GHOSTPROTOCOL

Web Application Exploit SQLi

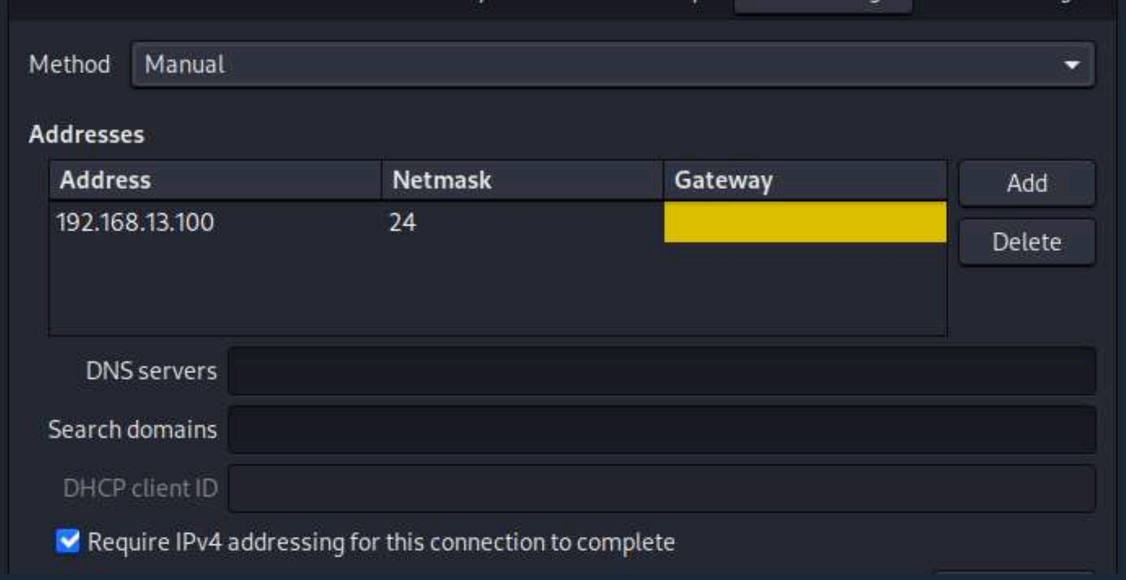
SQL injection Vulnerability

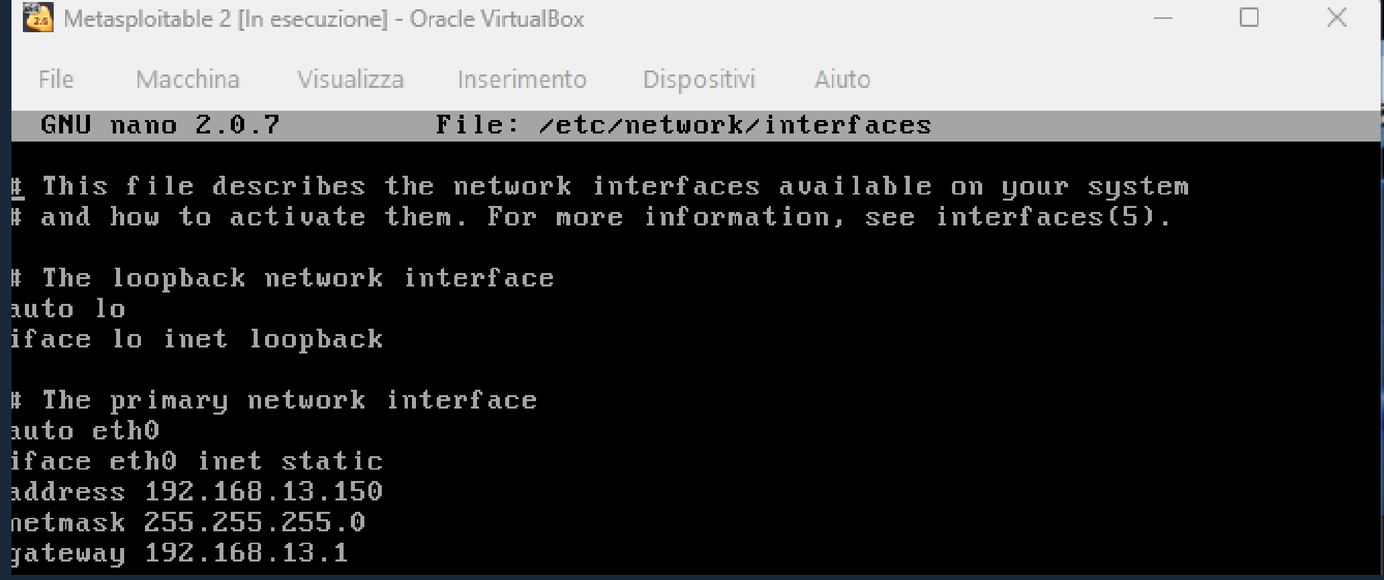
A Comprehensive Guide for New Employees

Presented by:

Sergio Astolfi
Luca Pani
Angelo De Santis
Daniele Castello
Federico Giannini
Alessandro P. Salerno
Alessandro Mammucari

Date:
10/11/2025





WORK ENVIRONMENT SETUP

GHOSTPROTOCOL

In questa fase è stato predisposto l'ambiente virtuale per le attività di penetration testing. Sono state configurate due macchine virtuali (kali Linux e Metasploitable2) nella stessa sottorete mediante indirizzi IP statici per garantire connettività stabile e riproducibilità dei test.

