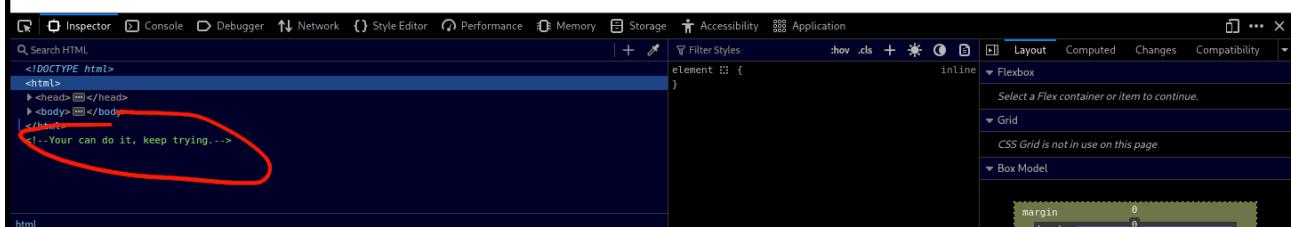
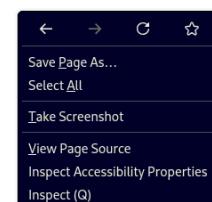
**Cos'è e come si analizza il Codice Sorgente:** Il codice sorgente HTML è il linguaggio di programmazione che descrive la struttura e il contenuto di una pagina web. Per analizzarlo, si utilizza la funzione "Ispeziona" (Inspect) o "Visualizza sorgente pagina", accessibile cliccando con il tasto destro del mouse in un punto qualsiasi della pagina e selezionando la voce corrispondente dal menu contestuale. Questo apre i "Developer Tools" del browser, permettendo di leggere i tag HTML, gli script e i commenti inseriti dagli sviluppatori.

**All'interno del codice è stato rinvenuto un commento nascosto: ``. La presenza di questo messaggio personalizzato conferma che l'errore 404 è un "Falso 404": la directory esiste realmente, ma il server è configurato per nasconderne il contenuto (Security through Obscurity).**

**Spiegazione Semplice:** Siamo andati davanti alla porta della stanza `~myfiles` e abbiamo trovato un cartello con scritto "Stanza Inesistente". Invece di andarcene, abbiamo "smontato" il cartello (facendo tasto destro e scegliendo "Ispeziona") per vedere cosa c'era scritto dietro. Abbiamo trovato un messaggio segreto degli sviluppatori che diceva: "Ce la puoi fare, continua a provare". Questo ci ha dato la conferma definitiva che la stanza esiste davvero e che dobbiamo solo trovare il modo giusto per entrarci.



## Error 404



---

## FASE 3 ANALISI STEGANOGRAFICA

In questa fase abbiamo analizzato gli elementi multimediali reperiti sul server per verificare la presenza di dati nascosti, una tecnica spesso utilizzata per occultare credenziali o messaggi cifrati.

### 3.1 Fallimento dell'analisi con Steghide

**Spiegazione Tecnica:** Sulla pagina principale è stata individuata l'immagine `arsene_lupin.jpg`. Un primo tentativo di analisi è stato eseguito con **Steghide**.

**Cos'è la Steganografia e Steghide:** La steganografia è la pratica di nascondere informazioni (testo, altri file) all'interno di un file apparentemente innocuo, come un'immagine o un audio, senza alterarne visibilmente l'aspetto. **Steghide** è un tool open source che permette di "embeddare" (inserire) o estrarre dati cifrati utilizzando una password.

Il comando utilizzato è stato `steghide extract -sf arsene_lupin.jpg`. Tuttavia, l'operazione è fallita con l'errore: *"the file format of the file is not supported"*.

**Spiegazione Semplice:** Abbiamo trovato una foto di Arsenio Lupin e abbiamo sospettato che potesse nascondere un messaggio segreto "dentro" i pixel, un po' come un inchiostro simpatico digitale. Abbiamo usato un software chiamato Steghide, che serve proprio a leggere questi messaggi invisibili, ma il programma ci ha risposto che non riusciva a capire il formato della foto.

### Analisi del File e Differenza tra JPG e PNG

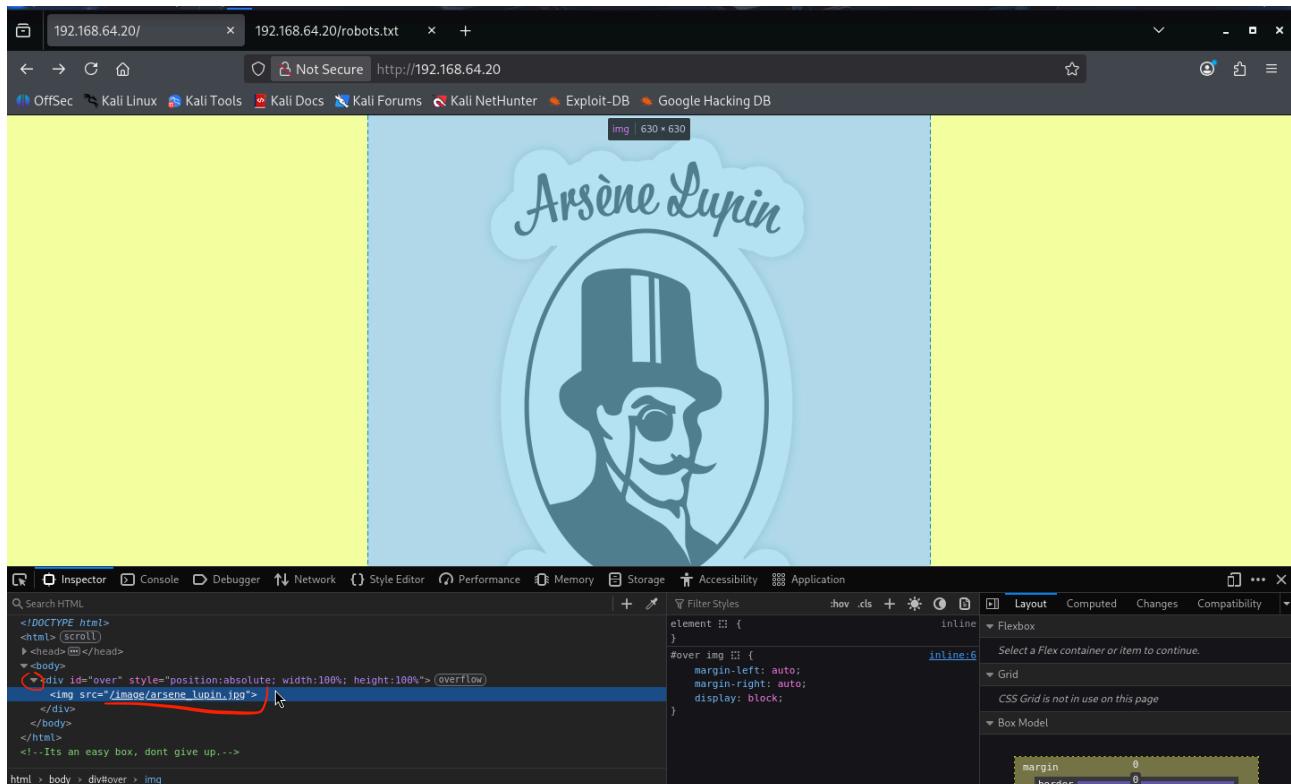
**Spiegazione Tecnica:** Per indagare il fallimento, abbiamo utilizzato il comando `file arsene_lupin.jpg`. Questo tool non legge l'estensione del file, ma analizza i **Magic Bytes** (l'intestazione esadecimale interna). L'analisi ha rivelato che il file, nonostante l'estensione `.jpg`, era in realtà un'immagine **PNG**.

