

BUILD WEEK - NETWORK FOR THETA

Il progetto della build week prevedeva l'implementazione di una rete per l'azienda Theta. La richiesta era di collegare 120 dispositivi, un server NAS, un web server, un firewall perimetrale e 3 IDS/IPS, utilizzando una corretta suddivisione in sottoreti. I dispositivi dovevano essere distribuiti su 6 piani, con 20 per ciascun piano.

Data la consegna, la fase di ingaggio riguardava più che altro la redazione di un budget secondo le richieste aziendali, e l'aggiunta della manodopera per la configurazione. La nostra redazione dei costi totali ammonta a un totale di **174.490,58€** di preventivo che con il budget massimo previsto di 285.000€ stabilito inizialmente rientra ampiamente nelle necessità del nostro cliente. Qui sotto una lista di link per i componenti scelti:

Preventivo per azienda Theta

SWITCH:

- [S3700-24T4F 24 PORTE](#)
- COSTO SINGOLO: 254,98€
- PER 8 DISPOSITIVI: 2.039,84€

CAVO PATCH IN FIBRA OTTICA

- COSTO TOTALE: 2.225,73€

SERVER

- [TOWER POWEREDGE T160](#)
- COSTO: 4.135,54€

PC:

- [HP PRO TOWER 400 G9](#)
- COSTO SINGOLO: 947,00€
- COSTO x 120 DISPOSITIVI: 113.640€

MONITOR:

- [SAMSUNG S31C](#)
- COSTO SINGOLO: 89,90€
- COSTO X 120 DISPOSITIVI: 10.788€

FIREWALL:

- [NSG-5220](#)
- COSTO: 5.936,52€

IPS/IDS:

- [CISCO 4260](#)
- COSTO SINGOLO: 1.696,50€
- COSTO X3 DISPOSITIVI: 5.089,50€

NAS:

- [SYNOLOGY DISKSTATION DS1823xs+](#)
- COSTO : 1.941,52€

ROUTER:

- [CISCO C891FW-E-K9](#)
- COSTO: 1.080,50€

MANODOPERA E PROVIDER INTERNET:

- 20.000€
- 20 GIORNI PER PROGETTAZIONE E MONTAGGIO RETE (TRAMITE DITTA ESTERNA)
- 7620€ PER 12 MESI DI RETE, COSTI FISSI INCLUSI

Abbiamo progettato una rete per un edificio di 6 piani, ipotizzando una suddivisione per mansioni aziendali. La distribuzione è la seguente:

- 1° Piano: Segreteria e Reception (VLAN-Piano-1);
- 2° Piano: Ufficio Marketing (VLAN-Piano-2);
- 3° Piano: Direzione (VLAN-Piano-3, VLAN-WEBSERVER, VLAN-NAS);
- 4° Piano: Area Commerciale (VLAN-Piano-4);
- 5° Piano: Logistica (VLAN-Piano-5);
- 6° Piano: Finanza (VLAN-Piano-6).

Ogni piano dispone di 20 dispositivi collegati a uno switch, per un totale di 120 dispositivi e 6 switch. Gli switch sono collegati a un router gateway, posizionato al 3° piano per minimizzare la lunghezza dei cavi e ridurre la dispersione. Al 3° piano si trovano anche il dispositivo NAS, il web server, il firewall hardware e i 3 sistemi IDS/IPS, in modo che siano vicini tra loro e centralizzati rispetto agli altri piani.

Gli IDS e IPS sono stati posizionati in punti strategici: due IDS proteggono il NAS e i dispositivi della Direzione, poiché contengono dati sensibili, permettendo comunque l'accesso ai dipendenti. Un IPS è stato collocato a difesa del web server per garantire la sicurezza contro attacchi esterni. Il firewall utilizzato è di tipo hardware, configurato a livello perimetrale per gestire l'elevato numero di dispositivi connessi, riducendo al minimo eventuali disservizi. È stato configurato per utilizzare il filtraggio per applicazioni, con una DMZ all'interno del firewall.