Dettagli di configurazione:

- Attaccante (Kali Linux): 192.168.13.100/24
- Bersaglio (Metasploitable 2): 192.168.13.150/24
- Gateway: 192.168.13.1

Lo schema assicura comunicazione diretta tra host e target, facilitando le successive fasi di scansione, sfruttamento e raccolta evidenze.

VERIFICA DELLA CONFIGURAZIONE DI RETE

In questa fase è stata verificata la corretta configurazione della rete interna tra la macchina attaccante (Kali Linux) e la macchina bersaglio (Metasploitable 2), entrambe operanti all'interno di un ambiente virtuale controllato.

Attraverso il comando ip a, è stato possibile confermare l'assegnazione degli indirizzi IP statici precedentemente impostati:

- Kali Linux: 192.168.13.100/24
- Metasploitable 2: 192.168.13.150/24

Entrambe le interfacce risultano attive (UP) e collegate alla stessa sottorete (192.168.13.0/24), garantendo così la possibilità di comunicazione diretta.

Successivamente, mediante l'utilizzo del comando ping, è stata testata la connettività bidirezionale tra i due host:

- Da Kali verso Metasploitable (ping 192.168.13.150)
- Da Metasploitable verso Kali (ping 192.168.13.100)

Le risposte ICMP ricevute confermano la piena raggiungibilità tra le due macchine, condizione necessaria per proseguire con le successive fasi di analisi, scansione e test di penetrazione.

GHOSTPROTOCOL

SETUP DEMONSTRATION

```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:1f:b7:23 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.100/24 brd 192.168.13.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::2a5d:c794:2dc7:a50a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:28:f6:45 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.150/24 brd 192.168.13.255 scope global eth0
    inet6 fe80::a00:27ff:fe28:f645/64 scope link
```

PING DEMONSTRATION

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=18.5 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=60.1 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=12.8 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=6.79 ms
```