#### Differenza tra JPG e PNG:

- **JPG (JPEG):** Utilizza una compressione "lossy" (con perdita di dati). È ottimo per le foto ma non preserva l'integrità perfetta dei singoli pixel, rendendo la steganografia più complessa e soggetta a corruzione.
- **PNG:** Utilizza una compressione "lossless" (senza perdita di dati). Questo formato preserva ogni singolo bit dell'immagine originale, rendendolo il candidato ideale per nascondere dati nei bit meno significativi (LSB - Least Significant Bit).

**Perché è importante:** Molti tool sono specifici per un formato. Steghide supporta JPG e BMP, ma non supporta i PNG. Rinominare un file non ne cambia la struttura interna; è come mettere l'etichetta "Acqua" su una bottiglia di vino: l'etichetta inganna l'occhio, ma il contenuto rimane lo stesso.

**Spiegazione Semplice:** Abbiamo scoperto che la foto ci stava "mentendo". Si chiamava **.jpg**, ma dentro era costruita come una **.png**. È come se qualcuno avesse preso una scatola di biscotti (JPG) e ci avesse messo dentro dei cioccolatini (PNG). Steghide sa aprire solo le scatole di biscotti, quindi si è bloccato. La differenza è importante perché le foto PNG sono molto più precise e permettono di nascondere segreti molto meglio rispetto alle JPG, che invece "rovinano" un po' i dati per occupare meno spazio.



## Utilizzo di Zsteg

### Acquisizione Manuale e Analisi del Formato

Durante la fase di acquisizione, l'impossibilità di scaricare il file tramite terminale ha richiesto un intervento manuale per garantire la continuità dell'assessment.

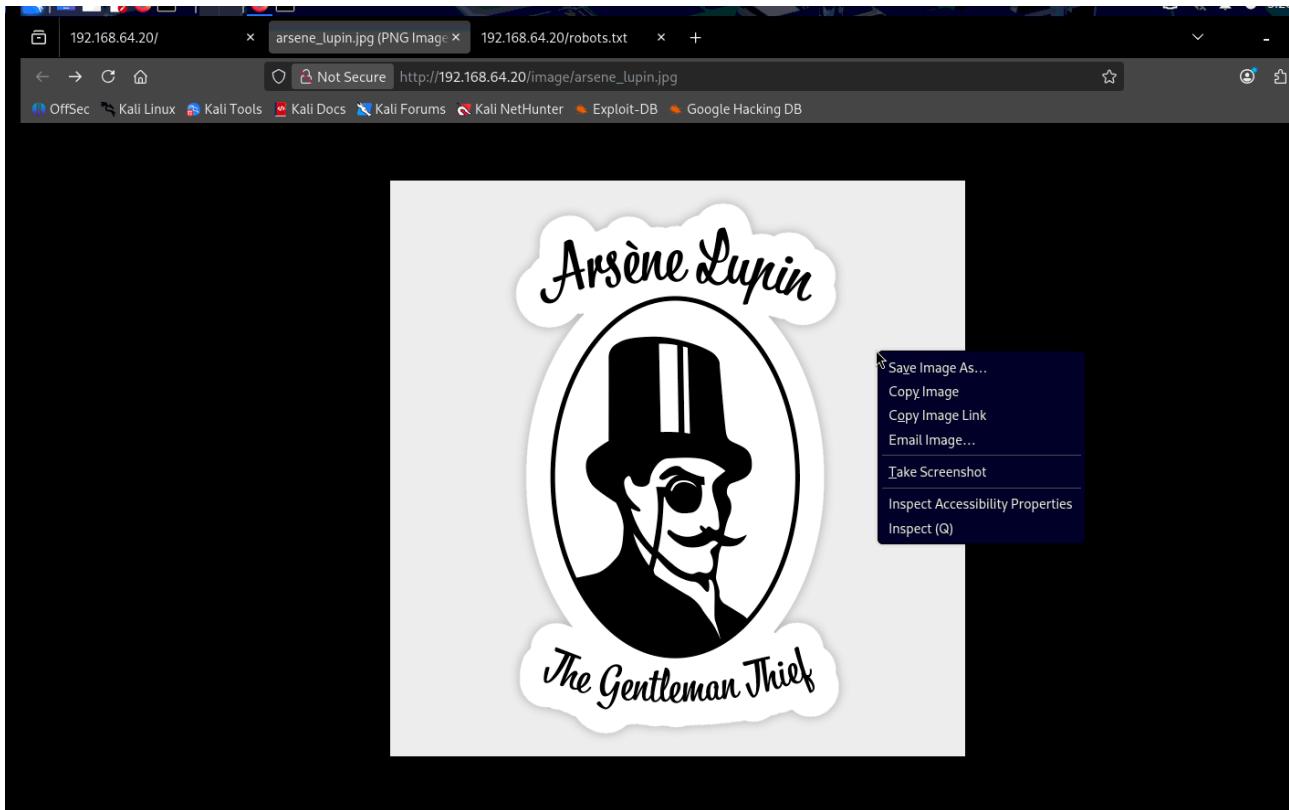
**Spiegazione Tecnica:** Dopo il riscontro di un errore di rete automatizzato, è stata eseguita un'acquisizione manuale della risorsa **arsene\_lupin.jpg** utilizzando l'interfaccia del browser. La procedura ha previsto l'apertura dell'immagine in una nuova scheda (**Open image in new tab**) per isolare la risorsa e il successivo salvataggio locale (**Save image as**). Una volta spostata la sessione operativa nella directory corretta (**cd Downloads**) e verificata la presenza del file (**ls**), è stata effettuata un'analisi dei **File Signature** (Magic Bytes). Questa ha rivelato che il file, nonostante l'estensione dichiarata, era strutturalmente un **PNG**. Di conseguenza, è stato eseguito il tool **zsteg**, specifico per questo formato, per l'estrazione di dati steganografici.

**Spiegazione Semplice:** Visto che il terminale non ne voleva sapere, abbiamo agito come farebbe un investigatore sul campo: abbiamo aperto la foto in una nuova finestra del browser e l'abbiamo salvata "a mano" nella cartella dei Download. Entrati nella cartella tramite riga di comando, ci

siamo accorti che la foto ci stava mentendo: portava il nome di un file `.jpg` ma era a tutti gli effetti un `.png`. Grazie a questa scoperta, abbiamo potuto usare lo strumento giusto (`zsteg`) per vedere cosa ci fosse nascosto dentro.

**Spiegazione Tecnica:** Identificata la reale natura del file (PNG), abbiamo utilizzato **zsteg**, un tool avanzato progettato specificamente per rilevare dati nascosti nei file PNG e BMP attraverso l'analisi dei vari livelli di bit (LSB, piani di colore, ecc.). Nonostante l'analisi approfondita (comando **zsteg -a arsene\_lupin.png**), l'output ha mostrato stringhe di testo casuali e rumore di fondo, senza fornire password o chiavi in chiaro.

**Spiegazione Semplice:** Abbiamo cambiato strumento e usato `zsteg`, che è come un microscopio potentissimo specifico per le immagini PNG. Abbiamo guardato dentro ogni singolo strato di colore della foto, ma purtroppo non abbiamo trovato nessuna password scritta chiaramente. Questo significa che il segreto non è in quella foto o, se c'è, è protetto in un modo che questo strumento non riesce a leggere.



## 3.2 PROBLEMI DI FUZZING E IL "FALSO 404"

In questa fase abbiamo cercato di forzare la serratura della directory `~myfiles` utilizzando tecniche di **Fuzzing**.

