COL8585/COL862 Assignment I: The Platinum Shield

Task 1 Report: Configuring a FreeBSD 13.4 L3 Forwarding Firewall using pf

Arindam Sal (Entry No. 2024JCS2041)
Sambit Paul (Entry No. 2024JCS2038)

1 Overview and Problem Statement

This report documents Task 1 of "The Platinum Shield" assignment. The task requires building a **Layer-3 (L3) forwarding firewall** using **FreeBSD 13.4** and its firewall framework **pf** (Packet Filter).

The firewall must sit between an Ubuntu Client (VM1) and an Ubuntu Server (VM3), and enforce the following traffic policy:

- **Default deny**: block all traffic by default.
- **Allow ICMP**: permit ping for diagnostics.
- **Allow HTTP**: permit TCP port 80 (web access) from client to server.
- **Block SSH**: deny TCP port 22 connectivity after firewall activation.

The assignment also requires evidence:

- Commands used to enable IPv4 forwarding (routing) on VM2.
- Firewall rules in `/etc/pf.conf`.
- Proof of success using screenshots including traceroute from VM1 to VM3.

2 Conceptual Background (Theory)

2.1 Why Layer-3 Forwarding is Needed

VM1 and VM3 are placed in **different subnets**:

- VM1: 10.0.1.0/24
- VM3: 10.0.2.0/24

Hosts in different subnets cannot directly communicate without a router/gateway. Therefore, VM2 must act as an L3 router:

- VM1 sends packets to VM2 (gateway) when destination is outside 10.0.1.0/24.
- VM2 forwards packets out of its other interface toward VM3.
- VM3 replies back via VM2.

This requires **IP forwarding** to be enabled on VM2, otherwise VM2 will receive packets but will not forward them.

2.2 Firewalling with pf (Packet Filter)

`pf` is FreeBSD's stateful firewall system. It processes packets based on rules that match:

- Interface (e.g., em0 or em1)
- Direction (inbound or outbound)
- Protocol (TCP/UDP/ICMP)
- Ports (e.g., 80, 22)
- Source/Destination IP

2.3 Default-Deny (Block All) Security Model

A default-deny policy means:

> "Everything is blocked unless explicitly allowed."

This is the standard enterprise security approach because:

- It minimizes attack surface.
- It prevents unexpected services from being reachable.
- It enforces least privilege (only required traffic passes).

2.4 Stateful Filtering and `keep state`

When pf uses `keep state`, it remembers a connection (e.g., a TCP session). This is important because:

- TCP uses a handshake (SYN, SYN-ACK, ACK).
- If the firewall only allows SYN packets but not reply packets, connections break.
- With state tracking, pf automatically allows return traffic for approved connections.

Thus, `keep state` is necessary for correct TCP/ICMP operation.

3 Network Topology and Addressing

3.1 Topology

- **VM1 (Ubuntu Client)**: 10.0.1.2/24 via enp0s3, Gateway 10.0.1.1
- **VM2 (FreeBSD Firewall)**:
  - em0: 10.0.1.1/24 (client side)
  - em1: 10.0.2.1/24 (server side)
- **VM3 (Ubuntu Server)**: 10.0.2.2/24 via enp0s3, Gateway 10.0.2.1

3.2 VirtualBox Internal Networks (Isolation)

We use VirtualBox **Internal Network** to isolate traffic from the host machine and internet. Two internal networks are created:

- `intnetA`: connects VM1 ↔ VM2(em0)
- `intnetB`: connects VM2(em1) ↔ VM3

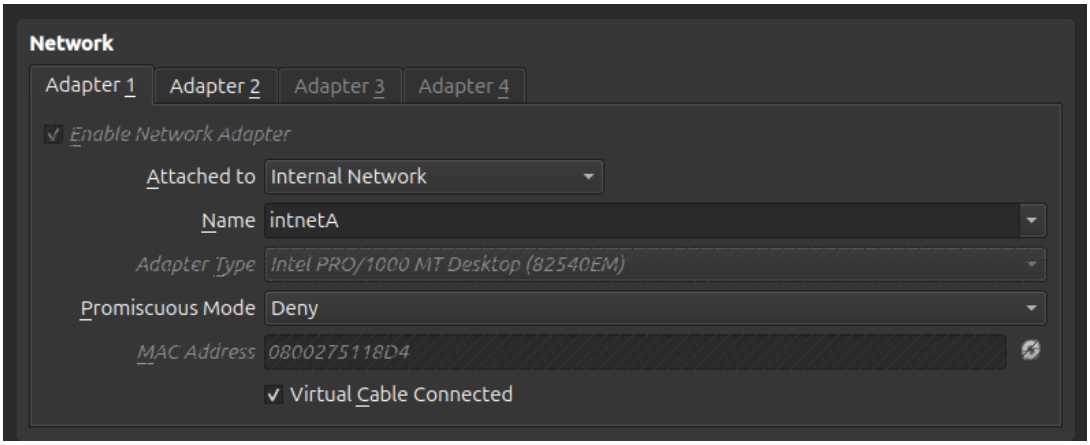

Figure 1: VM2 Firewall Adapter connected to Internal Network `intnetA`

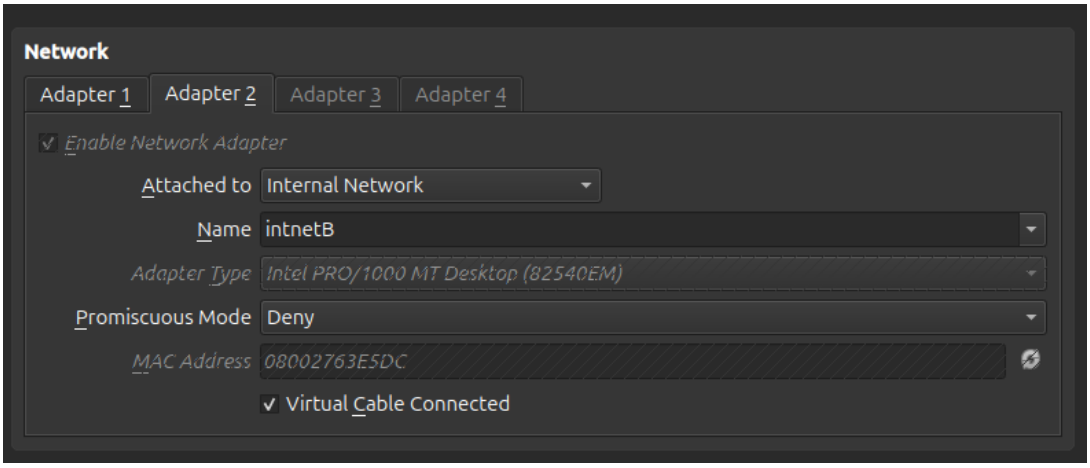

Figure 2: VM2 Firewall Adapter connected to Internal Network `intnetB`

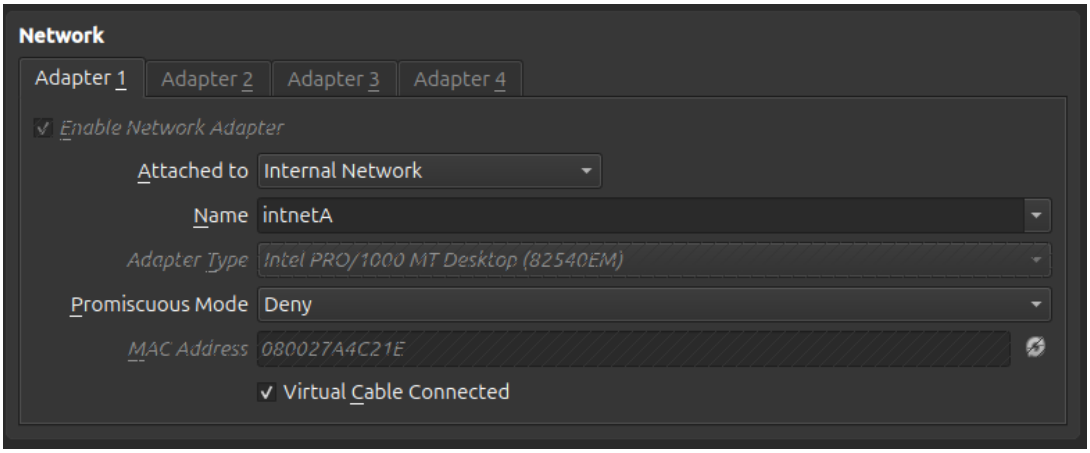

Figure 3: VM1 Client Adapter connected to Internal Network `intnetA`

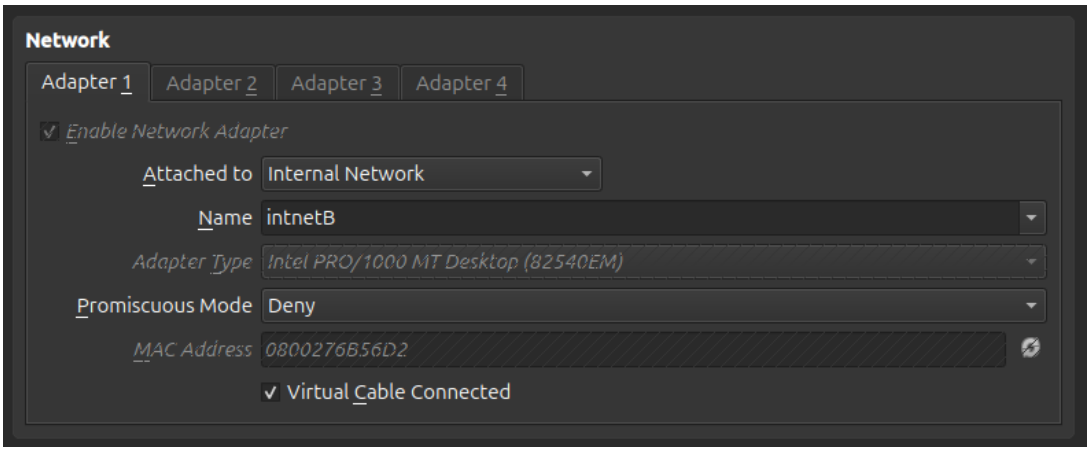

Figure 4: VM3 Server Adapter connected to Internal Network `intnetB`

4 Infrastructure Setup (Static IP and Routes)

4.1 Why Static IP?

Static IP ensures:

- predictable addressing (important for firewall rules)
- correct default gateway behavior
- reproducibility of experiments

4.2 VM1 Configuration (Ubuntu Client)

VM1 is configured with IP 10.0.1.2/24 and gateway 10.0.1.1.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
        - 10.0.1.2/24
      routes:
        - to: default
          via: 10.0.1.1
      nameservers:
        addresses:
          - 8.8.8.8
```

Listing 1: Netplan-style configuration intent for VM1 (static IP + gateway)

Explanation of fields:

- `addresses`: assigns a fixed IP and prefix
- `routes`: sets default route (gateway) to the firewall
- `nameservers`: DNS server (not mandatory for internal access but standard practice)

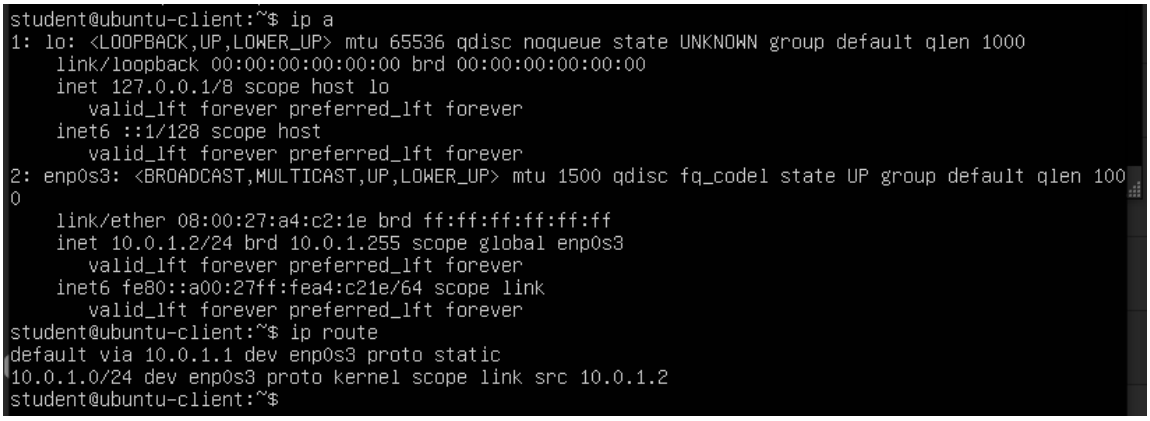

Figure 5: VM1 verification: `ip a` and `ip route` show IP 10.0.1.2/24 and default route via 10.0.1.1

4.3 VM3 Configuration (Ubuntu Server)

VM3 is configured with IP 10.0.2.2/24 and gateway 10.0.2.1.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
        - 10.0.2.2/24
      routes:
        - to: default
          via: 10.0.2.1
      nameservers:
        addresses:
          - 8.8.8.8
```