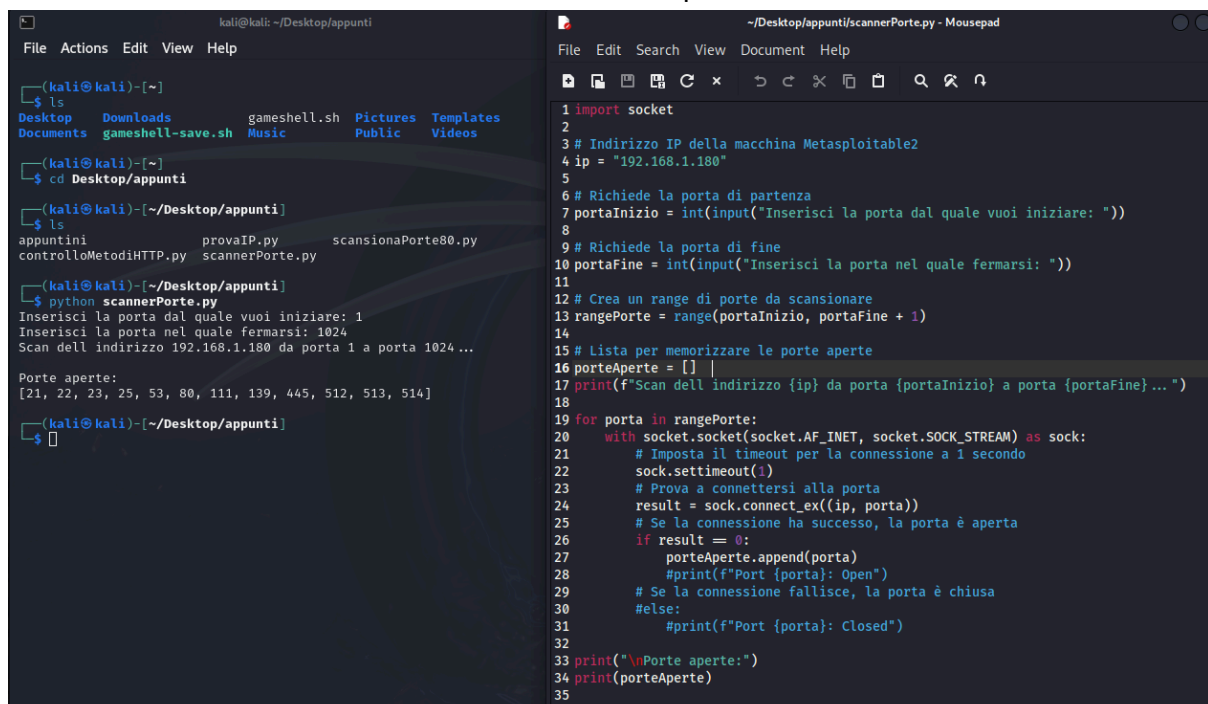
Per la segmentazione della rete, abbiamo scelto una subnet mask /27, che offre 32 indirizzi IP per subnet, di cui 30 utilizzabili per gli host, riservandone uno per il gateway. Questa scelta, basata sul teorema di Cisco, permette di supportare i 20 dispositivi per piano, lasciando spazio per futuri ampliamenti fino a un totale di 54 dispositivi aggiuntivi.

Abbiamo inoltre configurato VLAN per ciascuna subnet, in modo da compartimentare le informazioni e migliorare il flusso di traffico all'interno delle diverse aree operative. Gli switch gestiscono lo scambio di dati tra i dispositivi della stessa subnet, mentre il router gateway consente la comunicazione tra le diverse sottoreti. Data la mancanza di un tecnico informatico interno all'azienda stessa, è stato scelto volutamente di assegnare dei nomi intuitivi alle diverse VLAN per rendere più chiara e comprensibile la suddivisione delle stesse, anche per chi potrebbe non avere chiara la base della gestione della rete.