**Spiegazione Tecnica:** Il Fuzzing è una tecnica di **Black-Box Testing** che consiste nell'invio massivo di dati semi-strutturati (input) verso un target per osservarne la reazione. Nel contesto web, il fuzzing delle directory (come quello che stiamo facendo con `ffuf` o `gobuster`) utilizza una **wordlist** (un dizionario di termini comuni) per generare migliaia di richieste HTTP.

Se il server risponde con un codice **200 OK**, significa che la risorsa esiste; se risponde con **403 Forbidden**, la risorsa esiste ma non abbiamo i permessi; se risponde con **404 Not Found**, la parola non corrisponde a nessuna risorsa reale. Questa tecnica permette di mappare la struttura del server che non è visibile tramite i normali link della pagina.

**Spiegazione Semplice:** Immagina di trovarsi davanti a un palazzo con centinaia di porte senza targhetta. Il Fuzzing è come avere un robot velocissimo che corre per i corridoi e prova a girare ogni maniglia urlando nomi a caso: "*C'è la stanza 'Segreto'?*", "*C'è la stanza 'Password'?*", "*C'è la stanza 'Admin'?*".

Il robot non sa cosa c'è dentro, ma ogni volta che una porta si apre o emette un suono diverso, segna la posizione sulla mappa. In questo modo, riusciamo a scoprire stanze (cartelle e file) che il proprietario del palazzo voleva tenere nascoste.

### Gli Strumenti Utilizzati: Gobuster e Wfuzz

#### Spiegazione Tecnica:

- **Gobuster:** È un tool scritto in Go utilizzato per scoprire oggetti nascosti (directory, file, sottodomini) tramite attacchi basati su dizionario. È estremamente veloce grazie alla sua capacità di gestire molteplici connessioni simultanee (multi-threading).
- **Wfuzz:** È un framework di analisi web altamente flessibile. A differenza di Gobuster, permette di inserire il punto di "fuzzing" (il segnaposto FUZZ) in qualsiasi parte di una richiesta HTTP (parametri, header, cookie), rendendolo ideale per test complessi.

**Spiegazione Semplice:** Se dobbiamo indovinare il nome di un file segreto, non possiamo farlo a mano. Usiamo dei "robot" come **Gobuster** e **Wfuzz**. Immaginali come dei lettori ultra-veloci che provano migliaia di nomi al secondo da un elenco, bussando alla porta del server per vedere se qualcuno risponde.

#### Tentativo 1: Utilizzo della Wordlist "common.txt"

**Spiegazione Tecnica:** Abbiamo iniziato con la wordlist `common.txt`. Una **Wordlist** è un file di testo contenente migliaia di parole comuni utilizzate come nomi di file o cartelle (es: `admin`, `backup`, `test`).

- **Esito:** Nessun risultato. La lista era troppo piccola e non conteneva il termine specifico scelto dai creatori della macchina.

**Spiegazione Semplice:** Abbiamo provato a cercare la cartella segreta usando un dizionario di "parole comuni". È come cercare di aprire una cassaforte provando solo date di nascita semplici come "1234": se il proprietario è stato originale, non funzionerà mai.

```
(kali㉿kali)-[~/Downloads]
$ ffuf -u http://192.168.64.20/~myfiles/FUZZ -w /usr/share/wordlists/dirb/common.txt
[FFUF] v2.1.0-dev
[FFUF] :: Method      : GET
[FFUF] :: URL        : http://192.168.64.20/~myfiles/FUZZ
[FFUF] :: Wordlist   : FUZZ: /usr/share/wordlists/dirb/common.txt
[FFUF] :: Follow redirects : false
[FFUF] :: Calibration   : false
[FFUF] :: Timeout      : 10
[FFUF] :: Threads      : 40
[FFUF] :: Matcher      : Response status: 200-299,301,302,307,401,403,405,500
[FFUF] :: Progress: [4614/4614] :: Job [1/1] :: 840 req/sec :: Duration: [0:00:06] :: Errors: 0 ::

[FFUF] .hta           [Status: 200, Size: 147, Words: 11, Lines: 15, Duration: 247ms]
[FFUF] .htpasswd      [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 631ms]
[FFUF] .htaccess      [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 722ms]
[FFUF] index.html     [Status: 200, Size: 147, Words: 11, Lines: 15, Duration: 68ms]
[FFUF] :: Progress: [4614/4614] :: Job [1/1] :: 840 req/sec :: Duration: [0:00:06] :: Errors: 0 ::
```

**Spiegazione Tecnica:** L'attività di fuzzing tramite `ffuf` e la wordlist `common.txt` ha permesso di identificare la presenza di file di configurazione e indici nella directory `/~myfiles/`. In particolare, il riscontro di un codice **HTTP 200** per il file `.hta` e di codici **HTTP 403** (Forbidden) per file critici come `.htpasswd` indica che la directory è soggetta a criteri di controllo degli accessi. Il file `.htpasswd` è storicamente utilizzato per memorizzare credenziali di autenticazione (username e hash delle password), confermando che l'obiettivo dell'attacco deve spostarsi verso l'ottenimento di tali credenziali.

**Spiegazione Semplice:** Il nostro robot ha finito di bussare a tutte le porte. Ha trovato una porta chiamata `.htaccess` che è aperta e altre chiamate `.htpasswd` che sono chiuse a chiave. Il fatto che esistano queste "porte chiuse" ci dice che lì dietro c'è probabilmente la cassaforte con le password che stiamo cercando. È come aver trovato una porta blindata in un corridoio: sappiamo che quello è il posto giusto dove insistere.

A screenshot of a web browser window. The address bar shows the URL 'http://192.168.64.20/~myfiles/.hta'. The browser displays an '403 Forbidden' error page with the text 'You don't have permission to access this resource.' and 'Apache/2.4.48 (Debian) Server at 192.168.64.20 Port 80'. The browser interface includes tabs for 'arsene\_lupin.jpg (PNG Image)', '403 Forbidden', and a '+' button. The navigation bar includes icons for back, forward, and search. Below the address bar is a navigation bar with links to 'OffSec', 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', and 'Google Hacking DB'. A star icon is also present in the top right corner.

## Tentativo 2: Wordlist "big.txt" e il problema dei Falsi Positivi

**Spiegazione Tecnica:** Siamo passati a `big.txt` (una lista molto più vasta), ma abbiamo riscontrato il problema del **Falso Positivo**. Normalmente, se un file esiste, il server risponde con **Codice 200 OK**; se non esiste, risponde con **404 Not Found**. Il server di Lupin One, invece, è configurato per rispondere sempre con `200 OK`, inviando però una pagina che graficamente dice "Error 404". Questo inganna i tool, facendogli credere che ogni singola parola della lista sia un file esistente.

**Spiegazione Semplice:** Abbiamo preso un dizionario gigante, ma qui il server ha iniziato a "prenderci in giro". Ogni volta che chiedevamo di un file (anche inesistente), il server rispondeva: "Sì, ce l'ho!", ma poi ci mostrava una pagina vuota con scritto "Scherzavo, non c'è". Questo ha mandato in confusione i nostri strumenti, che hanno iniziato a segnalare migliaia di file trovati che in realtà non esistevano.

```
(kali㉿kali)-[~/Downloads]
$ ffuf -u http://192.168.64.20/~myfiles/FUZZ -w /usr/share/wordlists/dirb/big.txt -fs 147
.
.
.
v2.1.0-dev
_____
:: Method          : GET
:: URL            : http://192.168.64.20/~myfiles/FUZZ
:: Wordlist        : FUZZ: /usr/share/wordlists/dirb/big.txt
:: Follow redirects: false
:: Calibration    : false
:: Timeout         : 10
:: Threads         : 40
:: Matcher         : Response status: 200-299,301,302,307,401,403,405,500
:: Filter          : Response size: 147
_____
.htpasswd          [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 62ms]
.htaccess          [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 64ms]
:: Progress: [20469/20469] :: Job [1/1] :: 579 req/sec :: Duration: [0:00:27] :: Errors: 0 ::
```

---

## 4. FASE 4: EXPLOITATION

In questa fase abbiamo utilizzato le informazioni raccolte per individuare il vero punto di ingresso al sistema.