Listing 2: Netplan-style configuration intent for VM3 (static IP + gateway)

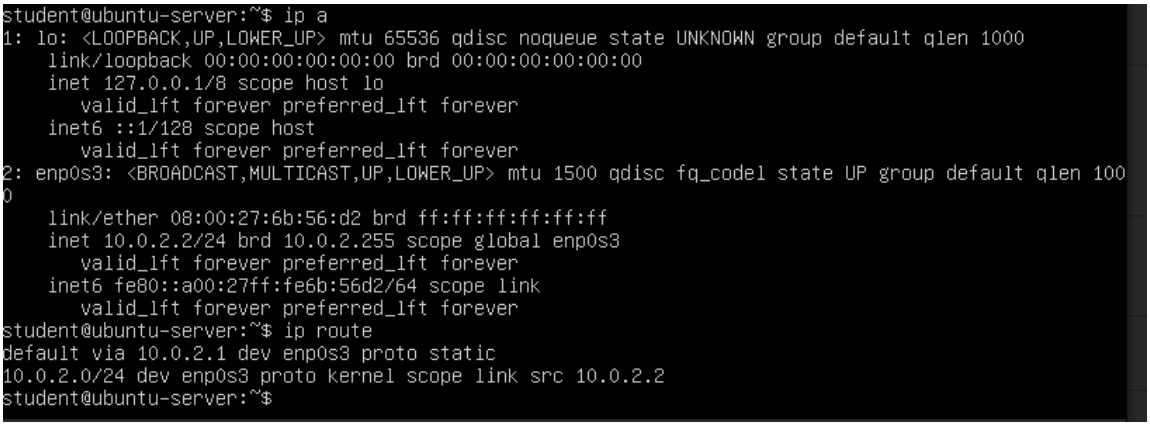

Figure 6: VM3 verification: `ip a` and `ip route` show IP 10.0.2.2/24 and default route via 10.0.2.1

5 Objective 1: Enable Routing (IPv4 Forwarding) on VM2

5.1 Why IPv4 Forwarding is Required

Even if VM2 has two interfaces, it will not act as a router unless forwarding is enabled. Without forwarding:

- VM2 receives packets on em0
- but drops them instead of forwarding to em1

5.2 Commands Used

```
sysctl net.inet.ip.forwarding=1
sysrc gateway_enable="YES"
```

Listing 3: Commands to enable IPv4 forwarding on FreeBSD VM2

Explanation:

- `sysctl net.inet.ip.forwarding=1`: enables forwarding immediately (runtime).
- `sysrc gateway`$_{e}nable = "YES"$: $makesforwardingpersistentafterrebootviarc.conf.$

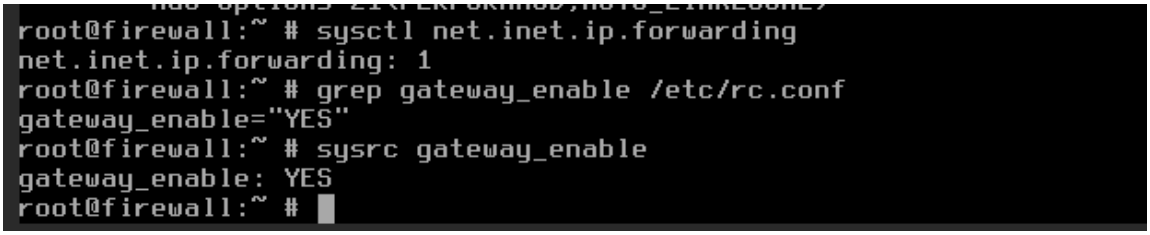

Figure 7: VM2 verification: `net.inet.ip.forwarding=1` and `gateway_enable="YES"`

6 Objective 2: Validate Basic L3 Connectivity (Before Firewall)

6.1 ICMP Testing (Ping)

Ping uses ICMP Echo Request/Reply. Allowing ping is useful because:

- It confirms that routing works.
- It helps isolate issues (IP, gateway, forwarding, firewall).