```
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=0.943 ms
64 bytes from 192.168.13.100: icmp_seq=2 ttl=64 time=0.889 ms
64 bytes from 192.168.13.100: icmp_seq=3 ttl=64 time=0.494 ms
64 bytes from 192.168.13.100: icmp_seq=4 ttl=64 time=0.781 ms
```

1

2

VERIFICATION AND FINDINGS

Security Level: Low

1.

Osservato:

landing page di Metasploitable2 raggiungibile via browser all'indirizzo http://192.168.13.150. Vengono mostrate le applicazioni vulnerabili presenti (TWiki, phpMyAdmin, Mutillidae, DVWA, WebDAV) e le credenziali predefinite suggerite (msfadmin/msfadmin).

Implicazione:

il sistema è predisposto per esercitazioni; molte applicazioni web presenti sono intenzionalmente vulnerabili e accessibili.

2. **DVWA — livello di sicurezza**

Osservato:

pagina di configurazione DVWA indica che il livello di sicurezza è impostato su low.

Implicazione:

il contenuto dell'applicazione è vulnerabile a tecniche comuni di attacco (es. SQL injection, XSS) in forma non mitigata, per scopi didattici.

GHOSTPROTOCOL

VERIFICATION AND FINDINGS

Security Level: Low

1.
Payload:
1*' union select user(), database() #*

Cosa mostra lo screenshot:

- Output che riporta l'utente del DB (es. root@localhost) e il nome del database corrente (dvwa).

Interpretazione tecnica:

- L'iniezione UNION SELECT permette l'esecuzione di una query che concatena risultati arbitrari alla risposta.
- L'applicazione restituisce direttamente il risultato della query senza sanificazione.

Impatto:

- Informazioni sul contesto privilegiato del DB (identità dell'utente e schema) disponibili per un attaccante: facilita escalation e orientamento dei successivi payload.

2.
Payload:
1*' union select table_name, null from information_schema.tables where table_schema="dvwa" #*

Cosa mostra lo screenshot:

- Elenco delle tabelle presenti nello schema dvwa (es. guestbook, users, ...).

Interpretazione tecnica:

- Uso di information_schema per enumerare la struttura del DB; l'applicazione consente query che leggono metadati del database.

Impatto:

- L'enumerazione delle tabelle consente a un attaccante di mappare risorse sensibili e pianificare estrazioni mirate (es. tabella users).

3.
Payload:
1*' union select table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #*

Cosa mostra lo screenshot:

- Mappatura delle colonne per tabella (es. guestbook.comment_id, users.user_id, users.password, users.first_name, ecc.).

Interpretazione tecnica:

- L'attaccante ora conosce esattamente quali colonne contengono dati potenzialmente sensibili (p.es. password, user).
- Consente di costruire payload UNION SELECT mirati per estrarre esattamente i campi interessati.

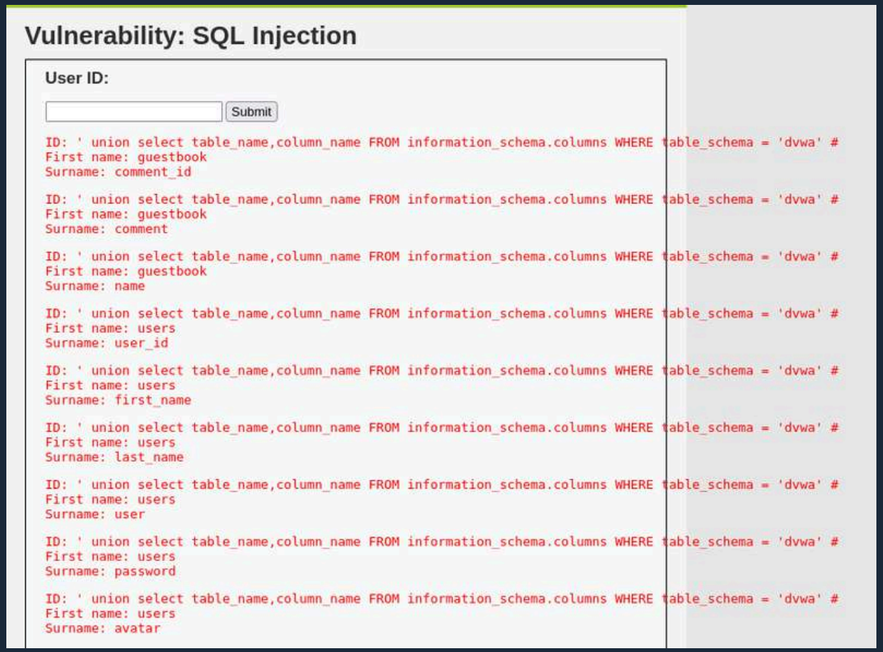
Impatto:

Aumento significativo del rischio: conoscendo nomi di colonne si può estrarre direttamente credenziali, email, ecc.

1

2

3



GHOSTPROTOCOL

VERIFICATION AND FINDINGS

Security Level: Low

1.
Payload:
1′ *union select user,last_name from dvwa.users #*

Cosa mostra lo screenshot:

- Righe estratte dalla tabella dvwa.users con username e cognomi (es. admin, gordonb, pablo, smithy, ecc.).

Interpretazione tecnica:
- L'iniezione consente di leggere record utente: conferma accesso in lettura alle tabelle utente.
- Possibile uso delle informazioni per attacchi di credenziali, spear-phishing o enumerazione account.

Impatto:
- Violazione della confidenzialità degli account (moderato-alto).

2.
Payload:
1′ *union select user,password from dvwa.users #*

Cosa mostra lo screenshot:

- Estratti username e password (in chiaro o in hash) per i record presenti (es. admin: 5f4dcc..., pablo: 0d107..., ecc.).

Interpretazione tecnica:

- Accesso a credenziali: se password in chiaro → compromissione immediata; se hash → possibile crack (attacco offline).
- Evidenzia l'assenza di protezioni come hashing sicuro/pepper o gestione corretta delle credenziali.

Impatto:

- Alto: esposizione di credenziali consente accesso agli account, potenziale pivoting e compromissione dell'applicazione/ambiente.

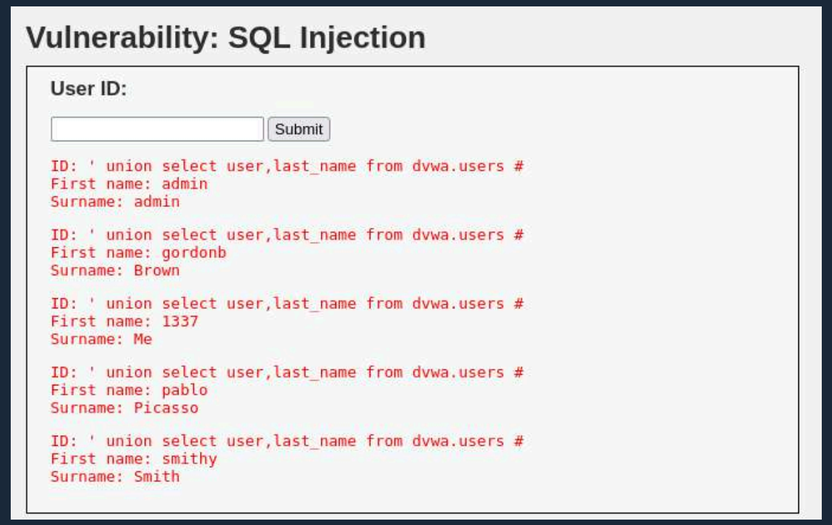
3.
Recupero password da hash

Comando eseguito:
john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt /home/kali/Desktop/PabloPass.txt

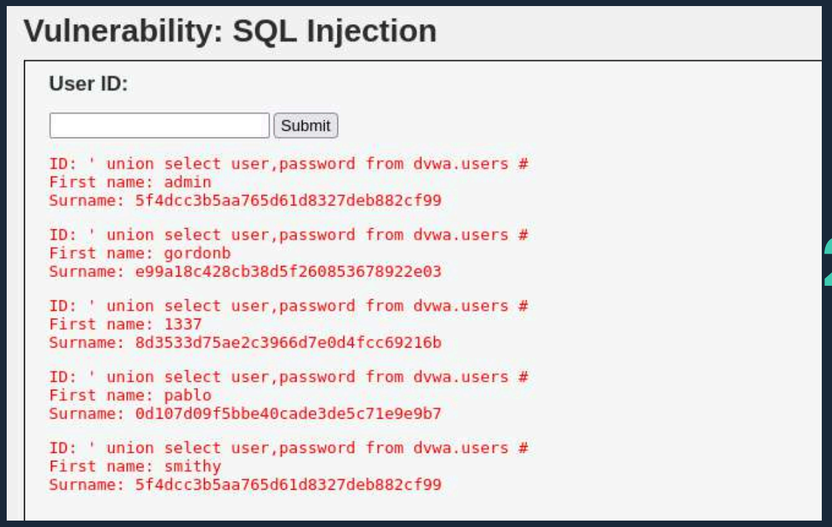
Osservato:
John ha caricato un hash in formato Raw-MD5 e, utilizzando il wordlist attack con rockyou.txt, ha completato la sessione segnalandone l'esito. Il file di output della sessione è stato creato nella home dell'utente (/home/kali/.john) e il tool suggerisce l'uso dell'opzione --show --format=Raw-MD5 per visualizzare le password recuperate.

Risultato:
hash craccato con successo tramite attacco di dizionario (wordlist). Evidenza: output di John (session completed / suggerimento per mostrare le password).

1



2



3

