

BUILD WEEK - NETWORK FOR THETA

The Build Week project involved implementing a network for the Theta company. The goal was to connect 120 devices, a NAS server, a web server, a perimeter firewall, and 3 IDS/IPS, organized into subnets. The devices were distributed across 6 floors, with 20 on each floor.

After the requirements were outlined, we focused on preparing a budget that included labor costs for configuration. The total estimated cost amounts to €174,490.58, well within the initial budget of €285,000. Below is a list of links to the selected components:

Budget for Theta Company

SWITCH:

- S3700-24T4F 24 PORT
- UNIT COST: €254.98
- TOTAL FOR 8 DEVICES: €2,039.84

OPTICAL FIBER CABLE

- TOTAL COST: €2,225.73

SERVER:

- TOWER POWEREDGE T160
- COST: €4,135.54

PC:

- HP PRO TOWER 400 G9
- UNIT COST: €947.00
- TOTAL FOR 120 DEVICES: €113,640

MONITOR:

- SAMSUNG S31C
- UNIT COST: €89.90
- TOTAL FOR 120 DEVICES: €10,788

FIREWALL:

- NSG-5220
- COST: €5,936.52

IPS/IDS:

- CISCO 4260
- UNIT COST: €1,696.50
- TOTAL FOR 3 DEVICES: €5,089.50

NAS:

- SYNOLOGY DISKSTATION DS1823xs+
- COST: €1,941.52

ROUTER:

- CISCO C891FW-E-K9
- COST: €1,080.50

LABOR AND INTERNET PROVIDER:

- €20,000
- 20 DAYS FOR NETWORK DESIGN AND INSTALLATION (THROUGH AN EXTERNAL COMPANY)
- €7,620 FOR 12 MONTHS OF INTERNET SERVICE, INCLUDING FIXED COSTS

Network Design and Distribution

The network is organized by floor as follows:

- 1st Floor: Reception and Front Desk (VLAN-Floor-1)
- 2nd Floor: Marketing (VLAN-Floor-2)
- 3rd Floor: Management (VLAN-Floor-3, VLAN-NAS, VLAN-WEBSERVER)
- 4th Floor: Sales Department (VLAN-Floor-4)
- 5th Floor: Logistics (VLAN-Floor-5)
- 6th Floor: Finance (VLAN-Floor-6)

Each floor has 20 devices connected to a switch, totaling 120 devices and 6 switches. These switches are linked to a router gateway positioned on the 3rd floor to minimize cable length and signal loss. Additionally, the 3rd floor houses the NAS, the web server, the firewall, and the 3 IDS/IPS units, for easy centralized access.

IDS and IPS are strategically placed: two IDS protect the NAS and Management devices, which contain sensitive data, while still allowing employee access. An IPS defends the web server, ensuring security against external threats. The firewall is hardware-based and configured at the perimeter to handle a high volume of connected devices, minimizing potential disruptions. It employs application filtering with a DMZ inside the firewall.