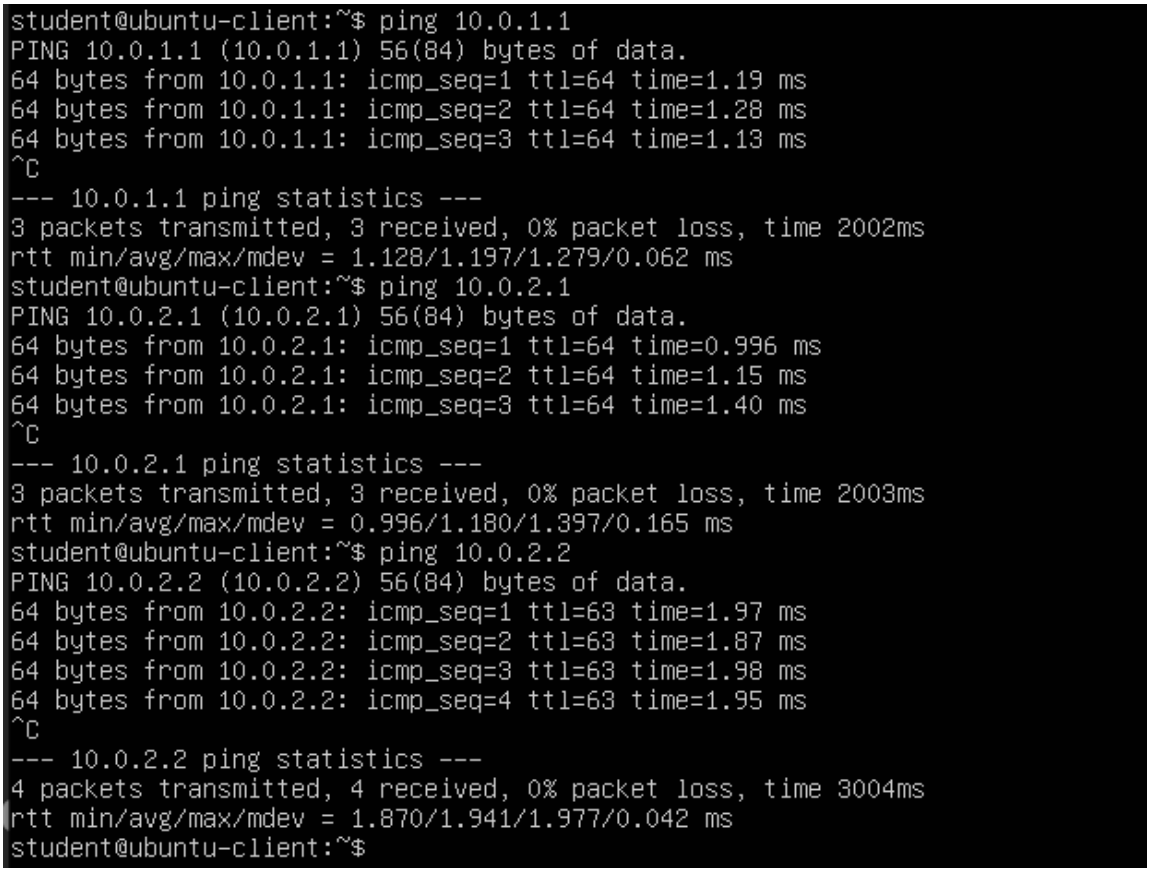

Figure 8: VM1 ping tests to 10.0.1.1 (firewall em0), 10.0.2.1 (firewall em1), and 10.0.2.2 (server)

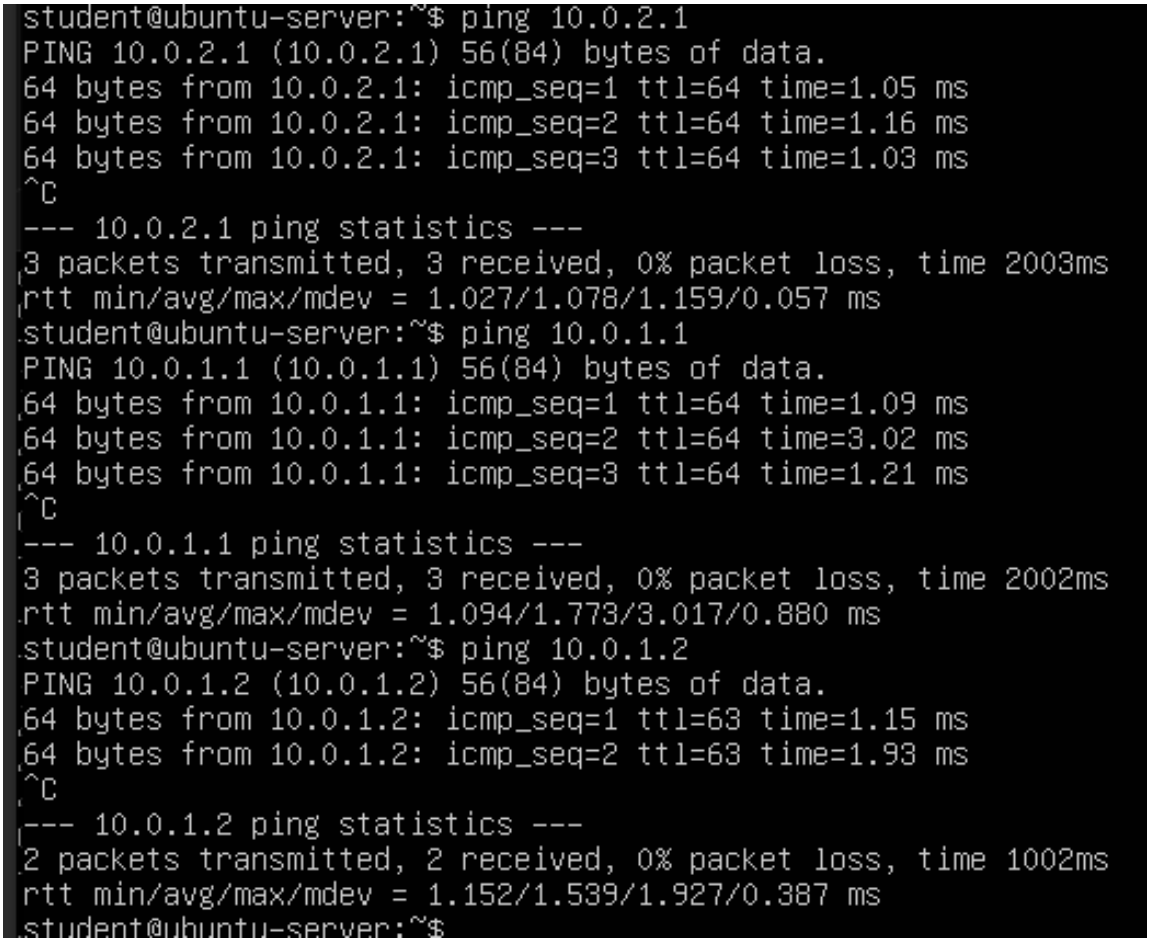

Figure 9: VM3 ping tests to 10.0.2.1 (firewall em1), 10.0.1.1 (firewall em0), and 10.0.1.2 (client)

6.2 Traceroute Requirement (Mandatory Evidence)

Traceroute helps confirm the path taken by packets. In a multi-hop path, traceroute reveals intermediate routers by using increasing TTL values.

How it works (simple explanation):

- TTL starts at 1. First router reduces TTL to 0 and sends ICMP Time Exceeded.
- TTL increases to 2, reaching the next hop, and so on.

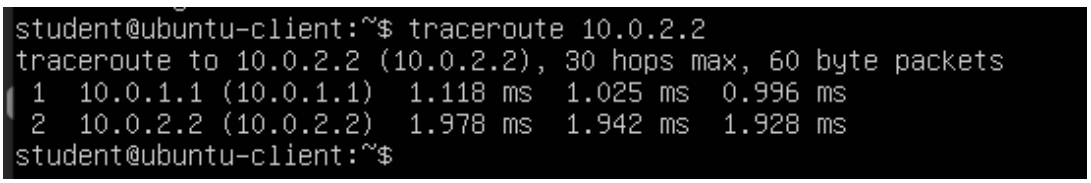

Figure 10: Traceroute from VM1 to VM3: hop 1 is firewall (10.0.1.1), hop 2 is server (10.0.2.2)

Inference: The traceroute output confirms VM2 is functioning as the intermediate L3 device.

7 Objective 3: Deploy Application Service (HTTP) on VM3

7.1 Why HTTP is Chosen

HTTP (port 80) is a realistic service that firewalls commonly allow for controlled access. We run Apache on VM3 so we can verify that the firewall allows port 80 traffic end-to-end.