I server NAS e web sono stati inseriti in una subnet separata con maschera /29, che offre 8 indirizzi IP, di cui 5 utilizzabili per gli host. Questa scelta è stata fatta per garantire l'isolamento dei server dal resto della rete, aumentando la sicurezza. Poiché il numero di dispositivi è inferiore rispetto agli altri piani, questa subnet fornisce il numero di indirizzi sufficiente senza sprechi.

Test di rete

Oltre al ping iniziale verso la macchina Metasploitable, da cui reperiamo l'indirizzo IP attraverso ifconfig, il primo test di rete mirava a verificare l'apertura delle porte all'interno della macchina di rete rappresentata in questo caso da Metasploitable: è stato eseguito attraverso la costruzione di uno scanner di rete che, impostando un range di porte da scannerizzare e dato un'indirizzo IP, fornisce una lista di tutte le porte correntemente aperte. Questo è utile sia per controllare che gli effettivi servizi necessari siano disponibili, sia per effettuare un check veloce su delle vulnerabilità che potrebbero essere critiche.



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

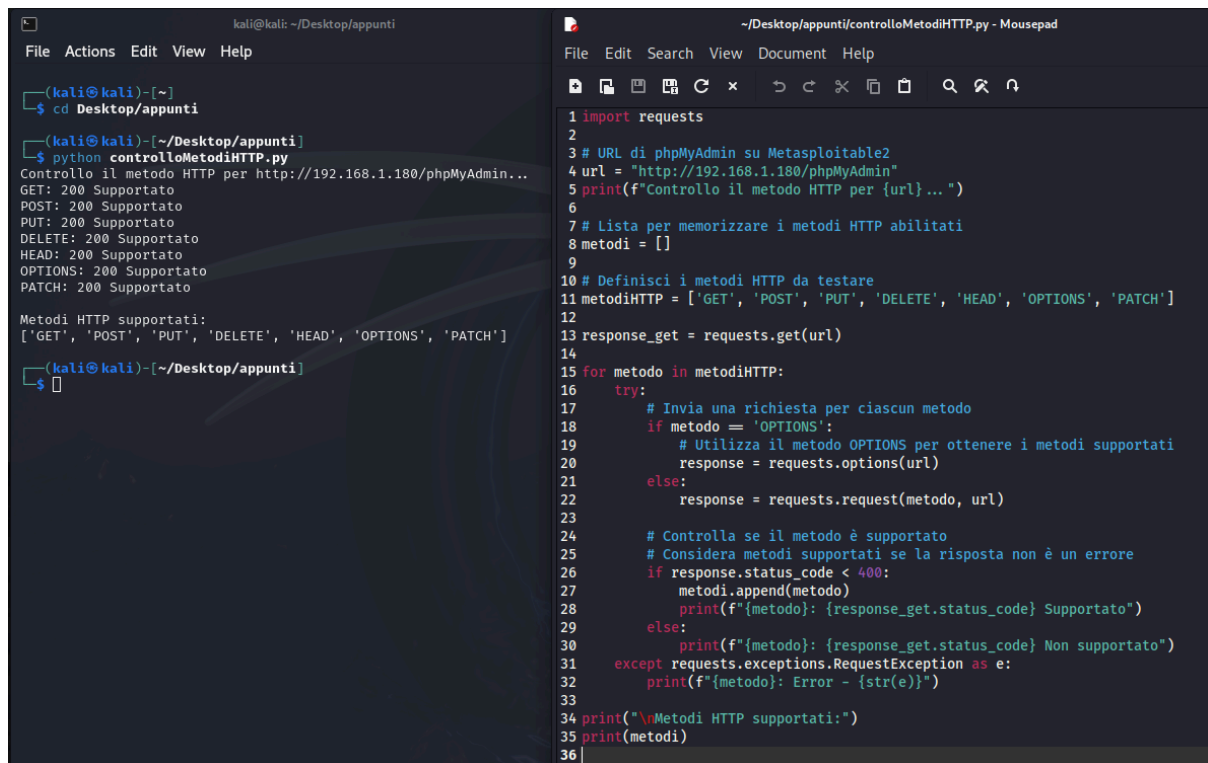
1 import socket
2
3 # Indirizzo IP della macchina Metasploitable2
4 ip = "192.168.1.180"
5
6 # Richiede la porta di partenza
7 portaInizio = int(input("Inserisci la porta dal quale vuoi iniziare: "))
8
9 # Richiede la porta di fine
10 portaFine = int(input("Inserisci la porta nel quale fermarsi: "))
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
```

In questo caso il test ci ha restituito come porte aperte la 21, 22, 23, 25, 53, 80, 111, 139, 445, 512, 513 e 514. Qui sotto una lista delle più rilevanti.

- 21 (FTP)** - Protocollo per il trasferimento file.
- 22 (SSH)** - Accesso remoto sicuro a sistemi.
- 23 (Telnet)** - Accesso remoto non sicuro (informazioni trasmesse in chiaro).
- 25 (SMTP)** - Invio di email.
- 80 (HTTP)** - Traffico web non sicuro.

Come previsto anche dalla traccia della build week stessa, abbiamo individuato la porta 80 (HTTP) aperta, che costituisce una vulnerabilità importante dal momento che è la versione “non sicura” di HTTPS.

Con questa considerazione dunque, ci muoviamo a questo punto sulla verifica dei metodi HTTP per lo stesso indirizzo IP della macchina Metasploitable; il programma definisce innanzitutto l'indirizzo IP, definisce una lista di memorizzazione dei risultati delle richieste con “metodo” e infine definisce anche quali saranno i metodi da testare. Successivamente, attraverso il modulo requests importato all'inizio (predisposto esattamente al nostro scopo, ovvero alla praticità e rapidità delle richieste HTTP), andiamo a definire un range con for e in relativi alle due liste definite prima, e attraverso i costrutti if/else definiamo delle condizioni in primo luogo per inviare una richiesta per ciascun metodo, e successivamente per controllare se il metodo è supportato. Come dai risultati visibili a sinistra, vediamo che tutti i metodi sono in questo caso supportati.



