

Report Progetto Build Week

19 dic. 2024

Team Leader: **Leonardo Catalano**

Team members: **Jacopo Benedetti, Jacopo Ciocca, Gianluca Contu, Nicolò Di Vittorio, Alessandra G. Fancello, Vincenzo Frau, Yuri Pedrana, Rokaia Sajie e Francesco Zaccione.**

Traccia

“Siamo stati ingaggiati dalla compagnia Theta per sviluppare un preventivo di spesa e un progetto di rete per la loro infrastruttura IT. Ecco i requisiti e i componenti necessari:

- *Struttura dell'edificio: 6 piani*
- *Dispositivi previsti: 20 computer per piano, per un totale di 120 computer*
- *Componenti aggiuntivi:*
 - *1 Web server (rappresentato dalla macchina DVWA di Metasploitable)*
 - *1 Firewall perimetrale*
 - *1 NAS (Network Attached Storage)*
 - *3 IDS/IPS (Intrusion Detection System / Intrusion Prevention System)* ”
inoltre

“Per concludere il progetto, effettueremo una serie di test sulla rete implementata. I test includeranno:

1. *Esercizio Traccia e requisiti Verifica dei Verbi HTTP: Scrivere un programma in Python per inviare richieste HTTP (GET, POST, PUT, DELETE) al web server e verificare le risposte.*
2. *Scansione delle Porte: Utilizzeremo un programma in Python per eseguire una scansione delle porte sui dispositivi di rete, verificando la sicurezza e l'accessibilità delle varie porte di comunicazione.”*
ed ancora

“Alla conclusione dei test, redigeremo un report dettagliato che includerà:

- *Risultati dei Test HTTP: Documentazione delle risposte ricevute dal web server per ogni verbo HTTP testato.*
- *Risultati della Scansione delle Porte : Elenco delle porte aperte e chiuse sui vari dispositivi, con raccomandazioni di sicurezza.”*

Report Progetto Build Week

Introduzione al progetto di rete per la compagnia Theta

In questa settimana di build team ci è stato chiesto di progettare una rete per l'ipotetica compagnia Theta e sviluppare un preventivo di spesa per l'infrastruttura necessaria. La traccia domanda, inoltre, di programmare due tool per eseguire test sulla rete, in linguaggio Python e infine implementare un web server simulato dalla macchina virtuale (da qui in poi anche sono V.M. o v.m.) Metasploitable2.

Dato di rilievo è che la ipotetica committente domanda la realizzazione di un progetto di rete per un edificio di 6 piani con una disposizione di 20 PC, od in generale end devices, per piano, per un totale di 120 dispositivi.

I dispositivi dovranno essere collegati a degli switch, più precisamente ne avremo uno per piano, i quali verranno poi collegati ad un router

Dovranno essere inseriti alcuni componenti ovvero:

- 1 Web server (rappresentato dalla macchina DVWA di Metasploitable), situato nella DMZ (o zona demilitarizzata);
- 1 Firewall perimetrale;
- 1 NAS (Network Attached Storage);
- 3 IDS/IPS (Intrusion Detection System / Intrusion Prevention System).

Ai fini di garantire una trasparenza anche sul piano metodologico organizzativo, si segnala che al fine di una più efficiente organizzazione interna al gruppo di lavoro, il progetto è stato suddiviso in numero tre fasi e che ognuna di esse è stata curata da sottogruppi scelti di comune accordo tra la leadership e i componenti del team.

Specificamente nel presente report sarà composto da numero tre fasi:

- Fase 1, progettazione della rete;
- Fase 2, programmazione dei tool per i test sulla rete;
- Fase 3, implementazione del web server.

Report Progetto Build Week

Fase 1

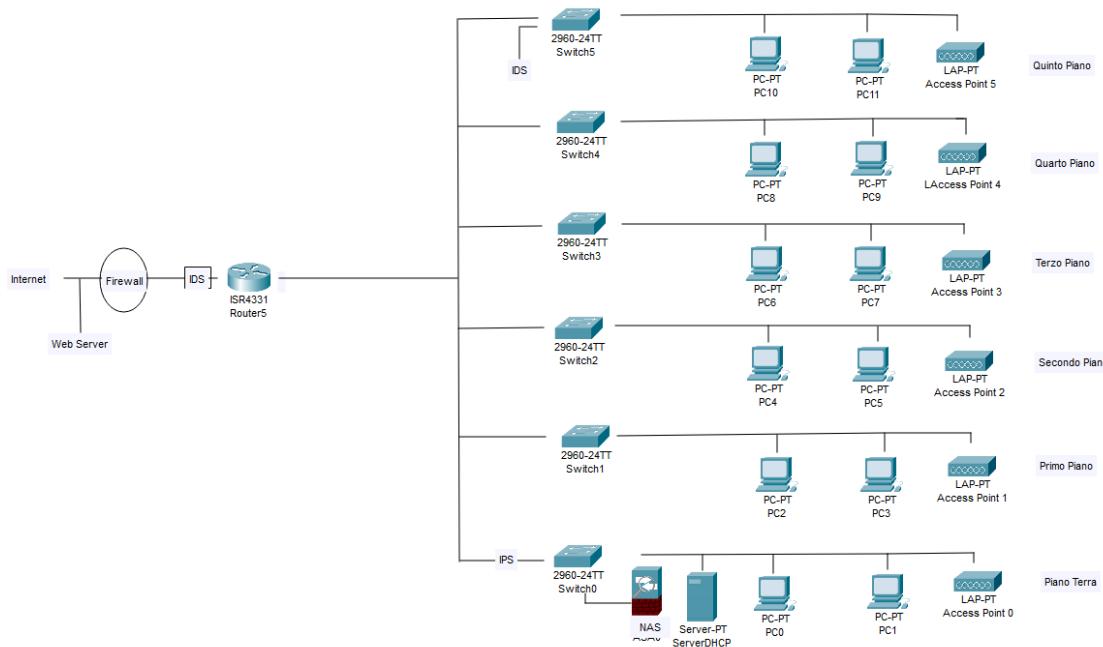
Abbiamo iniziato a impostare la suddetta rete nel modo a seguire:

Nel report grafico, utilizzeremo 2 host (PC) per piano come esempio di collegamento.

I PC andranno collegati agli switch specifici del loro piano, i quali, a loro volta, saranno collegati attraverso una canalina al Router.

In questa rete sarà inserito 1 web server (DVWA di Metasploitable) nella DMZ, 3 IDS/IPS, 1 firewall perimetrale e 1 NAS e un Server DHCP per permettere agli host di ottenere indirizzi ip dinamici.

A seguire si riporta una rappresentazione grafica della proposta di progetto ipotizzata:



Successivamente si eseguirà una disamina delle ragioni di convenienza che hanno portato alle scelte illustrate nell'immagine precedente.

Allo scopo di presentare un sistema ordinato ed organizzato, si è optato di utilizzare delle canaline, al cui interno passeranno i cavi ethernet (nello specifico si utilizzeranno cavi

Report Progetto Build Week

Straight-Through) per collegare gli host agli switches, ed una canalina principale per collegare gli switch al router.

Aggiungiamo, dunque, sei access point per la possibilità di connettersi Wi-Fi ad ogni piano.

Tra il router e la connessione a internet abbiamo collocato il firewall – come da consegna – ed un IDS, mentre tra il firewall e l’accesso a internet è presente la DMZ con il Web Server.

In questa medesima sede sono stati inseriti i primi tre componenti aggiuntivi, che andranno a gestire e rendere sicura la connessione ad internet.

A scopo dichiarativo, si fa memoria che firewall è un componente che può essere sia hardware che software, ed ha la funzione di gestire il traffico dei dati sia in entrata che in uscita. Per tale motivo è importante inserirlo tra l’accesso ad internet ed il router, in modo tale da poter permettere la gestione del traffico sia in entrata che in uscita dalla rete.

L’IDS in questo caso servirà come mero supporto al firewall aumentando il livello di sicurezza generale. Il suo compito sarà quindi quello di analizzare il traffico e sarà programmato in modo da inviare segnalazioni al firewall allorquando dovesse riscontrare anomalie nei pacchetti esaminati. Dunque, solamente il firewall andrà materialmente ad eseguire azioni di effettivo contenimento della minaccia.

La funzione della DMZ è una parte delle reti che espone servizi pubblici raggiungibili da internet, per maggior sicurezza è necessario mantenere questi servizi pubblici isolati dalla rete interna, riducendo il rischio di compromissione.

All’interno della DMZ troviamo il web server, ossia una macchina fisica o virtuale che ospita siti web o applicazioni web, e in questo caso si riferisce al server che esegue il DVWA (Damn Vulnerable Web Application) di Metasploitable, deputata all’esecuzione di test di sicurezza.

Questa è stata la base dell’impostazione strutturale della rete.

A questo punto si è passati allo studio del posizionamento dei NAS, IDS/IPS.

Report Progetto Build Week

La traccia ci chiedeva di inserire il NAS al piano terra, per cui si è assunto che tale piano fosse un punto nevralgico dell’organizzazione aziendale della committenza e che dunque richiedesse un focus maggiore in termini di messa in sicurezza.

Dopo di che bisogna munirsi di un Server DHCP per permettere a tutti gli host di avere un assegnazione automatica degli indirizzi ip, esso è un componente fondamentale nelle reti informatiche poiché garantisce la possibilità ai dispositivi di connettersi e comunicare correttamente all’interno di una rete.

Si precisa che con NAS si intende un dispositivo di archiviazione che immagazzina i file della rete aziendale.

Gli IPS/IDS sono, invece, dei sistemi che controllano il traffico sulla rete per rilevare attività sospette o malevole.

La principale differenza tra i due è che l’IPS lavora in “maniera attiva”, quindi oltre a rilevare un’attività sospetta esegue anche attività di contenzione della stessa. Tuttavia, questa funzione ulteriore eseguita dall’IPS conduce ad una maggiore latenza sulla rete.

Al contrario, l’IDS lavora “in maniera passiva” e si limita solamente ad inviare segnalazioni al firewall.

L’obiettivo principale di un IPS è impedire che gli attacchi informatici possano danneggiare il sistema o la rete. Ad ogni modo, nel caso degli IPS si rende di assoluta rilevanza la necessità di una corretta configurazione del dispositivo e la relativa costante manutenzione, allo scopo di evitare falsi positivi, ovvero il blocco di traffico legittimo.

Fase 1 sub 1: la distribuzione

Gli IDS/IPS andranno posizionati in punti strategici della rete, per massimizzare la loro efficacia.

Buona norma è metterli “a monte” del firewall, comunque vicino ai punti di ingresso/uscita del traffico di rete.

Dal momento che il NAS è il dispositivo più importante abbiamo deciso di installare un IPS allo switch del piano terra.

Anche al netto della latenza generata dall’IPS si è comunque deciso di inserirne uno a presidio del NAS, per garantire una maggiore sicurezza di un così importante dispositivo.

Report Progetto Build Week

Come terzo sistema proponiamo, infine, di posizionare un IDS nel piano o sezione dell'edificio dedicato all'amministrazione e alla gestione di dati sensibili. Nel progetto di rete si è deciso in modo arbitrario di prendere come assunto che tale reparto si collochi all'ultimo piano dell'edificio (il quinto). Quest'ultimo assunto, ovviamente, andrà ad adeguarsi alle diverse necessità della committenza.

Report Progetto Build Week

Fase 1 sub 2: il prezzario

Nell'immagine a seguire si evidenziano le risultanze dello studio relativo ai costi dei soli materiali impiegati nella proposta di progetto descritta nell'immagine di cui sopra.

Prezzario					
Componente	Quantità	Modello	Dove	Importo	Note
Switch	6	Cisco-Business-CBS220	Negozi	2.250,00 €	
Access-Point	6	Cisco-Business-240AC	Negozi	1.176,00 €	
Router	1	Mikrotik CCR2116	Negozi	956,00 €	
NAS	1	Synology-DiskStation-DS1621+	Negozi	941,00 €	
HDD-NAS	4	Synology-Hat3310-8T	Negozi	972,00 €	
Firewall	1	CiscoFirepower-Firewall-1120	Negozi	2.634,00 €	
NIDS	1	Snort	Sito	0,00 €	software gratuito
IPS	1	Fortinet-Fortigate-60F	Negozi	547,00 €	
Server-DHCP	1	HPE-Proliant-Microserver-Gen11	Negozi	975,00 €	
Cablaggio				350,00 €	
Totale				10.801,00 €	
Totale Con riserva Router+1/Switch+1/Access-Point+1				12.328,00 €	

Descrizione dei prodotti:

- gli switch che abbiamo scelto sono i Cisco-Business-CBS 220.
Sono switch da 48 porte con un'installazione semplice e un'ottima efficienza energetica. Riteniamo che siano i migliori dal punto di vista qualità prezzo
- gli access-point che abbiamo scelto sono i Cisco-Business-240AC.
Sono access-point che riescono a supportare fino a 200 dispositivi e danno una copertura massima di 3000 metri quadrati;
- il router scelto è un Cisco-RV345-K9-G5.
Questo router ha una connessione 4G che riteniamo sia sufficiente ed ha un ottimo prezzo;
- il firewall che abbiamo scelto è il Cisco Secure Firewall 1010, ha 8 porte GbE e un'ottima velocità di trasmissione;

Report Progetto Build Week

- il NAS che abbiamo scelto è il Synology DS1621+, è un NAS veloce e stabile con un'ottima capacità di archiviazione, con l'aggiunta degli hard disk ufficiali Synology Hat3310-BT, ottimizzati per il NAS stesso;
- l'IPS Fortinet-Fortigate-60F è un ottimo IPS con una latenza bassa e un'ottima velocità;
- un server DHCP HPE-Proliant-Microserver-Gen11, per gestire efficientemente gli indirizzi IP di ogni macchina host presente sulla rete.

Report Progetto Build Week

Fase 2

In questa fase ci occuperemo della programmazione in python dei tool per effettuare i test sulla rete.

Fase 2 sub 1: il programma di verifica dei verbi HTTP

Scriveremo un programma in Python per inviare richieste HTTP (GET, POST, PUT, DELETE) al web server e verificare le risposte.

La libreria `requests` è una libreria usata in Python per inviare facilmente richieste HTTP.

Con questa libreria, possiamo fare tutte le operazioni necessarie (GET, POST, PUT, DELETE).

Successivamente inseriamo l'indirizzo del server DVWA al quale inviare le richieste. Se il server DVWA è su una macchina diversa o su una porta diversa, dovremo modificare questo URL.

```
import requests  
#IMPOSTA L'URL DEL SERVER DVWA  
url = "inserire indirizzo ip con http:// davanti" #MODIFICA L'URL IN BASE AL TUO SERVE DVWA
```

La funzione che andremo ad utilizzare è la parte più importante del programma. Gestisce tutte le richieste HTTP (GET, POST, PUT, DELETE) in modo unificato, evitando di scrivere codice duplicato per ogni tipo di richiesta.

Method: Una stringa che indica il tipo di richiesta HTTP da inviare. Può essere "GET", "POST", "PUT" o "DELETE".

Url: L'indirizzo del server al quale inviare la richiesta.

data: (Opzionale) I dati da inviare nel corpo della richiesta, utilizzato solo per i metodi POST e PUT. Per esempio, nel caso di POST, potresti voler inviare un nome utente e una password.

