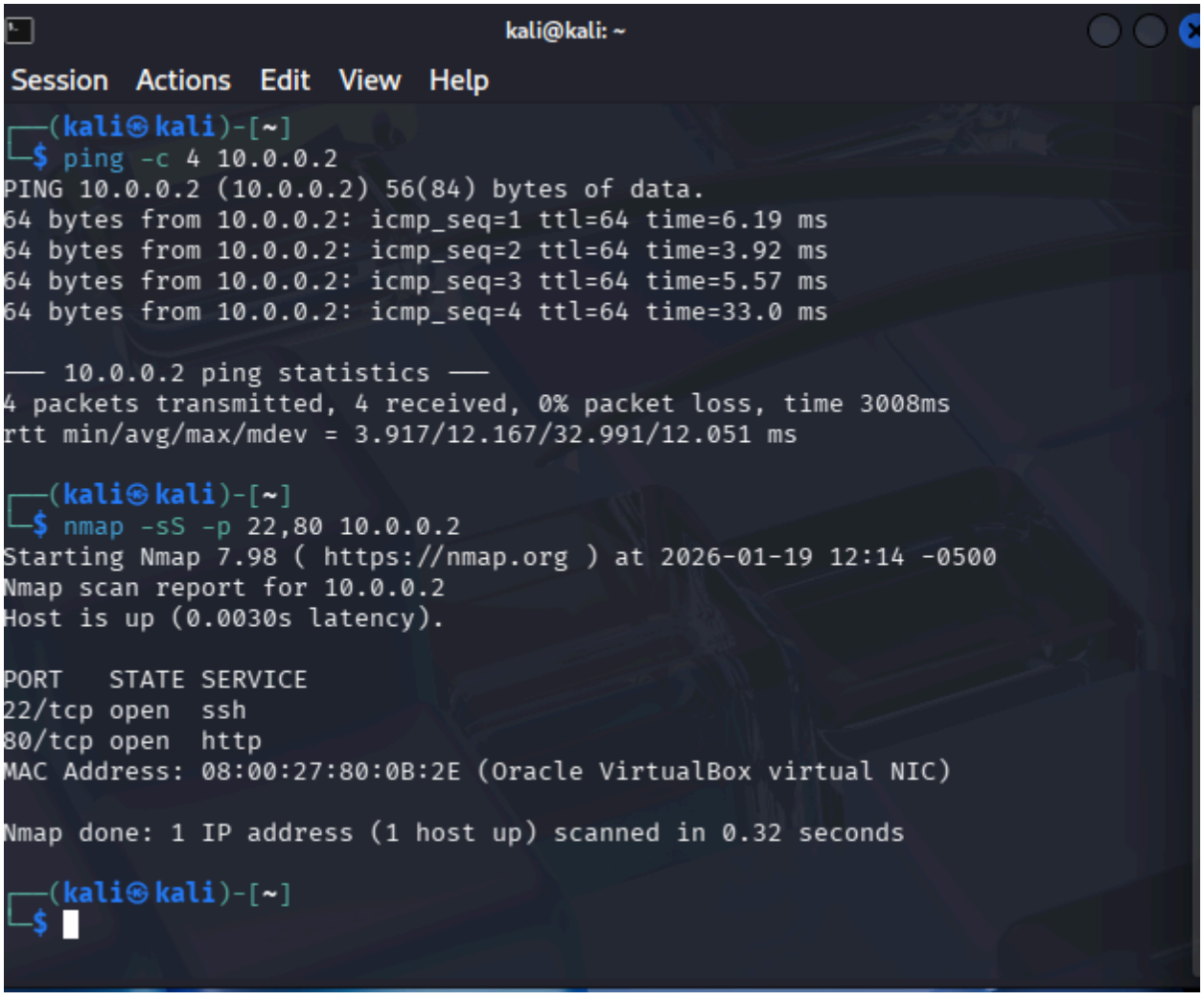
Day 3 – Traffic Control & Firewall Hardening (UFW)

Mindset today:
You are defending a server in a real company.
Nothing is trusted. Everything must be proven.