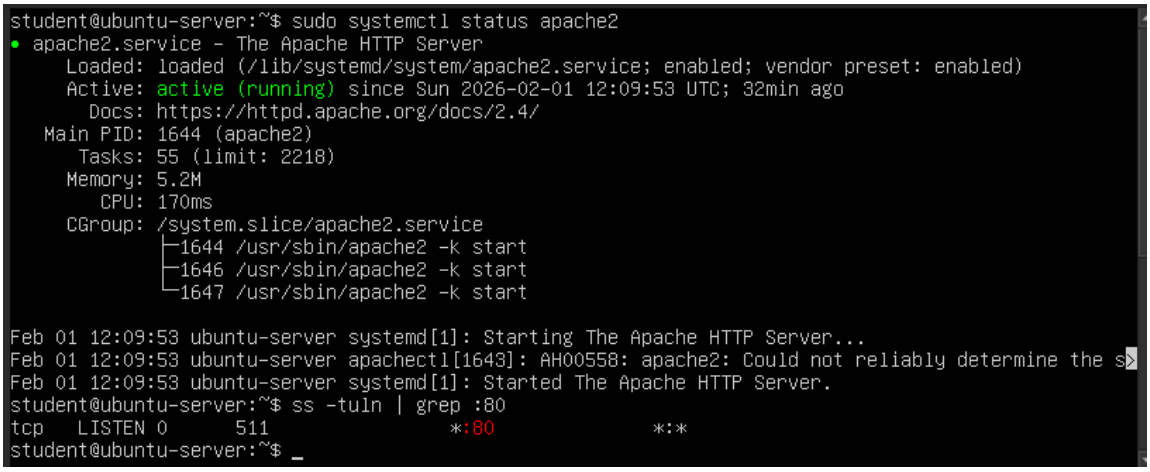

Figure 11: Apache2 status is active and `ss -tuln` confirms port 80 listener on VM3

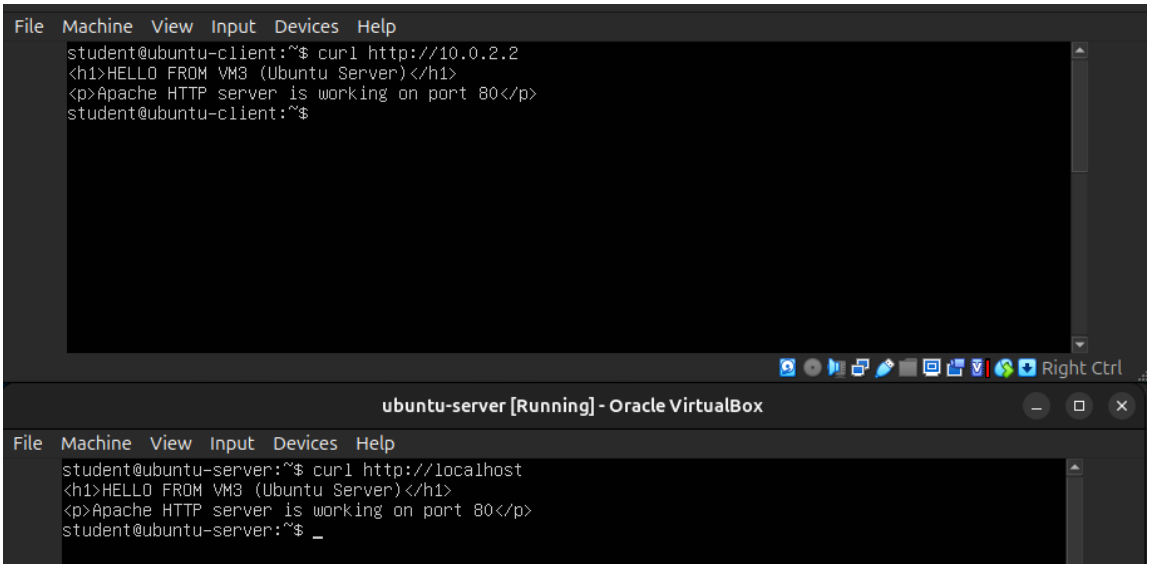

Figure 12: curl from VM1 successfully fetches the HTTP page from VM3 (10.0.2.2)

8 Objective 4: Show SSH Works Before Firewall (Baseline)

Before enforcing firewall rules, we establish a baseline showing SSH connectivity exists.

Why this is important:

- It proves the network path is correct.
- Later, if SSH fails, we can attribute it to firewall enforcement (not routing misconfig).

