OWASP JUICE SHOP

Web application security analisys with Burp suite

1. Introduction

The OWASP foundation
The OWASP foundation works to improve security of software through its community-led opne source software projects, hundreds of chapters worldwide, ten od thousands of members, and by hosting locak and global cobferences.

OWASP top 10
The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.
Companies should adopt this document and start the process of ensuring that their web applications minimize these risks. Using the OWASP Top 10 is perhaps the most effective first step towards changing the software development culture within your organization into one that produces more secure code.

The Juice Shop
OWASP Juice Shop is probably the most modern and sophisticated insecure web application! It can be used in security trainings, awareness demos, CTFs and as a guinea pig for security tools! Juice Shop encompasses vulnerabilities from the entire OWASP Top Ten along with many other security flaws found in real-world applications!

2. Task

The aim of this exercise is to find and exploit some of the *Juice Shop* vulnerabilities. *Juice Shop* is a large application and so I've only covered some of the topics:

- **Injection**
- **Broken Authentication**
- **Sensitive Data Exposure**
- **Broken Access Control**
- **Cross-Site Scripting XSS**

For the completition of this task I made use of the Kali linux distribution hosted on Virtual Machine on *VM Workstation 16.*

Burp suite
Burp Suite Professional is one of the most popular penetration testing and vulnerability finder tools, and is often used for checking web application security. "Burp," as it is commonly known, is a proxy-based tool used to evaluate the security of web-based applications and do hands-on testing.

Foxy Proxy
The Burp Proxy works by opening a web interface on 127.0.0.1:8080(by default). As implied by the fact that this is a "proxy", we need to redirect all of our browser traffic through this port before we can start intercepting it with Burp. We can do this by altering our browser settings or, more commonly, by using a Firefox browser extension called FoxyProxy. FoxyProxy allows us to save proxy profiles, meaning we can quickly and easily switch to our "Burp Suite" profile in a matter of clicks, then disable the proxy just as easily.

3. Execution

The first step is to simply open a browser and search for the Ip Address. We can now browse the web app and have a walking through.

SQL Injection
There are many type of *Injection*. Some of them are *SQL Injection, Command Injection and Email Injection.*
SQL Injection is when an attacker enters a malicious or malformed query to either retrieve or tamper data from a database. And in some cases, log into accounts .

After we navigate to the login page we enter some data in the email and password fields, and with **Burp suite Intercept mode On** we can now see which data is sent to the server.
In the SQL query, we substitute the email field with: **' or 1=1--**

- The character " **'** " will close the brackets od the **SQL statement**
- '**OR**' in a SQL statement will return true if either side of it is true. As **1=1 is always true**, the whole statement is true. Thus it will tell the server that the email is valid, and log us into **user id 0**, which happens to be the administrator account.
- The "**--**" character is used in SQL to comment out data, any restrictions on the login will no longer work as they are interpreted as a comment. This is like the **#** and **//** comment in python and javascript respectively.

And we successfully gain access to the **Admin** account. But we still don't know his password.

Broken Authentication
Authentication is “broken” when attackers are able to compromise passwords, keys or session tokens, user account information, and other details to assume user identities.

We now intercept again the login request but this time we send it to the ***Intruder,*** which we’ll use to perform a ***bruteforce*** attack.
We’ll make use of a short password wordlist: **Seclists** (*/usr/share/seclists/Passwords/Common-credential/best1050.txt*).
We load the following wordlist into the intruder and we run the attack. We can shortly see the password **admin123** returing a **200 status code**, which means we found the correct password.

Sensitive Data Exposure
Most of the time, data protection is not applied consistently across the web application making certain pages accessible to the public. Other times information is leaked to the public without the knowledge of the developer, making the web application vulnerable to an attack.

There different ways to search for sensitive data accrss the web app. Two of these are: **Dirb** and a simple **walking through**.
In fact, if we navigate to the **About us** page and drag the mouse over the *terms and condition* link we can see that there’s an **FTP** directory, and it’s accesible! Here we find interesting files such as: **acquisitions.md** and **package.json.bak** (a backup file!).

When we try to download the backup file our request is denied as we can only download **.md** and **.pdf** can be downloaded.
We can get around this making use a **Poison Null Byte**. If we add to our request: **%2500.md**, the string will tell the server to terminate at that point, nulling the rest of thre string.

```
10.10.107.153/ftp/packgage.json.bak%2500.md
```

Broken Access Control
Broken access control vulnerabilities exist when a user can in fact access some resource or perform some action that they are not supposed to be able to access.

We can access some of these resources using the **Debugger** tool in Firefox. If we look through the javascript we can see a path called **/#/administration**. We can navigate to this folder logging in as **admin** and submitting the path in the Firefox search bar.

Another example of *broken access control* is visible using once again **burp**. If we navigate to the **basket** page and we intercept the data, we can modify the **GET request id,** from " 1 " to " 2 " , to view others users's baskets.

GET *rest/*basket/**2** http/1.1

XSS – Cross Site Scripting
XSS or Cross-site scripting is a vulnerability that allows attackers to run javascript in web applications. These are one of the most found bugs in web applications. Their complexity ranges from easy to extremely hard, as each web application parses the queries in a different way.
There are three major type of XXS: **DOM**, **Persistent** and **Reflected**.

DOM (*Document Object Model-based Cross-site Scripting*)
uses the HTML environment to execute malicious javascript. This type of attack commonly uses the <*script*></*script*> HTML tag.

We'll be using the *Iframe* element with a javascript alert tag:
<iframe src="javascript:alert('xss')">
Inputting the Iframe in the search bar will trigger the alert ans shoe '**xss**'.

Another method to perform an *XSS* is using **Burp Header Tab**. If we log in as admin, we navigate to **Last Login IP** and we activate our interceptor, we can execute the same Iframe simply adding a new Header. Now, when accessing again and navigating to the Last login IP we'll trigger the alert.

Reflected
Reflected XSS is javascript that is run on the client-side end of the web application. These are most commonly found when the server doesn't sanitise search data.

Again, we Log in as **admin**, we navigate to **Order History** page and we can see the URL query:

10-10.107.153/truck-result?id=6484-f730006b3b6

If we substitute ***6484-f730006b3b6*** with the same Iframe and we refresh the page we can see the alert.

4. Conclusions
The OWASP Juice Shop web App is purposely weak and full of vulnerabilities. This is the reason why it is a very usefull tool for web application pentesting. After completing this exercise I was able to detect and exploit some of the most common web application vulnerabilities. I was also able to use some of the funda-

mental tools in **Burp suite**, such as: Proxy, Intruder and Header. I've injected malicious SQL data, I've bruteforce the admin password and performed a XSS attack.

Resources

1. Cyber security learning platform
https://tryhackme.com/room/rpburpsuite
2. Burp Documentation
https://portswigger.net/burp/documentation
3. W3 School
https://www.w3schools.com/cybersecurity/index.php
4. Owasp foundation
https://owasp.org/

credit:
Alessio Ragazzi,
London.