```
kali@kali: ~/Desktop/appunti
File Actions Edit View Help

(kali@kali)~[~]
$ cd Desktop/appunti

(kali@kali)~[~/Desktop/appunti]
$ python controlloMetodiHTTP.py
Controllo il metodo HTTP per http://192.168.1.180/phpMyAdmin...
GET: 200 Supportato
POST: 200 Supportato
PUT: 200 Supportato
DELETE: 200 Supportato
HEAD: 200 Supportato
OPTIONS: 200 Supportato
PATCH: 200 Supportato

Metodi HTTP supportati:
['GET', 'POST', 'PUT', 'DELETE', 'HEAD', 'OPTIONS', 'PATCH']

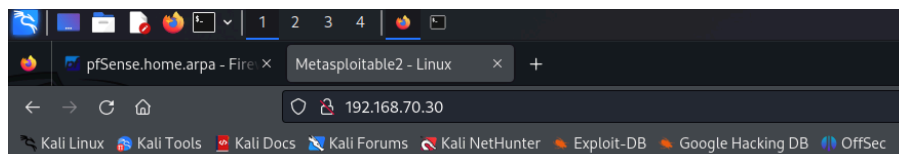
(kali@kali)~[~/Desktop/appunti]
$

~/Desktop/appunti/controlloMetodiHTTP.py - Mousepad
File Edit Search View Document Help

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.status_code} Supportato")
29         else:
30             print(f"{metodo}: {response.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
```

Infine, come previsto dalla traccia, abbiamo innanzitutto verificato in una simulazione con macchina virtuale un ping la comunicazione fra i diversi dispositivi (pfsense, Metasploitable e Kali), che trasmettevano pacchetti senza alcuna perdita.