```
┌──(kali㉿kali)-[~]
└─$ john --format=raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt /home/kali/Desktop/PabloPass.txt
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (?)
1g 0:00:00:00 DONE (2025-11-10 04:35) 25.00g/s 19200p/s 19200c/s 19200C/s jeffrey..james1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

GHOSTPROTOCOL

EXECUTIVE SUMMARY

Sintesi delle attività svolte:

1. **C**onfigurazione ambiente di rete
   - È stata verificata la corretta connessione tra le due macchine tramite IP statici nella stessa sottorete (192.168.13.0/24).
   - I test di connettività (ping) hanno confermato la comunicazione bidirezionale.
2. Verifica accesso e funzionalità del target
   - L'interfaccia web di Metasploitable 2 è risultata raggiungibile via browser.
   - L'applicazione DVWA è stata configurata a livello di sicurezza Low, consentendo di replicare comportamenti vulnerabili.
3. Attività di attacco simulato
   - È stato eseguito un test di SQL Injection sulla sezione User ID di DVWA.
   - Attraverso una serie di query UNION SELECT, sono state progressivamente enumerate:
   - L'utenza del database e il nome dello schema attivo.
   - Le tabelle e le colonne contenute nello schema dvwa.
   - I record della tabella users, comprensivi di username, cognome e password/hash.
4. Raccolta e validazione delle evidenze
   - Tutti i passaggi sono stati documentati tramite screenshot (inclusi output SQL, dati estratti e risposte applicative).
   - Le informazioni ottenute dimostrano che l'applicazione non implementa alcuna forma di validazione dell'input o protezione contro query manipolate.
5. Verifica di cracking offline (John the Ripper)
   - Una password hash è stata sottoposta a test con l'utility John the Ripper, utilizzando la wordlist rockyou.txt.
   - L'attacco ha avuto esito positivo, confermando la debolezza delle password utilizzate e l'assenza di politiche di sicurezza robuste.

EXECUTIVE SUMMARY

Risultati e osservazioni

1. L’applicazione DVWA, in configurazione “low”, è deliberatamente vulnerabile: l’iniezione SQL è stata eseguita con successo fino all’estrazione di credenziali.
2. Le vulnerabilità riscontrate includono:
   - Assenza di sanitizzazione dell’input utente.
   - Accesso diretto al database tramite query concatenate.
   - Visualizzazione di messaggi d’errore SQL che rivelano dettagli di sistema.
   - Password archiviate con hashing debole (MD5).
3. L’ambiente ha permesso di dimostrare come, in un contesto reale, un attacco analogo potrebbe compromettere confidenzialità, integrità e autenticità dei dati aziendali.

IMPATTO POTENZIALE

AREA	DESCRIZIONE	LIVELLO DI RISCHIO
CONFIDENZIALITÀ	Esposizione di credenziali e struttura del DB	**ALTO**
INTEGRITÀ	Possibilità di alterare dati o account utente	**MEDIO-ALTO**
DISPONIBILITÀ	Potenziale interruzione servizi tramite query distruttive	**MEDIO**
REPUTAZIONE AZIENDALE	Impatto su fiducia clienti e compliance normativa	**ALTO**

EXECUTIVE SUMMARY

L’attività ha dimostrato come una configurazione non sicura e l’assenza di controlli sugli input possano portare alla completa compromissione di un’applicazione web.