### 4.1 Discovery della Directory Utente Nascosta

**Spiegazione Tecnica:** L'osservazione del pattern `~myfiles` nel file `robots.txt` ci ha indirizzati verso una vulnerabilità logica legata al modulo **Apache mod\_userdir**. Questo modulo permette agli utenti del sistema operativo di avere una propria directory web pubblica (solitamente localizzata in `/home/utente/public_html`).

In termini di sicurezza, questo è un rischio perché:

1. **Enumerazione Utenti:** Conferma i nomi degli utenti presenti nel sistema (es. se esiste `~secret`, esiste un utente chiamato `secret`).
2. **Configurazione Debole:** Spesso queste cartelle personali sono meno controllate dagli amministratori rispetto alla directory web principale (`/var/www/html`).

**Spiegazione Semplice:** Abbiamo notato che il nome della cartella iniziava con una tilde (`~`). Nei computer che usano Linux e il server web Apache, quel simbolo è come una firma: dice che quella cartella appartiene a un utente specifico del computer. Invece di cercare cartelle generiche come "Immagini" o "Dati", abbiamo capito che dovevamo cercare le "stanze private" degli utenti.

### 4.2 L'Attacco di Fuzzing Mirato

**Spiegazione Tecnica:** Per confermare l'ipotesi, abbiamo utilizzato **Wfuzz** con una strategia specifica. Invece di cercare file comuni, abbiamo inserito il segnaposto **FUZZ** immediatamente dopo la tilde, testando i nomi degli utenti più probabili.

**Analisi del Comando:** `wfuzz -c -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --hc 404 http://192.168.64.20/~FUZZ`

- **-c:** Colora l'output per distinguere visivamente i risultati.
- **-w [percorso]:** Utilizza la wordlist `directory-list-2.3-medium.txt`. È una lista molto più ampia e professionale rispetto a `common.txt`, basata su nomi di directory reali trovati sul web.
- **--hc 404:** (Hide Code) Istruisce il programma a nascondere tutte le risposte "404 Not Found". Questo pulisce l'output, mostrandoci solo ciò che esiste veramente.

- **-FUZZ:** Questa è la chiave dell'attacco. Wfuzz sostituirà ogni parola della lista dopo la tilde (es. `~admin`, `~user`, `~secret`).

**Spiegazione Semplice:** Abbiamo preso un dizionario molto più grande e professionale e abbiamo detto al nostro robot: "Prova a inserire ogni parola di questa lista subito dopo il simbolo della tilde (~)". Abbiamo anche aggiunto un filtro: "Non dirmi quando fallisci, dimmi solo quando trovi una porta che si apre davvero". Questo ci permette di ignorare il rumore e concentrarci solo sulle cartelle personali reali.

### 4.3 Risultato: La scoperta di `-secret`

**Spiegazione Tecnica:** L'attacco ha avuto successo, identificando la directory `http://192.168.64.20/-secret/`. Questo conferma non solo l'esistenza della cartella, ma anche che esiste un utente nel sistema che probabilmente ha privilegi o file sensibili necessari per la fase di accesso iniziale (Initial Access).

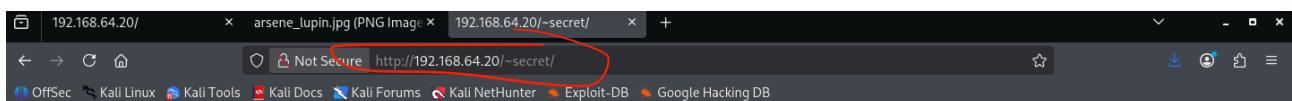
**Spiegazione Semplice:** Il nostro robot ha trovato quello che cercavamo! Esiste una cartella nascosta chiamata `-secret`. Questo è il nostro nuovo obiettivo: è come aver trovato la porta sul retro della casa che il proprietario pensava fosse invisibile.

```
(kali㉿kali)-[~]
└─$ wfuzz -c -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --hc 404 http://192.168.64.20/~FUZZ
/usr/lib/python3/dist-packages/wfuzz/_init_.py:34: UserWarning:Wfuzz is not compiled against OpenSSL. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
Target: http://192.168.64.20/~FUZZ
Total requests: 220560

ID      Response  Lines   Word    Chars  Payload
=====
000005155:  301      9 L    28 W   316 Ch  "-secret"

```

### CI DIRIGIAMO SUBITO NELLA PAGINA



Hello Friend, I'm happy that you found my secret directory. I created like this to share with you my create ssh private key file, Its hidde somewhere here, so that hackers dont find it and crack my passphrase with fasttrack.  
I'm smart I know that.  
Any problem let me know

Your best friend icex64

Questa fase rappresenta il cuore dell'attività di **Exploitation**. Dopo aver trovato la "stanza segreta", abbiamo dovuto cercare minuziosamente al suo interno per trovare la chiave d'accesso al sistema.

## 4.4 Recupero del Payload Cifrato

L'analisi della directory `~secret` ha rivelato la presenza di un utente denominato "**icex64**". Il messaggio lasciato dall'autore suggeriva che la chiave per procedere fosse nascosta proprio lì. In un sistema Linux, i file sensibili (come le chiavi SSH) sono spesso "nascosti".

### La Chiave Privata SSH: Cos'è?

**Spiegazione Tecnica:** Una chiave privata SSH fa parte di un sistema di crittografia asimmetrica. A differenza di una password tradizionale, è un file di testo contenente una lunga stringa di caratteri cifrati. Chi possiede la chiave privata può autenticarsi sul server (che possiede la chiave pubblica corrispondente) senza digitare una password. Se un attaccante sottrae una chiave privata, ottiene l'accesso diretto al sistema come se fosse l'utente legittimo.

**Spiegazione Semplice:** Immagina che la password sia una parola d'ordine che devi ricordare, mentre la chiave privata SSH sia una **chiave fisica magnetica** molto complessa. Se trovi la chiave che l'utente "icex64" ha lasciato per sbaglio nel suo cassetto, puoi entrare nel computer senza che nessuno ti chieda chi sei.

### Fuzzing Avanzato per File Nascosti

**Spiegazione Tecnica:** Per trovare il file, abbiamo dovuto eseguire un attacco di fuzzing più sofisticato. Nei sistemi Linux, i file che iniziano con un punto (.) sono considerati "**Hidden Files**" (file nascosti) e non vengono mostrati dai comandi standard come `ls`. Abbiamo quindi configurato `Wfuzz` per cercare file che iniziassero con il punto e avessero diverse estensioni possibili.

**Analisi del Comando:** `wfuzz -c -w /usr/share/wordlists/.../directory-list-2.3-medium.txt -z list,txt-key-rsa --hc 404,403 http://192.168.64.20/~secret/.FUZZ.FUZZ`

- **.FUZZ**: Il primo punto indica che stiamo cercando specificamente file nascosti (es. `.config`, `.backup`).
- **-z list,txt-key-rsa**: Questo parametro (payload secondario) dice a Wfuzz di provare diverse estensioni per ogni parola trovata, come `.txt`, `.key` o `.rsa`.
- **.FUZZ**: È il segnaposto per la seconda lista (le estensioni). Wfuzz proverà combinazioni come `.secret.txt`, `.secret.key`, ecc.
- **--hc 404,403**: Nasconde sia gli errori di "File non trovato" (404) sia quelli di "Accesso negato" (403), mostrandoci solo ciò che è realmente accessibile.