Report Progetto Build Week

```
#FUNZIONE PER INVIARE UNA RICHIESTA HTTP GENERICA (GET, POST, PUT, DELETE)

def send_request (method, url, data = None):

    try:
        if method == "GET":
            response=requests.get(url)
        elif method == "POST":
            response=requests.post(url,data=data)
        elif method == "PUT":
            response=requests.put(url,data=data)
        elif method == "DELETE":
            response=requests.delete(url)
        else:
            print("metodo http non supportato")
```

In base al tipo di richiesta (method):

Se il method è "GET", la funzione invia una richiesta GET al server con requests.get(url).

Se il method è "POST", invia una richiesta POST con requests.post(url, data=data), includendo i dati passati come data.

Se il method è "PUT", invia una richiesta PUT con requests.put(url, data=data), includendo i dati passati come data.

Se il method è "DELETE", invia una richiesta DELETE con requests.delete(url).

Se il method non è uno di quelli supportati, la funzione genera un errore tramite raise ValueError("Metodo HTTP non supportato").

```
#STAMPA LA RISPOSTA

    print("\n{method} Request:")
    print(f"status code: {response.status_code}")
    print(f"response text: {response.text[:200]}...") #MOSTRA SOLO I PRIMI 200 CARATTERI

except requests.exceptions.RequestException as e:
    print(f"errore nella richiesta {method}:{e}")

#FUNZIONE PRINCIPALE

def main ():
    #INVIO DELLE RICHIESTE CON I METODI HTTP

        send_request("GET", url)
        send_request("POST", url, data={"username": "admin","password": "password"})
        send_request("PUT", url, data={"key": "value"})
        send_request("DELETE", url)
```

Una volta inviata la richiesta, il programma stampa:

Il codice di stato HTTP (response.status_code), che indica se la richiesta ha avuto successo (ad esempio, 200 per OK).

Report Progetto Build Week

I primi 200 caratteri della risposta del server (response.text[:200]), per dare un'idea di cosa è stato restituito senza stamparlo tutto.

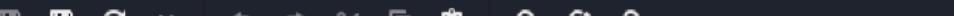
La funzione main() è dove vengono effettivamente inviate le richieste HTTP. All'interno di questa funzione, chiamiamo la funzione send_request per ogni tipo di richiesta (GET, POST, PUT, DELETE) con i dati appropriati.

Dettaglio:

- `send_request("GET", url)`: Invia una richiesta GET al server (recupera la pagina o le informazioni).
 - `send_request("POST", url, data={"username": "admin", "password": "password"})`: Invia una richiesta POST al server con dei dati (in questo caso, un nome utente e una password).
 - `send_request("PUT", url, data={"key": "value"})`: Invia una richiesta PUT per aggiornare o modificare qualcosa sul server.
 - `send_request("DELETE", url)`: Invia una richiesta DELETE per eliminare qualcosa dal server.

Fase 2 sub 2: il programma di scansione delle porte

In questa seconda parte delle presente fase, invece, progetteremo un programma in Python per eseguire una scansione delle porte sui dispositivi di rete, verificando la sicurezza e l'accessibilità delle varie porte di comunicazione.



```
1 import socket
2 import time
3
```

Importazione dei Moduli:

Per iniziare abbiamo importato i due moduli `socket` e `time`

socket: permette di creare connessioni di rete e la usiamo in questo caso per tentare di connettersi alle porte dei dispositivi

time: Utilizzato per calcolare il tempo necessario, e quindi della durata alla scansione.

Report Progetto Build Week

```
3 #Funzione per scansionare una singola porta
4 def scan_port(target, port):
5     try:
6         #Creazione di un socket
7         s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8         s.settimeout(1)      #Timeout per ogni connessione
9         result = s.connect_ex((target, port)) #Tentativo di connessione
10        if result == 0:
11            print (f"[+] Porta {port} è APERTA su {target}")
12        else:
13            print (f"[-] Porta {port} è CHIUSA su {target}")
14            s.close()
15    except Exception as e:
16        print (f"Errore nella scansione della porta {port}: {e}")
17
18
```

Funzione scan_port

- `scan_port` : è la funzione che si occupa di scansionare una singola porta

Parametri

- `target` : indirizzo IP dell'host da scansionare
- `port` : porta specifica da testare nella scansione

Azioni

- Creiamo un socket TCP (`Socket.socket`)
- Impostiamo un timeout di 1 secondo per evitare attese infinite (`settimeout(1)`)
- Proviamo a connetterci alla porta dell'host tramite `connect_ex`.
- Se la connessione è riuscita (ritorna `0`), la porta è considerata **aperta**

Output: Stampa lo stato della porta (APERTA).

- In caso contrario, la porta è considerata **chiusa**.

Output: Stampa lo stato della porta (CHIUSA).

- Il socket dopo ogni tentativo, viene chiuso
- Gestione degli errori: Se si verifica un'eccezione, stampa un messaggio d'errore

```
18
19 #Funzione principale per la scansione delle porte
20 def scan_ports(target, ports):
21     print (f"[*] Avvio scansione delle porte su {target} ... ")
22     start_time = time .time()
23
24     for port in ports:
25         scan_port(target, port)
26
27     end_time = time .time()
28     print (f"[*] Scansione completata in {round(end_time - start_time, 2)} secondi.\n")
29
```

Funzione scan_ports

Report Progetto Build Week

Parametri:

- **target**: Indirizzo IP dell'host da scansionare.
- **ports**: Lista di porte da controllare (intervallo).

Azioni:

- **Scan_ports()** : scansiona un range di porte su un dispositivo specifico.
- **Start-time**: registra il tempo di inizio della scansione
- **For port in ports**: scorre tutte le porte specificate e chiama scan_port() per ciascuna di esse.
- **End_time** : calcola il tempo impiegato per completare la scansione.
- **Print()**: stampa il tempo totale della scansione, calcolando la differenza tra la fine e l'inizio della scansione.

```
#Target e range di porte
targets = ["192.168.1.129"]
port_range = range(1, 1025) #Scansione porte da 1 a 1024

#Esecuzione della scansione per ogni target
for target in targets:
    scan_ports(target, port_range)
```

Configurazione dei Target e Range di Porte

- **targets**: Lista contenente gli indirizzi IP degli host da scansionare
- **port_range**: Intervallo di porte da scansionare (da 1 a 1024).

Esecuzione del Codice

- Per ogni target nella lista, viene avviata la scansione tramite la funzione **scan_ports**.

Ciclo for per la scansione

Il seguente codice scansiona un elenco di Ip target; per ciascun dispositivo prova a connettersi su ogni porta da 1 a 1024, se la connessione riesce, stampa che la porta è aperta e infine al termine della scansione mostra il tempo totale impiegato.

Il codice utilizza **socket** per eseguire una scansione delle porte TCP su più dispositivi e permette di identificare le porte aperte. Un socket TCP (socket.SOCK_STREAM) viene creato per ogni porta che si vuole analizzare su un determinato IP. La funzione **connect_ex**

Report Progetto Build Week

del socket tenta di stabilire una connessione: Se il risultato è 0, significa che la porta è aperta e accetta connessioni. Un valore diverso da 0 indica che la porta è chiusa o non accessibile. Esso è una connessione diretta al dispositivo target per verificare lo stato delle porte; è altamente personalizzabile, consentendo di configurare timeout e protocolli di rete (ad esempio, TCP o UDP). Fornisce un'astrazione semplice per operazioni di rete complesse. Considerazioni di sicurezza: Utilizzare socket per la scansione delle porte su dispositivi non autorizzati può essere considerato un'attività malevola (ad esempio, ricognizione in un attacco informatico). È importante rispettare le normative legali e scansionare solo dispositivi di cui si possiede il controllo o l'autorizzazione.

Report Progetto Build Week

Fase 3

In questa fase si procederà alla implementazione del Web server nella DMZ del progetto di rete.

Sul piano procedimentale, si precisa che in questa fase di progettazione sono state impiegate tre macchine virtuali, quali:

- una v.m. con un sistema operativo “Kali Linux”;
- una v.m. con installata la distribuzione “Pfsense”;
- una v.m. con installato il software “Metasploitable 2”.

Allo scopo di impostare un firewall per schermare la costituenda rete per la Società committente, si è proceduto in via preliminare ad assegnare una serie di indirizzi IP, netmask e gateway ai tre strumenti impiegati allo scopo di prima mettere in comunicazione le tre v.m.

In primo luogo, si specifica che ai fini di interesse sia la macchina Kali-Linux che la macchina Metasploitable sono munite di una sola scheda di rete, ossia una scheda di rete interna.

Al contrario, la macchina Pfsense è provvista di ben tre schede di rete, la prima di tipo NAT e le altre sono schede di rete interna. Le schede di rete interna di Pfsense andranno impostate, una come rete LAN e associata ad un IP che insista sulla medesima rete della macchina Kali Linux. Similmente, la seconda scheda di rete interna sarà impostata sulla medesima rete di Metasploitable, nella rete chiamata “OPT1”.

A seguire si propone una tabella esplicativa, nella quali si riportano i dati di interesse.

	IPv4 Address	Netmask	Gateway
Kali	192.168.5.5	24	192.168.5.1
Pfsense LAN	192.168.5.1	24	192.168.5.1
Pfsense OPT1	192.168.1.1	24	192.168.1.1
Metasploitable	192.168.1.3	24	192.168.1.3

Report Progetto Build Week

Come passaggio immediatamente successivo abbiamo proceduto ad agire direttamente a livello di Firewall.

Attraverso la Gui di Pfsense si sono formulate regole apposite allo scopo di consentire la comunicazione tra Kali e Metasploitable.

Precisamente, nella rete LAN si sono inserite due regole

- una che consente la trasmissione dati dalla subnet di OPT1 alla rete LAN,
- una seconda che consente la trasmissione dati dalla subnet della rete LAN a quella della OPT1 (regola speculare).

The screenshot shows the 'Edit Firewall Rule' configuration page. It is divided into several sections:

- Action:** Set to "Pass". A note below says: "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:** An unchecked checkbox labeled "Disable this rule". A note below says: "Set this option to disable this rule without removing it from the list."
- Interface:** Set to "LAN". A note below says: "Choose the interface from which packets must come to match this rule."
- Address Family:** Set to "IPv4". A note below says: "Select the Internet Protocol version this rule applies to."
- Protocol:** Set to "Any". A note below says: "Choose which IP protocol this rule should match."
- Source:** A section with "Source" dropdown set to "LAN subnets" and "Source Address" field.
- Destination:** A section with "Destination" dropdown set to "OPT1 subnets" and "Destination Address" field.

Similmente si è proceduto anche per le regole della OPT1, consentendone il dialogo con la rete LAN con due regole apposite.

Report Progetto Build Week

Edit Firewall Rule

Action	Pass	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 type="checkbox"/> Disable this rule	Set this option to disable this rule without removing it from the list.
Interface	OPT1	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	Any	Choose which IP protocol this rule should match.
Source		
Source	<input type="checkbox"/> Invert match	OPT1 subnets
Destination		
Destination	<input type="checkbox"/> Invert match	LAN subnets

Edit Firewall Rule

Action	Pass	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 type="checkbox"/> Disable this rule	Set this option to disable this rule without removing it from the list.
Interface	OPT1	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	Any	Choose which IP protocol this rule should match.
Source		
Source	<input type="checkbox"/> Invert match	LAN subnets
Destination		
Destination	<input type="checkbox"/> Invert match	OPT1 subnets

Ciò fatto si sono eseguite le verifiche in ordine alla comunicatività da parte Metasploitable e Kali in verso le altre direzioni.

Nell'immagine a seguire vi sono riportati i ping test eseguiti sia su Kali (a sinistra) sia su Metasploitable (a destra).

Report Progetto Build Week

The screenshot shows a terminal window with several tabs open, displaying network traffic analysis and configuration steps. The tabs include:

- File / Actions / Edit / View / Help**: The main menu.
- zsh: corrupt history file /home/kali/.zsh_history**: A command-line history corruption attempt.
- [kali@kali] -~]**: The current terminal session.
- \$ ping 192.168.5.1**: Pings the gateway.
- PING 192.168.5.1 (192.168.5.1) 56(84) bytes of data.**: The response from the ping command.
- 64 bytes from 192.168.5.1: icmp_seq=1 ttl=64 time=0.304 ms**
- 64 bytes from 192.168.5.1: icmp_seq=2 ttl=64 time=0.393 ms**
- 64 bytes from 192.168.5.1: icmp_seq=3 ttl=64 time=0.311 ms**
- ^C**: The interrupt key (^C).
- 192.168.5.1 ping statistics --**
- 3 packets transmitted, 3 received, 0% packet loss, time 2052ms**
- rtt min/avg/max/mdev = 0.311/0.356/0.393/0.033 ms**
- [kali@kali] -~]**: Another terminal session.
- \$ ping 192.168.1.1**: Pings the Kali Linux host.
- PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.**
- 64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.350 ms**
- 64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.380 ms**
- 64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.288 ms**
- ^C**
- 192.168.1.1 ping statistics --**
- 3 packets transmitted, 3 received, 0% packet loss, time 2048ms**
- rtt min/avg/max/mdev = 0.288/0.339/0.380/0.038 ms**
- [kali@kali] -~]**: Another terminal session.
- \$ ping 192.168.1.3**: Pings the Metasploitable host.
- PING 192.168.1.3 (192.168.1.3) 56(84) bytes of data.**
- 64 bytes from 192.168.1.3: icmp_seq=1 ttl=63 time=4.58 ms**
- 64 bytes from 192.168.1.3: icmp_seq=2 ttl=63 time=0.736 ms**
- 64 bytes from 192.168.1.3: icmp_seq=3 ttl=63 time=0.798 ms**
- ^C**
- 192.168.1.3 ping statistics --**
- 3 packets transmitted, 3 received, 0% packet loss, time 2008ms**
- rtt min/avg/max/mdev = 0.736/2.039/4.584/1.799 ms**
- [kali@kali] -~]**: Another terminal session.
- \$**: The prompt.

On the right side of the terminal window, there is a status bar with various icons and text, including:

- kali@kali: ~**
- Metasploitable [In esecuzione] - Oracle VirtualBox**
- File Macchina Visualizza Inserimento Dispositivi Aiuto**
- 64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.432 ms**
- 64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.304 ms**
- 192.168.1.1 ping statistics --**
- 3 packets transmitted, 3 received, 0% packet loss, time 1998ms**
- rtt min/avg/max/mdev = 0.304/0.368/0.432/0.052 ms**
- msfadmin@metasploitable: ~\$ ping 192.168.5.1**
- PING 192.168.5.1 (192.168.5.1) 56(84) bytes of data.**
- 64 bytes from 192.168.5.1: icmp_seq=1 ttl=64 time=0.448 ms**
- 64 bytes from 192.168.5.1: icmp_seq=2 ttl=64 time=0.379 ms**
- 64 bytes from 192.168.5.1: icmp_seq=3 ttl=64 time=0.356 ms**
- 192.168.5.1 ping statistics --**
- 3 packets transmitted, 3 received, 0% packet loss, time 1998ms**
- rtt min/avg/max/mdev = 0.356/0.394/0.448/0.042 ms**
- msfadmin@metasploitable: ~\$ ping 192.168.5.5**
- PING 192.168.5.5 (192.168.5.5) 56(84) bytes of data.**
- 64 bytes from 192.168.5.5: icmp_seq=1 ttl=63 time=1.27 ms**
- 64 bytes from 192.168.5.5: icmp_seq=2 ttl=63 time=0.762 ms**
- 64 bytes from 192.168.5.5: icmp_seq=3 ttl=63 time=0.716 ms**
- 192.168.5.5 ping statistics --**
- 3 packets transmitted, 3 received, 0% packet loss, time 1998ms**
- rtt min/avg/max/mdev = 0.716/0.918/1.278/0.257 ms**
- msfadmin@metasploitable: ~\$**
- 2.7.2-RELEASE**
- built on Wed Jul 10 14:44:20 UTC 2019**
- FreeBSD 14.0-CURRENT**
- You also may use**
- Reboot**
- Support subsonic**
- Recovery**
- Ctrl (Destra)**
- CTRL (DESTRA)**

Successivamente si è proceduto con la implementazione di una regola per impedire la comunicazione dal server nella DMZ (La rete di Metas) verso la rete LAN (di Kali).

