



# Progetto di Rete Compagnia Theta

Build Week 1

BY TEAM AEGIS

# Indice

- [Chi siamo](#)
- [Obiettivo del progetto](#)
- [Disegno di rete](#)
- [Dispositivi di rete](#)
- [Suddivisione della rete](#)
- [Subnetting della rete](#)
  - [Scelta della subnet mask](#)
  - [Calcolo della subnet mask](#)
  - [Tabella indirizzi IP](#)
- [Scansione delle vulnerabilità](#)
  - [Scansione dei metodi HTTP](#)
  - [Scansione delle porte di rete](#)
- [Funzionamento programmi python](#)
  - [Scanner porte di rete](#)
  - [Scanner metodi HTTP](#)
  - [Scambio di dati tramite TCP](#)
- [Blocco del traffico con PFsense](#)
  - [Obiettivo del firewall](#)
  - [Configurazione VirtualBox](#)
  - [Configurazione PFsense](#)
  - [Configurazione regola del firewall](#)
- [Preventivo](#)

# Chi siamo

# Team Aegis

Il tuo scudo personale a difesa del tuo sistema informatico

## DISEGNO DI RETE E PROGRAMMI PYTHON

Luca Calvigioni  
Francesco Proto

## TEAM LEADER E SETUP FIREWALL

Federico Cuccu

## CONTABILITA'

Carmine Malangone

## SUBNETTING DI RETE

Silvia Arnetta  
Sara Amato  
Nichol Galessiere

# Siamo Aegis

Un gruppo di **esperti appassionati di cybersecurity**, uniti dalla missione di proteggere le aziende nel vasto oceano digitale.

Con anni di esperienza combinata, il nostro team è pronto a trasformare le sfide della sicurezza informatica in opportunità di crescita e innovazione per la tua azienda.



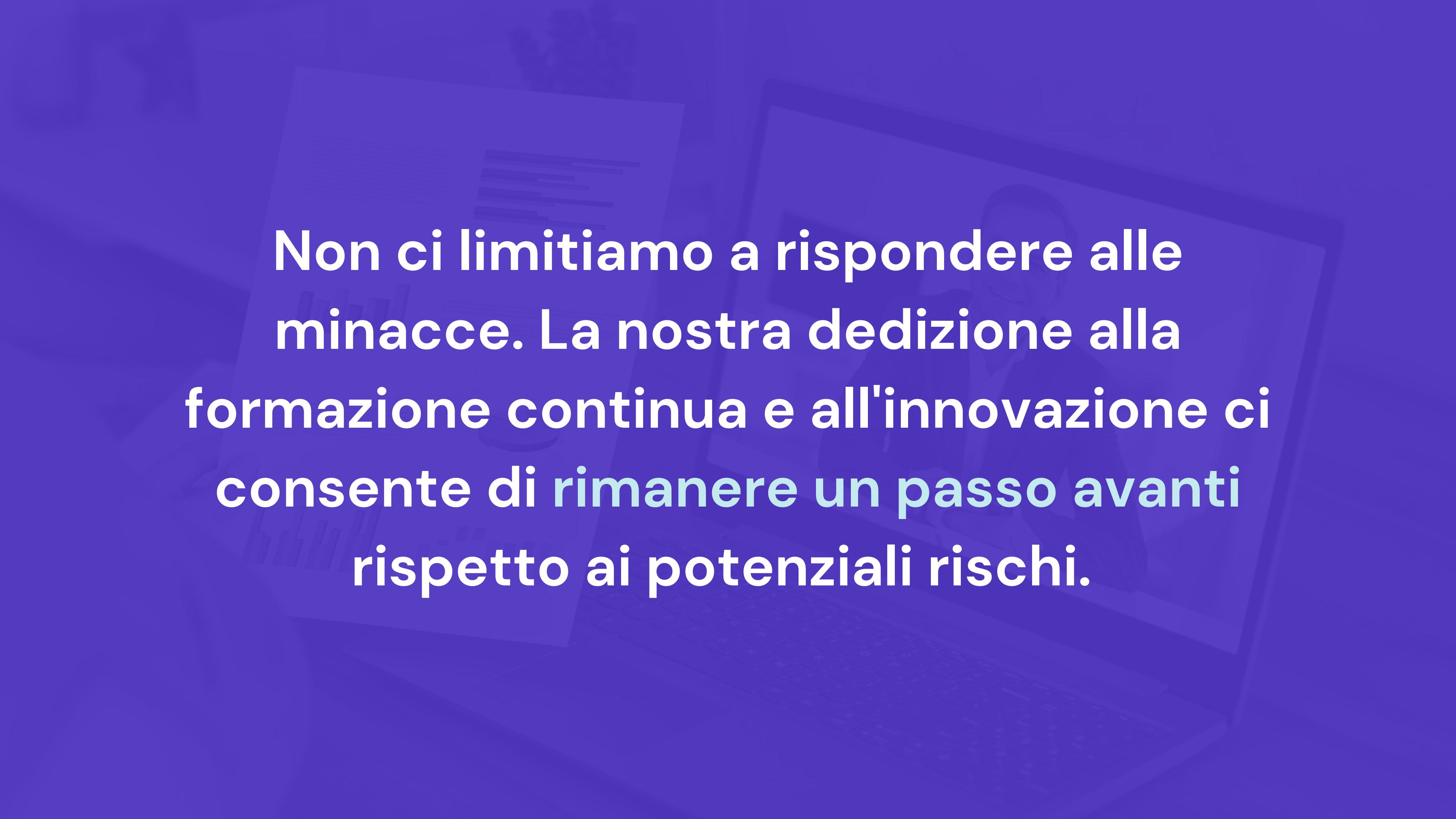
# Come nasce Aegis

Il nome Aegis affonda le sue radici nell'antica mitologia greca, dove rappresentava **lo scudo protettivo di Zeus e Atena**, simbolo di forza invincibile e difesa assoluta.

La nostra visione è un mondo in cui ogni azienda, grande o piccola, possa prosperare in un **ambiente sicuro**, libero da preoccupazioni.