**Spiegazione Semplice:** Dato che i file importanti spesso sono invisibili a occhio nudo, abbiamo chiesto al nostro robot di fare un lavoro doppio: non solo doveva provare migliaia di nomi, ma per ognuno doveva aggiungere un punto all'inizio (per vedere i file nascosti) e provare diverse "estensioni" alla fine (come `.testo`, `.chiave`, `.segreto`). È come se stessimo setacciando la sabbia con un filtro finissimo per trovare un ago specifico.

## Risultato: Individuazione di **.mysecret.txt**

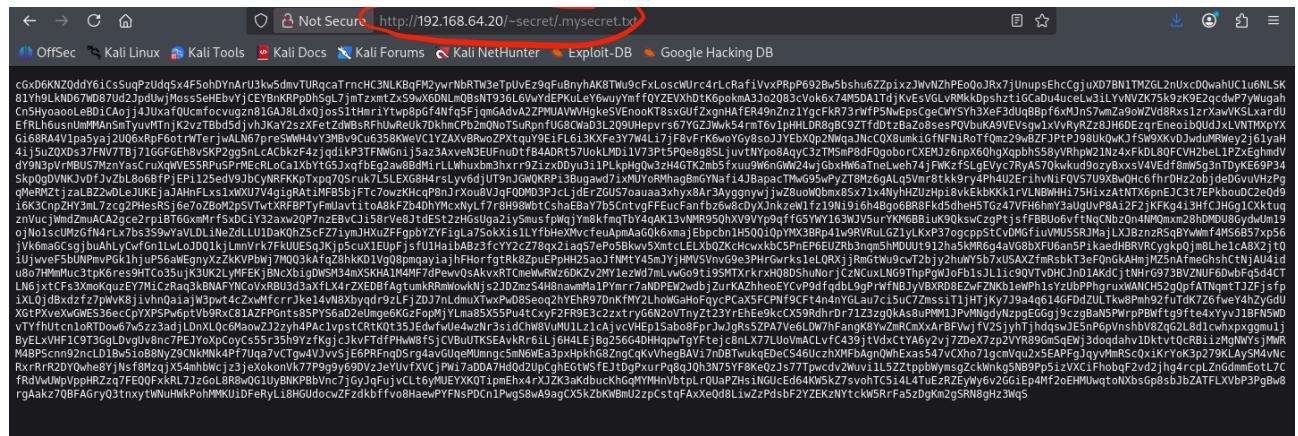
**Esito:** L'attacco ha avuto successo, individuando il file: **http://192.168.64.20/-secret/.mysecret.txt**.

Questo file non conteneva direttamente la chiave SSH, ma un **payload cifrato**. Questo è un ulteriore livello di difesa: anche se l'attaccante trova il file, deve ancora capire come decifrarne il contenuto.

| ID         | Response | Lines | Word | Chars   | Payload  |
|------------|----------|-------|------|---------|--|
| 000000001: | 200      | 5 L   | 54 W | 331 Ch  | "# directory-list-2.3-medium.txt - txt"  |
| 000000003: | 200      | 5 L   | 54 W | 331 Ch  | "# directory-list-2.3-medium.txt - rsa"  |
| 000000007: | 200      | 5 L   | 54 W | 331 Ch  | "# Copyright 2007 James Fisher - txt"  |
| 000000031: | 200      | 5 L   | 54 W | 331 Ch  | "# Priority ordered case sensitive list, where entries were found - txt"   |
| 000000015: | 200      | 5 L   | 54 W | 331 Ch  | "# This work is licensed under the Creative Commons - rsa"   |
| 000000038: | 200      | 5 L   | 54 W | 331 Ch  | "# - txt"  |
| 000000037: | 200      | 5 L   | 54 W | 331 Ch  | "# - key"  |
| 000000036: | 200      | 5 L   | 54 W | 331 Ch  | "# on atleast 2 different hosts - rsa"   |
| 000000035: | 200      | 5 L   | 54 W | 331 Ch  | "# on atleast 2 different hosts - key"   |
| 000000033: | 200      | 5 L   | 54 W | 331 Ch  | "# Priority ordered case sensitive list, where entries were found - rsa"   |
| 000000032: | 200      | 5 L   | 54 W | 331 Ch  | "# Priority ordered case sensitive list, where entries were found - key"   |
| 000000030: | 200      | 5 L   | 54 W | 331 Ch  | "# - rsa"  |
| 000000029: | 200      | 5 L   | 54 W | 331 Ch  | "# - key"  |
| 000000039: | 200      | 5 L   | 54 W | 331 Ch  | "# - rsa"  |
| 000000028: | 200      | 5 L   | 54 W | 331 Ch  | "# - txt"  |
| 000000034: | 200      | 5 L   | 54 W | 331 Ch  | "# on atleast 2 different hosts - txt"   |
| 000000027: | 200      | 5 L   | 54 W | 331 Ch  | "# Suite 300, San Francisco, California, 94105, USA. - rsa"  |
| 000000025: | 200      | 5 L   | 54 W | 331 Ch  | "# Suite 300, San Francisco, California, 94105, USA. - txt"  |
| 000000024: | 200      | 5 L   | 54 W | 331 Ch  | "# or send a letter to Creative Commons, 171 Second Street, - rsa"   |
| 000000019: | 200      | 5 L   | 54 W | 331 Ch  | "# license, visit <a href="http://creativecommons.org/licenses/by-sa/3.0/">http://creativecommons.org/licenses/by-sa/3.0/</a> - txt" |
| 000000021: | 200      | 5 L   | 54 W | 331 Ch  | "# license, visit <a href="http://creativecommons.org/licenses/by-sa/3.0/">http://creativecommons.org/licenses/by-sa/3.0/</a> - rsa" |
| 000000023: | 200      | 5 L   | 54 W | 331 Ch  | "# or send a letter to Creative Commons, 171 Second Street, - key"   |
| 000000022: | 200      | 5 L   | 54 W | 331 Ch  | "# or send a letter to Creative Commons, 171 Second Street, - txt"   |
| 000000017: | 200      | 5 L   | 54 W | 331 Ch  | "# Attribution-Share Alike 3.0 license. To view a copy of this - key"  |
| 000000018: | 200      | 5 L   | 54 W | 331 Ch  | "# Attribution-Share Alike 3.0 license. To view a copy of this - rsa"  |
| 000000020: | 200      | 5 L   | 54 W | 331 Ch  | "# license, visit <a href="http://creativecommons.org/licenses/by-sa/3.0/">http://creativecommons.org/licenses/by-sa/3.0/</a> - key" |
| 000000026: | 200      | 5 L   | 54 W | 331 Ch  | "# Suite 300, San Francisco, California, 94105, USA. - key"  |
| 000000014: | 200      | 5 L   | 54 W | 331 Ch  | "# This work is licensed under the Creative Commons - key"   |
| 000000016: | 200      | 5 L   | 54 W | 331 Ch  | "# Attribution-Share Alike 3.0 license. To view a copy of this - txt"  |
| 000000010: | 200      | 5 L   | 54 W | 331 Ch  | "# - txt"  |
| 000000008: | 200      | 5 L   | 54 W | 331 Ch  | "# Copyright 2007 James Fisher - key"  |
| 000000002: | 200      | 5 L   | 54 W | 331 Ch  | "# directory-list-2.3-medium.txt - key"  |
| 000000012: | 200      | 5 L   | 54 W | 331 Ch  | "# - rsa"  |
| 000000005: | 200      | 5 L   | 54 W | 331 Ch  | "# - key"  |
| 000000009: | 200      | 5 L   | 54 W | 331 Ch  | "# Copyright 2007 James Fisher - rsa"  |
| 000000013: | 200      | 5 L   | 54 W | 331 Ch  | "# <b>This work</b> is licensed under the Creative Commons - txt"  |
| 000000011: | 200      | 5 L   | 54 W | 331 Ch  | "# - key"  |
| 000000006: | 200      | 5 L   | 54 W | 331 Ch  | "# - rsa"  |
| 000000004: | 200      | 5 L   | 54 W | 331 Ch  | "# - txt"  |
| 000221107: | 200      | 1 L   | 1 W  | 4689 Ch | "mysecret - txt"   |

## 4.3 DECODIFICA DEL PAYLOAD E RECUPERO CHIAVE

Dopo aver scaricato il file **.mysecret.txt**, ci siamo trovati di fronte a una lunga stringa di caratteri apparentemente casuali in formato **ASCII**.