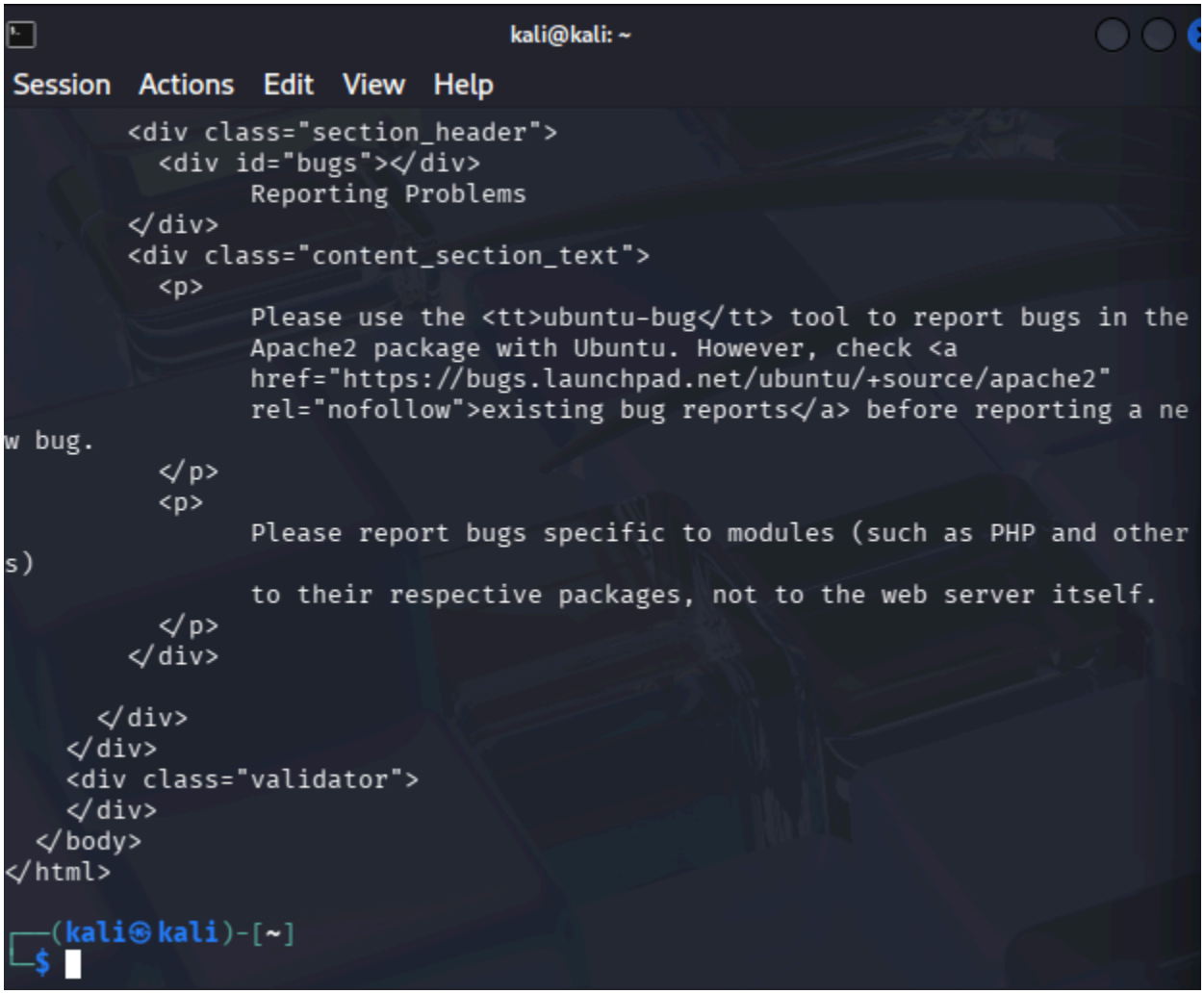
STEP 1: Baseline Scan (PROOF OF INSECURITY):
The command we have use in step 1 is nmap(network mapper) to find the open ports (Nmap -sS -p 22,80 10.0.0.2) :(-sS is stealth scan (fast n quite) -p this tell nmap to check only two ports 20,80,and my target is (10.0.0.2)ubuntu

Proof:



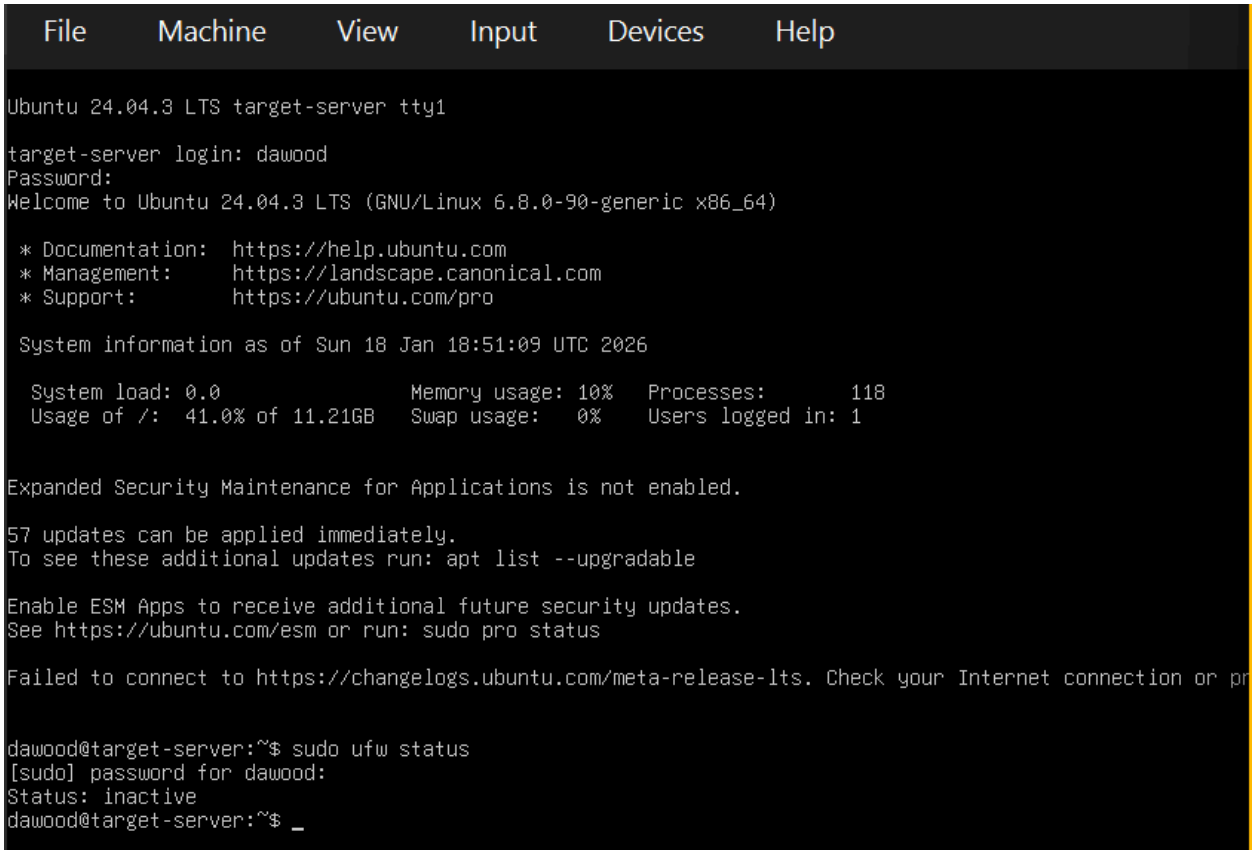
STEP 2: Confirm Web Access (Kali)
The command we are using in this is(curl http://10.0.0.2)
(curl) client url ,this tools act like a web browser in terminal to capture websites data

Proof



STEP 3: Move to Defense (Ubuntu)
The command we are using is (sudo ufw status)
ufw(uncomplicated firewall)
Mission Report: Run that command on Ubuntu now. **Does it say `Status: inactive`?**

Proof:

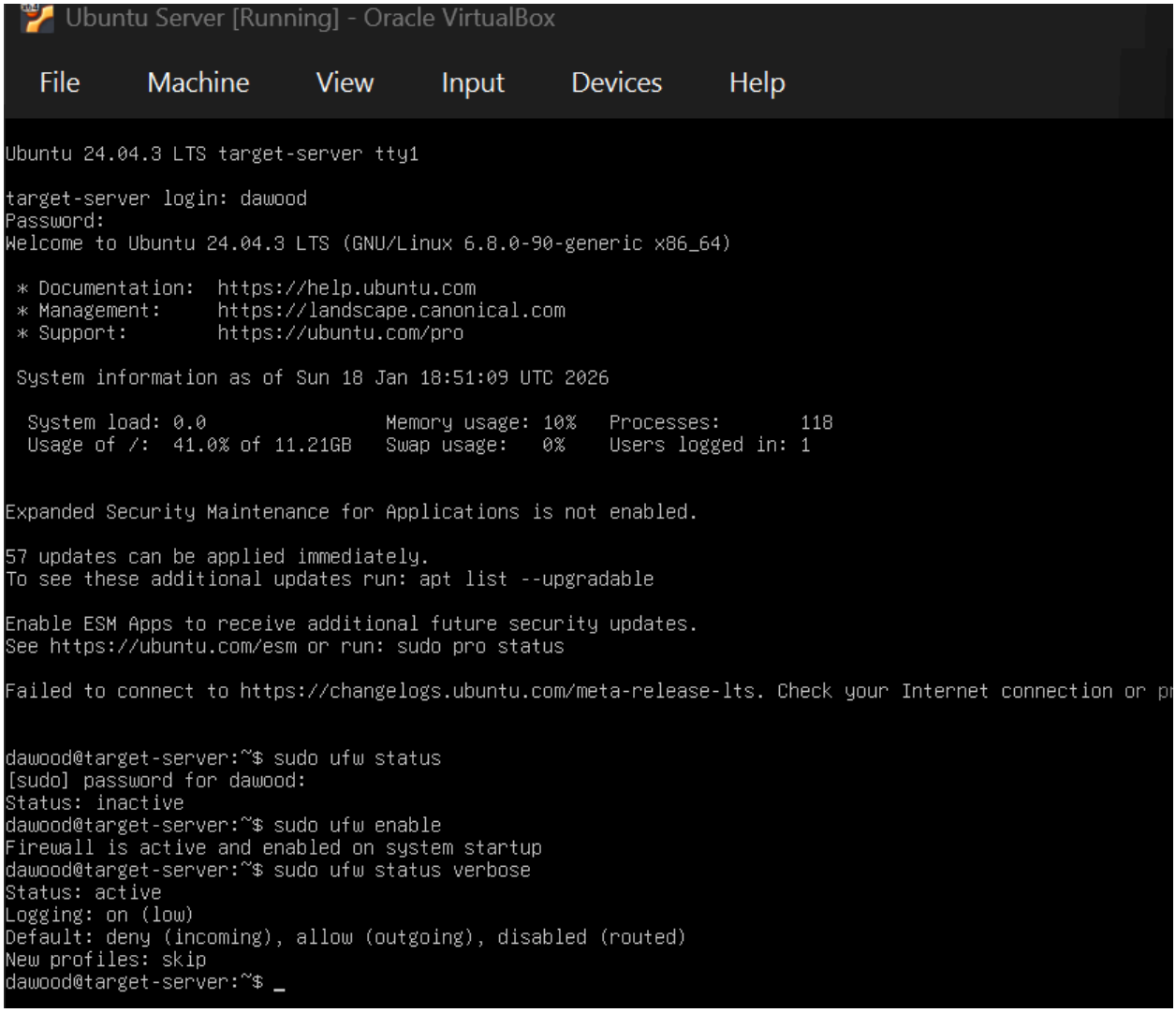


STEP 4: Enable Firewall (CRITICAL MOMENT)
The command we are using is (sudo ufw enable)
To see full detail use this command (sudo ufw status verbose)

Look closely at the output to confirm these three things:

- **Status:** `active`
- **Logging:** `on (low)` **(usually)**
- **Default:** `deny (incoming)`, `allow (outgoing)`, `disabled (routed)`

