|  |
| --- |
|  |
| Capstone Experience IST 894  Carl Laneave |
| Lab 7 Report |

Table of Contents

[1.0 Introduction – Web Vulnerabilities 3](#_Toc148300852)

[1.1 Lab Results – Web Vulnerabilities 5](#_Toc148300853)

[2.0 Introduction – Secure and Insecure Protocols 14](#_Toc148300854)

[2.1 Lab Results – Secure and Insecure Protocols 15](#_Toc148300855)

[3.0 Introduction – Network Security and Diagnostics 24](#_Toc148300856)

[3.1 Lab Results – Network Security and Diagnostics 25](#_Toc148300857)

[4.0 References: 28](#_Toc148300858)

[5.0 Activity Log 29](#_Toc148300859)

# Introduction – Web Vulnerabilities

During the execution of this lab, an evaluation was done to test the ability of different types of web vulnerabilities on a website. Web vulnerabilities are a critical path that many attackers can take in trying to access your servers and instances. There are several types of different web vulnerabilities including XSS reflection attacks, XSS stored attacks, SQL injections and file traversing.

XXS, or cross site forgery attacks, fall into many subcategories. In testing the first, which is the XSS reflection attacks, attackers will feed HTML code into a form since there is no protection against it. Through this, attackers can make server requests, execute malicious code, and more. Sanitization of said forms is critical as without it, it leaves the web application vulnerable to attacks.

Cross-site forgery stored attacks are like reflection attacks except that they are stored values. This could include something such as a blog, in which a potential attacker could add information beyond simple text. An example used in this lab was around submitting malicious links through a stored attack. The said attacker can use HTML to create a phishing link that will be stored on the site and used to phish other visitors to the site. Since it is on a blog of the site itself, it appears to be part of the site, which the attackers can use to their advantage.

Another attack reviewed is the SQL attack. SQL attacks were at one time one of the most common methods of web vulnerability attacks. This attack is accomplished by using illegal chars in a form such as ‘to get out of the text box and move into SQL querying language of the actual database. Since again, the data is not sanitized prior to doing the query, attackers can escape and run their own queries against the database. A simple query, such as 1=1 will return all true values, which is all the items in the database itself. It is critical to eliminate the ability to send non-alphanumeric characters as well as sanitize data prior to doing any type of query.

The last attack is file traversing, which is changing the HTML crumb of a called file on the server to a new crumb and using the system directories in the call. This is done when improper file and file structure permissions are done on the site. Attackers can change through files and directories by changing the slug in the web URL. Many URLs point to files on a system, and if not properly configured can be manipulated to traverse throughout the entire server's web directories. These types of attacks allow attackers to not only see critically important secret files but also explore and add in their own shell remote commands.

# 1.1 Lab Results – Web Vulnerabilities

A screenshot of a computer

Description automatically generated

Figure 1.0 – Log into web-vulns.com website using admin/admin.

A screenshot of a computer

Description automatically generated

Figure 1.1 – pass in html into field to test for XSS reflection attack capabilities.

A screenshot of a computer

Description automatically generated

Figure 1.2 - put in a script alert pop up to test XSS reflection attack further.

A screenshot of a computer

Description automatically generated

Figure 1.3 – Attempt a stored XSS attack, which is successful using html bold tags.

A screenshot of a computer

Description automatically generated

Figure 1.4 – Input a malicious link into the stored blog XSS attack.

A screenshot of a login form

Description automatically generated

Figure 1.5 – Attempt and test a DOM based XSS attack.

A computer code with black text

Description automatically generated

Figure 1.6 – Open page source, find the innerHTML where the request is going.

A screenshot of a computer

Description automatically generated

Figure 1.7 – change the PHP to do a get call and send a script alert to the request.

A screenshot of a computer

Description automatically generated

Figure 1.8 – Open SQL injection GET/Search.

A screenshot of a movie

Description automatically generated

Figure 1.9 – Input sql command ‘ or 1=1-‘ to return all items that are true

A screenshot of a computer

Description automatically generated

Figure 1.10 – Open Directory Traversal Files url.

A screenshot of a computer

Description automatically generated

Figure 1.11 – Change the text file director to pull page /etc/passwd

A screenshot of a computer

Description automatically generated

Figure 1.12 – Move through the different server directories.

A screenshot of a computer

Description automatically generated

Figure 1.13 – Load up RFI for local file inclusion.

A black screen with white text

Description automatically generated

Figure 1.14 – Lab complete

# 2.0 Introduction – Secure and Insecure Protocols

In this attack, the focus is on using properly secure protocols when making calls versus having insecure protocols. The protocols focused on here are ssh, ftp and sftp as each are important and commonly used protocols. To monitor the traffic and calls, we set up Wireshark to start sniffing and watching packets coming through our server. By using insecure protocols, we can capture the tcp packets inside of Wireshark and see information such as username as well as password in plain text. This lack of securing your protocols leaves transmissions between the host and client wide open in plain text to be ingested and seen inside our Wireshark packet sniffer. Attackers can use these insecure protocols and monitoring tools to take that plain text and steal valuable information. On the other side, in creating a secure connection, we can see that critical packet information will be protected and censored. This ensures that if an attacker was watching our packet transmissions, for example through a man-in-the-middle attack, the information still would be coming in encrypted and through a secure protocol.