Nel laboratorio, la vulnerabilità è stata sfruttata con successo fino all’estrazione di dati sensibili, confermando la criticità delle SQL Injection.

In un contesto produttivo, un simile scenario comporterebbe:

- violazione delle policy GDPR e dei principi di data protection by design;
- rischio di perdita o manipolazione di dati strategici;
- esposizione dell’infrastruttura ad attacchi di escalation o lateral movement.

Raccomandazioni principali

1. Implementare parametrizzazione delle query e prepared statements per tutti gli accessi al database.
2. Applicare filtri e validazioni lato server sugli input utente.
3. Utilizzare hashing sicuro (es. bcrypt, scrypt) per le password.
4. Limitare i privilegi del database user utilizzato dall’applicazione.
5. Attivare monitoraggio e logging delle query sospette.
6. Integrare test di sicurezza periodici (penetration test e code review).

VERIFICATION AND FINDINGS

Security Level: Medium

1.
Payload:
1 UNION SELECT user(), null

Cosa mostra lo screenshot:

- L'applicazione restituisce l'utente del database (es. root@localhost) e l'account applicativo (admin) nella stessa schermata.

Interpretazione tecnica:

- Il campo è vulnerabile a UNION SELECT. Nonostante DVWA sia impostata su medium (che applica filtri/escaping di base), la query è stata costruita in modo compatibile con la struttura SQL dell'applicazione (numero e tipo di colonne) e ha consentito la concatenazione di risultati arbitrari.
- Il fatto che l'output mostri sia admin sia root@localhost indica che l'attaccante ha ottenuto informazioni sia sull'utente dell'applicazione sia sul contesto DB.

Impatto:

- Informazioni contestuali utili per attacchi successivi: tipologia di account, possibili privilegi, target per esploit mirati.

2.
Payload:
1 *union select table_name, null from information_schema.tables where table_schema=0x64677661*

Cosa mostra lo screenshot:

- Elenco delle tabelle trovate nello schema dvwa (es. admin, guestbook, users), restituito in rosso dall'applicazione.

Interpretazione tecnica:

- DVWA medium applica filtri su stringhe e caratteri speciali; l'uso della rappresentazione esadecimale/hex (0x64677661 = 'dvwa') è una tecnica di bypass che evita filtri che cercano la stringa 'dvwa' in chiaro.
- L'applicazione consente accesso a information_schema, permettendo l'enumerazione delle tabelle.

Impatto:

- L'enumerazione delle tabelle consente il mapping della struttura del DB, facilitando payload successivi mirati a tabelle sensibili (e.g. users).

3.
Payload:
1 *union select table_name, column_name FROM information_schema.columns WHERE table_schema = 0x64677661*

Cosa mostra lo screenshot:

- Mappatura delle colonne per ciascuna tabella (es. guestbook.comment, users.user_id, users.password, users.first_name, ecc.).

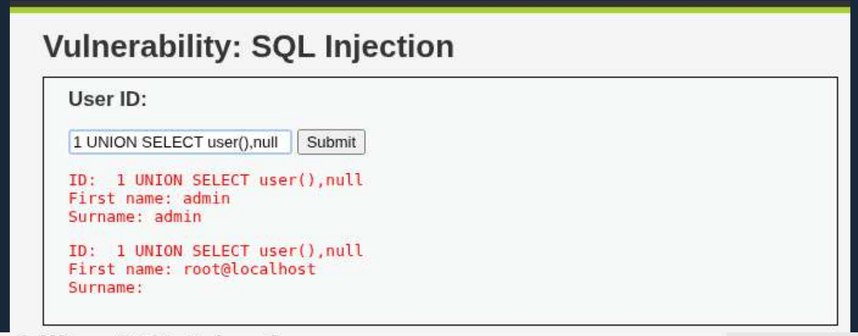
Interpretazione tecnica:

- Conoscendo nomi di colonne, un attaccante può costruire UNION SELECT specifici per estrarre esattamente i campi sensibili (p.es. users.password). Il fatto che l'enumerazione funzioni dimostra che i filtri applicati non impediscono query su information_schema né il pattern UNION se opportunamente costruito/mascherato.

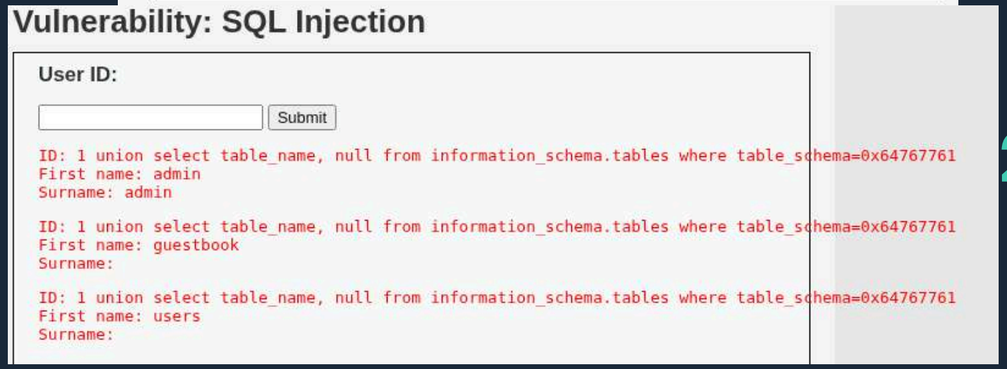
Impatto:

- Aumento esponenziale della capacità di estrazione dati sensibili: si passa da una visione generale a estrazioni mirate di credenziali o dati personali.