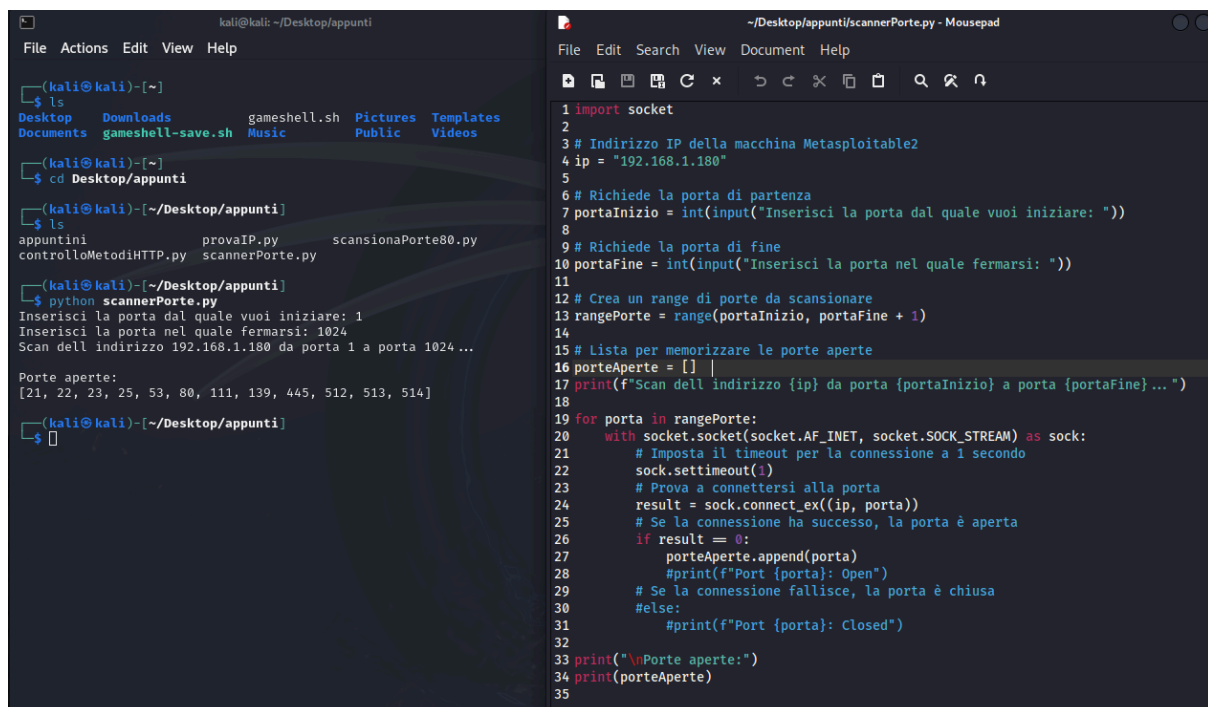
For network segmentation, we chose a /27 subnet mask, providing 32 IP addresses per subnet, with 30 available for hosts, reserving one for the gateway. This decision, based on Cisco's guidelines, supports the 20 devices per floor while allowing for potential future expansions of up to 54 additional devices.

VLANs were configured for each subnet to compartmentalize information and optimize traffic flow across departments. Switches handle data exchange within the same subnet, while the gateway router allows communication between subnets. To simplify network management for the company, which lacks an in-house IT technician, we assigned intuitive names to the VLANs for easy identification.

The NAS and web servers are placed in a separate subnet with a /29 mask, providing 8 IP addresses, of which 5 are available for hosts. This choice isolates the servers from the rest of the network, enhancing security. The lower number of devices means this subnet is appropriately sized without excess.

Network Testing

Beyond the initial ping to the Metasploitable machine to identify its IP address, the first network test verified open ports on this device, represented by the Metasploitable machine. Using a custom network scanner to scan a port range for a given IP address, we compiled a list of currently open ports. This is useful for verifying that necessary services are available and identifying any critical vulnerabilities.



```
kali@kali: ~/Desktop/appunti
File Actions Edit View Help
$ ls
Desktop Downloads gameshell.sh Pictures Templates
Documents gameshell-save.sh Music Public Videos

(kali@kali)~$ cd Desktop/appunti

(kali@kali)~/Desktop/appunti$ ls
appuntini provaIP.py scansionaPorte80.py
controlloMetodiHTTP.py scannerPorte.py

(kali@kali)~/Desktop/appunti$ python scannerPorte.py
Inserisci la porta dal quale vuoi iniziare: 1
Inserisci la porta nel quale fermarsi: 1024
Scan dell'indirizzo 192.168.1.180 da porta 1 a porta 1024 ...

Porte aperte:
[21, 22, 23, 25, 53, 80, 111, 139, 445, 512, 513, 514]

(kali@kali)~/Desktop/appunti$
```

```
~/Desktop/appunti/scannerPorte.py - Mousepad
File Edit Search View Document Help
1 import socket
2
3 # Indirizzo IP della macchina Metasploitable2
4 ip = "192.168.1.180"
5
6 # Richiede la porta di partenza
7 portaInizio = int(input("Inserisci la porta dal quale vuoi iniziare: "))
8
9 # Richiede la porta di fine
10 portaFine = int(input("Inserisci la porta nel quale fermarsi: "))
11
12 # Crea un range di porte da scansionare
13 rangePorte = range(portaInizio, portaFine + 1)
14
15 # Lista per memorizzare le porte aperte
16 porteAperte = []
17 print(f"Scan dell'indirizzo {ip} da porta {portaInizio} a porta {portaFine}...")
18
19 for porta in rangePorte:
20     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
21         # Imposta il timeout per la connessione a 1 secondo
22         sock.settimeout(1)
23         # Prova a connettersi alla porta
24         result = sock.connect_ex((ip, porta))
25         # Se la connessione ha successo, la porta è aperta
26         if result == 0:
27             porteAperte.append(porta)
28             #print(f"Port {porta}: Open")
29         # Se la connessione fallisce, la porta è chiusa
30         else:
31             #print(f"Port {porta}: Closed")
32
33 print("\nPorte aperte:")
34 print(porteAperte)
35
```