Per fare ciò è stata creata una regola apposita attraverso la Gui di PfSense, nella sezione della rete OPT1 (rete della DMZ), imponendo il comando di blocco di tutti i protocolli TCP / Anche any per essere più sicuri; come source Network, con indirizzo della rete della DMZ (192.168.1.0/24) e come destination LAN subnet (ossia la rete interna).

Report Progetto Build Week

The screenshot shows a configuration interface for a network rule. The rule is currently set to **Action: Block**. It is **Disabled** and applies to **Interface: OPT1**, **Address Family: IPv4**, and **Protocol: TCP**. The source is set to **Source: 192.168.1.0 / 24** and the destination port range is **Destination Port Range: any / any**. There is also an **Extra Options** section.

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: OPT1
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 **Network:** LAN subnets **Address:** 192.168.1.0 / 24 **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.

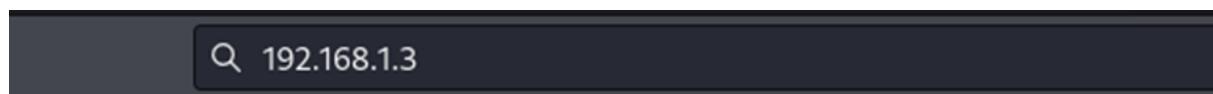
Destination:

Destination: Invert match **LAN subnets:** LAN subnets **Address:** Destination Address / **From:** any **To:** any **Custom:** Custom
Specify the destination port or port range for this rule. The "To" field may be left empty if only filtering a single port.

Extra Options:

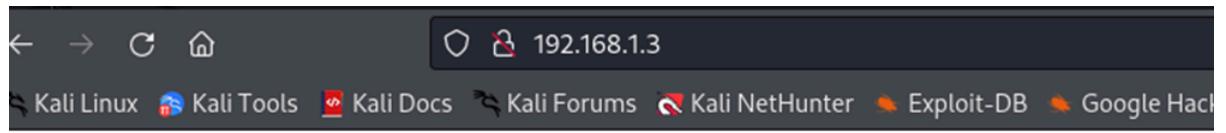
Passaggio successivo consiste nell'eseguire il login su phpMyadmin, dalla Gui di Metasploitable.

A tale scopo è necessario inserire l'indirizzo IP di Metas nella barra di ricerca del browser di Kali.



Dunque avviare la ricerca ed entrare nella Gui di Metas.

Report Progetto Build Week



```
arning: Never expose this VM to an untrusted network!
```

```
ontact: msfdev[at]metasploit.com
```

```
ogin with msfadmin/msfadmin to get started
```

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

A questo punto si rende necessario eseguire due attività:

- la prima è stata quella di settare la funzione DVWA di Metasploitable, cliccando sulla apposita sezione nella Gui e seguendo i passaggi guidati dal programma.

Report Progetto Build Week

The screenshot shows the DVWA Security interface. On the left is a sidebar menu with the following items:

- Home
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored
- DVWA Security** (highlighted in green)
- PHP Info
- About
- Logout

The main content area has a title "DVWA Security" with a padlock icon. Below it is a section titled "Script Security". It contains the following text and form:

Security Level is currently **high**.
You can set the security level to low, medium or high.
The security level changes the vulnerability level of DVWA.

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.
You can enable PHPIDS across this site for the duration of your session.
PHPIDS is currently **disabled**. [[enable PHPIDS](#)]
[[Simulate attack](#)] - [[View IDS log](#)]

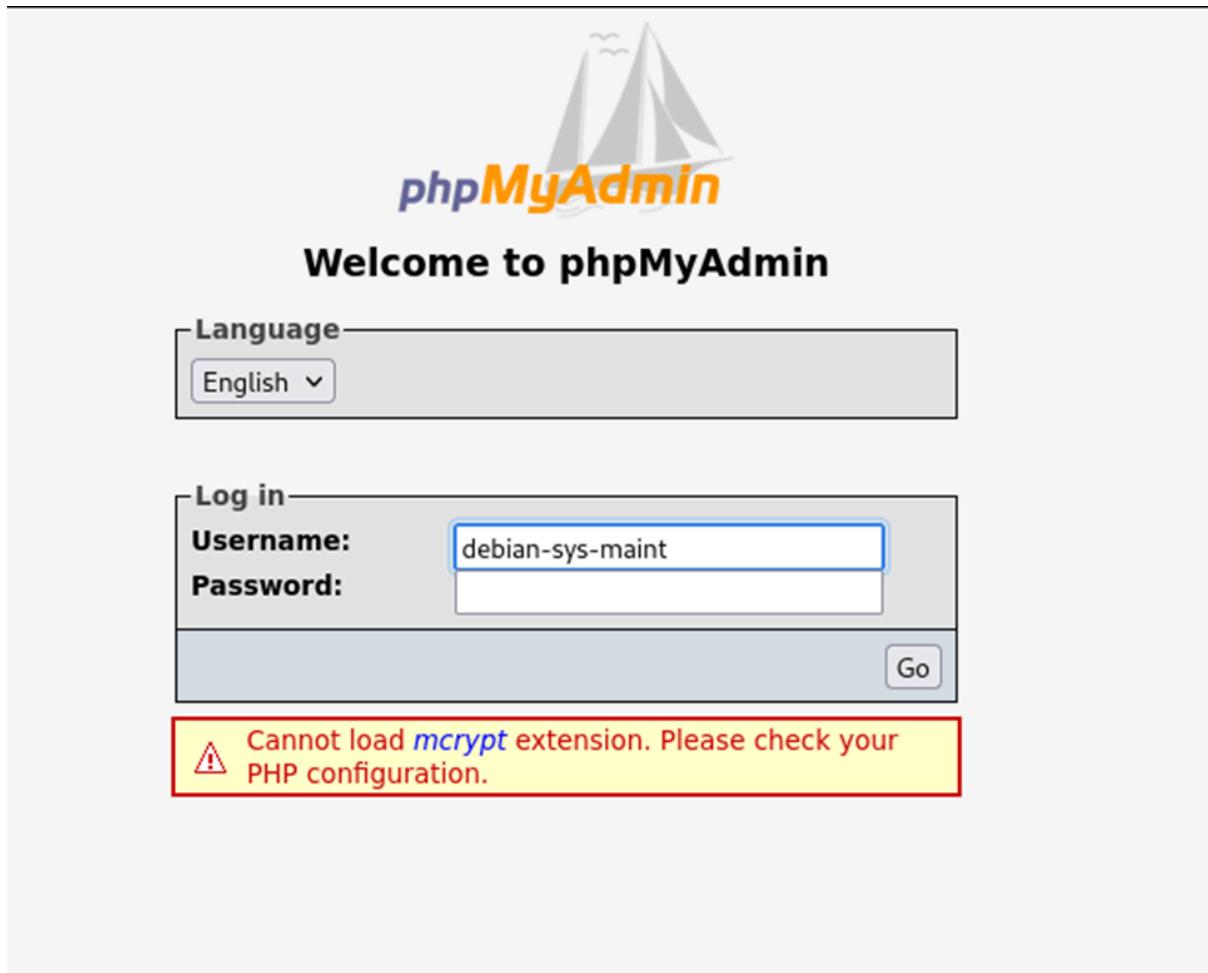
Nel caso che ci interessa abbiamo provveduto a settare la DVWA su “high”.

Questa funzione di Metasploitable, materialmente collocata all’interno della DMZ della rete, in una posizione inoffensiva per la stessa in quanto impossibilitata a comunicarvi, consentirà in futuro di eseguire i penetration test.

Test che si raccomanda di eseguire almeno una o due volte l’anno, allo scopo di tenere sotto controllo lo stato di sicurezza dell’interno sistema di rete

- La seconda attività è stata invece quella di cliccare su phpMyAdmin ed eseguire il login con le credenziali di default, se non precedentemente modificate.

Report Progetto Build Week



A questo punto si sarà finalmente in grado di accedere alla home di phpMyAdmin.

A screenshot of the phpMyAdmin home page. The top navigation bar shows "Server: localhost" with links for Databases, SQL, Status, Variables, Charsets, Engines, Privileges, Processes, Export, and Import. The left sidebar lists databases: dwva (2), information_schema (17), metasploit, mysql (17), owncloud (6), tikiwiki (194), and tikiwiki195 (194). A message says "Please select a database". The main content area shows "Actions" with "Create new database" and "MySQL localhost" sections. It includes a "Collation" dropdown and a "Create" button. The "Interface" section has dropdowns for "Language" (English), "Theme / Style" (Original), "Custom color" (Reset), and "Font size" (82%). On the right, there are sections for "MySQL", "Web server", and "phpMyAdmin". The "MySQL" section shows the server version as 5.0.51a-3ubuntu5. The "Web server" section shows Apache2.2.8 (Ubuntu) DAV/2. The "phpMyAdmin" section shows the version as 3.1.1. A red warning box at the bottom contains the text "⚠ Cannot load mcrypt extension. Please check your PHP configuration." and "⚠ The configuration file now needs a secret passphrase (blowfish, secret.)". A "phpMyAdmin" logo is at the bottom right.

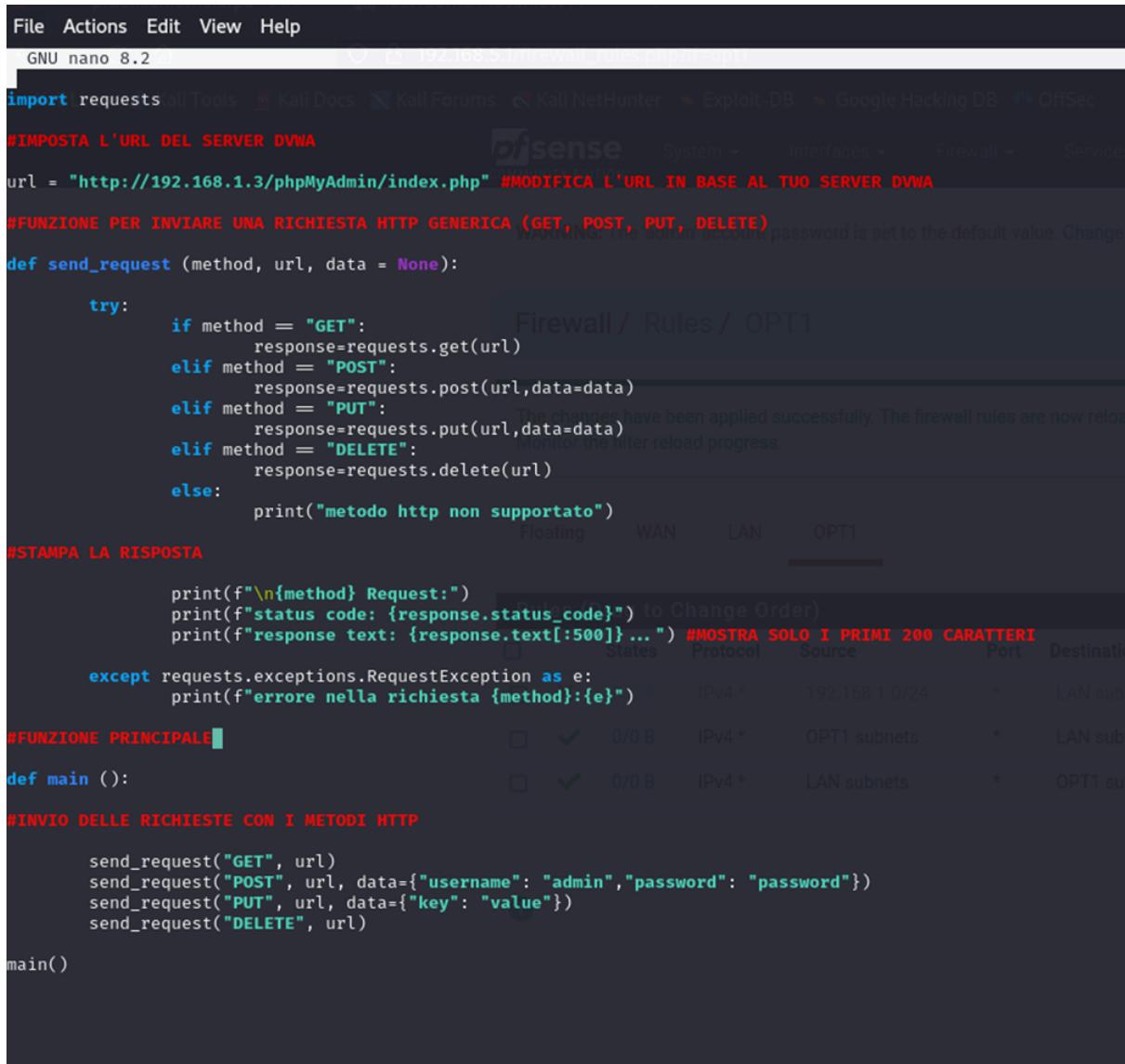
Report Progetto Build Week

Fase 3 sub 1: test dei programmi python

Ciò fatto, procediamo con il lancio dei programmi python per la verifica dei verbi HTTP e la scansione delle porte.

Per il primo test, la verifica dei verbi http si allegano le due immagini a seguire.