## Cos'è il formato ASCII?

**Spiegazione Tecnica:** L'ASCII (American Standard Code for Information Interchange) è un sistema di codifica dei caratteri che assegna un valore numerico a lettere, numeri e simboli. In termini di sicurezza, vedere un file in "puro ASCII" senza caratteri binari strani ci indica che i dati sono stati intenzionalmente codificati per essere facilmente trasportabili via testo, ma non sono immediatamente leggibili.

**Spiegazione Semplice:** Immagina che il computer parli una lingua fatta di soli numeri. L'ASCII è il "traduttore" che trasforma quei numeri in lettere e simboli che possiamo leggere sulla tastiera. Quando diciamo che un file è ASCII, significa che possiamo aprirlo con un semplice Blocco Note e vedere del testo, anche se quel testo sembra non avere senso.

## 1. Decodifica con CyberChef

**Spiegazione Tecnica:** Per la decodifica è stato utilizzato **CyberChef**.

**Cos'è CyberChef:** Definito come "Il coltellino svizzero dell'analista", è una web-app (spesso usata offline in ambito cybersecurity) che permette di concatenare diverse operazioni logiche e crittografiche (chiamate "Recipe") per analizzare, decodificare o deoffuscare dati.

Inserendo la stringa e selezionando l'operazione "From Base58", abbiamo ottenuto una **OpenSSH Private Key**. Come spiegato in precedenza, questa chiave rappresenta l'identità digitale dell'utente e permette l'accesso remoto senza password.

**Spiegazione Semplice:** Abbiamo usato uno strumento chiamato **CyberChef**, che è come un laboratorio di chimica digitale: tu gli dai un ingrediente misterioso e lui prova tutte le "ricette" per capire cos'è. Quando gli abbiamo detto di provare la ricetta "Base58", il messaggio segreto si è trasformato magicamente in una **Chiave Privata SSH**, ovvero la "chiave magnetica" che ci serve per entrare nel computer della vittima.

## 2. Analisi: Da Base64 a Base58

**Spiegazione Tecnica:** Inizialmente è stata ipotizzata una codifica **Base64**, lo standard più comune per trasformare dati binari (come le chiavi) in testo. Tuttavia, il tentativo di decodifica ha prodotto un output incoerente. L'analisi visiva della stringa ha rivelato l'assenza di:

- **Padding:** I simboli = alla fine della stringa.
- **Caratteri speciali:** I simboli + e /.
- **Caratteri ambigui:** Cifre e lettere che si confondono facilmente (come lo 0 e la O, o la I e la 1).

Queste caratteristiche sono tipiche della codifica **Base58**. La Base58 è una variante della Base64 progettata originariamente per Bitcoin; elimina i caratteri che potrebbero causare errori di lettura umana, rendendo la stringa più "pulita" ma comunque capace di contenere dati complessi.

**Spiegazione Semplice:** All'inizio pensavamo fosse un codice "standard" (Base64), ma qualcosa non tornava: mancavano dei simboli strani come il "+" o l'uguale "=" . Abbiamo capito che si trattava di **Base58**, un sistema di scrittura usato spesso nelle criptovalute. È un modo di scrivere messaggi segreti pensato per evitare errori: ad esempio, elimina lo zero "0" perché assomiglia troppo alla lettera "O". È un codice più "ordinato" e difficile da confondere.

The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar on the left listing various conversion options: FROM BASE, FROM BASE32, FROM BASE45, FROM BASE58, FROM BASE62, FROM BASE64, FROM BASE85, FROM BASE92, Fork, To Base58, Favourites, Data format, Encryption / Encoding, Public Key, and Arithmetic / Logic.
- Recipe:** A main area titled "From Base58" with the input "haber".
- Input:** The text "haber" is highlighted with a red box and a red arrow pointing to it.
- Output:** The resulting Base64 string is highlighted with a red box and a red arrow pointing to it.
- Code:** Below the input, the Base58 string "23456789ABCDEFGHJKLMNPQRSTUVWXYZ" is shown, and below the output, the Base64 string "-----BEGIN OPENSSH PRIVATE KEY-----" is shown.

Una volta ottenuta la chiave decodificata da CyberChef, è necessario configurarla correttamente sul sistema Kali Linux per poterla utilizzare come mezzo di autenticazione.

**Spiegazione Tecnica:** Il contenuto della chiave è stato copiato e salvato in un file di testo denominato `id_rsa`. Prima di procedere alla connessione, è obbligatorio eseguire il comando `chmod 600 id_rsa`.

**Cos'è chmod 600:** Il comando `chmod` (Change Mode) modifica i permessi di accesso ai file. Il valore **600** imposta i permessi in modo che solo il proprietario del file possa leggere e scrivere (Read/Write), negando qualsiasi accesso a gruppi o altri utenti (No Access). Il protocollo SSH, per design di sicurezza, impedisce l'avvio della connessione se la chiave privata ha permessi troppo permissivi (es. 644 o 777), poiché verrebbe considerata "compromessa" o insicura.

**Spiegazione Semplice:** Abbiamo preso la nostra "chiave magnetica" digitale e l'abbiamo messa in un file chiamato `id_rsa`. Ma c'è un problema: SSH è molto pignolo. Se vede che quella chiave può essere toccata o letta da chiunque altro sul computer, si rifiuta di usarla. Usando il comando `chmod 600`, abbiamo messo un "lucchetto" al file, dicendo al sistema: "Solo io posso vedere questo file". Solo a questo punto la serratura SSH accetterà la chiave.

## 4.5 Tentativo di Accesso SSH

**Spiegazione Tecnica:** Con la chiave pronta, abbiamo tentato l'accesso al server utilizzando l'utente identificato precedentemente, **icex64**, con il seguente comando: `ssh -i id_rsa icex64@192.168.64.20`

- **-i (Identity file):** Specifica il file da cui leggere la chiave privata per l'autenticazione.

**Esito:** Il sistema ha accettato la chiave ma ha richiesto una **passphrase** per sbloccarla. Questo indica che la chiave privata è ulteriormente protetta da una password (crittografia della chiave stessa).

## Conversione della chiave in Hash

**Spiegazione Tecnica:** Gli strumenti di cracking non possono lavorare direttamente sul file della chiave SSH (**id\_rsa**). Abbiamo quindi utilizzato lo script **ssh2john.py** per convertire la chiave privata in un formato leggibile dal software di cracking. Il comando ha estratto i parametri crittografici della chiave e li ha salvati in un file chiamato **ssh.hash**.

**Comando utilizzato:** `/usr/share/john/ssh2john.py id_rsa > ssh.hash`

**Spiegazione Semplice:** Immagina che la nostra "chiave magnetica" sia dentro una scatola trasparente ma chiusa a chiave. Non possiamo usarla finché non apriamo la scatola. Per farlo, dobbiamo prima "fotografare" il lucchetto della scatola in un modo che il nostro programma di scasso possa capire. Il comando **ssh2john** serve proprio a fare questa "fotografia" (l'hash).

```
(kali㉿kali)-[~]
$ /usr/share/john/ssh2john.py id_rsa > ssh.hash
```

## Attacco a Dizionario Mirato

**Spiegazione Tecnica:** Invece di procedere con un attacco casuale, abbiamo sfruttato l'**OSINT (Open Source Intelligence)**. Nel codice sorgente della directory **~secret**, un commento dell'utente "icex64" forniva un indizio fondamentale: la passphrase poteva essere violata usando la wordlist **fasttrack**. Abbiamo quindi configurato **John the Ripper** per tentare solo le password contenute in quel dizionario specifico.