```
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)-[~]  
$
```



metasploitable2

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

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

Una volta giunti alla conclusione che la configurazione di rete era corretta, passando attraverso la GUI di pfsense stesso abbiamo posto una regola di blocco sulla porta 80 per impedire la comunicazione fra Kali e la 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

☒ 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

☐ Invert match

Address or Alias

192.168.60.30

/

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

☐ Invert match

Address or Alias

192.168.70.30

/

Destination Port Range

HTTP (80)

From

Custom

HTTP (80)

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.

FloatingWANLANLAN2

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input 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

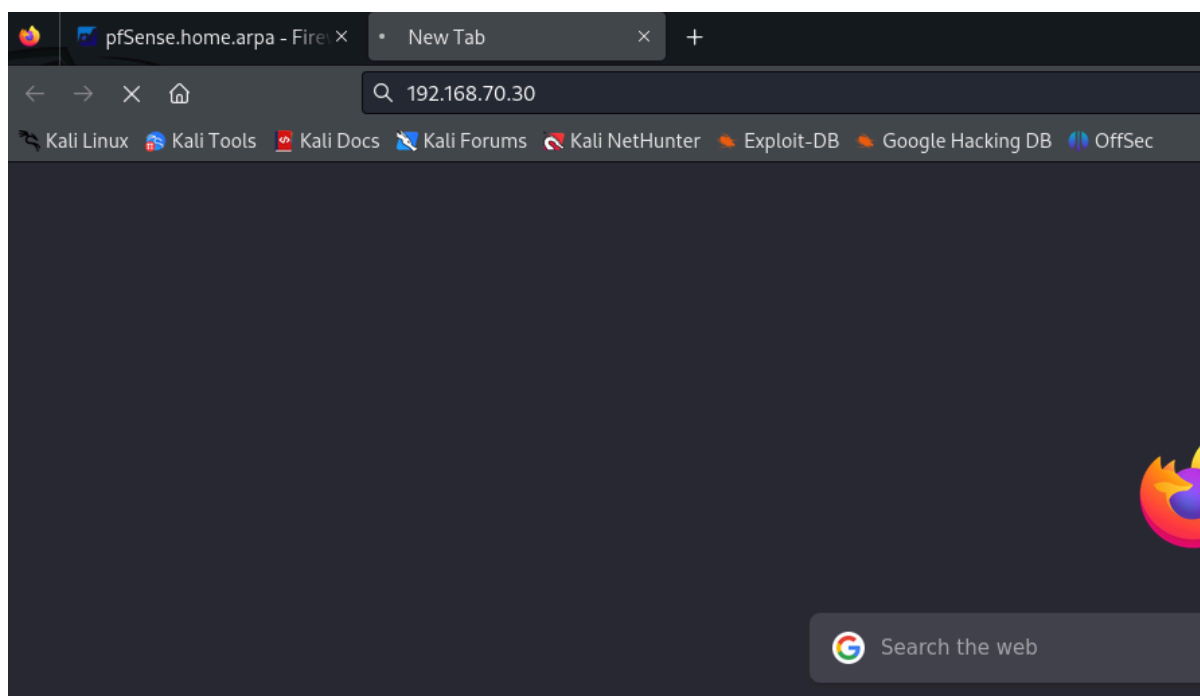
Save

Separator

i

A questo punto testiamo l'effettivo blocco delle regole attraverso un ultimo 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  
ping in the background  
(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)-[~]  
$
```



Come ci aspettavamo, la comunicazione è stata interrotta, e possiamo affermare di aver bloccato la vulnerabilità richiesta sulla porta 80.

Possiamo inoltre considerare, oltre alla porta 80 di cui già ci siamo occupati via firewall, come forti vulnerabilità la presenza dei verbi HTTP PUT e DELETE che sono al momento attivi, permettendo a qualunque utente di aggiungere o cancellare codice a piacimento. Questo comporterebbe grossi problemi qualora qualcuno decidesse di eliminare o aggiungere righe a scopo malevolo. Si potrebbe raccomandare in questo caso all'azienda di mitigare i rischi disattivando i verbi in questione.