La prima riporta il codice finale impiegato nei test.



```
File Actions Edit View Help
GNU nano 8.2
import requests
#IMPOSTA L'URL DEL SERVER DVWA
url = "http://192.168.1.3/phpMyAdmin/index.php" #MODIFICA L'URL IN BASE AL TUO SERVER DVWA
#FUNZIONE PER INVIARE UNA RICHIESTA HTTP GENERICA (GET, POST, PUT, DELETE)
def send_request (method, url, data = None):
    try:
        if method == "GET":
            response=requests.get(url)
        elif method == "POST":
            response=requests.post(url,data=data)
        elif method == "PUT":
            response=requests.put(url,data=data)
        elif method == "DELETE":
            response=requests.delete(url)
        else:
            print("metodo http non supportato")
    except requests.exceptions.RequestException as e:
        print(f"errore nella richiesta {method}:{e}")
#STAMPA LA RISPOSTA
def main ():
    #INVIO DELLE RICHIESTE CON I METODI HTTP
    send_request("GET", url)
    send_request("POST", url, data={"username": "admin","password": "password"})
    send_request("PUT", url, data={"key": "value"})
    send_request("DELETE", url)
main()
```

Report Progetto Build Week

```
$ python 1234.py

GET Request:
status code: 200
response text: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link rel="icon" href=".//favicon.ico" type="image/x-icon" />
  <link rel="shortcut icon" href=".//favicon.ico" type="image/x-icon" />
  <title>phpMyAdmin </title>
  <link rel="stylesheet" type="text/css" hr ...

POST Request:
status code: 200
response text: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link rel="icon" href=".//favicon.ico" type="image/x-icon" />
  <link rel="shortcut icon" href=".//favicon.ico" type="image/x-icon" />
  <title>phpMyAdmin </title>
  <link rel="stylesheet" type="text/css" hr ...

PUT Request:
status code: 200
response text: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link rel="icon" href=".//favicon.ico" type="image/x-icon" />
  <link rel="shortcut icon" href=".//favicon.ico" type="image/x-icon" />
  <title>phpMyAdmin </title>
  <link rel="stylesheet" type="text/css" hr ...

DELETE Request:
status code: 200
response text: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link rel="icon" href=".//favicon.ico" type="image/x-icon" />
  <link rel="shortcut icon" href=".//favicon.ico" type="image/x-icon" />
  <title>phpMyAdmin </title>
  <link rel="stylesheet" type="text/css" hr ...

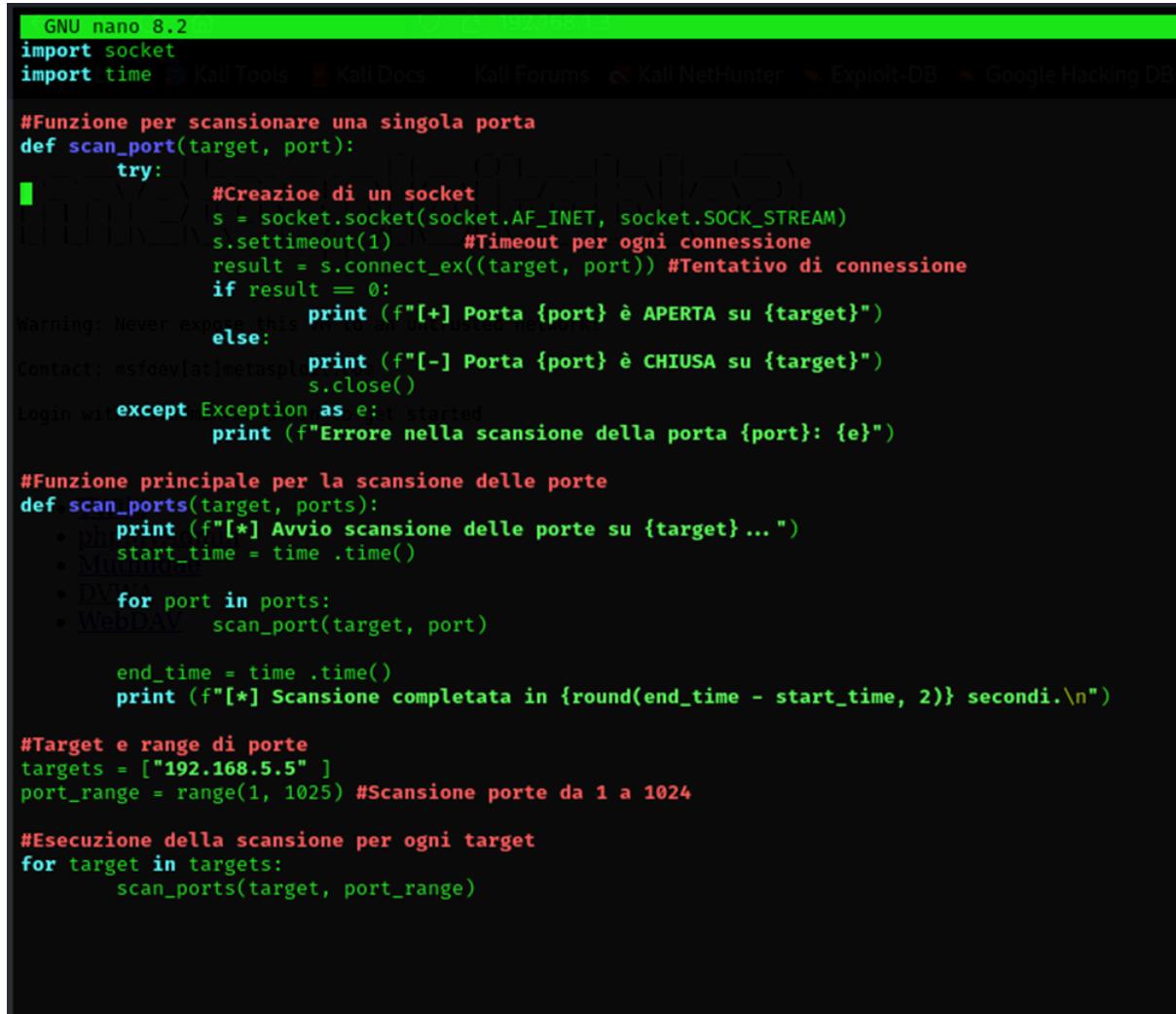

(kali㉿kali)-[~]
```

Mentre l'immagine che precede presenta l'output estratto dal codice lanciato. Come è possibile vedere, si riscontrano risultati positivi su ogni verbo HTTP, come evidenziato dall'esito “status code 200”, riportato dopo ogni richiesta.

Report Progetto Build Week

Fase 3 sub 2: test sulla scansione delle porte

In via analoga a quanto fatto per il primo test, a seguire si riporta il testo del codice utilizzato per la prova e successivamente gli esiti.



```
GNU nano 8.2
import socket
import time
#Funzione per scansionare una singola porta
def scan_port(target, port):
    try:
        #Creazione di un socket
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(1)      #Timeout per ogni connessione
        result = s.connect_ex((target, port)) #Tentativo di connessione
        if result == 0:
            print(f"[+] Porta {port} è APERTA su {target}")
        else:
            print(f"[-] Porta {port} è CHIUSA su {target}")
            s.close()
    except Exception as e:
        print(f"Errore nella scansione della porta {port}: {e}")

#Funzione principale per la scansione delle porte
def scan_ports(target, ports):
    print(f"[*] Avvio scansione delle porte su {target} ... ")
    start_time = time.time()
    for port in ports:
        scan_port(target, port)

    end_time = time.time()
    print(f"[*] Scansione completata in {round(end_time - start_time, 2)} secondi.\n")

#Target e range di porte
targets = ["192.168.5.5"]
port_range = range(1, 1025) #Scansione porte da 1 a 1024

#Esecuzione della scansione per ogni target
for target in targets:
    scan_ports(target, port_range)
```

Si osservi che nell'immagine di cui sopra si scansionano le porte della macchina Kali-Linux (IP 192.168.5.5), a titolo meramente esemplificativo.

Gli esiti di cui a seguire sono relativi anche ai test eseguiti sulla macchina Metasploitable 2 (IP 192.168.1.3).

Report Progetto Build Week

Esiti test sulle porte Metasploitable 2:

```
(kali㉿kali)-[~]
$ python porte.py
[*] Avvio scansione delle porte su 192.168.1.3 ...
[-] Porta 1 è CHIUSA su 192.168.1.3
[-] Porta 2 è CHIUSA su 192.168.1.3
[-] Porta 3 è CHIUSA su 192.168.1.3
[-] Porta 4 è CHIUSA su 192.168.1.3
[-] Porta 5 è CHIUSA su 192.168.1.3
[-] Porta 6 è CHIUSA su 192.168.1.3
[-] Porta 7 è CHIUSA su 192.168.1.3
[-] Porta 8 è CHIUSA su 192.168.1.3
[-] Porta 9 è CHIUSA su 192.168.1.3
[-] Porta 10 è CHIUSA su 192.168.1.3
[-] Porta 11 è CHIUSA su 192.168.1.3
[-] Porta 12 è CHIUSA su 192.168.1.3
[-] Porta 13 è CHIUSA su 192.168.1.3
[-] Porta 14 è CHIUSA su 192.168.1.3
[-] Porta 15 è CHIUSA su 192.168.1.3
[-] Porta 16 è CHIUSA su 192.168.1.3
[-] Porta 17 è CHIUSA su 192.168.1.3
[-] Porta 18 è CHIUSA su 192.168.1.3
[-] Porta 19 è CHIUSA su 192.168.1.3
[-] Porta 20 è CHIUSA su 192.168.1.3
[+] Porta 21 è APERTA su 192.168.1.3
[+] Porta 22 è APERTA su 192.168.1.3
[+] Porta 23 è APERTA su 192.168.1.3
[-] Porta 24 è CHIUSA su 192.168.1.3
[+] Porta 25 è APERTA su 192.168.1.3
[-] Porta 26 è CHIUSA su 192.168.1.3
[-] Porta 27 è CHIUSA su 192.168.1.3
[-] Porta 28 è CHIUSA su 192.168.1.3
[-] Porta 29 è CHIUSA su 192.168.1.3
[-] Porta 30 è CHIUSA su 192.168.1.3
[-] Porta 31 è CHIUSA su 192.168.1.3
[-] Porta 32 è CHIUSA su 192.168.1.3
[-] Porta 33 è CHIUSA su 192.168.1.3
[-] Porta 34 è CHIUSA su 192.168.1.3
[-] Porta 35 è CHIUSA su 192.168.1.3
[-] Porta 36 è CHIUSA su 192.168.1.3
[-] Porta 37 è CHIUSA su 192.168.1.3
[-] Porta 38 è CHIUSA su 192.168.1.3
[-] Porta 39 è CHIUSA su 192.168.1.3
[-] Porta 40 è CHIUSA su 192.168.1.3
[-] Porta 41 è CHIUSA su 192.168.1.3
[-] Porta 42 è CHIUSA su 192.168.1.3
[-] Porta 43 è CHIUSA su 192.168.1.3
[-] Porta 44 è CHIUSA su 192.168.1.3
[-] Porta 45 è CHIUSA su 192.168.1.3
[-] Porta 46 è CHIUSA su 192.168.1.3
[-] Porta 47 è CHIUSA su 192.168.1.3
[-] Porta 48 è CHIUSA su 192.168.1.3
```

Report Progetto Build Week

The screenshot shows a terminal window on a Kali Linux system (kali㉿kali) performing a port scan on 192.168.1.3 using the command `python porte.py`. The output lists ports 1 through 48 as closed ('CHIUSA'). Simultaneously, a browser window is open to the pfSense firewall rules configuration page (192.168.5.1/firewall_rules.php?if=opt1). A modal dialog in the browser indicates that changes have been applied successfully. The main table lists three existing rules, all of which have been disabled ('X') and are set to IPv4.

	States	Protocol
<input type="checkbox"/>	X	0/0 B IPv4 *
<input type="checkbox"/>	✓	0/0 B IPv4 *
<input type="checkbox"/>	✓	0/0 B IPv4 *

Report Progetto Build Week

The screenshot shows a terminal window displaying a list of port status entries. The entries are as follows:

- [-] Porta 48 è CHIUSA su 192.168.1.3
- [-] Porta 49 è CHIUSA su 192.168.1.3
- [-] Porta 50 è CHIUSA su 192.168.1.3
- [-] Porta 51 è CHIUSA su 192.168.1.3
- [-] Porta 52 è CHIUSA su 192.168.1.3
- [+] Porta 53 è APERTA su 192.168.1.3
- [-] Porta 54 è CHIUSA su 192.168.1.3
- [-] Porta 55 è CHIUSA su 192.168.1.3
- [-] Porta 56 è CHIUSA su 192.168.1.3
- [-] Porta 57 è CHIUSA su 192.168.1.3
- [-] Porta 58 è CHIUSA su 192.168.1.3
- [-] Porta 59 è CHIUSA su 192.168.1.3
- [-] Porta 60 è CHIUSA su 192.168.1.3
- [-] Porta 61 è CHIUSA su 192.168.1.3
- [-] Porta 62 è CHIUSA su 192.168.1.3
- [-] Porta 63 è CHIUSA su 192.168.1.3
- [-] Porta 64 è CHIUSA su 192.168.1.3
- [-] Porta 65 è CHIUSA su 192.168.1.3
- [-] Porta 66 è CHIUSA su 192.168.1.3
- [-] Porta 67 è CHIUSA su 192.168.1.3
- [-] Porta 68 è CHIUSA su 192.168.1.3
- [-] Porta 69 è CHIUSA su 192.168.1.3
- [-] Porta 70 è CHIUSA su 192.168.1.3
- [-] Porta 71 è CHIUSA su 192.168.1.3
- [-] Porta 72 è CHIUSA su 192.168.1.3
- [-] Porta 73 è CHIUSA su 192.168.1.3
- [-] Porta 74 è CHIUSA su 192.168.1.3
- [-] Porta 75 è CHIUSA su 192.168.1.3
- [-] Porta 76 è CHIUSA su 192.168.1.3
- [-] Porta 77 è CHIUSA su 192.168.1.3
- [-] Porta 78 è CHIUSA su 192.168.1.3
- [-] Porta 79 è CHIUSA su 192.168.1.3
- [+] Porta 80 è APERTA su 192.168.1.3
- [-] Porta 81 è CHIUSA su 192.168.1.3
- [-] Porta 82 è CHIUSA su 192.168.1.3
- [-] Porta 83 è CHIUSA su 192.168.1.3
- [-] Porta 84 è CHIUSA su 192.168.1.3
- [-] Porta 85 è CHIUSA su 192.168.1.3
- [-] Porta 86 è CHIUSA su 192.168.1.3
- [-] Porta 87 è CHIUSA su 192.168.1.3
- [-] Porta 88 è CHIUSA su 192.168.1.3
- [-] Porta 89 è CHIUSA su 192.168.1.3
- [-] Porta 90 è CHIUSA su 192.168.1.3
- [-] Porta 91 è CHIUSA su 192.168.1.3
- [-] Porta 92 è CHIUSA su 192.168.1.3
- [-] Porta 93 è CHIUSA su 192.168.1.3
- [-] Porta 94 è CHIUSA su 192.168.1.3
- [-] Porta 95 è CHIUSA su 192.168.1.3
- [-] Porta 96 è CHIUSA su 192.168.1.3
- [-] Porta 97 è CHIUSA su 192.168.1.3
- [-] Porta 98 è CHIUSA su 192.168.1.3
- [-] Porta 99 è CHIUSA su 192.168.1.3