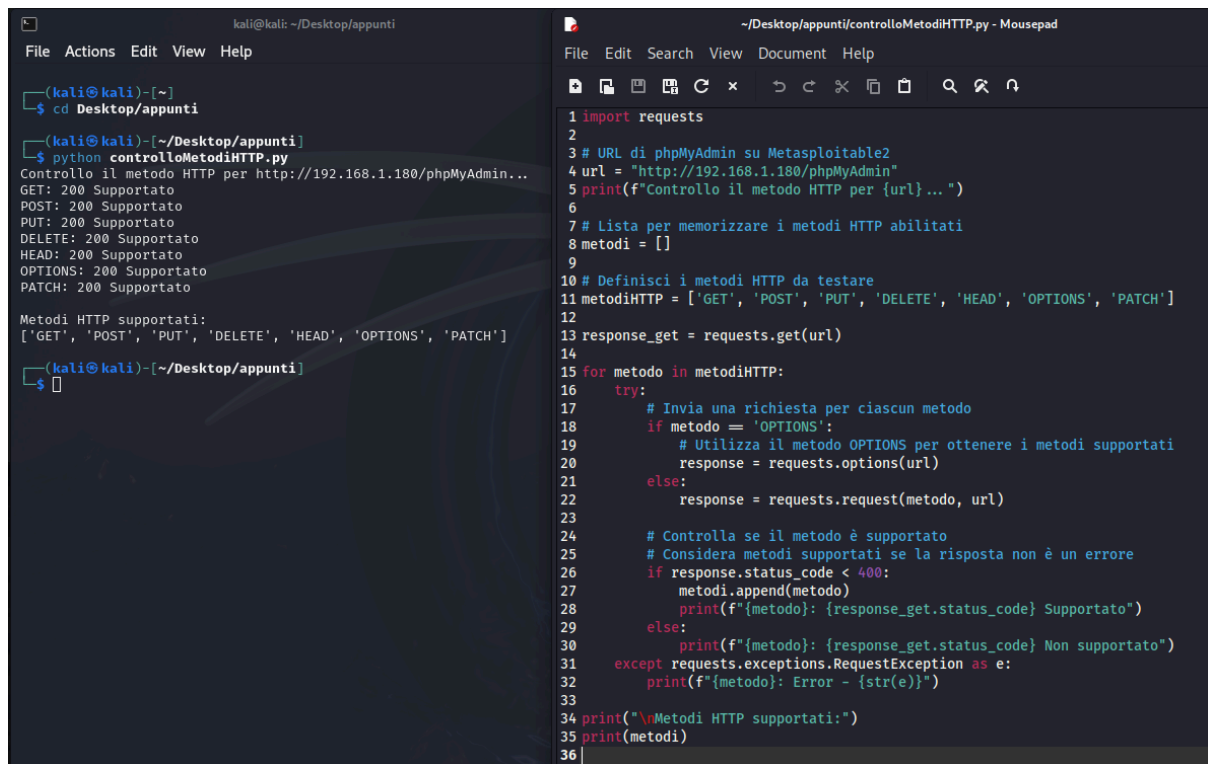
The test found the following ports open: 21, 22, 23, 25, 53, 80, 111, 139, 445, 512, 513, and 514. Below is a list of the most relevant:

- 21 (FTP) - File Transfer Protocol
- 22 (SSH) - Secure Remote Access
- 23 (Telnet) - Unsecured Remote Access (data transmitted in plain text)
- 25 (SMTP) - Email Transmission
- 80 (HTTP) - Unsecured Web Traffic

As expected, we identified port 80 (HTTP) as open, representing a vulnerability since it is the "insecure" version of HTTPS.