Come lo scudo divino che proteggeva i suoi portatori dalle minacce più insidiose, ci poniamo come la barriera che **tutela ciò che è più prezioso**: i tuoi dati, le tue operazioni e la tua fiducia.

Difendiamo il tuo **futuro digitale** da ogni minaccia, con soluzioni su misura che riflettono il valore e la forza di chi ci sceglie.



Non ci limitiamo a rispondere alle minacce. La nostra dedizione alla formazione continua e all'innovazione ci consente di rimanere un passo avanti rispetto ai potenziali rischi.

# Obiettivo del progetto

La compagnia Theta ci ha ingaggiato per la realizzazione della loro infrastruttura IT.

L'edificio è suddiviso in **6 piani** e per ogni piano saranno presenti 20 computer, per un totale di **120 computer**.

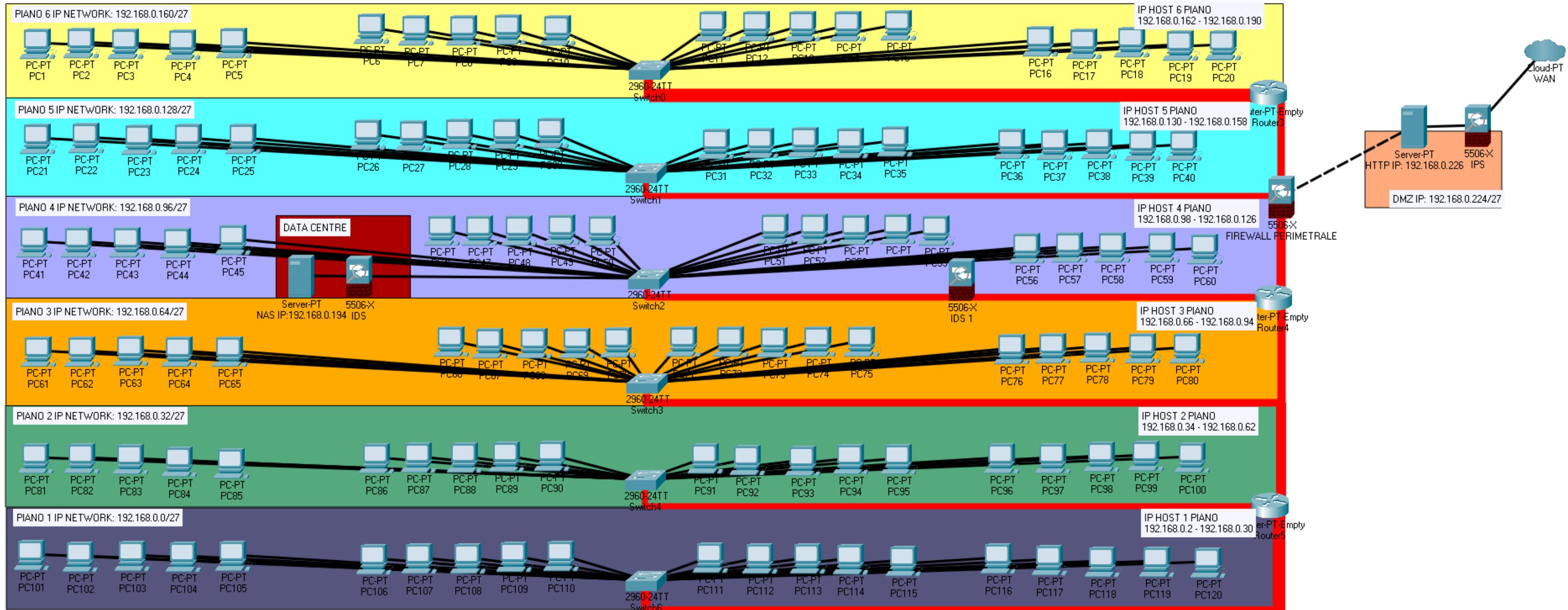
Inoltre la rete sarà strutturata da:

- 1 Web server
- 1 Firewall perimetrale
- 1 NAS (Network Attached Storage)
- 3 IDS/IPS (Intrusion Detection System / Intrusion Prevention System)

Legenda:

- canaline
- cavi di rete

# Disegno di rete



# Dispositivi di rete

# Dispositivi di rete

Ogni piano, nello specifico, prevede **20 postazioni** e ognuna di queste comprende:

- 1 computer desktop
- 1 mouse e 1 tastiera come dispositivi di input
- 1 cuffia con microfono come dispositivi di input/output
- 1 monitor come dispositivo di output

Allo stesso tempo, su ogni piano sarà presente **1 switch** che metterà in comunicazione tutti i computer con il router gateway e saranno forniti di connessione internet.

I **router gateway** saranno 3 per garantire la continuità del funzionamento della rete (in caso un router andasse in down) e metteranno in comunicazione tutti gli host della rete con internet, passando prima per gli switch, poi per il router e infine per il firewall.

# Dispositivi di rete

Il **NAS** conterrà tutti i dati aziendali e avrà una sottorete dedicata per aumentare il livello di sicurezza, mantenendo comunque l'accessibilità a tutti i dipendenti.

Il **web server** fornirà il servizio al quale accederanno le persone dall'esterno della rete, ad esempio i clienti dell'azienda.

Il **firewall perimetrale a filtraggio dinamico** respingerà automaticamente tutte le connessioni inizializzate dall'esterno, proteggendo la rete interna da connessioni non autorizzate provenienti da internet.

# Dispositivi di rete

Affinché sia accessibile pubblicamente, il web server sarà all'interno di una **zona demilitarizzata**, in modo da consentire tutte le connessioni in entrata e in uscita.

I 2 **IDS**, che proteggeranno sia il NAS e che la rete interna, lanceranno l'allarme nel caso in cui un pacchetto sospetto sia arrivato fino a questo punto della rete, bypassando il firewall perimetrale.

L'**IPS** che protegge il web server, invece, bloccherà anche il pacchetto malevolo, oltre che lanciare l'allarme.