Report Progetto Build Week

The screenshot shows the pfSense Firewall Rules configuration interface. The title bar indicates the window is titled "pfSense.home.arpa - Firewall" and the URL is "192.168.1.3 / localhost | pfSense". The menu bar includes File, Actions, Edit, View, and Help. The main content area displays a list of 150 firewall rules, all of which are currently closed (CHIUSA). The list starts with Port 99 and ends with Port 150. The rules are listed in the following format: [] Porta 99 è CHIUSA su 192.168.1.3, followed by a series of 149 more entries. A floating window titled "Firewall / Rules (Draft)" is visible on the right side of the screen, containing a message about changes being made and a link to "Monitor the file".

```
[ ] Porta 99 è CHIUSA su 192.168.1.3
[ ] Porta 100 è CHIUSA su 192.168.1.3
[ ] Porta 101 è CHIUSA su 192.168.1.3
[ ] Porta 102 è CHIUSA su 192.168.1.3
[ ] Porta 103 è CHIUSA su 192.168.1.3
[ ] Porta 104 è CHIUSA su 192.168.1.3
[ ] Porta 105 è CHIUSA su 192.168.1.3
[ ] Porta 106 è CHIUSA su 192.168.1.3
[ ] Porta 107 è CHIUSA su 192.168.1.3
[ ] Porta 108 è CHIUSA su 192.168.1.3
[ ] Porta 109 è CHIUSA su 192.168.1.3
[ ] Porta 110 è CHIUSA su 192.168.1.3
[+] Porta 111 è APERTA su 192.168.1.3
[ ] Porta 112 è CHIUSA su 192.168.1.3
[ ] Porta 113 è CHIUSA su 192.168.1.3
[ ] Porta 114 è CHIUSA su 192.168.1.3
[ ] Porta 115 è CHIUSA su 192.168.1.3
[ ] Porta 116 è CHIUSA su 192.168.1.3
[ ] Porta 117 è CHIUSA su 192.168.1.3
[ ] Porta 118 è CHIUSA su 192.168.1.3
[ ] Porta 119 è CHIUSA su 192.168.1.3
[ ] Porta 120 è CHIUSA su 192.168.1.3
[ ] Porta 121 è CHIUSA su 192.168.1.3
[ ] Porta 122 è CHIUSA su 192.168.1.3
[ ] Porta 123 è CHIUSA su 192.168.1.3
[ ] Porta 124 è CHIUSA su 192.168.1.3
[ ] Porta 125 è CHIUSA su 192.168.1.3
[ ] Porta 126 è CHIUSA su 192.168.1.3
[ ] Porta 127 è CHIUSA su 192.168.1.3
[ ] Porta 128 è CHIUSA su 192.168.1.3
[ ] Porta 129 è CHIUSA su 192.168.1.3
[ ] Porta 130 è CHIUSA su 192.168.1.3
[ ] Porta 131 è CHIUSA su 192.168.1.3
[ ] Porta 132 è CHIUSA su 192.168.1.3
[ ] Porta 133 è CHIUSA su 192.168.1.3
[ ] Porta 134 è CHIUSA su 192.168.1.3
[ ] Porta 135 è CHIUSA su 192.168.1.3
[ ] Porta 136 è CHIUSA su 192.168.1.3
[ ] Porta 137 è CHIUSA su 192.168.1.3
[ ] Porta 138 è CHIUSA su 192.168.1.3
[+] Porta 139 è APERTA su 192.168.1.3
[ ] Porta 140 è CHIUSA su 192.168.1.3
[ ] Porta 141 è CHIUSA su 192.168.1.3
[ ] Porta 142 è CHIUSA su 192.168.1.3
[ ] Porta 143 è CHIUSA su 192.168.1.3
[ ] Porta 144 è CHIUSA su 192.168.1.3
[ ] Porta 145 è CHIUSA su 192.168.1.3
[ ] Porta 146 è CHIUSA su 192.168.1.3
[ ] Porta 147 è CHIUSA su 192.168.1.3
[ ] Porta 148 è CHIUSA su 192.168.1.3
[ ] Porta 149 è CHIUSA su 192.168.1.3
[ ] Porta 150 è CHIUSA su 192.168.1.3
```

Report Progetto Build Week

```
File Actions Edit View Help
[-] Porta 417 è CHIUSA su 192.168.1.3
[-] Porta 418 è CHIUSA su 192.168.1.3
[-] Porta 419 è CHIUSA su 192.168.1.3
[-] Porta 420 è CHIUSA su 192.168.1.3
[-] Porta 421 è CHIUSA su 192.168.1.3
[-] Porta 422 è CHIUSA su 192.168.1.3
[-] Porta 423 è CHIUSA su 192.168.1.3
[-] Porta 424 è CHIUSA su 192.168.1.3
[-] Porta 425 è CHIUSA su 192.168.1.3
[-] Porta 426 è CHIUSA su 192.168.1.3
[-] Porta 427 è CHIUSA su 192.168.1.3
[-] Porta 428 è CHIUSA su 192.168.1.3
[-] Porta 429 è CHIUSA su 192.168.1.3
[-] Porta 430 è CHIUSA su 192.168.1.3
[-] Porta 431 è CHIUSA su 192.168.1.3
[-] Porta 432 è CHIUSA su 192.168.1.3
[-] Porta 433 è CHIUSA su 192.168.1.3
[-] Porta 434 è CHIUSA su 192.168.1.3
[-] Porta 435 è CHIUSA su 192.168.1.3
[-] Porta 436 è CHIUSA su 192.168.1.3
[-] Porta 437 è CHIUSA su 192.168.1.3
[-] Porta 438 è CHIUSA su 192.168.1.3
[-] Porta 439 è CHIUSA su 192.168.1.3
[-] Porta 440 è CHIUSA su 192.168.1.3
[-] Porta 441 è CHIUSA su 192.168.1.3
[-] Porta 442 è CHIUSA su 192.168.1.3
[-] Porta 443 è CHIUSA su 192.168.1.3
[-] Porta 444 è CHIUSA su 192.168.1.3
[+] Porta 445 è APERTA su 192.168.1.3
[-] Porta 446 è CHIUSA su 192.168.1.3
[-] Porta 447 è CHIUSA su 192.168.1.3
[-] Porta 448 è CHIUSA su 192.168.1.3
[-] Porta 449 è CHIUSA su 192.168.1.3
[-] Porta 450 è CHIUSA su 192.168.1.3
[-] Porta 451 è CHIUSA su 192.168.1.3
[-] Porta 452 è CHIUSA su 192.168.1.3
[-] Porta 453 è CHIUSA su 192.168.1.3
[-] Porta 454 è CHIUSA su 192.168.1.3
[-] Porta 455 è CHIUSA su 192.168.1.3
[-] Porta 456 è CHIUSA su 192.168.1.3
[-] Porta 457 è CHIUSA su 192.168.1.3
[-] Porta 458 è CHIUSA su 192.168.1.3
[-] Porta 459 è CHIUSA su 192.168.1.3
[-] Porta 460 è CHIUSA su 192.168.1.3
[-] Porta 461 è CHIUSA su 192.168.1.3
[-] Porta 462 è CHIUSA su 192.168.1.3
[-] Porta 463 è CHIUSA su 192.168.1.3
[-] Porta 464 è CHIUSA su 192.168.1.3
[-] Porta 465 è CHIUSA su 192.168.1.3
[-] Porta 466 è CHIUSA su 192.168.1.3
[-] Porta 467 è CHIUSA su 192.168.1.3
[-] Porta 468 è CHIUSA su 192.168.1.3
```

Report Progetto Build Week

```
[+] Porta 483 è CHIUSA su 192.168.1.3
[+] Porta 484 è CHIUSA su 192.168.1.3
[+] Porta 485 è CHIUSA su 192.168.1.3
[+] Porta 486 è CHIUSA su 192.168.1.3
[+] Porta 487 è CHIUSA su 192.168.1.3
[+] Porta 488 è CHIUSA su 192.168.1.3
[+] Porta 489 è CHIUSA su 192.168.1.3
[+] Porta 490 è CHIUSA su 192.168.1.3
[+] Porta 491 è CHIUSA su 192.168.1.3
[+] Porta 492 è CHIUSA su 192.168.1.3
[+] Porta 493 è CHIUSA su 192.168.1.3
[+] Porta 494 è CHIUSA su 192.168.1.3
[+] Porta 495 è CHIUSA su 192.168.1.3
[+] Porta 496 è CHIUSA su 192.168.1.3
[+] Porta 497 è CHIUSA su 192.168.1.3
[+] Porta 498 è CHIUSA su 192.168.1.3
[+] Porta 499 è CHIUSA su 192.168.1.3
[+] Porta 500 è CHIUSA su 192.168.1.3
[+] Porta 501 è CHIUSA su 192.168.1.3
[+] Porta 502 è CHIUSA su 192.168.1.3
[+] Porta 503 è CHIUSA su 192.168.1.3
[+] Porta 504 è CHIUSA su 192.168.1.3
[+] Porta 505 è CHIUSA su 192.168.1.3
[+] Porta 506 è CHIUSA su 192.168.1.3
[+] Porta 507 è CHIUSA su 192.168.1.3
[+] Porta 508 è CHIUSA su 192.168.1.3
[+] Porta 509 è CHIUSA su 192.168.1.3
[+] Porta 510 è CHIUSA su 192.168.1.3
[+] Porta 511 è CHIUSA su 192.168.1.3
[+] Porta 512 è APERTA su 192.168.1.3
[+] Porta 513 è APERTA su 192.168.1.3
[+] Porta 514 è APERTA su 192.168.1.3
[+] Porta 515 è CHIUSA su 192.168.1.3
[+] Porta 516 è CHIUSA su 192.168.1.3
[+] Porta 517 è CHIUSA su 192.168.1.3
[+] Porta 518 è CHIUSA su 192.168.1.3
[+] Porta 519 è CHIUSA su 192.168.1.3
[+] Porta 520 è CHIUSA su 192.168.1.3
[+] Porta 521 è CHIUSA su 192.168.1.3
[+] Porta 522 è CHIUSA su 192.168.1.3
[+] Porta 523 è CHIUSA su 192.168.1.3
[+] Porta 524 è CHIUSA su 192.168.1.3
[+] Porta 525 è CHIUSA su 192.168.1.3
[+] Porta 526 è CHIUSA su 192.168.1.3
[+] Porta 527 è CHIUSA su 192.168.1.3
[+] Porta 528 è CHIUSA su 192.168.1.3
[+] Porta 529 è CHIUSA su 192.168.1.3
[+] Porta 530 è CHIUSA su 192.168.1.3
[+] Porta 531 è CHIUSA su 192.168.1.3
[+] Porta 532 è CHIUSA su 192.168.1.3
[+] Porta 533 è CHIUSA su 192.168.1.3
[+] Porta 534 è CHIUSA su 192.168.1.3
```

Report Progetto Build Week

Gli esiti dei test sulla scansione delle porte di Metasploitable rilevano che vi sono un numero consistente di porte aperte.

Report Progetto Build Week

A seguire gli esiti dei test sulla v.m. di Kali-Linux

```
└# python porte.py
[*] Avvio scansione delle porte su 192.168.5.5 ...
[!] Porta 1 è CHIUSA su 192.168.5.5
[!] Porta 2 è CHIUSA su 192.168.5.5
[!] Porta 3 è CHIUSA su 192.168.5.5
[!] Porta 4 è CHIUSA su 192.168.5.5
[!] Porta 5 è CHIUSA su 192.168.5.5
[!] Porta 6 è CHIUSA su 192.168.5.5
[!] Porta 7 è CHIUSA su 192.168.5.5
[!] Porta 8 è CHIUSA su 192.168.5.5
[!] Porta 9 è CHIUSA su 192.168.5.5
[!] Porta 10 è CHIUSA su 192.168.5.5
[!] Porta 11 è CHIUSA su 192.168.5.5
[!] Porta 12 è CHIUSA su 192.168.5.5
[!] Porta 13 è CHIUSA su 192.168.5.5
[!] Porta 14 è CHIUSA su 192.168.5.5
[!] Porta 15 è CHIUSA su 192.168.5.5
[!] Porta 16 è CHIUSA su 192.168.5.5
[!] Porta 17 è CHIUSA su 192.168.5.5
[!] Porta 18 è CHIUSA su 192.168.5.5
[!] Porta 19 è CHIUSA su 192.168.5.5
[!] Porta 20 è CHIUSA su 192.168.5.5
[!] Porta 21 è CHIUSA su 192.168.5.5
[!] Porta 22 è CHIUSA su 192.168.5.5
[!] Porta 23 è CHIUSA su 192.168.5.5
[!] Porta 24 è CHIUSA su 192.168.5.5
[!] Porta 25 è CHIUSA su 192.168.5.5
[!] Porta 26 è CHIUSA su 192.168.5.5
[!] Porta 27 è CHIUSA su 192.168.5.5
[!] Porta 28 è CHIUSA su 192.168.5.5
[!] Porta 29 è CHIUSA su 192.168.5.5
[!] Porta 30 è CHIUSA su 192.168.5.5
[!] Porta 31 è CHIUSA su 192.168.5.5
[!] Porta 32 è CHIUSA su 192.168.5.5
[!] Porta 33 è CHIUSA su 192.168.5.5
[!] Porta 34 è CHIUSA su 192.168.5.5
[!] Porta 35 è CHIUSA su 192.168.5.5
[!] Porta 36 è CHIUSA su 192.168.5.5
[!] Porta 37 è CHIUSA su 192.168.5.5
[!] Porta 38 è CHIUSA su 192.168.5.5
[!] Porta 39 è CHIUSA su 192.168.5.5
[!] Porta 40 è CHIUSA su 192.168.5.5
[!] Porta 41 è CHIUSA su 192.168.5.5
[!] Porta 42 è CHIUSA su 192.168.5.5
[!] Porta 43 è CHIUSA su 192.168.5.5
[!] Porta 44 è CHIUSA su 192.168.5.5
[!] Porta 45 è CHIUSA su 192.168.5.5
[!] Porta 46 è CHIUSA su 192.168.5.5
[!] Porta 47 è CHIUSA su 192.168.5.5
[!] Porta 48 è CHIUSA su 192.168.5.5
[!] Porta 49 è CHIUSA su 192.168.5.5
[!] Porta 50 è CHIUSA su 192.168.5.5
[!] Porta 51 è CHIUSA su 192.168.5.5
```

Report Progetto Build Week

```
[+] Porta 53 è CHIUSA su 192.168.5.5
[+] Porta 54 è CHIUSA su 192.168.5.5
[+] Porta 55 è CHIUSA su 192.168.5.5
[+] Porta 56 è CHIUSA su 192.168.5.5
[+] Porta 57 è CHIUSA su 192.168.5.5
[+] Porta 58 è CHIUSA su 192.168.5.5
[+] Porta 59 è CHIUSA su 192.168.5.5
[+] Porta 60 è CHIUSA su 192.168.5.5
[+] Porta 61 è CHIUSA su 192.168.5.5
[+] Porta 62 è CHIUSA su 192.168.5.5
[+] Porta 63 è CHIUSA su 192.168.5.5
[+] Porta 64 è CHIUSA su 192.168.5.5
[+] Porta 65 è CHIUSA su 192.168.5.5
[+] Porta 66 è CHIUSA su 192.168.5.5
[+] Porta 67 è CHIUSA su 192.168.5.5
[+] Porta 68 è CHIUSA su 192.168.5.5
[+] Porta 69 è CHIUSA su 192.168.5.5
[+] Porta 70 è CHIUSA su 192.168.5.5
[+] Porta 71 è CHIUSA su 192.168.5.5
[+] Porta 72 è CHIUSA su 192.168.5.5
[+] Porta 73 è CHIUSA su 192.168.5.5
[+] Porta 74 è CHIUSA su 192.168.5.5
[+] Porta 75 è CHIUSA su 192.168.5.5
[+] Porta 76 è CHIUSA su 192.168.5.5
[+] Porta 77 è CHIUSA su 192.168.5.5
[+] Porta 78 è CHIUSA su 192.168.5.5
[+] Porta 79 è CHIUSA su 192.168.5.5
[+] Porta 80 è CHIUSA su 192.168.5.5
[+] Porta 81 è CHIUSA su 192.168.5.5
[+] Porta 82 è CHIUSA su 192.168.5.5
[+] Porta 83 è CHIUSA su 192.168.5.5
[+] Porta 84 è CHIUSA su 192.168.5.5
[+] Porta 85 è CHIUSA su 192.168.5.5
[+] Porta 86 è CHIUSA su 192.168.5.5
[+] Porta 87 è CHIUSA su 192.168.5.5
[+] Porta 88 è CHIUSA su 192.168.5.5
[+] Porta 89 è CHIUSA su 192.168.5.5
[+] Porta 90 è CHIUSA su 192.168.5.5
[+] Porta 91 è CHIUSA su 192.168.5.5
[+] Porta 92 è CHIUSA su 192.168.5.5
[+] Porta 93 è CHIUSA su 192.168.5.5
[+] Porta 94 è CHIUSA su 192.168.5.5
[+] Porta 95 è CHIUSA su 192.168.5.5
[+] Porta 96 è CHIUSA su 192.168.5.5
[+] Porta 97 è CHIUSA su 192.168.5.5
[+] Porta 98 è CHIUSA su 192.168.5.5
[+] Porta 99 è CHIUSA su 192.168.5.5
[+] Porta 100 è CHIUSA su 192.168.5.5
[+] Porta 101 è CHIUSA su 192.168.5.5
[+] Porta 102 è CHIUSA su 192.168.5.5
[+] Porta 103 è CHIUSA su 192.168.5.5
[+] Porta 104 è CHIUSA su 192.168.5.5
[+] Porta 105 è CHIUSA su 192.168.5.5
```