Proof

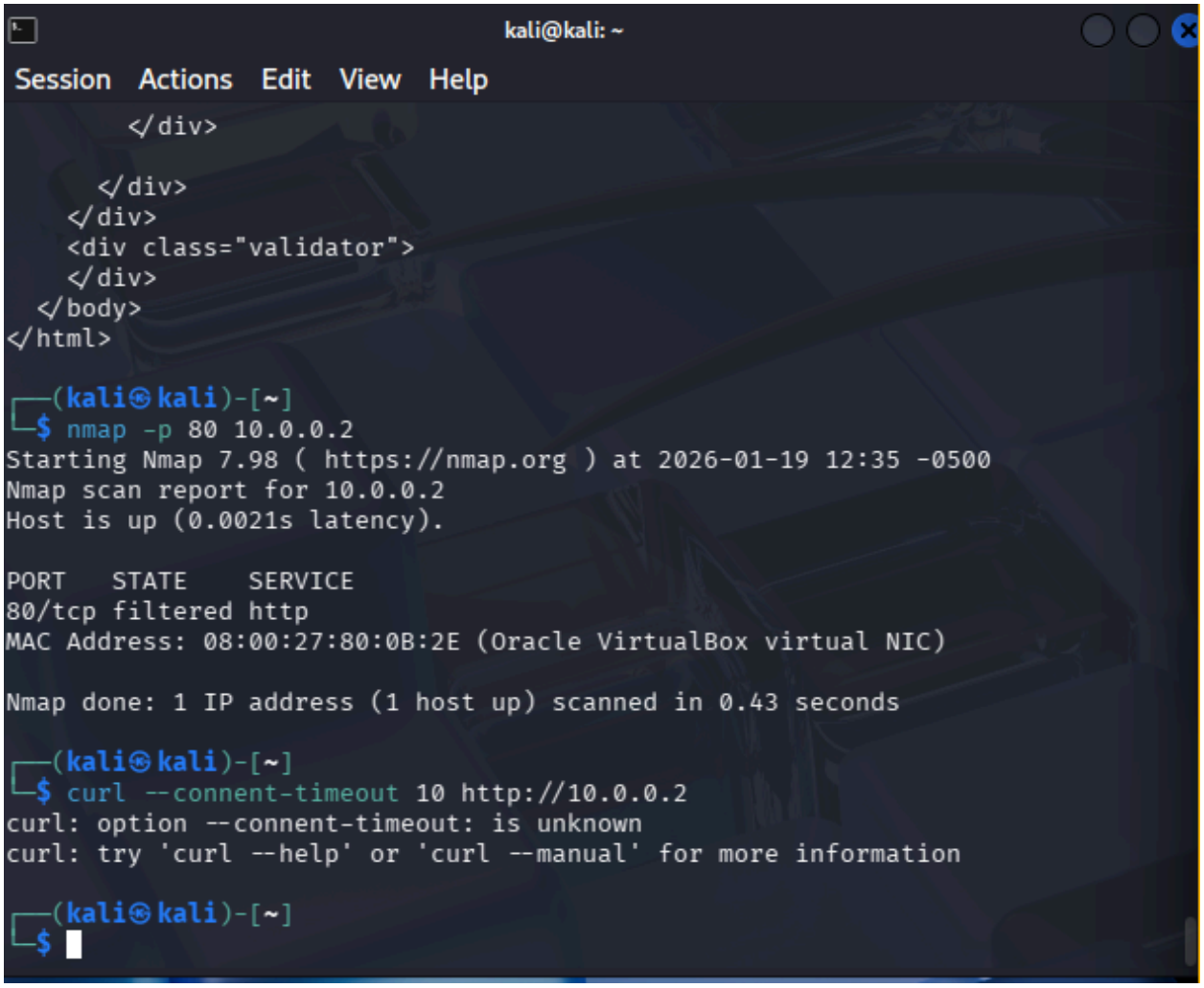


STEP 5: Test the Block (ATTACK FAILS)
The command we are using in this is (nmap -p 80 10.0.0.2)

The Tool Nmap & Curl — We use the same tools as before to show the difference between "Unprotected" and "Protected".

Now user experience verification(curl) command is (curl --connect-timeout 10 http://10.0.0.2)

Proof

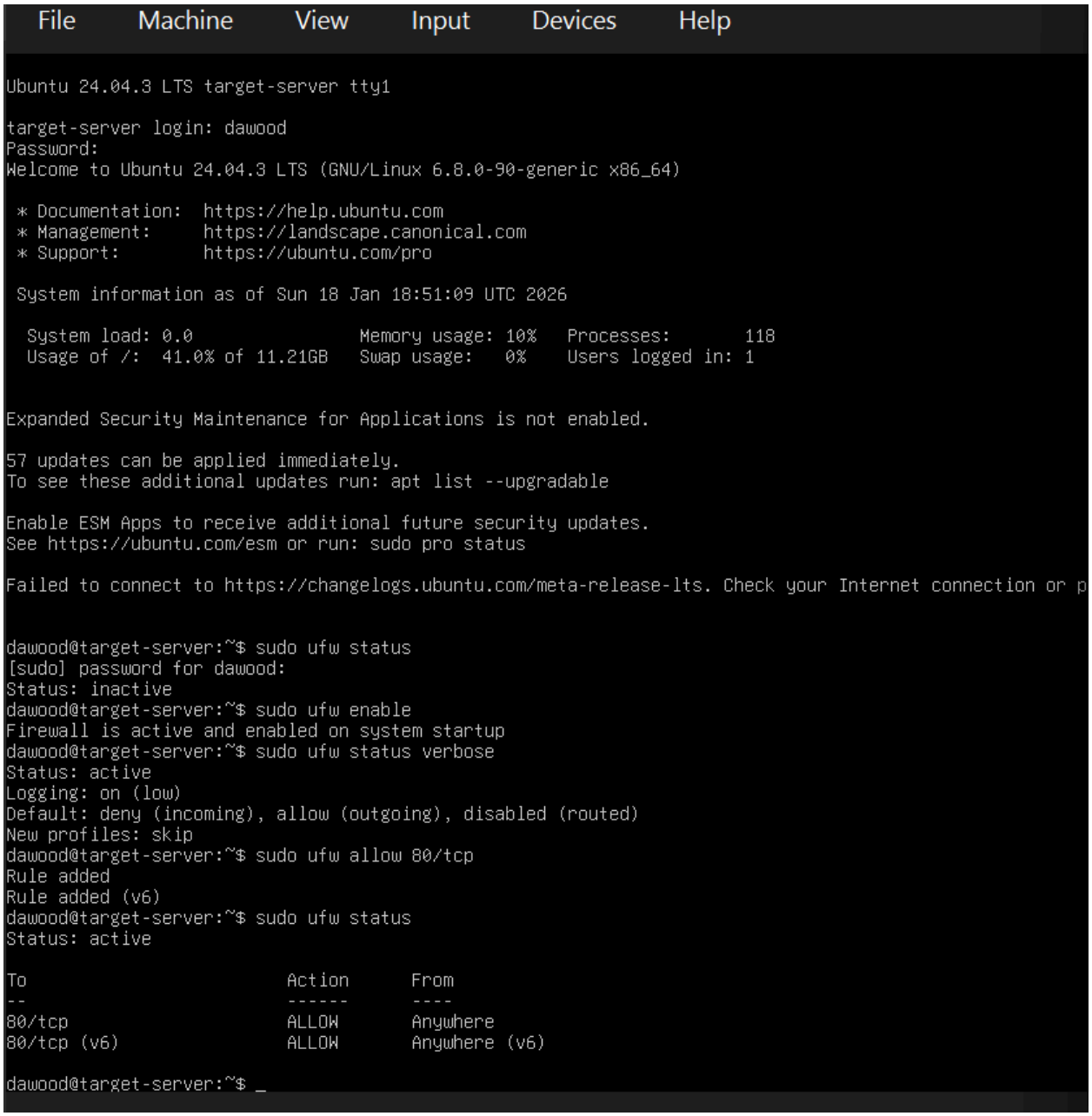


STEP 6: Allow HTTP (CONTROLLED ACCESS)

The command we are using in this is (sudo ufw allow 80/tcp)
The command we use for active rule is (sudo ufw status)

📍 **The Tool UFW (Uncomplicated Firewall)** — We are adding a specific "Allow" exception to the "Default Deny" wall you built in Step 4.