Based on this finding, we proceeded to verify the HTTP methods for the same IP address on the Metasploitable machine. The program first defines the IP address, creates a list to store request results by "method," and sets the HTTP methods to test. The `requests` module, imported at the beginning (chosen for its practicality and speed with HTTP requests), is used

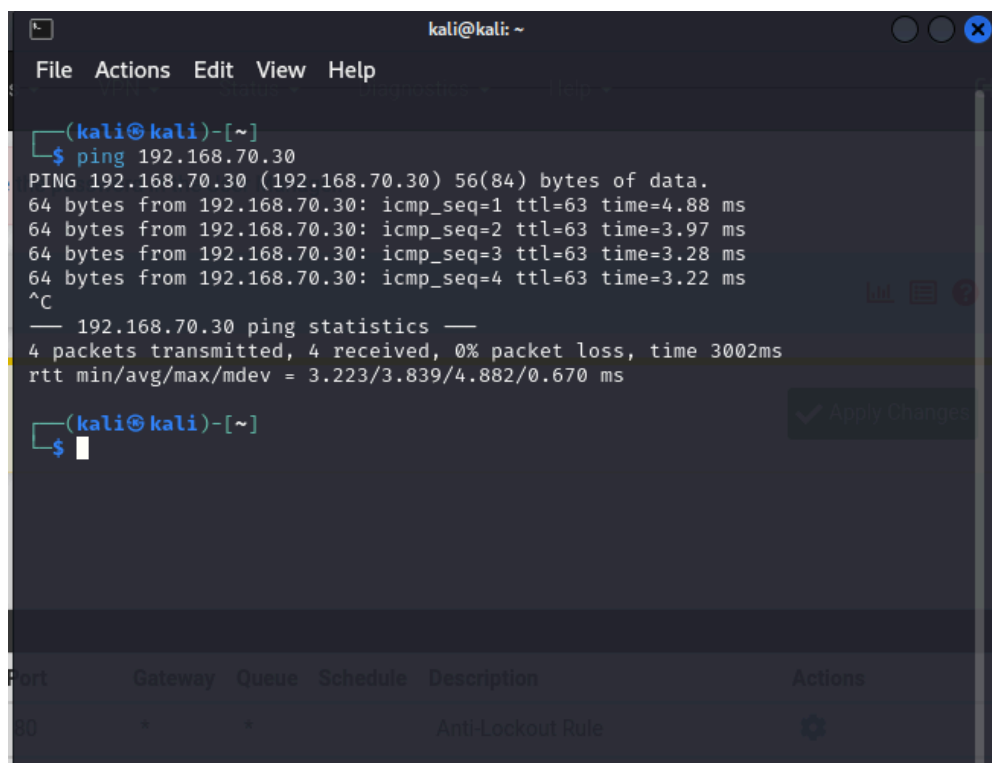
within a for-loop that iterates through both defined lists. Using if/else constructs, we set conditions to send requests for each method and check if the method is supported. The results show that all methods are supported in this case.



The image shows two side-by-side terminal windows. The left window is a Kali Linux terminal with the prompt 'kali@kali: ~/Desktop/appunti'. It shows the execution of a Python script 'controlloMetodiHTTP.py' which tests various HTTP methods (GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH) against a target URL 'http://192.168.1.180/phpMyAdmin...'. The output shows that all methods are supported (200 Supportato). The right window is a Mousepad editor showing the source code of 'controlloMetodiHTTP.py'. The code imports the 'requests' library, defines a target URL, and iterates through a list of HTTP methods. For each method, it sends a request and checks the status code. If the status code is less than 400, the method is added to a list of supported methods. The script prints the list of supported methods at the end.

```
1 import requests
2
3 # URL di phpMyAdmin su Metasploitable2
4 url = "http://192.168.1.180/phpMyAdmin"
5 print(f"Controllo il metodo HTTP per {url}...")
6
7 # Lista per memorizzare i metodi HTTP abilitati
8 metodi = []
9
10 # Definisci i metodi HTTP da testare
11 metodiHTTP = ['GET', 'POST', 'PUT', 'DELETE', 'HEAD', 'OPTIONS', 'PATCH']
12
13 response_get = requests.get(url)
14
15 for metodo in metodiHTTP:
16     try:
17         # Invia una richiesta per ciascun metodo
18         if metodo == 'OPTIONS':
19             # Utilizza il metodo OPTIONS per ottenere i metodi supportati
20             response = requests.options(url)
21         else:
22             response = requests.request(metodo, url)
23
24         # Controlla se il metodo è supportato
25         # Considera metodi supportati se la risposta non è un errore
26         if response.status_code < 400:
27             metodi.append(metodo)
28             print(f"{metodo}: {response_get.status_code} Supportato")
29         else:
30             print(f"{metodo}: {response_get.status_code} Non supportato")
31     except requests.exceptions.RequestException as e:
32         print(f"{metodo}: Error - {str(e)}")
33
34 print("\nMetodi HTTP supportati:")
35 print(metodi)
36
```

Finally, as required, we simulated device communication (pfsense, Metasploitable, and Kali) in a virtual environment. Packet transmission occurred without loss.