Report Progetto Build Week

```
[+] Porta 260 è CHIUSA su 192.168.5.5
[+] Porta 261 è CHIUSA su 192.168.5.5
[+] Porta 262 è CHIUSA su 192.168.5.5
[+] Porta 263 è CHIUSA su 192.168.5.5
[+] Porta 264 è CHIUSA su 192.168.5.5
[+] Porta 265 è CHIUSA su 192.168.5.5
[+] Porta 266 è CHIUSA su 192.168.5.5
[+] Porta 267 è CHIUSA su 192.168.5.5
[+] Porta 268 è CHIUSA su 192.168.5.5
[+] Porta 269 è CHIUSA su 192.168.5.5
[+] Porta 270 è CHIUSA su 192.168.5.5
[+] Porta 271 è CHIUSA su 192.168.5.5
[+] Porta 272 è CHIUSA su 192.168.5.5
[+] Porta 273 è CHIUSA su 192.168.5.5
[+] Porta 274 è CHIUSA su 192.168.5.5
[+] Porta 275 è CHIUSA su 192.168.5.5
[+] Porta 276 è CHIUSA su 192.168.5.5
[+] Porta 277 è CHIUSA su 192.168.5.5
[+] Porta 278 è CHIUSA su 192.168.5.5
[+] Porta 279 è CHIUSA su 192.168.5.5
[+] Porta 280 è CHIUSA su 192.168.5.5
[+] Porta 281 è CHIUSA su 192.168.5.5
[+] Porta 282 è CHIUSA su 192.168.5.5
[+] Porta 283 è CHIUSA su 192.168.5.5
[+] Porta 284 è CHIUSA su 192.168.5.5
[+] Porta 285 è CHIUSA su 192.168.5.5
[+] Porta 286 è CHIUSA su 192.168.5.5
[+] Porta 287 è CHIUSA su 192.168.5.5
[+] Porta 288 è CHIUSA su 192.168.5.5
[+] Porta 289 è CHIUSA su 192.168.5.5
[+] Porta 290 è CHIUSA su 192.168.5.5
[+] Porta 291 è CHIUSA su 192.168.5.5
[+] Porta 292 è CHIUSA su 192.168.5.5
[+] Porta 293 è CHIUSA su 192.168.5.5
[+] Porta 294 è CHIUSA su 192.168.5.5
[+] Porta 295 è CHIUSA su 192.168.5.5
[+] Porta 296 è CHIUSA su 192.168.5.5
[+] Porta 297 è CHIUSA su 192.168.5.5
[+] Porta 298 è CHIUSA su 192.168.5.5
[+] Porta 299 è CHIUSA su 192.168.5.5
[+] Porta 300 è CHIUSA su 192.168.5.5
[+] Porta 301 è CHIUSA su 192.168.5.5
[+] Porta 302 è CHIUSA su 192.168.5.5
[+] Porta 303 è CHIUSA su 192.168.5.5
[+] Porta 304 è CHIUSA su 192.168.5.5
[+] Porta 305 è CHIUSA su 192.168.5.5
[+] Porta 306 è CHIUSA su 192.168.5.5
[+] Porta 307 è CHIUSA su 192.168.5.5
[+] Porta 308 è CHIUSA su 192.168.5.5
[+] Porta 309 è CHIUSA su 192.168.5.5
[+] Porta 310 è CHIUSA su 192.168.5.5
[+] Porta 311 è CHIUSA su 192.168.5.5
[+] Porta 312 è CHIUSA su 192.168.5.5
```

Report Progetto Build Week

```
[+] Porta 554 è CHIUSA su 192.168.5.5
[+] Porta 555 è CHIUSA su 192.168.5.5
[+] Porta 556 è CHIUSA su 192.168.5.5
[+] Porta 557 è CHIUSA su 192.168.5.5
[+] Porta 558 è CHIUSA su 192.168.5.5
[+] Porta 559 è CHIUSA su 192.168.5.5
[+] Porta 560 è CHIUSA su 192.168.5.5
[+] Porta 561 è CHIUSA su 192.168.5.5
[+] Porta 562 è CHIUSA su 192.168.5.5
[+] Porta 563 è CHIUSA su 192.168.5.5
[+] Porta 564 è CHIUSA su 192.168.5.5
[+] Porta 565 è CHIUSA su 192.168.5.5
[+] Porta 566 è CHIUSA su 192.168.5.5
[+] Porta 567 è CHIUSA su 192.168.5.5
[+] Porta 568 è CHIUSA su 192.168.5.5
[+] Porta 569 è CHIUSA su 192.168.5.5
[+] Porta 570 è CHIUSA su 192.168.5.5
[+] Porta 571 è CHIUSA su 192.168.5.5
[+] Porta 572 è CHIUSA su 192.168.5.5
[+] Porta 573 è CHIUSA su 192.168.5.5
[+] Porta 574 è CHIUSA su 192.168.5.5
[+] Porta 575 è CHIUSA su 192.168.5.5
[+] Porta 576 è CHIUSA su 192.168.5.5
[+] Porta 577 è CHIUSA su 192.168.5.5
[+] Porta 578 è CHIUSA su 192.168.5.5
[+] Porta 579 è CHIUSA su 192.168.5.5
[+] Porta 580 è CHIUSA su 192.168.5.5
[+] Porta 581 è CHIUSA su 192.168.5.5
[+] Porta 582 è CHIUSA su 192.168.5.5
[+] Porta 583 è CHIUSA su 192.168.5.5
[+] Porta 584 è CHIUSA su 192.168.5.5
[+] Porta 585 è CHIUSA su 192.168.5.5
[+] Porta 586 è CHIUSA su 192.168.5.5
[+] Porta 587 è CHIUSA su 192.168.5.5
[+] Porta 588 è CHIUSA su 192.168.5.5
[+] Porta 589 è CHIUSA su 192.168.5.5
[+] Porta 590 è CHIUSA su 192.168.5.5
[+] Porta 591 è CHIUSA su 192.168.5.5
[+] Porta 592 è CHIUSA su 192.168.5.5
[+] Porta 593 è CHIUSA su 192.168.5.5
[+] Porta 594 è CHIUSA su 192.168.5.5
[+] Porta 595 è CHIUSA su 192.168.5.5
[+] Porta 596 è CHIUSA su 192.168.5.5
[+] Porta 597 è CHIUSA su 192.168.5.5
[+] Porta 598 è CHIUSA su 192.168.5.5
[+] Porta 599 è CHIUSA su 192.168.5.5
[+] Porta 600 è CHIUSA su 192.168.5.5
[+] Porta 601 è CHIUSA su 192.168.5.5
[+] Porta 602 è CHIUSA su 192.168.5.5
[+] Porta 603 è CHIUSA su 192.168.5.5
[+] Porta 604 è CHIUSA su 192.168.5.5
[+] Porta 605 è CHIUSA su 192.168.5.5
[+] Porta 606 è CHIUSA su 192.168.5.5
```

Report Progetto Build Week

```
[+] Porta 977 è CHIUSA su 192.168.5.5
[+] Porta 978 è CHIUSA su 192.168.5.5
[+] Porta 979 è CHIUSA su 192.168.5.5
[+] Porta 980 è CHIUSA su 192.168.5.5
[+] Porta 981 è CHIUSA su 192.168.5.5
[+] Porta 982 è CHIUSA su 192.168.5.5
[+] Porta 983 è CHIUSA su 192.168.5.5
[+] Porta 984 è CHIUSA su 192.168.5.5
[+] Porta 985 è CHIUSA su 192.168.5.5
[+] Porta 986 è CHIUSA su 192.168.5.5
[+] Porta 987 è CHIUSA su 192.168.5.5
[+] Porta 988 è CHIUSA su 192.168.5.5
[+] Porta 989 è CHIUSA su 192.168.5.5
[+] Porta 990 è CHIUSA su 192.168.5.5
[+] Porta 991 è CHIUSA su 192.168.5.5
[+] Porta 992 è CHIUSA su 192.168.5.5
[+] Porta 993 è CHIUSA su 192.168.5.5
[+] Porta 994 è CHIUSA su 192.168.5.5
[+] Porta 995 è CHIUSA su 192.168.5.5
[+] Porta 996 è CHIUSA su 192.168.5.5
[+] Porta 997 è CHIUSA su 192.168.5.5
[+] Porta 998 è CHIUSA su 192.168.5.5
[+] Porta 999 è CHIUSA su 192.168.5.5
[+] Porta 1000 è CHIUSA su 192.168.5.5
[+] Porta 1001 è CHIUSA su 192.168.5.5
[+] Porta 1002 è CHIUSA su 192.168.5.5
[+] Porta 1003 è CHIUSA su 192.168.5.5
[+] Porta 1004 è CHIUSA su 192.168.5.5
[+] Porta 1005 è CHIUSA su 192.168.5.5
[+] Porta 1006 è CHIUSA su 192.168.5.5
[+] Porta 1007 è CHIUSA su 192.168.5.5
[+] Porta 1008 è CHIUSA su 192.168.5.5
[+] Porta 1009 è CHIUSA su 192.168.5.5
[+] Porta 1010 è CHIUSA su 192.168.5.5
[+] Porta 1011 è CHIUSA su 192.168.5.5
[+] Porta 1012 è CHIUSA su 192.168.5.5
[+] Porta 1013 è CHIUSA su 192.168.5.5
[+] Porta 1014 è CHIUSA su 192.168.5.5
[+] Porta 1015 è CHIUSA su 192.168.5.5
[+] Porta 1016 è CHIUSA su 192.168.5.5
[+] Porta 1017 è CHIUSA su 192.168.5.5
[+] Porta 1018 è CHIUSA su 192.168.5.5
[+] Porta 1019 è CHIUSA su 192.168.5.5
[+] Porta 1020 è CHIUSA su 192.168.5.5
[+] Porta 1021 è CHIUSA su 192.168.5.5
[+] Porta 1022 è CHIUSA su 192.168.5.5
[+] Porta 1023 è CHIUSA su 192.168.5.5
[+] Porta 1024 è CHIUSA su 192.168.5.5
[*] Scansione completata in 0.03 secondi.
```

```
└─(root㉿kali)-[/home/kali]
# █
```

Report Progetto Build Week

Come riscontrato negli esiti dei test sulla macchina Kali-Linux, la totalità delle porte esaminate sono chiuse.

Fase 3 sub 3: raccomandazioni

In chiusura della presente fase osserviamo le raccomandazioni a seguire.

Sarebbe opportuno procedere con la chiusura di tutte le porte non necessarie/non utilizzate allo scopo di garantire una migliore e più efficace messa in sicurezza del sistema di rete. Ciò in quanto ogni porta aperta rappresenta sempre una vulnerabilità ed una fonte di accesso per agenti malevoli. Ne consegue che, per “vincere” un trade off in ordine all’allocazione dei fisiologici rischi informatici, ma manovra più accorta rimane sempre quella di eliminare tutto ciò che è superfluo.

Sempre in ordine alla manutenzione delle porte, si raccomanda inoltre di procedere periodicamente all’esecuzione di test con tool o programmi appositamente progettati per la scansione delle porte (come ad esempio il programma da noi proposto o lo strumento nmap di Metasploitable).

Ulteriore raccomandazione spendibile, può essere quella di imporre una serie di regole di blocco sul firewall, allo scopo di impedire il traffico dati verso direzioni che la dirigenza intende proibire, come specifici siti web.

Ovviamente il firewall, come anche gli altri dispositivi, come IPS dovranno essere doverosamente manutenuti e tenuti sempre aggiornati.

Report Progetto Build Week

Fase Bonus

Sub 1: Esercizio Bonus Rete

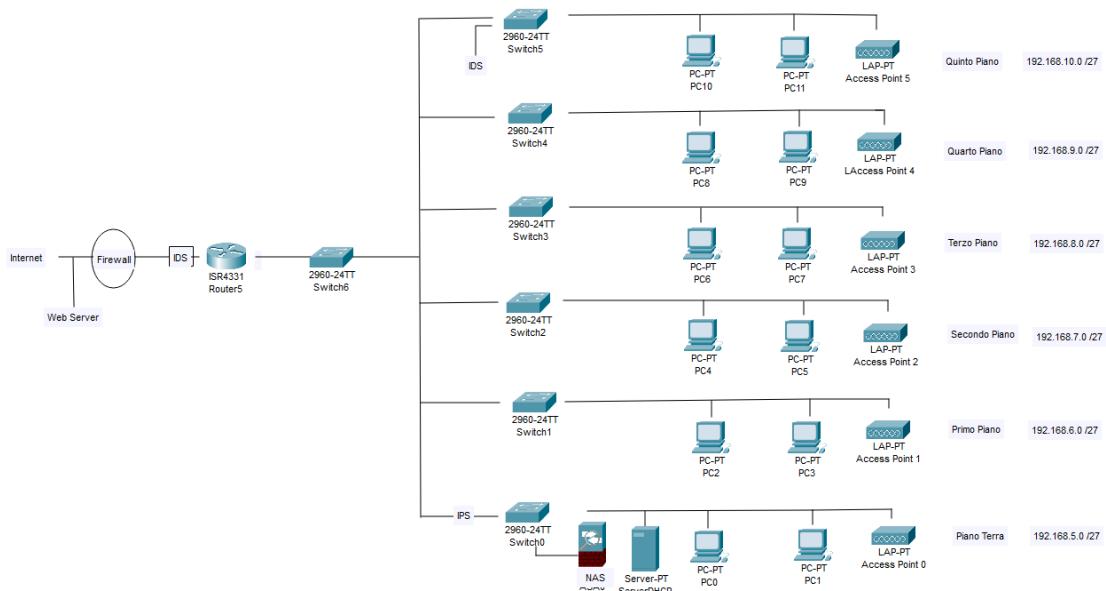
Per costituire le subnet come da consegna, si renderà necessario alterare lo schema generale del progetto di rete da noi proposto.

In primo luogo inserendo uno switch in più dopo il router, che ne diventerà ausiliario allo scopo di ridimensionare l'altrimenti troppo gravoso carico di lavoro.

Gli switch preesistenti, quindi, verranno collegati al router attraverso la mediazione del nuovo switch, che chiameremo “centralizzatore”.