# 2.1 Lab Results – Secure and Insecure Protocols

A computer screen shot of a black screen

Description automatically generated

Figure 2.0 – Load up LXTerminal.

A screen shot of a computer code

Description automatically generated

Figure 2.1 – Run flag help for tcpdump.

A screenshot of a computer

Description automatically generated

Figure 2.2 – Load wireshark up.

A screenshot of a computer

Description automatically generated

Figure 2.3 – Run a filter against tcp port 23.

A screenshot of a computer

Description automatically generated

Figure 2.4 – telnet into the ip address of the server and see captured packets on wireshark.

A screenshot of a computer

Description automatically generated

Figure 2.5 – Follow the packet, inside the packet is the username and password.



Figure 2.6 – do a tcpdump on eth0.

A computer screen with white text

Description automatically generated

Figure 2.7 – ssh into the target machine using the root information.

A screenshot of a computer

Description automatically generated

Figure 2.8 – Investigate the packet for ssh and we can see the key exchange.

A screenshot of a computer

Description automatically generated

Figure 2.9 – Capture FTP port and save the echo’d text to ftp\_test.txt.

A screenshot of a computer

Description automatically generated

Figure 2.10 – ftp into target, packets are captured in wireshark.

A screenshot of a computer

Description automatically generated

Figure 2.11 – Upload a file.txt and we can see it captured in wireshark.

A screenshot of a computer

Description automatically generated

Figure 2.12 – Save the TCP Stream from FTP 150 packets in wireshark.

A screenshot of a computer

Description automatically generated

Figure 2.13 – Filter in wireshark by tcp port 22.

A screenshot of a computer

Description automatically generated

Figure 2.14 – SFTP login is captured inside of wireshark as it’s unencrypted.



Figure 2.15 – Upload the file using sftp

A black and white screen with a check mark in a circle

Description automatically generated

Figure 2.16 – Lab is complete.

# 3.0 Introduction – Network Security and Diagnostics

In this lab, the different types of firewall rules and how they are added and modified are tested. The configuration of a firewall is critical to the defense of a network. Furthermore, it is important to not only be able to add rules, but also understand what each rule is doing. To do this, technology such as traceroute as well as Nmap are used to attempt to test the configuration of each ruleset. When looking at firewall rules, the important part to track is around what each port handles. There are several ‘universal’ ports that are designated to different types of technologies. For example, 443 is for https while 80 is for http. These common ports are at the highest criticality to protect when doing firewall configuration on a local server.

Firewall configurations are based on inbound and outbound rulesets. Inbound reflects the requestor and if they have the permissions to access said port. In allowing a port to be open, a ruleset can have a direct IP, a CIDR range of IPs, or a completely open rule. The other side is around the outbound rulesets, which represent the server itself and the response to said actions. If these are not open, many services such as web access and file transfer can be blocked and unusable from the server.

Iptables host the rules that are used across the board for firewall requests and are critical in definition and importance. Throughout, understanding how, why, and application of said rules in the table are crucial for the defense of a server's network and network connections.

# 3.1 Lab Results – Network Security and Diagnostics

A screenshot of a computer

Description automatically generated

Figure 3.0 – load up firewall rules for filtering traffic at layer 7.

A screenshot of a computer

Description automatically generated

Figure 3.1 – Capture routes out to route-out.txt file.

A computer screen shot of a computer program

Description automatically generated

Figure 3.2 – save ifconfig eth0 to file ifconfig-out.txt file.

A black screen with white text

Description automatically generated

Figure 3.3 – get the iptables available options using -S.

A screen shot of a computer

Description automatically generated

Figure 3.4 – show the option list of the rules in the chain using -L.



Figure 3.5 – adding rule to allow for icmp pings to be accepted.

A screen shot of a computer

Description automatically generated

Figure 3.6 – run nmap against 40 and 443 ports.

A screenshot of a computer program

Description automatically generated

Figure 3.7 - Ping the Apache ip, since icmp is open we can see it works.

A screen shot of a computer

Description automatically generated

Figure 3.8 – Curl against the Apache ip on port 443, it is reachable.

A black screen with white text

Description automatically generated

Figure 3.9 – add in new rules to the firewall and run a traceroute.

A screen shot of a computer

Description automatically generated

Figure 3.10 – List all the new firewall rules that have been added.

A screen shot of a computer

Description automatically generated

Figure 3.11 – Add in new rules to the firewall and out put them.

# 4.0 References:

[1] Khess. (2020, January 27). Sysadmin tools: How to use iptables. Enable Sysadmin. https://www.redhat.com/sysadmin/iptables

‌ [2] OWASP. (2013). SQL Injection. OWASP. <https://owasp.org/www-> community/attacks/SQL\_Injection

[3] PortSwigger. (2019). What is reflected XSS (cross-site scripting)? Tutorial & Examples. Portswigger.net. https://portswigger.net/web-security/cross-site-scripting/reflected

[4] 3 Types of Cross-Site Scripting (XSS) Attacks. (2023, May 11). Trend Micro. https://www.trendmicro.com/en\_za/devops/23/e/cross-site-scripting-xss-attacks.html

‌

‌

‌

# 5.0 Activity Log

| **Member Name** | **Task Date** | **Task Details** |
| --- | --- | --- |
| Carl Laneave | 10/15/2023 | Created Template, executed all labs, took screenshots, and completed report |
|  |  |  |
|  |  |  |