DOCUMENTATION: PROJECT: BUILD AND CONFIGURE A FIREWALL

Objective: The goal of this project is to build and configure a firewall to protect a network from unauthorized access and potential security threats.

➔ I used an **Ubuntu virtual machine** to carry out this project.

Step 1: Update your system

Here in this step, I ensured that the system is up to date. The commands that I used are as follows:

sudo apt update

sudo apt upgrade -y

apt list - -upgradable. This command is used to upgrade all the packages that are ready for

Step 2: Install UFW

I installed the ufw by using the command given below:

sudo apt install ufw upgrade.

Step 3: Enable UFW

sudo ufw enable. This command activates the Uncomplicated Firewall (UFW) on a Linux system, applying all configured firewall rules to control incoming and outgoing network traffic. This ensures that the firewall starts enforcing security policies, typically blocking incoming connections while allowing outgoing connections unless specified otherwise.

Step 4: Allow SSH Connections

sudo ufw allow ssh. This command configures the Uncomplicated Firewall (UFW) to allow incoming SSH (Secure Shell) connections on port 22. This is essential for remote access to the

Step 5: Allow Specific Services and Ports

a. Allow HTTP and HTTPS traffic:

sudo ufw allow http

sudo ufw allow https

The commands `sudo ufw allow http` and `sudo ufw allow https` configure the Uncomplicated Firewall (UFW) to allow incoming web traffic:

- sudo ufw allow http: Allows incoming HTTP connections on port 80, enabling access to websites served over HTTP.

- sudo ufw allow https: Allows incoming HTTPS connections on port 443, enabling access to websites served over HTTPS, which is encrypted and more secure.

These commands ensure that your web server can handle both standard and secure web traffic system, ensuring that SSH traffic is permitted while the firewall is active.

b. Allow other specific ports: So here for this command, I allowed traffic on a specific port 8080. The command used is **sudo ufw allow 8080/tcp**.

c. Allow a range of ports:

sudo ufw allow 1000:2000/tcp

This command configures the Uncomplicated Firewall (UFW) to allow incoming TCP traffic on all ports in the range from 1000 to 2000. This is useful when you need to open multiple ports for a specific application or service that requires a range of TCP ports

d. Allow specific IP addresses:

sudo ufw allow from ip address. This command allows incoming traffic from IP address

e. Allow specific subnets:

sudo ufw allow from subnets. This command allows incoming traffic from the subnets specified subnets.

Step 6: Deny Specific Services and Ports

a. Deny a specific port:

sudo ufw deny 23/tcp. The command `sudo ufw deny 23/tcp` blocks incoming TCP traffic on port 23. In brief, it denies access to the Telnet service, which operates on port 23, enhancing security by preventing potential unauthorized access through Telnet.

Step 7: View UFW Status and Rules: In this section, I checked the status of UFW and viewed the current rules. The command I used is sudo ufw status verbose. In brief, it shows whether UFW is active or inactive, and lists all configured firewall rules with more detailed descriptions.

Step 8: Delete UFW Rules: In this section, I checked the rules, and tried deleting a rule by specifying the number.

a. **Using rule number:** listed the rules with numbers. The command used is: **sudo ufw status numbered.** I deleted a specific rule by specifying the number of the rule. The command used is sudo ufw delete 1.

b. I deleted a port number, the command used is **sudo ufw delete allow 8080/tcp**. This command removes a specific firewall rule that allowed incoming TCP traffic on port 8080. This action blocks any new connections to port 8080, which is commonly used for web servers and other services. By deleting this rule, you are enhancing the security of your system by restricting access to services running on this port.

Step 9: Advanced UFW Configuration

a. Logging: Enabling logging to monitor UFW activity, command used sudo ufw logging on. The command `sudo ufw logging on` enables the logging feature of the Uncomplicated Firewall (UFW). This logs firewall activity, including both allowed and denied connections, and stores the logs typically in `/var/log/ufw.log`. Enabling logging helps with monitoring network traffic, diagnosing issues, and enhancing security by keeping track of potential threats and unauthorized access attempts.

b. Default Policies: Setting policies to deny all incoming and allow all outgoing traffic. The commands used are: sudo ufw default deny incoming and **sudo default allow outgoing**. **sudo ufw default deny incoming:** Blocks all incoming network traffic by default, unless explicitly allowed by specific rules. This enhances security by preventing unauthorized access sudo ufw default allow outgoing: Allows all outgoing network traffic by default, ensuring that applications and services can communicate outward without restrictions.

c. Application Profiles: Here I included profiles for some common applications. The commands used are: sudo ufw app list and sudo ufw allow 'CUPS'. The application that I had in the app list was CUPS, so I added it to the ufw. The command **`sudo ufw app list`** displays a list of all available application profiles that UFW can manage, providing a simplified way to apply firewall rules specific to certain applications. When you use `sudo ufw allow 'CUPS'`, it configures UFW to allow incoming traffic for the Common UNIX Printing System (CUPS), enabling network connections required for remote printing services. This allows you to easily manage and apply firewall rules for CUPS, enhancing the system's security while maintaining necessary functionality.

Step 10: Testing the Firewall

a. In this step, I am testing the firewall connection, by first port scanning using namp, the command I used is nmap -v -A

b. Ping Scan: Purpose: The ping scan is used to determine if a host is responsive and reachable on the network. It sends ICMP echo requests and waits for ICMP echo replies to confirm the presence of the host.

c. Connect Scan: **Purpose**: The connect scan (also known as TCP connect scan) is a basic TCP port scanning technique where Nmap tries to connect to each port on the target

machine to see if it gets a response. It is more through than a ping scan and provides information on which ports are open, closed, or filtered.

d. **Service Scan (NSE - Nmap Scripting Engine): Purpose**: Nmap's Scripting Engine (NSE) allows for more advanced network exploration and vulnerability detection. It runs scripts against the target to gather more detailed information about services, operating systems, or potential vulnerabilities.

Interpretation

* **Closed Ports:** The device is present on the network and responding to ping requests, but it does not have any services accessible through the network (at least on the scanned ports).
* **Security Implications:** Closed ports are generally less of a security risk compared to open or filtered ports, as they do not expose any services to potential attackers.SEP However, further investigation might be necessary to fully understand the security posture of the device and its network configuration