In questo modo tutti i dispositivi collegati alla rete della committente avranno dunque il medesimo indirizzo di rete (nell'esempio a seguire si imporrà un IP 192.168.5.0 /24).

Per una razionalizzazione ai fini esemplificativi, ma anche per garantire una maggiore efficienza nella gestione e sicurezza si può procedere ad effettuare le subnet per ogni piano.



Report Progetto Build Week

Nome Subnet	Range ip indirizzi	Piano	Host minimi necessari
Subnet 1	192.168.5.0 - 192.168.5.31 /27	Piano Terra	22
Subnet 2	192.168.6.0 - 192.168.6.31 /27	Primo Piano	21
Subnet 3	192.168.7.0 - 192.168.7.31 /27	Secondo Piano	21
Subnet 4	192.168.8.0 - 192.168.8.31 /27	Terzo Piano	21
Subnet 5	192.168.9.0 - 192.168.9.31 /27	Quarto Piano	21
Subnet 6	192.168.10.0 - 192.168.10.31 /27	Quinto Piano	21

Come si evince dalla tabella che immediatamente precede, si è optato la costituzione di una subnet con /27. Questa decisione non è arbitraria, ma frutto dei calcoli e valutazioni che si vanno di seguito a esplicare.

In via preliminare occorre svolgere una valutazione sul piano procedurale:

- basandosi su quanti host deve avere ogni subnet, quindi verificare se la capienza materiale di ogni subnet allo scopo di comprendere quante subnet dovranno essere realizzate
- nel caso che ci occupa 21/22 host minimi necessari, e 6 subnet.

Ora dobbiamo procedere con il calcolo potenziale:

- prendiamo infatti le potenze del 2
- poiché 2^5 (due alla quinta) è pari a 32, ne consegue che è il caso perfetto per gli host di cui abbiamo necessità.

Report Progetto Build Week

A questi 32 bit dell'IP della nostra proposta di rete, sarà poi necessario detrarre un numero pari a quello degli IP dedicati alla rete stessa (in questo caso 27).

Il resto ($32 - 27$) sarà pari a 5, ossia i bit dedicati agli host (2^5).

A questo punto possiamo procedere a strutturare le diverse subnet nel modo a seguire:

1) 192.168.5.0 /27 → Rete

192.168.5.1 |

192.168.5.30 |--> Host

192.168.5.31 → Broadcast

2) 192.168.6.0 /27 → Rete

192.168.6.1 |

192.168.6.30 |--> Host

192.168.6.31 → Broadcast

3) 192.168.7.0 /27 → Rete

192.168.7.1 |

192.168.7.30 |--> Host

192.168.7.31 → Broadcast

4) 192.168.8.0 /27 → Rete

192.168.8.1 |

192.168.8.30 |--> Host

192.168.8.31 → Broadcast

5) 192.168.9.0 /27 → Rete

192.168.9.1 |

192.168.9.30 |--> Host

192.168.9.31 → Broadcast

6) 192.168.10.0 /27 → Rete

192.168.10.1 |

192.168.10.30 |--> Host

192.168.10.31 → Broadcast

Report Progetto Build Week

Fase bonus

Sub 2: programma python bonus di cattura socket

Per lo svolgimento di quest’ulteriore consegna si è proceduto alla realizzazione di un codice apposito che nella sostanza consiste in uno “sniffer”, in grado di intercettare tutto il traffico che attraversa il socket di rete, consentendo quindi di monitorare tutti i pacchetti in entrata sulla rete.

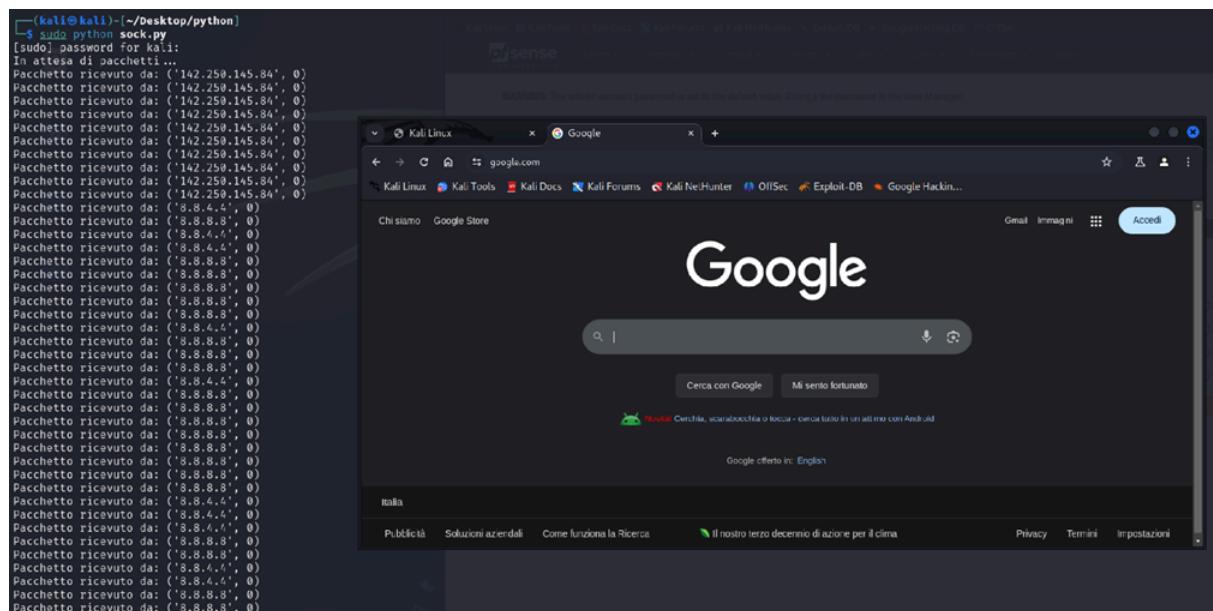
```
File Actions Edit View Help
GNU nano 8.1
import socket

def sniffer():
    # Crea un socket con lo scopo di catturare pacchetti
    s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
    s.bind(("0.0.0.0", 0)) # Si mette in ascolto su tutte le interfacce di rete
    s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1) #Includi l'header IP

    print("In attesa di pacchetti ... ")
    try:
        while True:
            pacchetto, indirizzo = s.recvfrom(65535) # Ricevi pacchetti di dimensione 65535max
            print(f"pacchetto ricevuto da: {indirizzo}")
            #print(f"Payload (esadecimale): {pacchetto.hex()[:50]} ... ") # Mostra i primi 50 caratteri del payload
    except KeyboardInterrupt:
        print("\nSniffer interrotto.")
    finally:
        s.close()

if __name__ == "__main__":
    sniffer()
python
```

A seguire l'esito del test in cui si è provato a fare una ricerca sulla barra di ricerca del browser di Kali-Linux.



Report Progetto Build Week

Fase bonus

Sub 3: Scrivere un programma in Python che utilizzi scapy per catturare e analizzare il traffico di rete.

“Il programma deve:

- 1) Catturare i pacchetti in tempo reale su un'interfaccia di rete specificata.
- 2) Filtrare i pacchetti in base al protocollo (ad esempio: TCP, UDP, ICMP).
- 3) Visualizzare i dettagli dei pacchetti, inclusi indirizzi IP sorgente/destinazione, porte e protocollo utilizzato.
- 4) Salvare i pacchetti catturati in un file .pcap per un'analisi successiva tramite Wireshark o strumenti simili.”

In primo luogo, per lo svolgimento della presente consegna sarà necessario procedere con la scrittura di un apposito programma in linguaggio Python.

Lo scopo del codice sarà quello di catturare i pacchetti in base ai protocolli TCP, UDP, ICMP in tempo reale e salvarli in un file “.pcap”.

L'output atteso dal presente codice sarà quello di restituire:

- gli indirizzi IP,
- la sorgente,
- la destinazione,
- le porte e
- il protocollo.

Il programma sarà composto da tre parti principali:

1. **Import delle librerie** necessarie per gestire la cattura e l'analisi del traffico di rete.
2. **Definizione delle funzioni** che gestiscono la cattura, l'analisi e il salvataggio dei pacchetti.

Report Progetto Build Week

3. Il vero e proprio blocco principale, dato dalla **raccolta dei parametri** dall'utente ed esecuzione delle funzioni principali.

Di qui a seguire si analizzeranno nello specifico le tre parti del programma

1. Import delle librerie

```
from scapy.all import sniff, wrpcap  
  
from scapy.layers.inet import IP, TCP, UDP, ICMP  
  
• scapy.all:  
  
    • sniff(): Permette di catturare pacchetti di rete in tempo reale.  
  
    • wrpcap(): Consente di salvare i pacchetti catturati in un file .pcap, compatibile con Wireshark.  
  
• scapy.layers.inet:  
  
    • Contiene i moduli per i protocolli IP, TCP, UDP e ICMP, che permettono di analizzare i pacchetti.
```

2. Funzione “analizza_pacchetto”

Questa funzione elabora ogni pacchetto catturato per estrarne e visualizzarne i dettagli.

Passaggi

1. **Controllo del livello IP:**
 - Ogni pacchetto catturato viene analizzato per verificare se contiene un livello IP.
 - Se sì, vengono estratti gli indirizzi IP sorgente e destinazione.
2. **Identificazione del protocollo di trasporto:**
 - Se il pacchetto contiene TCP o UDP, vengono estratte anche le porte sorgente e destinazione.
 - Per ICMP, poiché non usa porte, vengono mostrati simboli "-".
3. **Visualizzazione dei dettagli:**
 - I dati raccolti (IP, porta e protocollo) vengono stampati sulla console in formato leggibile.

Report Progetto Build Week

2.1. Funzione “cattura_pacchetti”

Questa funzione gestisce l’intero processo di cattura dei pacchetti.

Passaggi

1. Avvio della cattura:

- La funzione sniff() cattura i pacchetti sull’interfaccia specificata e applica un filtro (se fornito) per limitare i pacchetti catturati a un protocollo specifico (ad esempio, tcp o icmp).
- Ogni pacchetto catturato viene passato alla funzione analizza_pacchetto().

2. Memorizzazione dei pacchetti:

- I pacchetti catturati vengono memorizzati in una lista interna grazie all’opzione store=True.

3. Salvataggio su file:

- Al termine della cattura, i pacchetti memorizzati vengono salvati in un file .pcap tramite wrpcap().

4. Gestione degli errori:

- KeyboardInterrupt: Permette all’utente di interrompere il processo con Ctrl+C.
- **Altri errori:** Qualsiasi problema (ad esempio, un’interfaccia non valida) viene catturato e stampato.

3. Blocco principale

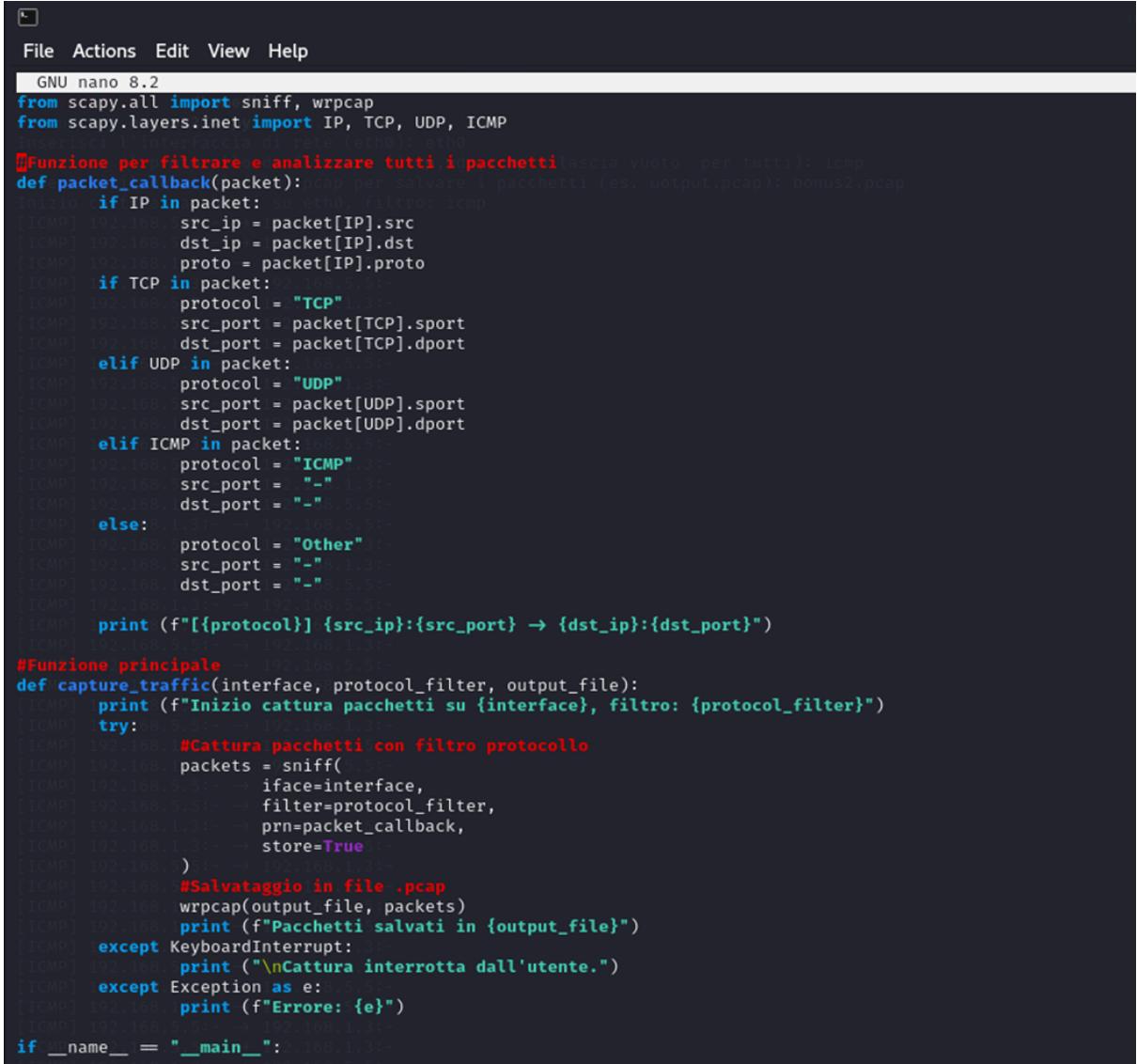
1. Raccolta dei parametri dall’utente:

- interfaccia: Nome dell’interfaccia di rete (esempio: eth0 o wlan0).
- filtro: Filtro opzionale per catturare pacchetti di un protocollo specifico (ad esempio, tcp, udp, icmp).
- file_output: Nome del file in cui salvare i pacchetti catturati.

2. Chiamata della funzione principale:

- i parametri raccolti vengono passati a cattura_pacchetti() per avviare la cattura.

