The Diamond Shield

Devashish Gosain

Deadline: Feb. 17 2025 (23 59hrs)

1 Description

There has been a sudden rise in periodic suspicious crawling of important web resources of the country "Vesperia" by a secret agency "obscura", tantamounting to remote surveillance. The state of Vesperia has thus initiated a project called "Diamond Shield" to develop a country-scale firewall and train cyber enthusiasts. You, the citizen of Vesperia, have been hired as a senior trainer in this project, and your first job is to develop a demo for firewalling in the sandbox environment. You need to develop a proof-of-concept (PoC) firewall where specific requests can be blocked by the firewall, which would eventually transform into the Diamond shield. Time is of the essence; the enemy is regularly probing the country's infrastructure, and we need to act fast. Thus, you are expected to create the first working PoC in the next 18 days. Act fast!

1.1 Task 1 (Points: 30)

Requirements:

1. One VM running FreeBSD 13.4 (preferred) with no X window. You could use a very lean configuration — e.g., 2 cores and 2 GB RAM. This should be acting as a L3 forwarding firewall. You need to enable the IP routing for the same. We would rely on FreeBSD `pf` firewall for this exercise.
2. Two other VMs running any OS of your choice (Linux or FreeBSD) with as minimal configuration as possible.
3. The three VMs must be connected using virtual networks.

Objective: The assignment is designed to allow you to get your hands dirty with FreeBSD `pf` (an industry standard firewall framework) to filter traffic. You would require configuring the three VMs using the configurational setup shown in Fig. 1.

a) Assign private IP addresses to each interface so the VMs can ping one another. VM1 (eth0) and VM2 (eth0) should be in the same subnet. VM2(eth1) and VM3 (eth0) should be in the same subnet. [**5 points**]

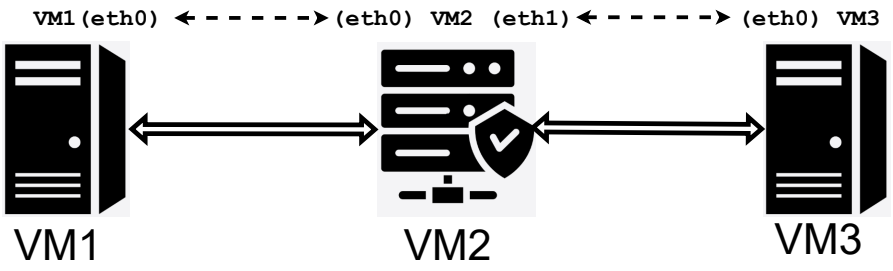

Figure 1: VM network arrangement.

b) Setup IP forwarding on VM2 such that VM1 can ping both interfaces IPs of VM2 and VM3. Make sure traffic from VM1 to VM3 goes via VM2 (can be seen via `traceroute`). [**5 points**]

c) On VM3, install an HTTP server (e.g., `Apache`) on the VM3. Create a web directory and add some files to it. Enable the webserver process on port 80. [**2 points**]

d) You need to use `pf` on VM2, to restrict access *only* to port 80 and not allow anything else. To test the correct functionality, you could install an SSH server on VM3 and try to access it. Correct firewall functionality should restrict access to port 22 (SSH). [**13 points**]

Help: The following links are a good starting point to learn more about FreeBSD `pf`.

- FreeBSD Firewall Book
- Digital Ocean Tutorial
- The book of `pf`: Chapters 2,3 and 5.

Artifact Submission: Please create a "professional" report describing the steps involved in achieving each objective, with corresponding screenshots. Importantly, please write the commands used to enable routing (IPv4 forwarding) on VM2 and firewalling rules with a screenshot showing that it did so successfully. You also show a screenshot where you run a traceroute from VM1 targeted to VM3. [**5 points**]

1.2 Task 2 (Points: 20)

The first task familiarized you with `pf` firewall. This exercise makes you write your own `pf` module to block specific packets. This is akin to `netfilter` kernel modules (in Linux), wherein you can add functionality to add/remove specific

firewall functions by inspecting the header files. For this, you would use the setup used in the first task itself, i.e., with the three VMs.

You would need to write a kernel hook function that blocks inbound `ICMP Echo request` packets. This hook function should be loaded as a kernel module and invoked when appropriate packets arrive on the firewall VM. When the module drops the packets, it should print the number of ICMP packets dropped along with their size on the terminal. To test the firewall functions, `ping` VM3 from VM1 (i.e., via the firewall).

Note: Pings bear `ICMP Echo request` messages to which standard implementations would respond with `ICMP Echo response messages` (i.e., ping responses). Before loading the hook module, you must ensure no existing firewall rules are present. The best practice is to flush the rules before you begin.

Artifact Submission:

1. Submit a makefile that successfully compiles the code. [**5 points**]
2. Submit your source code with a correctly written logic/code execution path and correct functioning. [**10 points**]
3. Description of the systems, commands to execute and test the program, and your assumptions. [**5 points**]