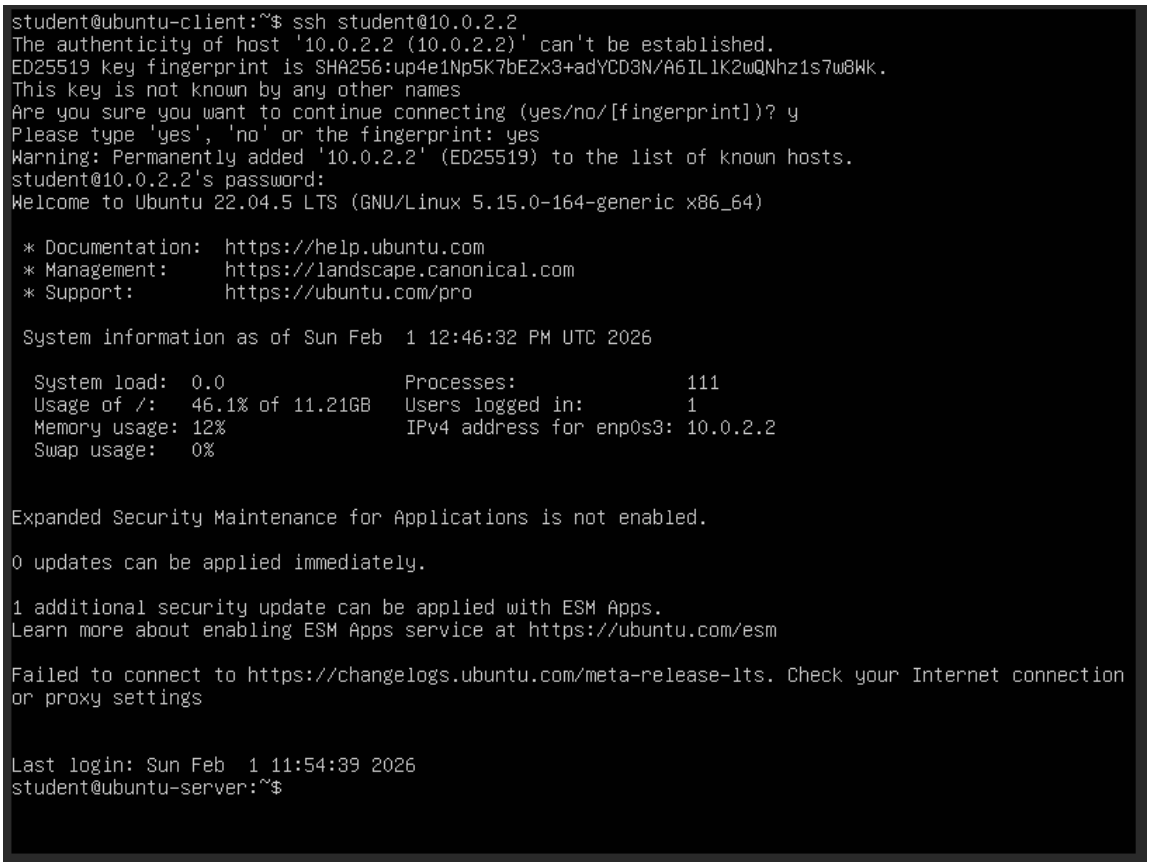

Figure 13: SSH connection from VM1 to VM3 successfully established (baseline before firewall)

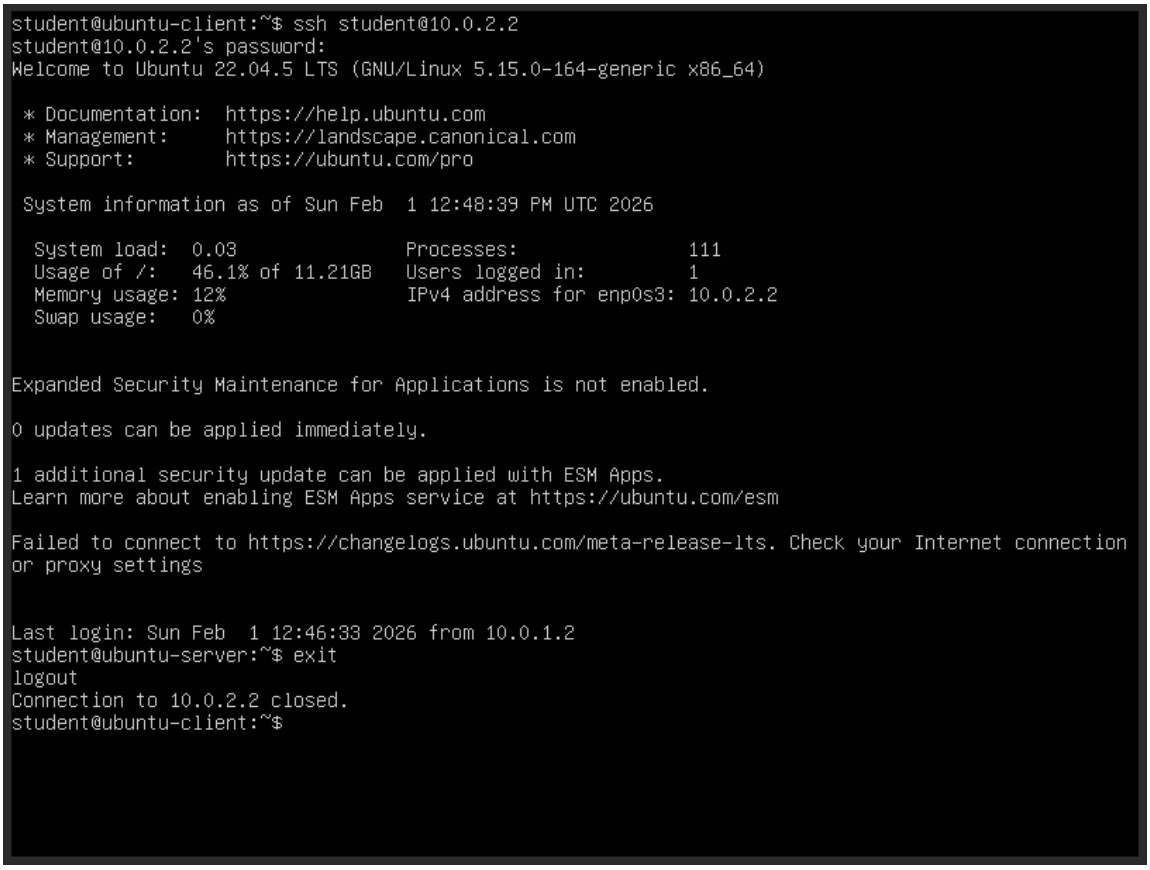

Figure 14: Successful SSH login session and exit (shows SSH works pre-firewall)

9 Objective 5: Implement Firewall Rules using pf on VM2

9.1 Firewall Policy (Default Block + Allow Only Required)

We implement:

- block all by default
- allow ICMP (ping)
- allow TCP/80 (HTTP)
- do not allow TCP/22 (SSH remains blocked)

9.2 pf Rule-set

```
block all

pass inet proto icmp all keep state
```

```
pass in on em0 inet proto tcp from any to 10.0.2.2 port 80 flags
   S/SA keep state
pass out on em1 inet proto tcp from any to 10.0.2.2 port 80 flags
    S/SA keep state
```

Listing 4: /etc/pf.conf rules used in Task 1

9.3 Explanation of Each Rule in Detail

- `block all`: denies everything unless a later rule allows it (default deny).
- `pass inet proto icmp all keep state`: allows ICMP in IPv4 for diagnostics.
- `pass in on em0 ... port 80`: allows client-side incoming HTTP requests towards VM3.
- `pass out on em1 ... port 80`: allows those HTTP packets to leave toward server-side network.
- `flags S/SA`: ensures the rule matches TCP connection initiation correctly.
- `keep state`: ensures reply traffic is allowed automatically.