Report Progetto Build Week



```
GNU nano 8.2
from scapy.all import sniff, wrpcap
from scapy.layers.inet import IP, TCP, UDP, ICMP
#inserisci l'interfaccia di rete (eth0): eth0
#Funzione per filtrare e analizzare tutti i pacchetti lascia vuoto per tutti): icmp
def packet_callback(packet):
    inizio = packet[IP].src
    ICMP] 192.168.5.5:->192.168.1.3: ICMP
    ICMP] 192.168.5.5:->192.168.1.3: src_ip = packet[IP].src
    ICMP] 192.168.5.5:->192.168.1.3: dst_ip = packet[IP].dst
    ICMP] 192.168.5.5:->192.168.1.3: proto = packet[IP].proto
    ICMP] 192.168.5.5:->192.168.1.3: if TCP in packet:
    ICMP] 192.168.5.5:->192.168.1.3:     protocol = "TCP"
    ICMP] 192.168.5.5:->192.168.1.3:     src_port = packet[TCP].sport
    ICMP] 192.168.5.5:->192.168.1.3:     dst_port = packet[TCP].dport
    ICMP] 192.168.5.5:->192.168.1.3: elif UDP in packet:
    ICMP] 192.168.5.5:->192.168.1.3:     protocol = "UDP"
    ICMP] 192.168.5.5:->192.168.1.3:     src_port = packet[UDP].sport
    ICMP] 192.168.5.5:->192.168.1.3:     dst_port = packet[UDP].dport
    ICMP] 192.168.5.5:->192.168.1.3: elif ICMP in packet:
    ICMP] 192.168.5.5:->192.168.1.3:     protocol = "ICMP"
    ICMP] 192.168.5.5:->192.168.1.3:     src_port = "-"
    ICMP] 192.168.5.5:->192.168.1.3:     dst_port = "-"
    ICMP] 192.168.5.5:->192.168.1.3: else:
    ICMP] 192.168.5.5:->192.168.1.3:     protocol = "Other"
    ICMP] 192.168.5.5:->192.168.1.3:     src_port = "-"
    ICMP] 192.168.5.5:->192.168.1.3:     dst_port = "-"
    ICMP] 192.168.5.5:->192.168.1.3: print(f"[{protocol}] {src_ip}:{src_port} → {dst_ip}:{dst_port}")
    ICMP] 192.168.5.5:->192.168.1.3: #Funzione principale
def capture_traffic(interface, protocol_filter, output_file):
    ICMP] 192.168.5.5:->192.168.1.3:     print(f"Inizio cattura pacchetti su {interface}, filtro: {protocol_filter}")
    ICMP] 192.168.5.5:->192.168.1.3:     try:
    ICMP] 192.168.5.5:->192.168.1.3:         #Cattura pacchetti con filtro protocollo
    ICMP] 192.168.5.5:->192.168.1.3:         packets = sniff(
    ICMP] 192.168.5.5:->192.168.1.3:             iface=interface,
    ICMP] 192.168.5.5:->192.168.1.3:             filter=protocol_filter,
    ICMP] 192.168.5.5:->192.168.1.3:             prn=packet_callback,
    ICMP] 192.168.5.5:->192.168.1.3:             store=True)
    ICMP] 192.168.5.5:->192.168.1.3:         #Salvataggio in file .pcap
    ICMP] 192.168.5.5:->192.168.1.3:         wrpcap(output_file, packets)
    ICMP] 192.168.5.5:->192.168.1.3:         print(f"pacchetti salvati in {output_file}")
    ICMP] 192.168.5.5:->192.168.1.3:     except KeyboardInterrupt:
    ICMP] 192.168.5.5:->192.168.1.3:         print("\nCattura interrotta dall'utente.")
    ICMP] 192.168.5.5:->192.168.1.3:     except Exception as e:
    ICMP] 192.168.5.5:->192.168.1.3:         print(f"Errore: {e}")
    ICMP] 192.168.5.5:->192.168.1.3: if __name__ == "__main__":
    ICMP] 192.168.5.5:->192.168.1.3:
```

A questo punto possiamo procedere con l' esecuzione del programma.

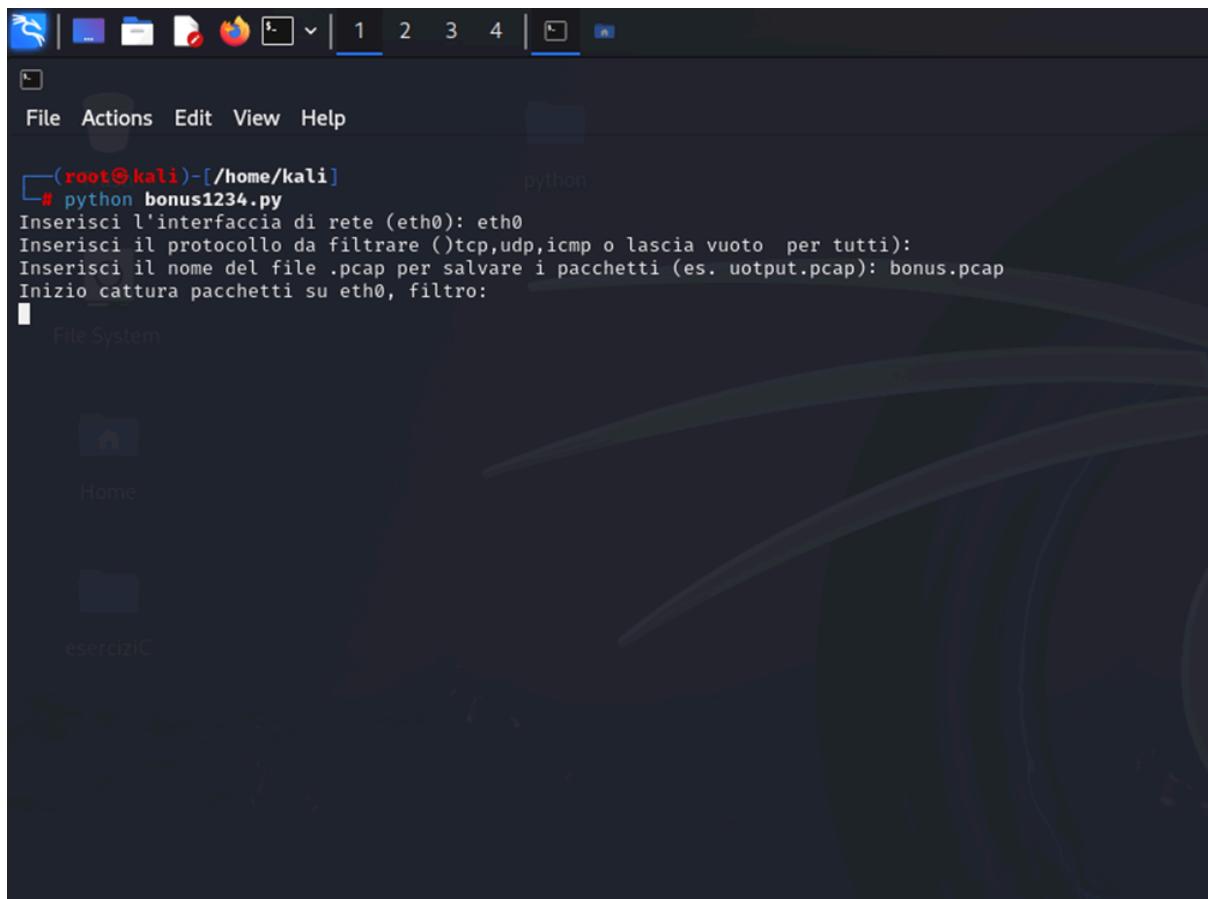
Si precisa che per poter lanciare questo programma sarà necessario utilizzare i privilegi di root (amministratore) in Kali-Linux.

Quindi procedere in via preliminare con il comando: “sudo su” ed inserire la password di default“kali”, qualora non si sia precedentemente personalizzata.

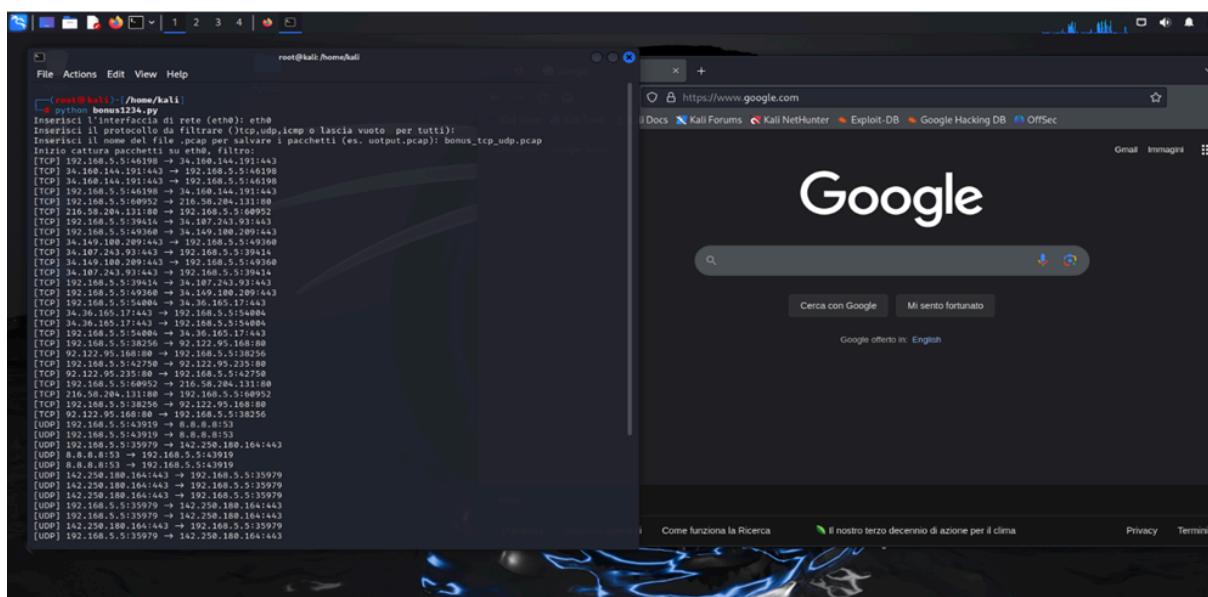
Andiamo quindi ad inserire:

- L'interfaccia di rete (eth0)
- Il protocollo da filtrare tra tcp, udp e icmp o lasciamo il campo vuoto per filtrarli tutti.
- Il nome del file .pcap dove salveremo i pacchetti.

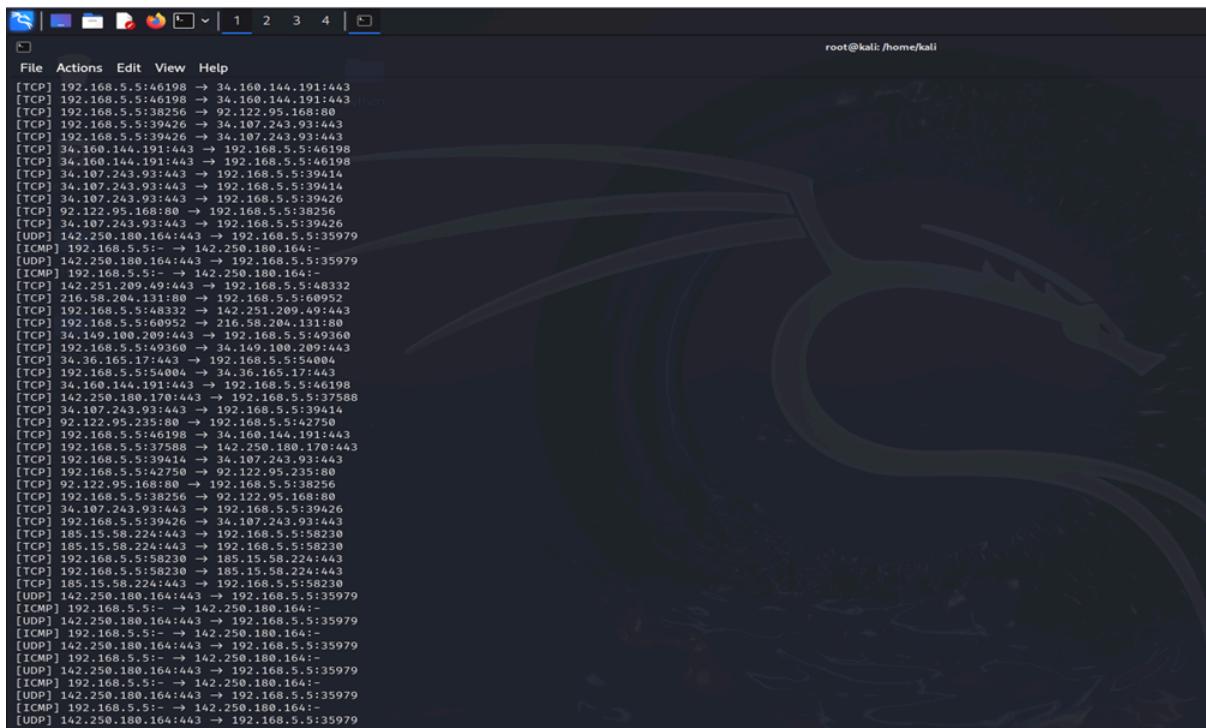
Report Progetto Build Week



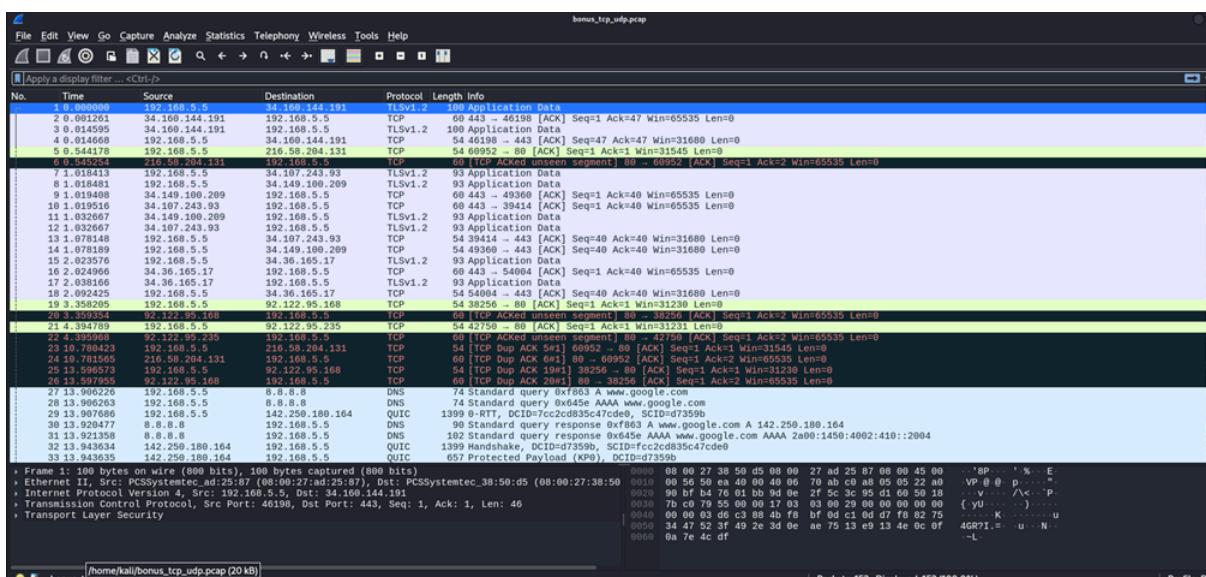
La cattura dei dati avrà inizio apprendendo una pagina web come possiamo vedere dalle immagini sottostanti.



Report Progetto Build Week



A questo punto andiamo a controllare i pacchetti catturati aprendo il file creato con il tool “Wireshark”.



Report Progetto Build Week