**Comando utilizzato:** `john --wordlist=/usr/share/wordlists/fasttrack.txt ssh.hash`

**Cos'è John the Ripper:** È uno dei software di password cracking più famosi al mondo. Funziona prendendo ogni parola da un elenco (wordlist), applicandogli lo stesso algoritmo di cifratura del bersaglio e confrontando i risultati. Se le "impronte" coincidono, la password è trovata.

**Spiegazione Semplice:** Abbiamo usato un programma chiamato **John the Ripper**, che è come un ladro velocissimo che prova migliaia di chiavi al secondo. Invece di provare combinazioni a caso, gli abbiamo dato un elenco specifico di password (chiamato **fasttrack**) perché avevamo trovato un indizio che diceva che la password era proprio lì dentro.

```
(kali㉿kali)-[~/Desktop/lupin]
$ john --wordlist=/usr/share/wordlists/fasttrack.txt ssh.hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 16 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55w0rd! (id_rsa)
1g 0:00:00:02 DONE (2026-01-28 06:27) 0.3448g/s 33.10p/s 33.10c/s 33.10C/s Autumn2013.. testing123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

## Risultato: Password Identificata

**Esito:** L'attacco ha avuto successo, restituendo la passphrase corretta: **P@55w0rd!**

**Valutazione del Rischio:** L'uso di una password contenuta in una wordlist standard come *fasttrack* rappresenta una grave vulnerabilità. Nonostante l'uso di una chiave SSH (misura di sicurezza forte), la debolezza della passphrase scelta dall'utente ha permesso la completa compromissione della sua identità digitale.

---

## 5. ACCESSO AL SISTEMA (Initial Access)

Con la passphrase ottenuta, siamo finalmente pronti a entrare nel sistema.

**Azione da compiere:** Esegui il comando SSH finale: `ssh -i id_rsa icex64@192.168.64.20`

Quando il sistema ti chiederà la passphrase, inserisci: **P@55w0rd!**

**Sei dentro?** Una volta loggato, il tuo obiettivo è trovare la prima "flag" (solitamente un file chiamato `user.txt`) e poi iniziare la **Privilege Escalation**. Digita `sudo -l` non appena sei dentro per vedere se `icex64` ha permessi speciali!

## Recupero della Flag Utente (user.txt)

**Spiegazione Tecnica:** Una volta ottenuto l'accesso alla shell, è stata eseguita l'esplorazione della directory *home* dell'utente. Tramite il comando `ls -la` (che mostra tutti i file, inclusi quelli nascosti e i relativi permessi), è stato individuato il file `user.txt`.

Il contenuto del file è stato visualizzato con il comando `cat user.txt`. Oltre a un'elaborata **A S C I I A r t**, il file contiene il codice della flag: **3mp!  
r3{I\_See\_That\_You\_Manage\_To\_Get\_My\_Bunny}**

**Spiegazione Semplice:** Appena entrati, siamo andati a cercare il "premio" nella stanza dell'utente. Abbiamo trovato un file chiamato `user.txt`. Aprendolo, abbiamo trovato un disegno fatto di caratteri e una frase segreta che conferma che abbiamo superato la prima grande parte della sfida. È la prova del nostro successo.

```
→ ssh -i id_rsa icex64@192.168.50.16
The authenticity of host '192.168.50.16 (192.168.50.16)' can't be established.
ED25519 key fingerprint is: SHA256:GZOCytQu/pnSRRTMvJLagwz7ZPlJMDiyabwLvxTrKME
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.50.16' (ED25519) to the list of known hosts.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
Enter passphrase for key 'id_rsa':
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:~$ ls
user.txt
icex64@LupinOne:~$ ls -la
total 40
drwxr-xr-x 4 icex64 icex64 4096 Oct  7 2021 .
drwxr-xr-x 4 root  root  4096 Oct  4 2021 ..
-rw----- 1 icex64 icex64 115 Oct  7 2021 .bash_history
-rw-r--r-- 1 icex64 icex64 220 Oct  4 2021 .bash_logout
-rw-r--r-- 1 icex64 icex64 3526 Oct  4 2021 .bashrc
drwxr-xr-x 3 icex64 icex64 4096 Oct  4 2021 .local
-rw-r--r-- 1 icex64 icex64 807 Oct  4 2021 .profile
-rw----- 1 icex64 icex64 12 Oct   4 2021 .python_history
drwx----- 2 icex64 icex64 4096 Oct  4 2021 .ssh
-rw-r--r-- 1 icex64 icex64 2801 Oct  4 2021 user.txt
```



## 6. PRIVILEGE ESCALATION (Scalata dei Privilegi)

In questa fase, abbiamo analizzato il sistema operativo per individuare vulnerabilità intrinseche del Kernel che permettano di passare dall'utente **icex64** all'utente **root** (Amministratore).

### 6.1 Identificazione del Sistema e del Kernel

**Spiegazione Tecnica:** Il primo passo è stato l'utilizzo del comando **uname -a**.

```
icex64@LupinOne:~$ uname -a
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64 GNU/Linux
```

**Cos'è uname -a:** È un comando che restituisce informazioni dettagliate sul sistema. Nello specifico, mostra il nome del Kernel, la versione, l'architettura (amd64) e la data di compilazione.

Dallo screenshot si evince che la macchina esegue **Linux LupinOne 5.10.0-8-amd64**. Con una piccola ricerca su Google troviamo che la versione di Linux 5.1.0-8-amd64 è vulnerabile all'exploit **DirtyPipe (CVE-2022-0847)**.

**Spiegazione Semplice:** Abbiamo chiesto al computer: "Chi sei esattamente e che versione usi?". Usando il comando **uname -a**, il computer ci ha risposto mostrandoci la sua "carta d'identità" tecnica. Cercando su internet, abbiamo scoperto che quella specifica versione ha un difetto di fabbrica molto grave chiamato "DirtyPipe", che permette a chiunque sia già dentro di diventare il "capo" del computer.

The screenshot shows a web browser displaying the Exploit-DB website. The URL is [www.exploit-db.com/exploits/50806](https://www.exploit-db.com/exploits/50806). The page title is "Linux Kernel 5.8 < 5.16.11 - Local Privilege Escalation (DirtyPipe)". The exploit details are as follows:

| EDB-ID: | CVE:      | Author:           | Type: | Platform: | Date:      |
|---------|-----------|-------------------|-------|-----------|------------|
| 50806   | 2022-0847 | LANCE BIGGERSTAFF | LOCAL | LINUX     | 2022-03-08 |

Below the table, it says "EDB Verified: ✘". The "Exploit" section shows a red "Exploit" button and a green "PoC" button. The "Vulnerable App:" section is empty. The exploit code is listed at the bottom:

```
/* Exploit Title: [Linux Kernel 5.8 < 5.16.11] - Local Privilege Escalation (DirtyPipe)
// Exploit Author: blasty (@perturbmaxx.in)
// Original Author: Max Kellermann <max.kellermann@ionos.com>
// CVE: CVE-2022-0847