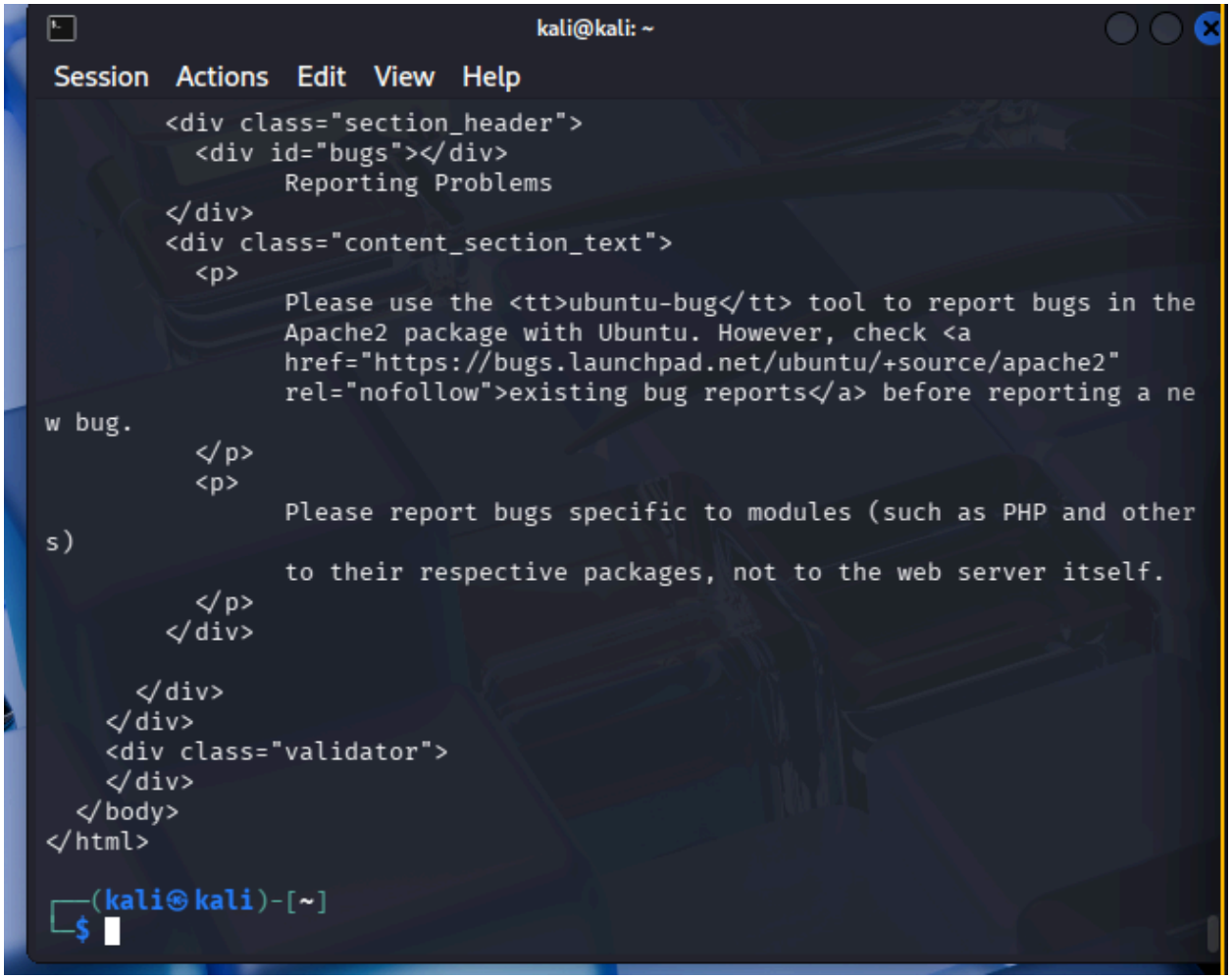
Proof



STEP 7: Retest Access (CONTROL RESTORED)
The command we use is (curl http://10.0.0.2)

📍 **The Tool curl** — You are using this to prove that the "Allow" rule you just created on Ubuntu is working.