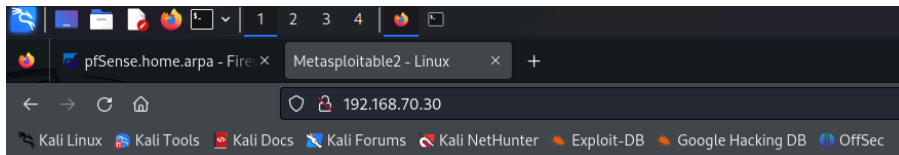


The image shows a terminal window with the prompt 'kali@kali: ~'. It shows the execution of a 'ping' command to the IP address '192.168.70.30'. The output shows that 4 packets were transmitted, 4 were received, and there was 0% packet loss. The round trip time (rtt) statistics are shown as 3.223/3.839/4.882/0.670 ms. Below the terminal window, there is a table with columns: Port, Gateway, Queue, Schedule, Description, and Actions. The table shows a single entry for port 80, with a description of 'Anti-Lockout Rule' and a gear icon in the Actions column.

```
(kali@kali)-[~]
$ ping 192.168.70.30
PING 192.168.70.30 (192.168.70.30) 56(84) bytes of data:
64 bytes from 192.168.70.30: icmp_seq=1 ttl=63 time=4.88 ms
64 bytes from 192.168.70.30: icmp_seq=2 ttl=63 time=3.97 ms
64 bytes from 192.168.70.30: icmp_seq=3 ttl=63 time=3.28 ms
64 bytes from 192.168.70.30: icmp_seq=4 ttl=63 time=3.22 ms
^C
— 192.168.70.30 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 3.223/3.839/4.882/0.670 ms

(kali@kali)-[~]
$
```

Port	Gateway	Queue	Schedule	Description	Actions
80	*	*		Anti-Lockout Rule	



Warning: Never expose this VM to an untrusted network!

Contact: [msfdev\[at\]metasploit.com](mailto:msfdev[at]metasploit.com)

Login with msfadmin/msfadmin to get started

- [TWiki](#)
- [phpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)

Once the network configuration was confirmed, we applied a blocking rule on port 80 through the pfsense GUI to prevent communication between Kali and the DVWA.

Action	Block		
Choose what to do with packets that match the criteria specified below. Hint: the difference between block and reject is that with reject, a packet (TCP RST or ICMP port unreachable for UDP) is returned to the sender, whereas with block the packet is dropped silently. In either case, the original packet is discarded.			
Disabled	<input checked="" type="checkbox"/> Disable this rule Set this option to disable this rule without removing it from the list.		
Interface	LAN		
Choose the interface from which packets must come to match this rule.			
Address Family	IPv4		
Select the Internet Protocol version this rule applies to.			
Protocol	TCP		
Choose which IP protocol this rule should match.			
Source			
Source	<input type="checkbox"/> Invert match	Address or Alias	192.168.60.30 /
<input type="button" value="Display Advanced"/>			
The Source Port Range for a connection is typically random and almost never equal to the destination port. In most cases this setting must remain at its default value, any .			
Destination			
Destination	<input type="checkbox"/> Invert match	Address or Alias	192.168.70.30 /
Destination Port Range	HTTP (80)	From	Custom
		To	Custom
Specify the destination port or port range for this rule. The "To" field may be left empty if only filtering a single port.			

Floating

WAN

LAN

LAN2

Rules (Drag to Change Order)

<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	✓ 1/339 KiB	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
<input type="checkbox"/>	✗ 0/0 B	IPv4 TCP	192.168.60.30	*	192.168.70.30	80 (HTTP)	*	none			
<input type="checkbox"/>	✓ 0/70 KiB	IPv4 *	LAN subnets	*	*	*	*	none		Default allow LAN to any rule	
<input type="checkbox"/>	✓ 0/0 B	IPv6 *	LAN subnets	*	*	*	*	none		Default allow LAN IPv6 to any rule	