/* SPDX-License-Identifier: GPL-2.0 */
/*
 * Copyright 2022 eXHall GmbH / IDAOS SE
 *
 * author: Max Kellermann <max.kellermann@ionos.com>
 *
 * = Proof-of-concept exploit for the Dirty Pipe
 * = vulnerability (CVE-2022-0847) caused by an uninitialized
 * = "pipe_buffer.flags" variable. It demonstrates how to overwrite any
 * = file contents in the page cache, even if the file is not permitted
 * = to be written, immutable or on a read-only mount.
 *
 * This exploit requires Linux 5.8 or later; the code path was made
```

## 6.2 Preparazione e Compilazione dell'Exploit

**Spiegazione Tecnica:** Una volta identificata la vulnerabilità, abbiamo reperito il codice sorgente dell'exploit (scritto in linguaggio C) da **Exploit-DB**, un archivio mondiale di vulnerabilità note.

Abbiamo proceduto come segue:

1. **Creazione del file:** nano `exploit.c` (abbiamo incollato il codice all'interno dell'editor di testo Nano).
2. **Compilazione:** `gcc exploit.c -o exploit`.

**Cos'è la Compilazione (gcc):** I computer non capiscono direttamente il codice scritto dagli umani (C). Il programma `gcc` (GNU Compiler Collection) funge da traduttore, trasformando il testo scritto in `exploit.c` in un file binario eseguibile che il processore può processare.

**Spiegazione Semplice:** Abbiamo trovato su internet le "istruzioni" per fabbricare una chiave universale. Le abbiamo copiate in un file chiamato `exploit.c`. Dato che il computer non può leggere le istruzioni così come sono, abbiamo usato un programma chiamato `gcc` (che funge da fabbro) per trasformare quelle istruzioni in una vera e propria "chiave" pronta all'uso.

## 6.3 Analisi del Vettore di Attacco: SUID Root

**Spiegazione Tecnica:** Perché l'exploit DirtyPipe funzioni, deve poter sovrascrivere dati in file che normalmente sono accessibili solo dall'amministratore. L'exploit richiede il percorso di un file con il bit **SUID** impostato.

**Cos'è il bit SUID:** È un permesso speciale che permette a un utente comune di eseguire un programma con i privilegi del proprietario del file (solitamente root). Esempi classici sono `/usr/bin/passwd` (che deve poter scrivere nel file protetto delle password quando un utente la cambia) o `/usr/bin/su`.

**Spiegazione Semplice:** Per far scattare la trappola e diventare amministratori, la nostra "chiave universale" ha bisogno di un punto di appoggio. Questo punto di appoggio deve essere un programma che ha già dei "superpoteri" (permessi SUID), come quello che serve per cambiare la password. Usando quel programma come leva, l'exploit riuscirà a scardinare l'intera sicurezza del sistema.

## 6.4 Funzionamento di DirtyPipe (CVE-2022-0847)

L'ultima fase dell'attacco ha sfruttato una vulnerabilità critica del Kernel Linux per elevare i privilegi da utente comune ad amministratore di sistema.

**Spiegazione Tecnica:** La vulnerabilità **DirtyPipe** (CVE-2022-0847) risiede nel modo in cui il Kernel gestisce le "pipe" (meccanismi di comunicazione tra processi). L'exploit permette di scrivere

dati in file di sola lettura presenti nella cache della pagina del Kernel, senza alterare il file originale sul disco in modo permanente.

Eseguendo `./exploit /usr/bin/passwd`, il programma ha iniettato del codice malevolo nella memoria del binario SUID scelto. Questo ha permesso di "dirottare" l'esecuzione del programma protetto per generare una **Root Shell** (una riga di comando con massimi privilegi). Come confermato dal comando `id`, l'utente corrente è ora `uid=0(root)`.

```
icex64@LupinOne:~$ ./exploit /usr/bin/passwd
[+] hijacking suid binary..
[+] dropping suid shell..
[+] restoring suid binary..
[+] popping root shell.. (dont forget to clean up /tmp/sh ;))
# ID A
/bin/sh: 1: ID: not found
# id
uid=0(root) gid=0(root) groups=0(root),1001(icex64)
```

**Spiegazione Semplice:** DirtyPipe è come un trucco di magia che inganna la memoria a breve termine del computer. Abbiamo detto al sistema: "Voglio usare il programma per cambiare le password", ma mentre il computer lo stava caricando, abbiamo sostituito velocemente un pezzetto del programma con le nostre istruzioni segrete. Il computer, confuso, ha eseguito i nostri ordini pensando che fossero quelli del "capo", regalandoci le chiavi totali della macchina (il potere di Root).

## RECUPERO FLAG FINALE

Una volta acquisiti i privilegi di root, l'ultima azione è stata l'estrazione della prova definitiva della compromissione.

**Azione Finale:** Navigando nella directory dell'amministratore (`cd /root`), è stato individuato il file `root.txt`. Il comando `cat /root/root.txt` ha rivelato la flag finale della CTF.

## 7. CONCLUSIONI E RACCOMANDAZIONI

## 7.1 Sintesi dell'Attività (Black Box Completion)

L'attività di Penetration Testing sul target **192.168.64.20** ("Empire: Lupin One") si è conclusa con la compromissione totale del sistema (**Root Compromise**).

**Spiegazione Tecnica:** La catena di attacco ha seguito un percorso metodico suddiviso in due macro-fasi:

- Accesso Iniziale (User):** Sfruttando la configurazione insicura del modulo `mod_userdir` del web server, sono state individuate directory personali nascoste (`~secret`) e file sensibili offuscati in Base58. Questo ha permesso di esfiltrare una chiave privata SSH, craccarne la passphrase debole (`P@55w0rd!`) tramite attacco a dizionario e accedere come utente `icex64`.
  - Privilege Escalation (Root):** L'enumerazione post-exploitation ha rivelato una versione del kernel Linux obsoleta (`5.10.0-8`). Sfruttando la vulnerabilità critica **DirtyPipe (CVE-2022-0847)** su un binario con bit SUID impostato (`/usr/bin/passwd`), è stato possibile iniettare codice malevolo in memoria, elevare i privilegi a root e recuperare la flag finale.

**Spiegazione Semplice:** Abbiamo completato la sfida prendendo il controllo totale del computer ("Root"). Per arrivarci abbiamo fatto due cose: prima siamo entrati come un normale utente rubando una chiave digitale nascosta male nel sito web. Poi, una volta dentro, abbiamo approfittato di un difetto nel "motore" del computer (il Kernel) per diventare i capi supremi del sistema. È come se fossimo entrati in un ufficio con una chiave rubata e poi avessimo usato un difetto della cassaforte per aprirla e prendere tutto.

## 7.2 Raccomandazioni di Sicurezza

Dal punto di vista del **Risk Management**, l'attuale configurazione presenta un rischio inaccettabile. Si raccomandano le seguenti azioni correttive:

- **Aggiornamento del Kernel (Priorità Critica):** È imperativo eseguire l'upgrade del kernel Linux. La versione attuale è vulnerabile all'exploit pubblico DirtyPipe, che annulla ogni altra misura di sicurezza locale permettendo a chiunque di diventare amministratore.
- **Rimozione Dati Sensibili dal Web (Priorità Alta):** Bisogna eliminare chiavi SSH e file di configurazione dalle directory accessibili via browser. Basare la sicurezza sul fatto che un file inizi con un punto (.) o una tilde (~) è un errore concettuale noto come *Security through Obscurity*.
- **Policy Password e Chiavi (Priorità Media):** Imporre l'uso di passphrase complesse (Long & Strong) per le chiavi crittografiche. La password attuale è stata individuata in pochi secondi perché presente in elenchi pubblici di password comuni.
- **Hardening del Web Server (Priorità Bassa):** Rimuovere i commenti "parlanti" nel codice HTML e disabilitare il modulo `mod_userdir` se non strettamente necessario per le finalità aziendali, riducendo così la superficie di attacco.