# Subnetting della rete

# Subnetting della rete

Utilizziamo la tecnica del subnetting per suddividere la rete per i seguenti motivi:

## Ottimizzazione degli indirizzi IP:

Permette di suddividere una rete grande in sottoreti più piccole, evitando lo spreco di indirizzi IP non utilizzati. Ci permette di assegnare solo gli indirizzi di rete e di host necessari a ciascuna rete (1 sottorete = 1 piano).

## Riduzione del traffico di rete:

Il traffico di rete resta confinato in ogni sottorete. Migliora le prestazioni della rete ne aumenta la sicurezza, in quanto i pacchetti di ogni sottorete non passeranno per le altre sottoreti.

## Migliore gestione della rete:

Facilita l'organizzazione logica della rete, separando i vari dipartimenti o piani in sottoreti specifiche. Risulta più semplice gestire e monitorare più facilmente ogni parte della rete.

# Scelta della subnet mask

Per progettare una rete ben strutturata e con margine di espansione futura, abbiamo considerato sia gli host attualmente presenti in ogni piano sia la possibilità di aggiungere ulteriori dispositivi in seguito, come server, NAS e dispositivi DMZ.

Ogni piano dell'edificio prevede 20 host fissi, ma con la subnet scelta abbiamo un ulteriore margine di 9 host aggiuntivi per ogni piano per garantire la **scalabilità e la flessibilità della rete**. Questo ci porta a un totale di 29 host possibili per piano.

# Scelta della subnet mask

Dato il numero di host, abbiamo scelto di utilizzare una rete di classe C (192.168.0.0/24).

Di norma, In una subnet di classe C, i primi 24 bit sono riservati per l'identificazione della rete, mentre gli ultimi 8 bit sono dedicati agli host.

Considerando che abbiamo bisogno di almeno 30 host per piano, abbiamo optato per una subnet con **27 bit dedicati alla rete**, lasciando **5 bit per gli host**, il che ci consente di avere fino a 32 indirizzi IP da dedicare agli host per ogni sottorete.

# Calcolo della subnet mask

# Calcolo della subnet mask

Per garantire un numero sufficiente di indirizzi IP per ogni piano, abbiamo scelto la potenza di 2 che più si avvicina al numero di host che ci serve, ovvero  **$2^5 = 32$  indirizzi totali**.

Di questi 32, 29 sono utilizzabili per gli host (escludendo l'indirizzo di network, di gateway e di broadcast).

L'indirizzo di rete 192.168.0.0/27 si rappresenta in binario come: 11111111.11111111.11111111.11100000

# Calcolo della subnet mask

Per ottenere la maschera di sottorete corretta, abbiamo effettuato il calcolo:

$$0 * 2^0 + 0 * 2^1 + 0 * 2^2 + 0 * 2^3 + 0 * 2^4 + 1 * 2^5 + 1 * 2^6 + 1 * 2^7 = 32 + 64 + 128 = 224$$

Di conseguenza, la maschera di sottorete è **255.255.255.224 (o /27)**.

Il numero 32 indica **l'intervallo di ogni subnet**, che ci consente di creare diverse subnet come viene mostrato nella tabella successiva.

# Tabella IPv4

# Tabella Indirizzi IP

Il NAS e la zona DMZ hanno una sottorete dedicata ciascuno per aumentare ulteriormente la sicurezza della rete interna.

Inoltre se volessimo aggiungere altri NAS o altri server, questi potranno essere aggiunti successivamente in modo semplice.

	Sottorete	Piano	IP Network	IP Gateway	Range Host	IP Broadcast
1	Sottorete	Piano	192.168.0.0/27	192.168.0.1	192.168.0.2 - 192.168.0.30	192.168.0.31
2	Sottorete	Piano	192.168.0.32/27	192.168.0.33	192.168.0.34 - 192.168.0.62	192.168.0.63
3	Sottorete	Piano	192.168.0.64/27	192.168.0.65	192.168.0.66 - 192.168.0.94	192.168.0.95
4	Sottorete	Piano	192.168.0.96/27	192.168.0.97	192.168.0.98 - 192.168.0.126	192.168.0.127
5	Sottorete	Piano	192.168.0.128/27	192.168.0.129	192.168.0.130 - 192.168.0.158	192.168.0.159
6	Sottorete	Piano	192.168.0.160/27	192.168.0.161	192.168.0.162 - 192.168.0.190	192.168.0.191
7	Sottorete	NAS	192.168.0.192/27	192.168.0.193	192.168.0.194 - 192.168.0.222	192.168.0.223
8	Sottorete	DMZ	192.168.0.224/27	192.168.0.225	192.168.0.226 - 192.168.0.254	192.168.0.255

# Scansione delle vulnerabilità

# Scansione delle vulnerabilità

Il Web Server di Theta verrà simulato dalla macchina virtuale di Metasploitable.

Andiamo a svolgere due scansioni sul web server: la scansione dei metodi HTTP e la scansione delle porte di rete aperte.

I **metodi HTTP** sono dei “comandi” che il protocollo HTTP definisce per specificare le azioni che un client (ad esempio, un browser) può eseguire su una risorsa (come una pagina web).

I **protocolli di rete** sono un insieme di regole che permettono a dispositivi di comunicare tra loro su una rete. Sono come linguaggi standardizzati che i dispositivi utilizzano per comunicare.

# Scansione delle vulnerabilità

Alcuni metodi HTTP sono molto pericolosi se lasciati disponibili. Ad esempio:

- **PUT**: Permette di aggiornare o creare una risorsa specifica sul server.
  - Pericoloso perché chiunque avrebbe il potere di modificare dati sul server.
- **DELETE**: Permette di eliminare una risorsa specifica sul server.
  - Chiunque avrebbe il potere di cancellare dati sul server.
- **OPTIONS**: Utilizzato per richiedere le opzioni i metodi abilitati sul web server.
  - Conviene disabilitarlo, in questo modo l'attaccante non potrà conoscere nell'immediato queste informazioni.
- **TRACE**: Utilizzato normalmente per il debug, è pericoloso in quanto l'attaccante potrebbe ottenere informazioni sensibili attraverso degli exploit di questo metodo.
  - Exploit: un programma contenente codice malevolo che sfrutta una vulnerabilità per ottenere accesso non autorizzato ed eseguire azioni dannose o ottenere informazioni sensibili.