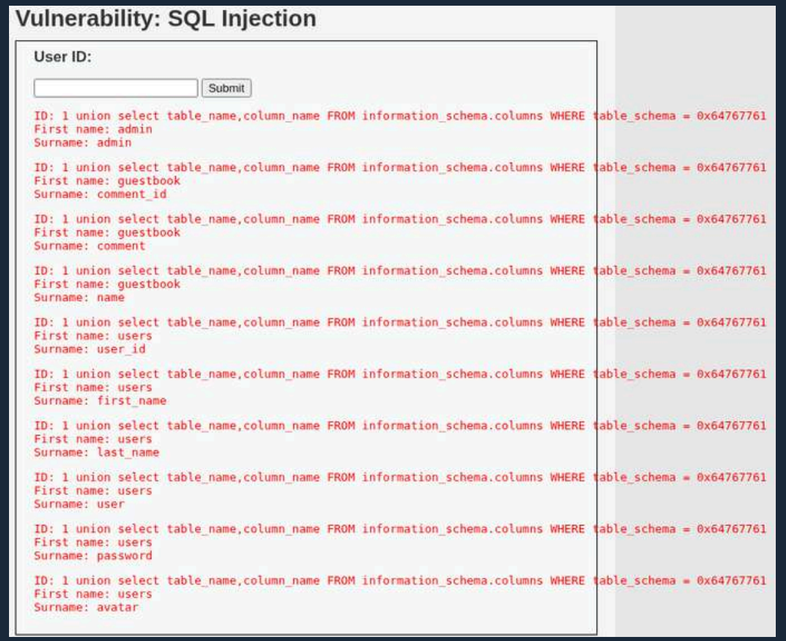
1



2



3



GHOSTPROTOCOL

VERIFICATION AND FINDINGS

Security Level: Medium

1.
Payload
1 *union select user, last_name from dvwa.users*

Cosa mostra lo screenshot:

- Lista degli utenti estratti dalla tabella dvwa.users con i rispettivi cognomi (es. admin, gordonb, pablo, smithy).

Interpretazione tecnica:

- L'applicazione ha consentito la lettura diretta della tabella utenti: l'input non è sufficientemente sanificato e la query costruita dall'applicazione è vulnerabile a UNION mirati.
- Anche con DVWA a medium, l'assenza di binding dei parametri e l'uso di query concatenate espone alla lettura diretta.

Impatto:

- Violazione della confidenzialità degli account: informazioni utilizzabili per social engineering, brute force, e attacchi mirati.

2.
Payload:
1 *union select user, password from dvwa.users*

Cosa mostra lo screenshot:

- Estrazione di username e password (alcune in hash MD5, altre apparentemente in testo o hash differenti), es.: admin: 5f4dcc3b5aa765d61d8327deb882cf99.

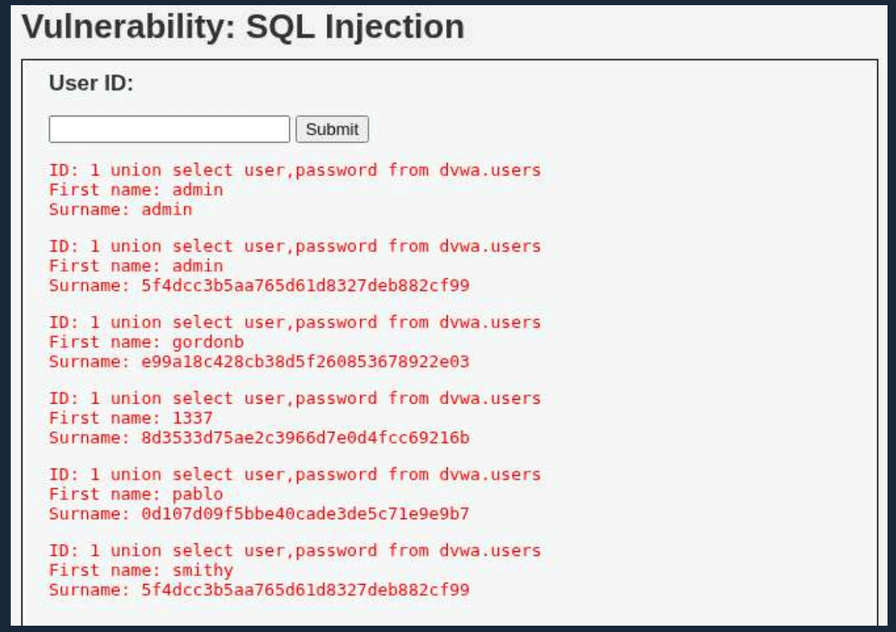
Interpretazione tecnica:

- L'estrazione di password conferma che i dati sensibili sono memorizzati in modo non adeguato (MD5 o valori reversibili/deboli). Anche se le password sono hashed, l'uso di algoritmi deboli (MD5) e l'assenza di salt/pepper rende possibile l'attacco offline (cracking), come dimostrato in precedenti passaggi con John the Ripper.
- In medium, DVWA spesso aggiunge filtri, ma non introduce pratiche di hashing sicuro: la logica dell'applicazione resta vulnerabile.

