Traccia: Sfruttamento delle Vulnerabilità XSS e SQL Injection sulla DVWA

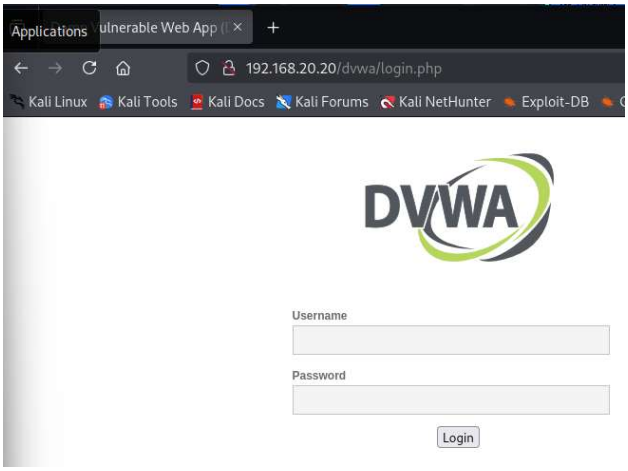
○ Verificate la comunicazione tra le due macchine utilizzando il comando ping.
○ Accedete alla DVWA dalla macchina Kali Linux tramite il browser.
○ Scegliete una vulnerabilità XSS reflected e una vulnerabilità SQL Injection (non blind).

PING DA MACCHINA KALI A MACCHINA META.

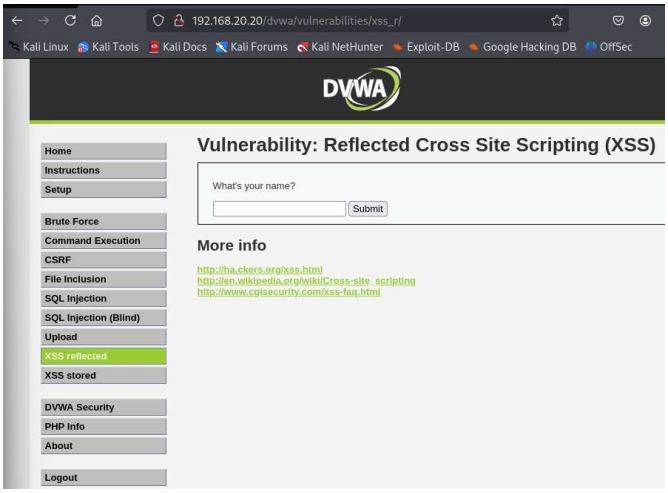
```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.20.20
PING 192.168.20.20 (192.168.20.20) 56(84) bytes of data.
64 bytes from 192.168.20.20: icmp_seq=1 ttl=64 time=2.06 ms
64 bytes from 192.168.20.20: icmp_seq=2 ttl=64 time=3.60 ms
64 bytes from 192.168.20.20: icmp_seq=3 ttl=64 time=4.94 ms
64 bytes from 192.168.20.20: icmp_seq=4 ttl=64 time=1.41 ms
64 bytes from 192.168.20.20: icmp_seq=5 ttl=64 time=0.972 ms
^C
── 192.168.20.20 ping statistics ──
5 packets transmitted, 5 received, 0% packet loss, time 4452ms
rtt min/avg/max/mdev = 0.972/2.596/4.941/1.472 ms
```

VULNERABILITA’

Accedo a DVWA.

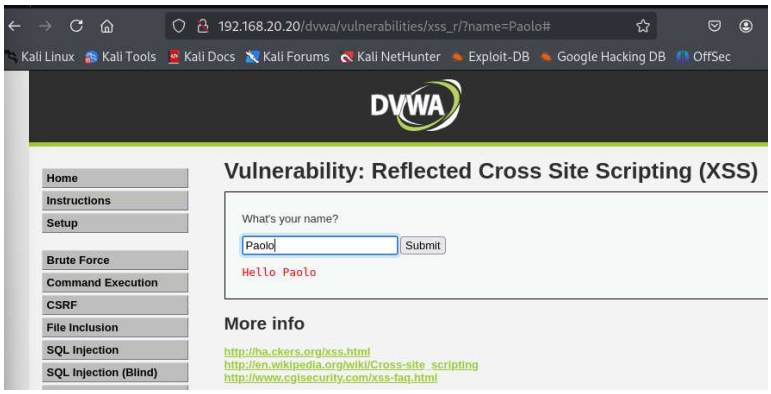


Una volta effettuato l’accesso, entro nella sezione *XSS REFLECTED*.