# Scansione delle vulnerabilità

Altri metodi HTTP:

- **GET:** Utilizzato per richiedere una risorsa specifica dal server. I dati richiesti sono inclusi nell'URL della richiesta
  - Esempio: una pagina web, un'immagine
  - I dati sono visibili in chiaro nell'URL (maggiore velocità)
  - Anche con GET si possono inviare dati, ma saranno in chiaro nell'URL
- **POST:** Utilizzato per inviare dati al server per l'elaborazione. I dati inviati sono inclusi nel corpo della richiesta, non nell'URL
  - Esempio: form, caricamento dati, invio di dati a un'API
  - I dati sono nascosti nel pacchetto (minore velocità ma maggiore riservatezza)
- **HEAD:** Simile al metodo GET, ma richiede solo i metadati della risorsa (header HTTP) senza il corpo del messaggio.
  - Viene utilizzato per testare le pagine web. Anziché scaricare tutta la pagina, ottieni solo una risposta dal web server che ci dice se la pagina funziona o meno

# Scansione delle vulnerabilità

La scansione delle porte aperte è utile per verificare se sono presenti eventuali porte aperte **non crittografate** che l'attaccante potrebbe sfruttare a suo vantaggio.

È meglio **disabilitare** tutte le porte che non vengono utilizzate, e spostare invece le porte che sono indispensabili per l'azienda.

Ad esempio: se l'azienda utilizza il protocollo telnet, possiamo utilizzare la porta 44444 anziché la porta di default (23).

# Scansione metodi HTTP

Grazie al programma che abbiamo scritto in python, possiamo notare che il server ci risponde.

I metodi abilitati sono GET, HEAD, POST, OPTIONS, TRACE.

Consigliamo di disabilitare **OPTIONS** e soprattutto **TRACE**, che risulta essere la vulnerabilità maggiore.

```
kali@kali: ~/Documents/Python/semplici  
File Actions Edit View Help  
└─(kali㉿kali)-[~/Documents/Python/semplici]  
$ python metodi_http_scanner.py  
ip host: 192.168.3.52/phpmyadmin  
porta (default:80):  
I metodi HTTP supportati sono: GET,HEAD,POST,OPTIONS,TRACE  
└─(kali㉿kali)-[~/Documents/Python/semplici]  
$ █
```

# Scansione porte aperte

```
kali㉿kali: ~/Documents/Python/semplici
File Actions Edit View Help
└─(kali㉿kali)-[~/Documents/Python/semplici]
└─$ python port_scanner.py
Inserisci l'indirizzo IP del dispositivo che vuoi scansionare: 192.168.3.52
Inserisci la porta iniziale (es. 1): 1
Inserisci la porta finale (es. 1024): 1024
Scansione di 192.168.3.52 dalla porta 1 alla porta 1024
+** Porta 21 APERTA +**
+** Porta 22 APERTA +**
+** Porta 23 APERTA +**
+** Porta 25 APERTA +**
+** Porta 53 APERTA +**
+** Porta 80 APERTA +**
+** Porta 111 APERTA +**
+** Porta 139 APERTA +**
+** Porta 445 APERTA +**
+** Porta 512 APERTA +*
+** Porta 513 APERTA +**
+** Porta 514 APERTA +*
```

Con l'altro programma scritto da noi in python, otteniamo la lista delle porte aperte sul web server.

Scansioniamo le **porte note** da 1 a 1024, che sono le più importanti, riservate ai servizi principali (le altre porte sono dinamiche o riservate a servizi minori).

Qui notiamo diverse vulnerabilità, come:

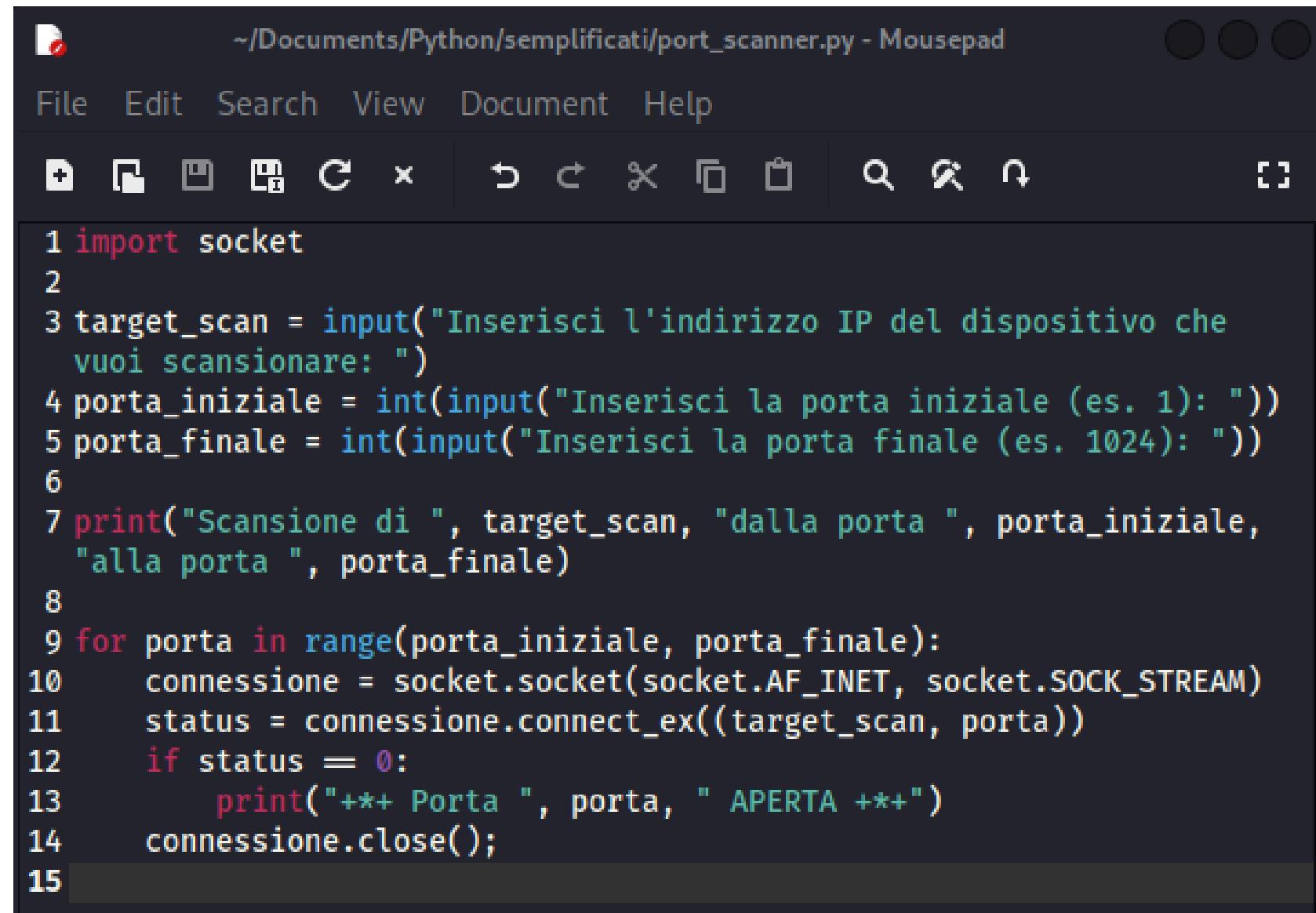
- porta 21: FTP
- porta 23: TELNET
- porta 80: HTTP
- porta 139: NetBIOS
- porta 445: Samba