↑ Add

↓ Add

🗑 Delete

🔄 Toggle

📄 Copy

💾 Save

➕ Separator

We tested this rule using a final ping:

```

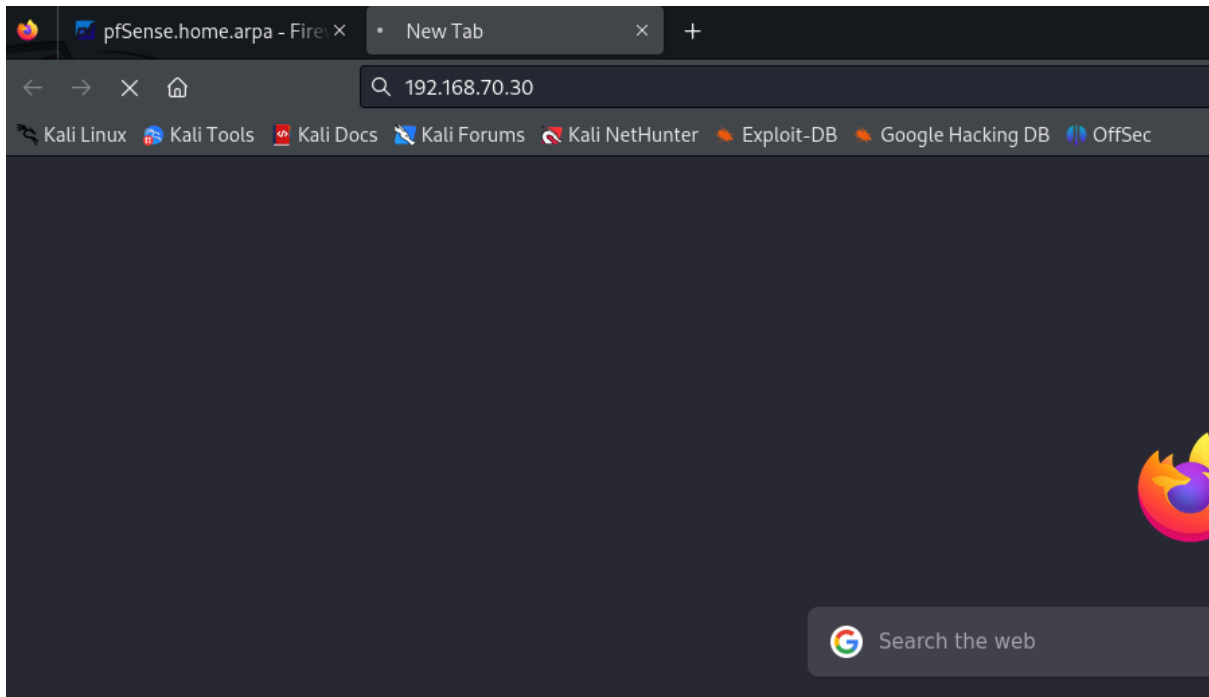
kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ ping 192.168.70.30
PING 192.168.70.30 (192.168.70.30) 56(84) bytes of data.
64 bytes from 192.168.70.30: icmp_seq=1 ttl=63 time=4.88 ms
64 bytes from 192.168.70.30: icmp_seq=2 ttl=63 time=3.97 ms
64 bytes from 192.168.70.30: icmp_seq=3 ttl=63 time=3.28 ms
64 bytes from 192.168.70.30: icmp_seq=4 ttl=63 time=3.22 ms
^C
— 192.168.70.30 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 3.223/3.839/4.882/0.670 ms

(kali@kali)-[~]
$ ping 192.168.70.30
PING 192.168.70.30 (192.168.70.30) 56(84) bytes of data.
64 bytes from 192.168.70.30: icmp_seq=1 ttl=63 time=3.24 ms
64 bytes from 192.168.70.30: icmp_seq=2 ttl=63 time=2.71 ms
64 bytes from 192.168.70.30: icmp_seq=3 ttl=63 time=3.30 ms
64 bytes from 192.168.70.30: icmp_seq=4 ttl=63 time=3.04 ms
^C
— 192.168.70.30 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 2.712/3.073/3.303/0.230 ms

(kali@kali)-[~]
$

```



As expected, communication was blocked, allowing us to confirm that we mitigated the vulnerability on port 80. Additionally, we noted that HTTP methods PUT and DELETE, which are currently active, pose significant vulnerabilities by allowing any user to freely add or delete code. This could lead to serious issues if malicious actions were taken. We recommend that the company mitigate risks by disabling these HTTP methods.