9.4 Enabling pf (Commands and Verification)

```
kldload pf
sysrc pf_enable="YES"
pfctl -nf /etc/pf.conf
pfctl -f /etc/pf.conf
pfctl -e
pfctl -sr
```

Listing 5: Commands to load and enable pf

What these commands do:

- `kldload pf`: loads pf kernel module.
- `sysrc pf`$_{e}nable = "YES"$: $enablespfserviceatboot.$
- `pfctl -nf: syntax check of the pf.conf file (safe validation).`
- `pfctl -f: loads the rules into the kernel.`
- `pfctl -e: enables packet filtering (if not already enabled).`

- `pfctl -sr:  prints active rule-set (mandatory proof).`

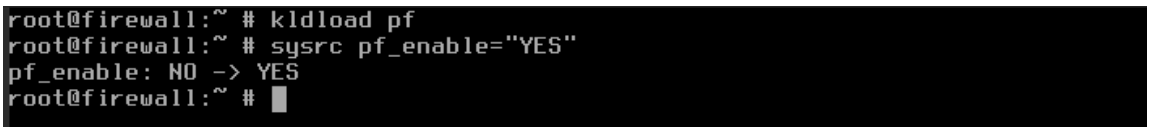

Figure 15: Loading pf module and enabling pf service on VM2

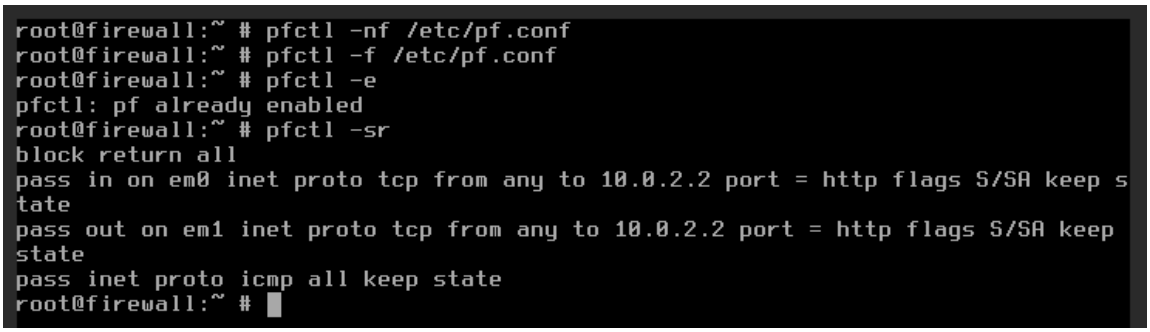

Figure 16: pf rule-set shown using `pfctl -sr` (proof that rules are active)

10 Objective 6: Verification After Firewall Enforcement

Now we test the required outcomes:

- HTTP should pass
- SSH should be blocked
- ICMP should still pass

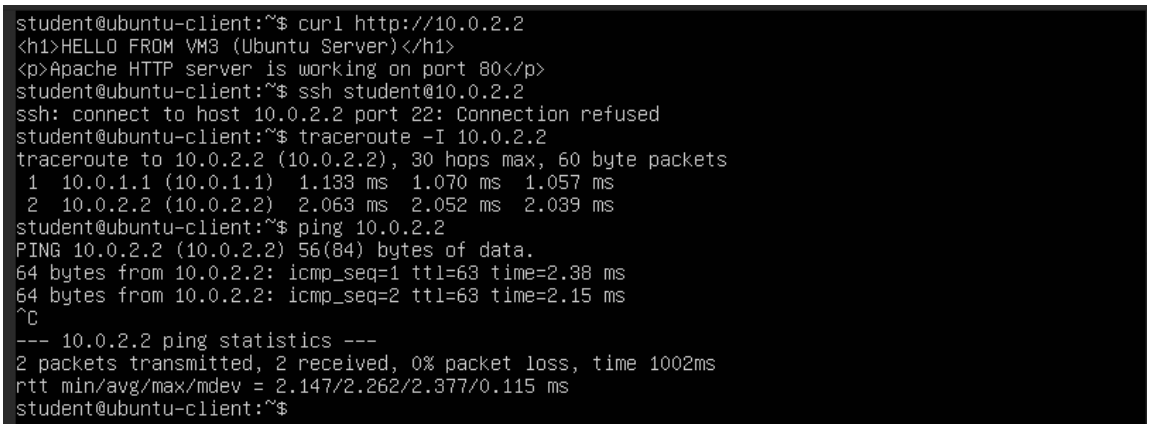

Figure 17: Verification: curl (HTTP) works; SSH fails; ICMP ping works

10.1 Why SSH Fails

SSH uses TCP port 22. Since we never explicitly allowed port 22 in pf.conf:

- `block all` applies
- SSH packets are dropped/blocked
- connection fails

This confirms correct enforcement of least privilege.

11 Objective 7: Firewall Logging (pflog) and Evidence

Logging provides visibility and forensic evidence. It shows exactly which packets are blocked and why.

What is pflog?

- `pflog` is a pseudo-interface used by pf for logging.
- Logs can be viewed via tcpdump on `pflog0`.

```
tcpdump -n -e -ttt -i pflog0
```