Bisognerebbe capire quali di queste porte sono indispensabili per l'azienda e di conseguenza spostarle su un **range di porte più alto** o se sostituire alcuni protocolli con dei protocolli crittografati.

Le altre porte, invece, vanno disabilitate.

# Funzionamento programmi python

# Port scanner



The screenshot shows a Python code editor window titled "port\_scanner.py - Mousepad". The code is a simple port scanner script:

```
1 import socket
2
3 target_scan = input("Inserisci l'indirizzo IP del dispositivo che
4 vuoi scansionare: ")
5 porta_iniziale = int(input("Inserisci la porta iniziale (es. 1): "))
6 porta_finale = int(input("Inserisci la porta finale (es. 1024): "))
7
8 print("Scansione di ", target_scan, "dalla porta ", porta_iniziale,
9      "alla porta ", porta_finale)
10
11 for porta in range(porta_iniziale, porta_finale):
12     connessione = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13     status = connessione.connect_ex((target_scan, porta))
14     if status == 0:
15         print("++ Porta ", porta, " APERTA ++")
16     connessione.close()
```

**import socket:** Per prima cosa importiamo la libreria socket, questa libreria servirà per creare e gestire connessioni di rete.

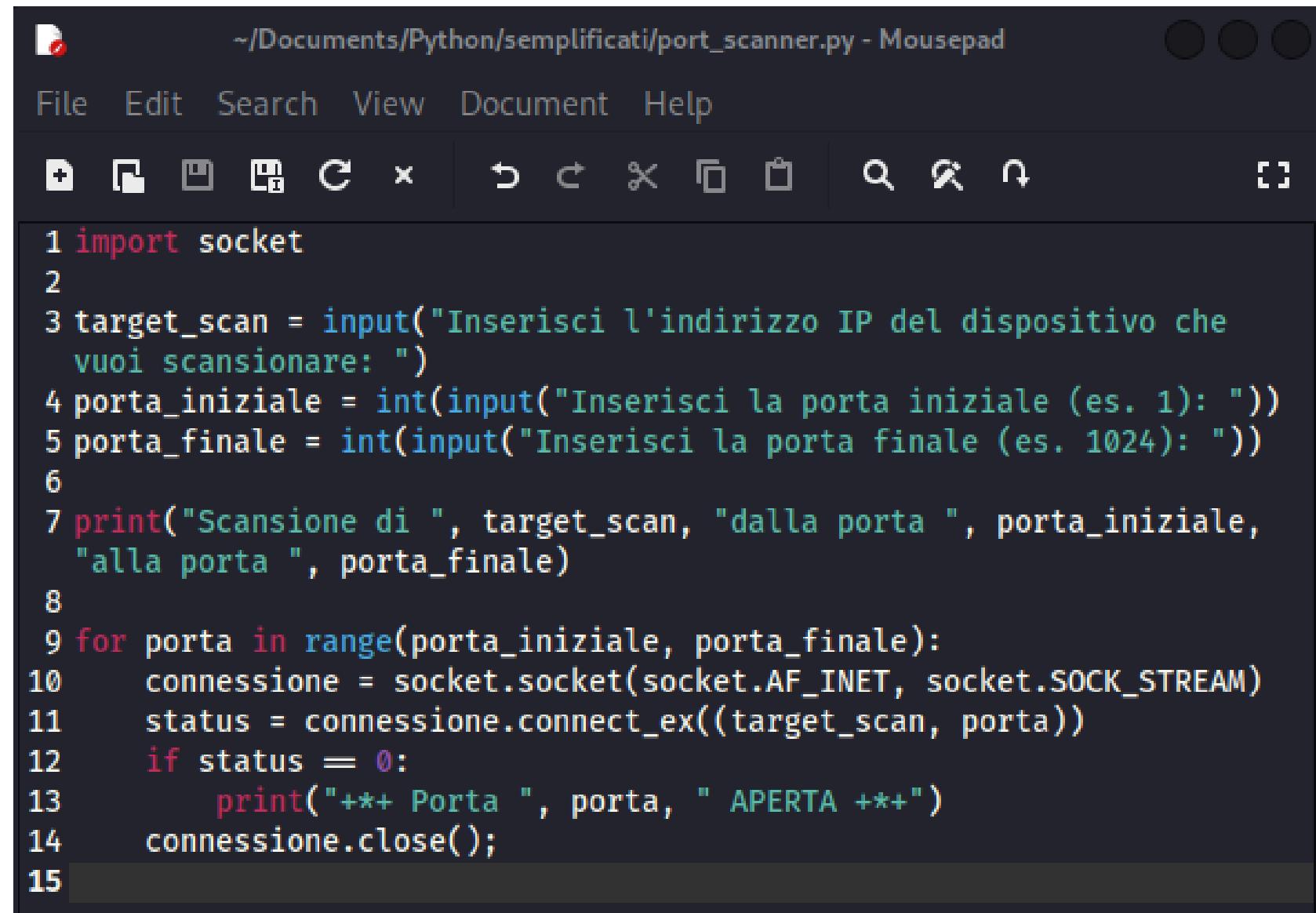
**Target\_scan:** è la prima variabile e chiede all'utente di inserire l'indirizzo ip del dispositivo da scansionare.