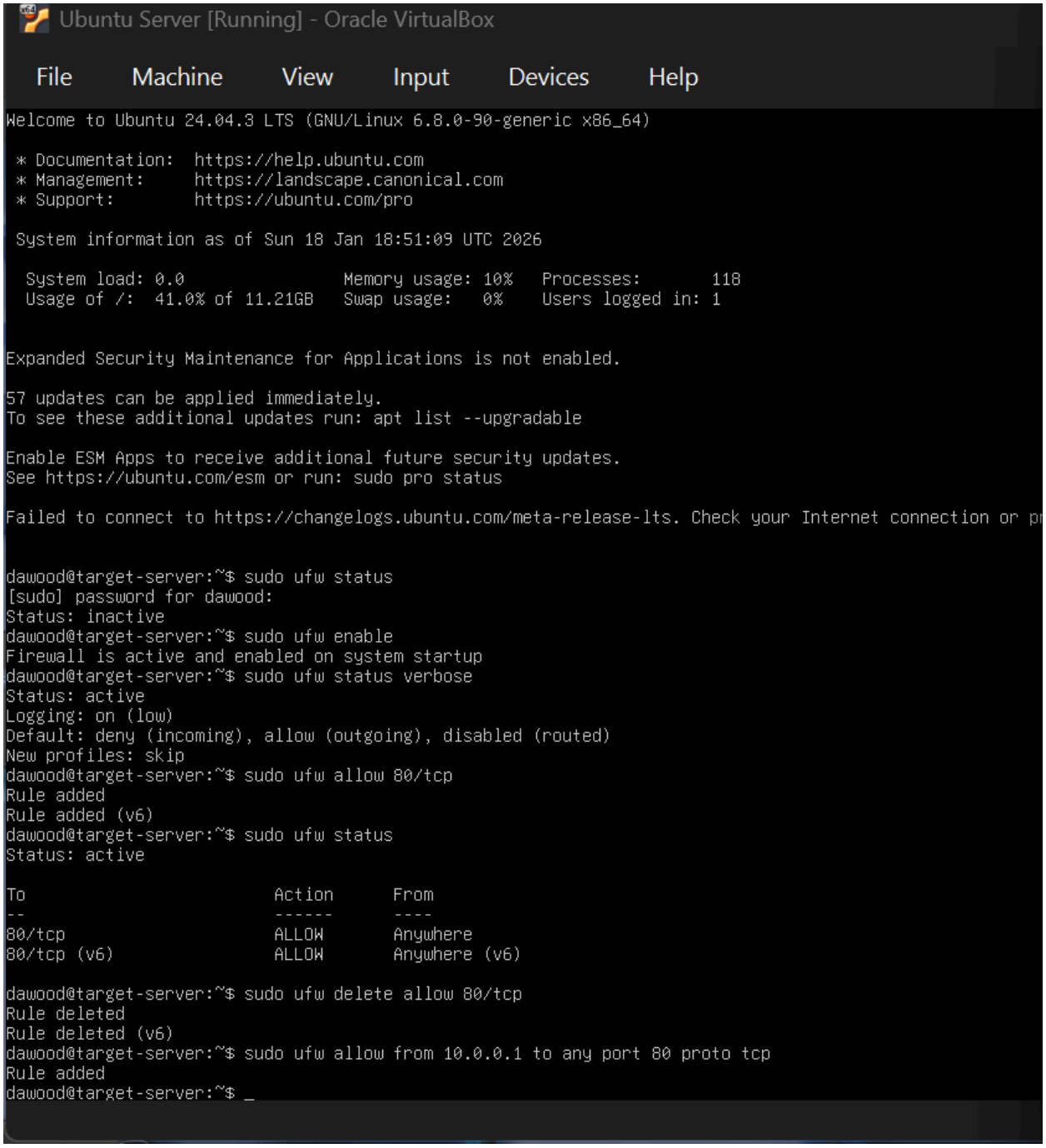
Proof



STEP 8: Restrict Access (ENGINEER MOVE)
The command we are using is (sudo ufw delete allow 80/tcp)
After this now we will create specialized rule that only kali vm to see (sudo ufw allow from 10.0.0.1 from any port 80 proto tcp)
The command to see configuration (sudo ufw status)

📍 **The Concept** Instead of saying "Everyone can see my website," you are saying "Only the administrator at IP `10.0.0.1` (Kali) can see my website". Everyone else on the network will still see a brick wall.