Listing 6: Typical command to view logs

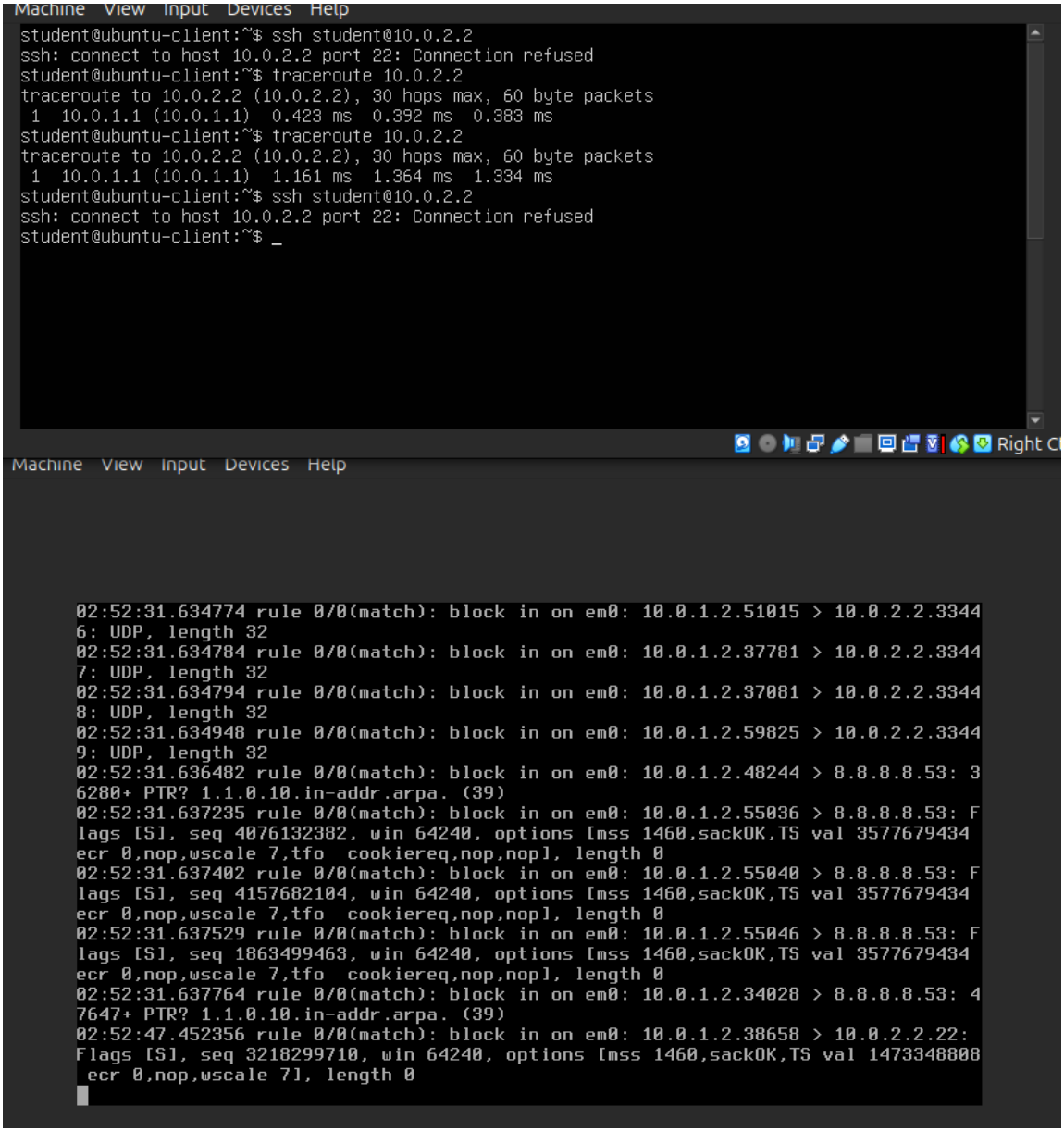

Figure 18: Live pflog output showing blocked SSH and other packets

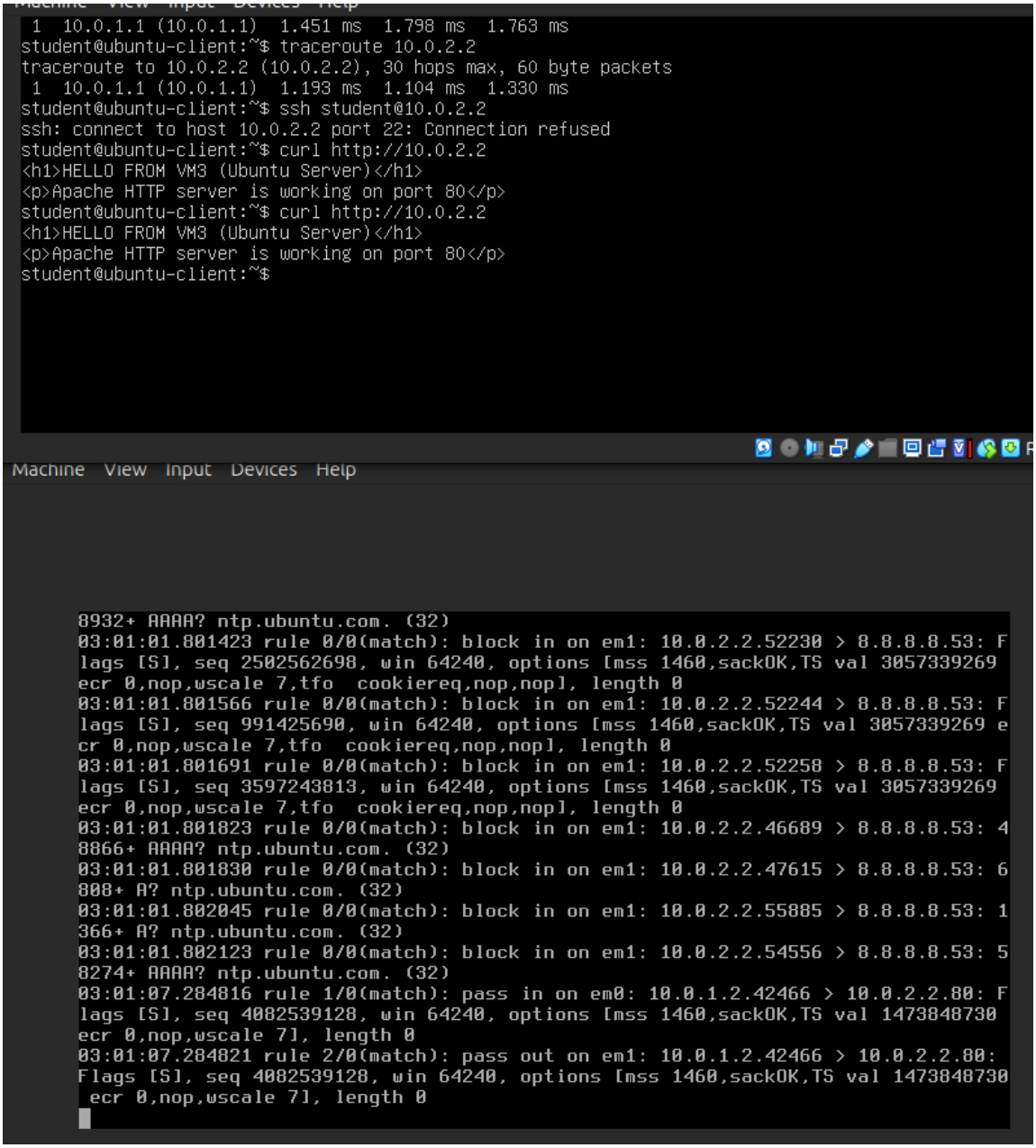

Figure 19: Additional pflog evidence: blocked traffic and allowed HTTP traffic

12 Conclusion

Task 1 was completed successfully by configuring VM2 (FreeBSD 13.4) as an L3 forwarding firewall using pf.

Final outcomes:

- L3 forwarding enabled and verified (`net.inet.ip.forwarding=1`)
- Routing path verified using traceroute (VM1 → 10.0.1.1 → VM3)
- pf deployed with a default deny posture

- ICMP allowed for diagnostics
- HTTP (TCP/80) allowed and verified using curl
- SSH (TCP/22) blocked after firewall activation
- Logging confirmed using pflog evidence

This setup reflects a real-world secure gateway firewall design enforcing least privilege.