**Porta\_iniziale** e **porta\_finale**: sono le altre due variabili in input, dove l'utente specificherà il range di porte da scansionare, convertendo l'input in interi.

**Print:** "" stampa che informa l'utente dell'inizio della scansione specificando target e range di porte.

**For:** Inizia poi il ciclo "for" che itera su ogni numero di porte nel range inserito dall'utente.

# Port scanner



The screenshot shows a terminal window titled "Mousepad" with the file path "~/Documents/Python/semplici/port\_scanner.py". The window contains a Python script for a port scanner. The code uses the `socket` module to scan a target IP address for open ports between a specified range.

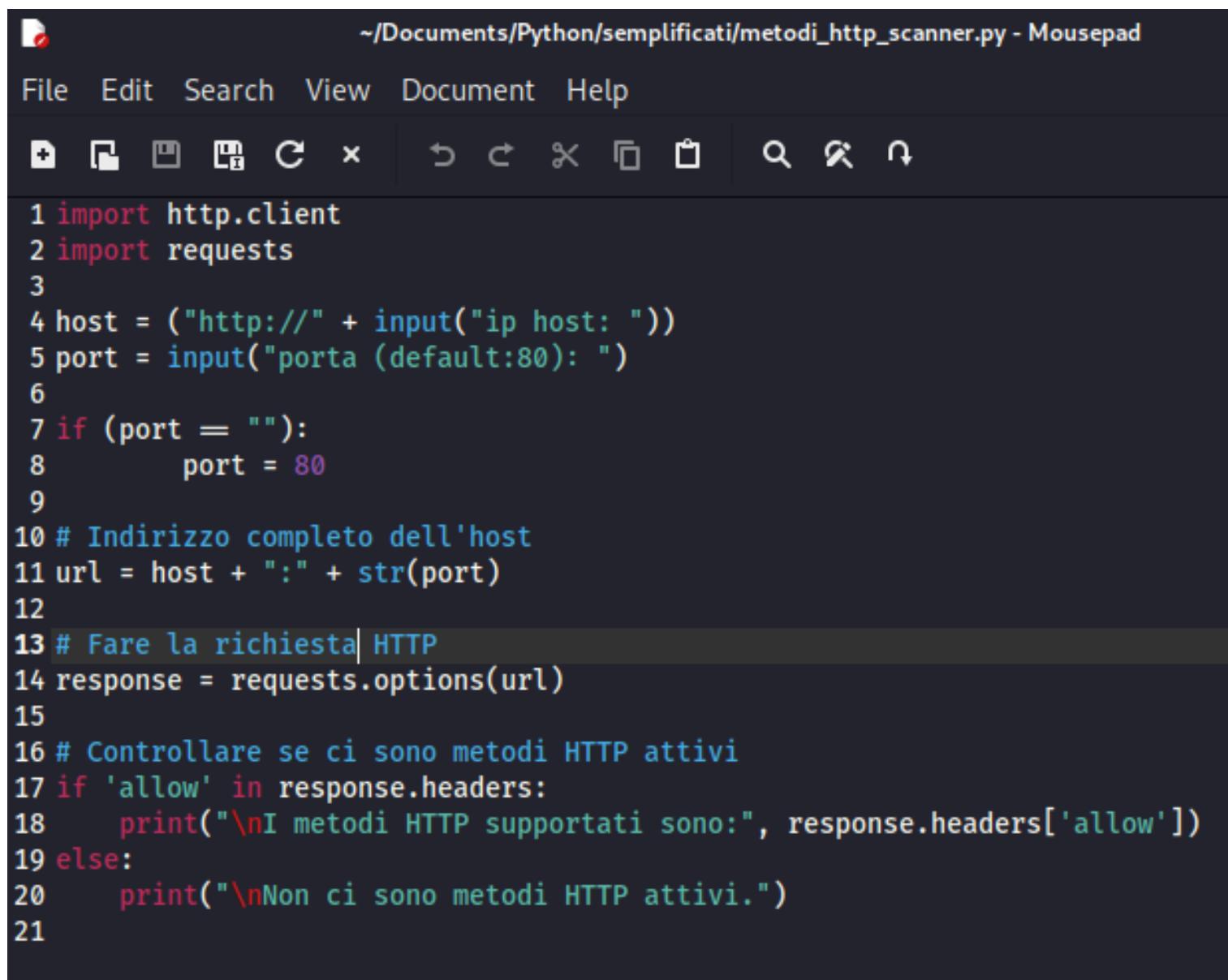
```
1 import socket
2
3 target_scan = input("Inserisci l'indirizzo IP del dispositivo che
4 vuoi scansionare: ")
5 porta_iniziale = int(input("Inserisci la porta iniziale (es. 1): "))
6 porta_finale = int(input("Inserisci la porta finale (es. 1024): "))
7
8 print("Scansione di ", target_scan, "dalla porta ", porta_iniziale,
9      "alla porta ", porta_finale)
10
11 for porta in range(porta_iniziale, porta_finale):
12     connessione = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13     status = connessione.connect_ex((target_scan, porta))
14     if status == 0:
15         print("++ Porta ", porta, " APERTA ++")
16     connessione.close()
```

Connessione = **socket.socket** è la funzione che crea un nuovo socket utilizzando l'ipv4 (AF\_INET) e il protocollo TCP (SOCK\_STREAM).