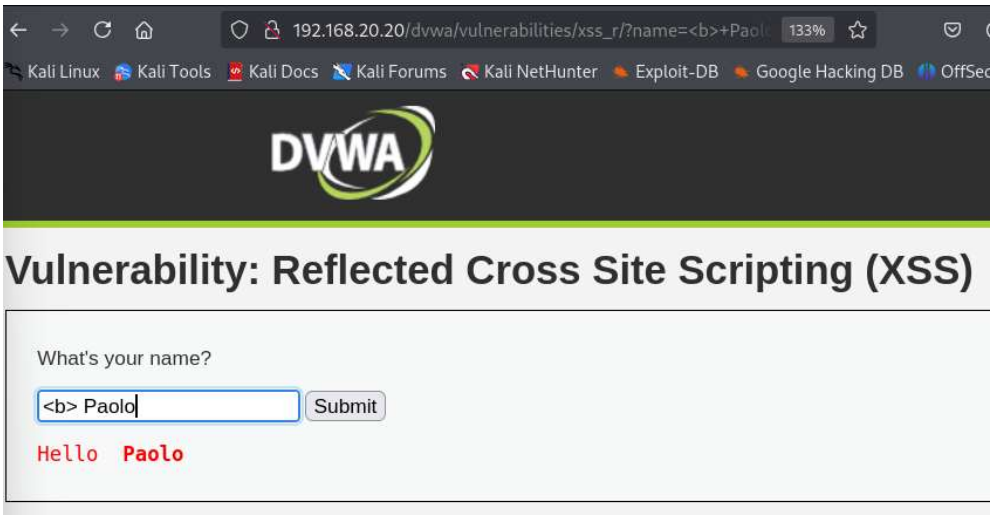


XSS REFLECTED

Corretto funzionamento

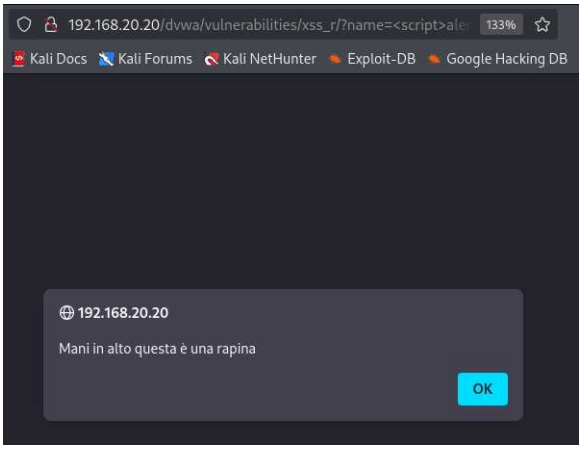


Cerco un *reflected point*. Utilizzo il tag <b> che serve a mettere in grassetto il testo. Trovato.



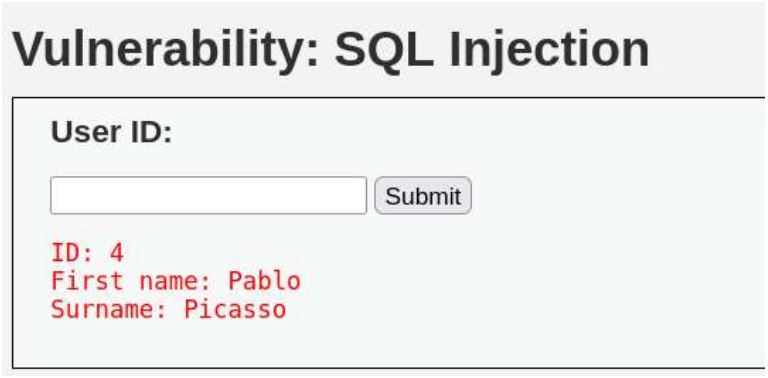
Eseguo l'attacco *XSS REFLECTED:* <script>alert("Mani in alto questa è una rapina")</script>

Risultato

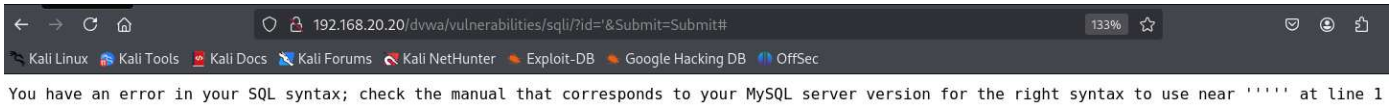


SQL INJECTION

Corretto funzionamento. Inserendo nel form l'ID 4, mi restituisce il nome e il cognome.



Inserendo nel form soltanto come carattere l'apice mi restituisce un errore, dimostrando che la query può utilizzarlo.



Una volta trovato la breccia inserisco nel form la stringa 1' OR '1'='1, per modificare la logica della query SQL in modo che ritorni sempre vera e di conseguenza l'app ci restituisce in output tutti i "First name" e Surname presenti dentro quel database.