Impatto:

- Critico se dovesse trattarsi di un ambiente reale: accesso immediato agli account (se password in chiaro) o possibilità concreta di recupero offline (se hash deboli). Potenziale pivoting e escalation.

1



2

VERIFICATION AND FINDINGS

Security Level: Medium

1
Payload
1 *UNION SELECT 1, schema_name FROM information_schema.schemata --*

Cosa mostra lo screenshot

Estrazione di record dalla tabella dvwa.users con coppie username : password/hash, ad esempio:
- admin : 5f4dcc3b5aa765d61d8327deb882cf99
- gordonb : e99a18c428cb38d5f260853678922e03
- 1337 : 8d3533d75ae2c3966d7e0d4fcc69216b

Interpretazione tecnica

- L'applicazione ha permesso la lettura diretta della tabella utenti tramite UNION SELECT, confermando che i campi sensibili sono accessibili tramite SQL injection.
- Le password risultano memorizzate con hash deboli (MD5) o formati facilmente attaccabili; assenti tecniche di rafforzamento quali salt o pepper.
- Anche con DVWA configurata su medium, i filtri non impediscono estrazioni mirate di credenziali quando la query è costruita opportunamente.

Impatto

- Critico: esposizione di credenziali abilita accesso agli account, potenziale pivoting e compromissione dell'intera infrastruttura.
- Hash deboli => possibilità concreta di cracking offline e riutilizzo delle credenziali.

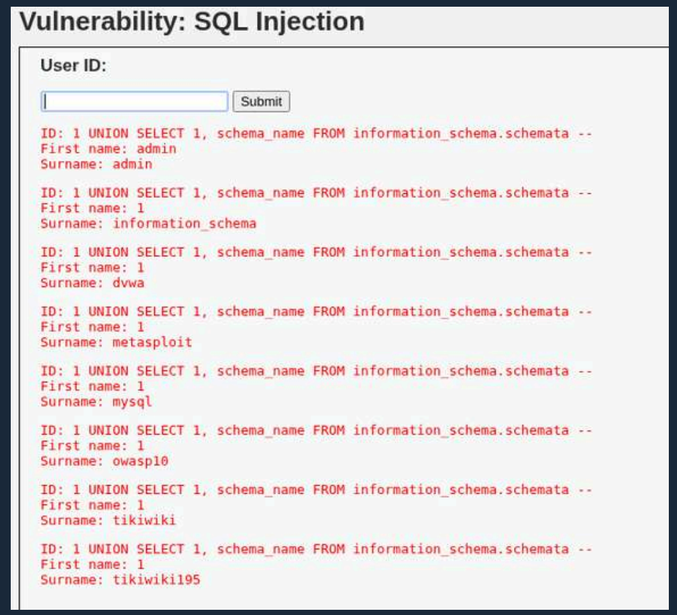
2.
Payload:
1 UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 0x6d7973716c

Cosa mostra:
- Elenco di tabelle presenti nello schema mysql (es. user, db, time_zone, procs_priv, ecc.), visualizzato direttamente nella pagina.

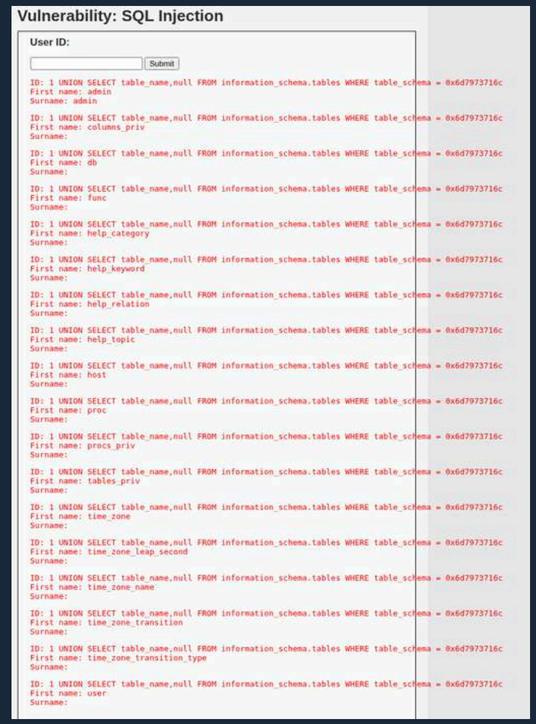
Interpretazione tecnica:
- È stata effettuata l'enumerazione di information_schema.tables usando la codifica esadecimale (0x6d7973716c = "mysql") per bypassare filtri; ciò conferma che la SQLi consente query su metadata del server.