**Status:** connessione.connect servirà per provare a connettersi all'indirizzo IPv4 specificato e alla corrispettiva porta.

**If status:** se lo status è uguale a 0, la funzione print ci stamperà un messaggio che indicherà che la porta è aperta, dopodichè chiuderà la connessione al socket.

# Scanner metodi HTTP



The screenshot shows a code editor window with the title bar reading "Documents/Python/semplicati/metodi\_http\_scanner.py - Mousepad". The menu bar includes File, Edit, Search, View, Document, and Help. Below the menu is a toolbar with various icons. The main area contains the following Python code:

```
1 import http.client
2 import requests
3
4 host = ("http://" + input("ip host: "))
5 port = input("porta (default:80): ")
6
7 if (port == ""):
8     port = 80
9
10 # Indirizzo completo dell'host
11 url = host + ":" + str(port)
12
13 # Fare la richiesta| HTTP
14 response = requests.options(url)
15
16 # Controllare se ci sono metodi HTTP attivi
17 if 'allow' in response.headers:
18     print("\nI metodi HTTP supportati sono:", response.headers['allow'])
19 else:
20     print("\nNon ci sono metodi HTTP attivi.")
21
```

**Import http.client e import request:** sono le librerie necessarie per inviare richieste http

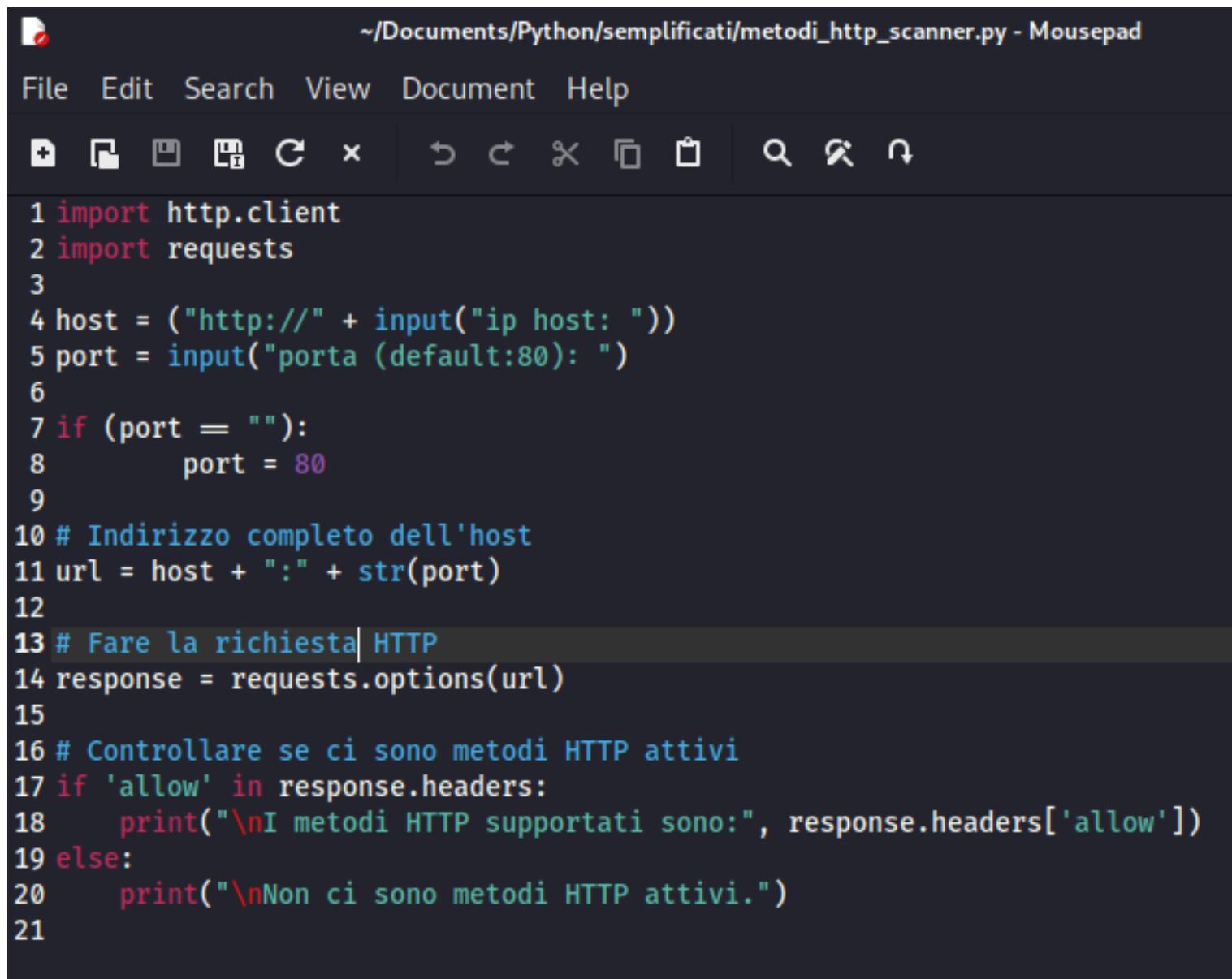
**Host:** chiede all'utente di inserire l'indirizzo ip del server che si vuole scansionare

**Port:** chiede all'utente di inserire la porta che si vuole raggiungere, se l'utente non inserisce nulla, di default il programma inserirà la porta 80

**If:** se l'input sarà vuoto verrà inserita automaticamente la porta 80

**Url:** forma un url completo, composto dall'indirizzo IP e la porta

# Scanner metodi HTTP



The screenshot shows a code editor window with a dark theme. The title bar reads "Documents/Python/semplicati/metodi\_http\_scanner.py - Mousepad". The menu bar includes File, Edit, Search, View, Document, and Help. Below the menu is a toolbar with various icons. The main area contains the following Python code:

```
1 import http.client
2 import requests
3
4 host = ("http://" + input("ip host: "))
5 port = input("porta (default:80): ")
6
7 if (port == ""):
8     port = 80
9
10 # Indirizzo completo dell'host
11 url = host + ":" + str(port)
12
13 # Fare la richiesta HTTP
14 response = requests.options(url)
15
16 # Controllare se ci sono metodi HTTP attivi
17 if 'allow' in response.headers:
18     print("\nI metodi HTTP supportati sono:", response.headers['allow'])
19 else:
20     print("\nNon ci sono metodi HTTP attivi.")
21
```