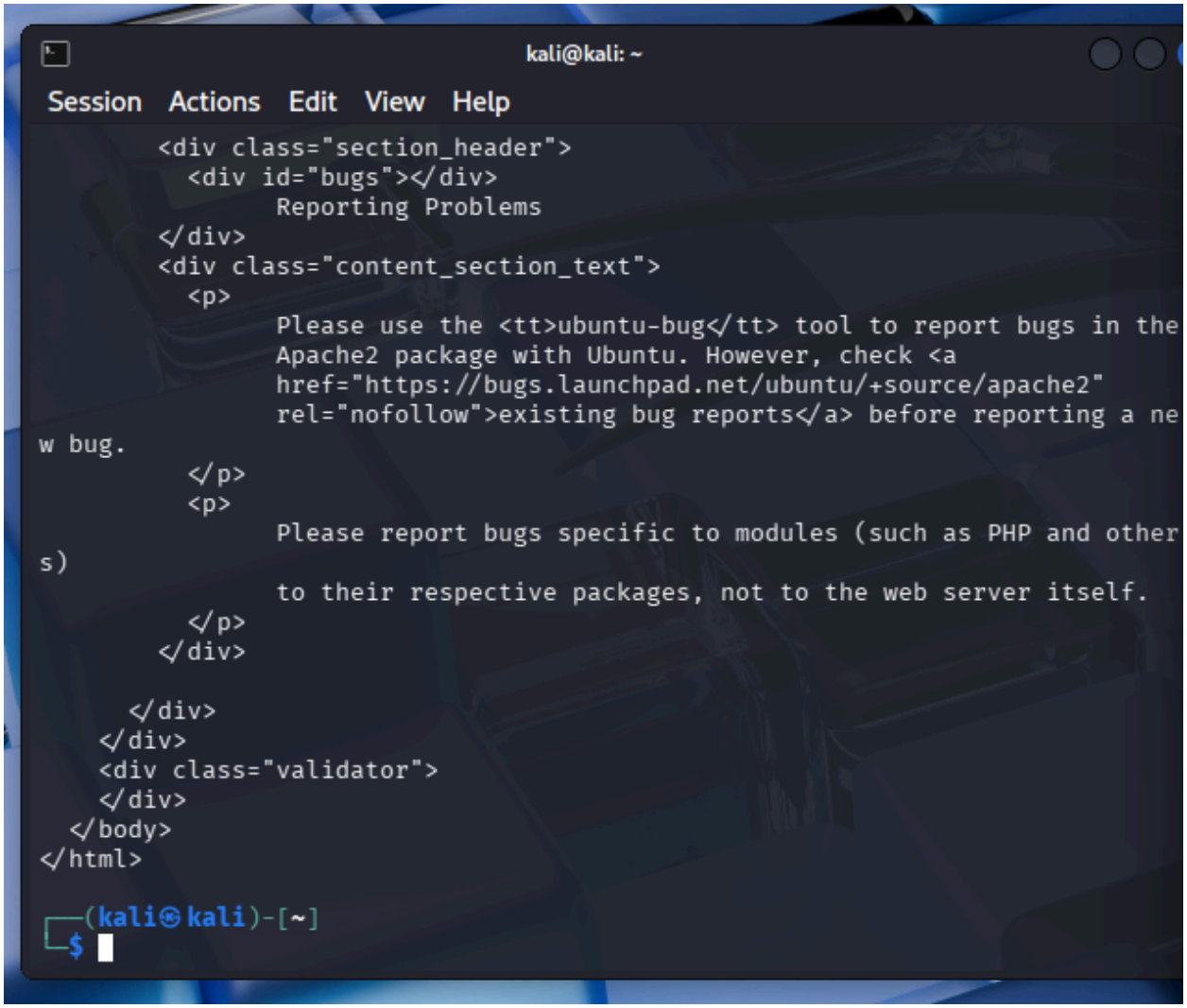
Proof



STEP 9: Final Proof (CONTROLLED SECURITY)
The command we are using is (curl http://10.0.0.2)

📍 **The Concept** In Step 8, you told the firewall: "Only talk to `10.0.0.1` (Kali)". Now, we confirm that Kali still has its special "VIP" access while everyone else is locked out.

Proof



Step	Task Name	Command Executed	Goal / Purpose	Result (Success Proof)

1	**Baseline Scan**	`nmap -sS -p 22,80 10.0.0.2`	Find open doors before security	**Success:** Port 22 & 80 shown as `open`
2	**Service Audit**	`curl http://10.0.0.2`	Confirm web server is reachable	**Success:** Raw HTML code received
3	**Health Check**	`sudo ufw status`	Check initial firewall state	**Success:** Status confirmed as `inactive`
4	**Shield Up**	`sudo ufw enable`	Activate the system's defense	**Success:** Firewall is `active` & `deny incoming`
5	**Attack Test**	`nmap -p 80 10.0.0.2`	Verify that firewall blocks access	**Success:** Port 80 shown as `filtered`
6	**Open Access**	`sudo ufw allow 80/tcp`	Allow everyone to see the site	**Success:** Rule added to the table
7	**Access Proof**	`curl http://10.0.0.2`	Prove site is back online	**Success:** HTML page loads again

8	**Engineering**	sudo ufw allow from 10.0.0.1 to any port 80	Restrict access to ONLY Kali IP	**Success:** Precision rule applied
9	**Final Victory**	curl http://10.0.0.2	Final proof of authorized access	**Success:** Site loads ONLY for Kali