Impatto:
- Conoscere le tabelle del DB di sistema permette di individuare risorse sensibili (es. mysql.user) e pianificare estrazioni di credenziali o privilegi. Rischio: alto.

1



2



GHOSTPROTOCOL

VERIFICATION AND FINDINGS

Security Level: Medium

1.
Payload:
1 UNION SELECT table_name, column_nameFROM information_schema.columnsWHERE table_schema = 0x6d7973716c -- AND table_name = 0x75736572 – (0x75736572 = "user")

Cosa mostra:

- Elenco delle colonne della tabella mysql.user (privilegi, colonne di controllo come Host, User, Password, Create_priv, Delete_priv, ecc.).

Interpretazione tecnica:

- L'attaccante ha ottenuto la struttura della tabella user del DB di sistema; con i nomi di colonna è possibile costruire query mirate per estrarre credenziali e privilegi.

Impatto:

- Elevata precisione negli attacchi futuri: conoscendo colonne di privilegi si possono estrarre informazioni per escalation di privilegi e pivoting. Rischio: molto alto.

2.
Payload:
1 UNION SELECT user, password FROM mysql.user

Cosa mostra:

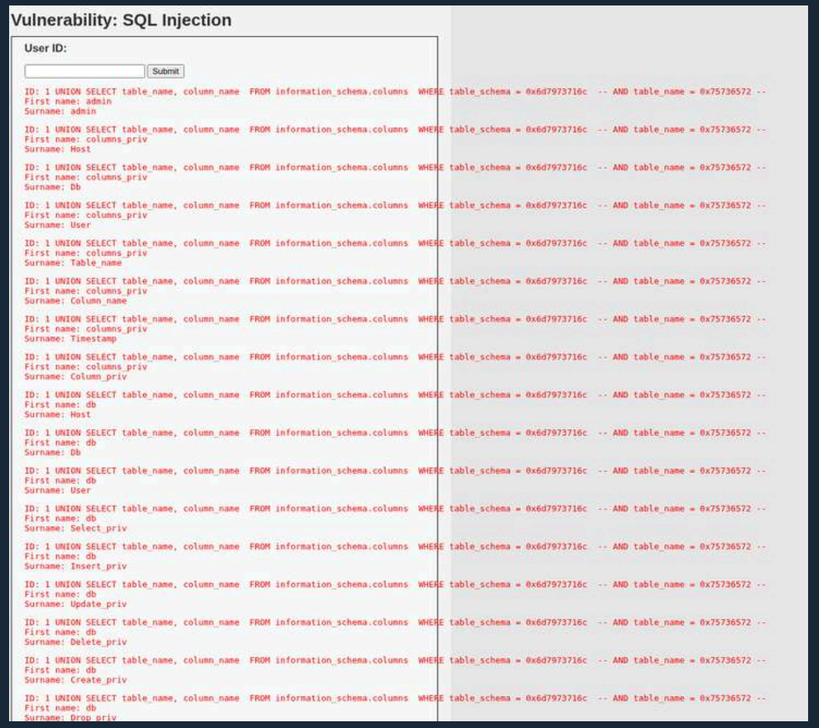
- Estrazione di righe dalla tabella mysql.user con nomi account di sistema (es. root, debian-sys-maint, guest, admin), e valori password/hash visibili.

Interpretazione tecnica:

- L'accesso alla tabella mysql.user consente di leggere account di amministrazione del DB: informazioni che, se sfruttate, permettono accesso diretto o offline cracking delle password.

Impatto:

- Critico — compromissione potenziale dell'intero server DB, con possibilità di takeover dell'istanza MySQL e accesso a tutte le applicazioni ospitate. Rischio: critico / immediato.



1

2

GHOSTPROTOCOL

GHOSTPROTOCOL

Web Application Exploit SQLi

Al termine delle attività previste, il team GhostProtocol ha completato con successo tutte le fasi di configurazione, verifica e test dell'ambiente di lavoro, dimostrando in modo pratico l'intero ciclo di un attacco controllato e la conseguente analisi delle vulnerabilità riscontrate.

L'esecuzione ha permesso di:

1. validare la corretta interoperabilità tra le macchine virtuali e la loro configurazione di rete;
2. individuare e documentare in modo sistematico una vulnerabilità di tipo SQL Injection;
3. comprendere in che modo un'applicazione non protetta possa esporre informazioni sensibili e compromettere la sicurezza complessiva di un sistema.

I risultati ottenuti confermano l'importanza di un approccio preventivo e consapevole alla sicurezza informatica, fondato su buone pratiche di sviluppo, controlli costanti e attività di penetration testing periodiche.

Il team ritiene che gli obiettivi e tecnici spiegati e che le competenze acquisite possano essere applicate efficacemente a futuri progetti di analisi e miglioramento della sicurezza delle infrastrutture informatiche.

Presented by:

Sergio Astolfi
Luca Pani
Angelo De Santis
Daniele Castello
Federico Giannini
Alessandro P. Salerno
Alessandro Mammucari

Date:
10/11/2025