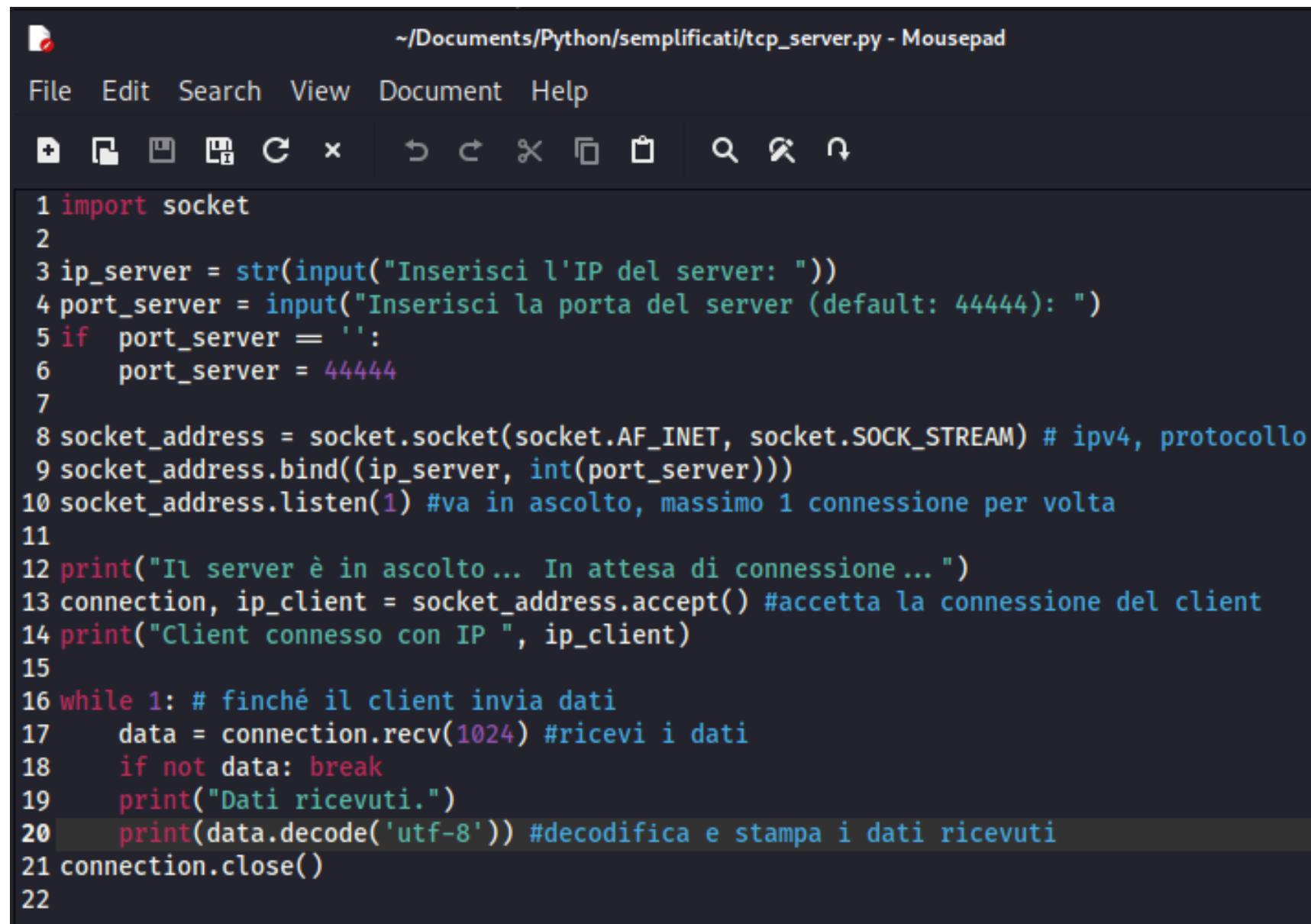
**Response:** la funzione `request.options` chiede al server quali metodi http può supportare.

**If allow in response.headers:** verifica se l'intestazione "allow" è presente nel server, questa funzione contiene una lista di tutti i metodi http che il server può supportare

**Print:** se l'intestazione allow è presente, la funzione `print` ci stamperà i metodi supportati (GET, POST, PUT, DELETE)

**Else/print:** se l'intestazione allow non è presente, verrà stampato a video il messaggio che dice che non sono presenti metodi http

# Scambio dati tramite TCP



The image shows a screenshot of a Python code editor. The title bar reads " ~/Documents/Python/semplicati/tcp\_server.py - Mousepad". The menu bar includes "File", "Edit", "Search", "View", "Document", and "Help". Below the menu is a toolbar with various icons. The main window displays the following Python code:

```
1 import socket
2
3 ip_server = str(input("Inserisci l'IP del server: "))
4 port_server = input("Inserisci la porta del server (default: 44444): ")
5 if port_server == '':
6     port_server = 44444
7
8 socket_address = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # ipv4, protocollo
9 socket_address.bind((ip_server, int(port_server)))
10 socket_address.listen(1) #va in ascolto, massimo 1 connessione per volta
11
12 print("Il server è in ascolto... In attesa di connessione...")
13 connection, ip_client = socket_address.accept() #accetta la connessione del client
14 print("Client connesso con IP ", ip_client)
15
16 while 1: # finché il client invia dati
17     data = connection.recv(1024) #ricevi i dati
18     if not data: break
19     print("Dati ricevuti.")
20     print(data.decode('utf-8')) #decodifica e stampa i dati ricevuti
21 connection.close()
22
```

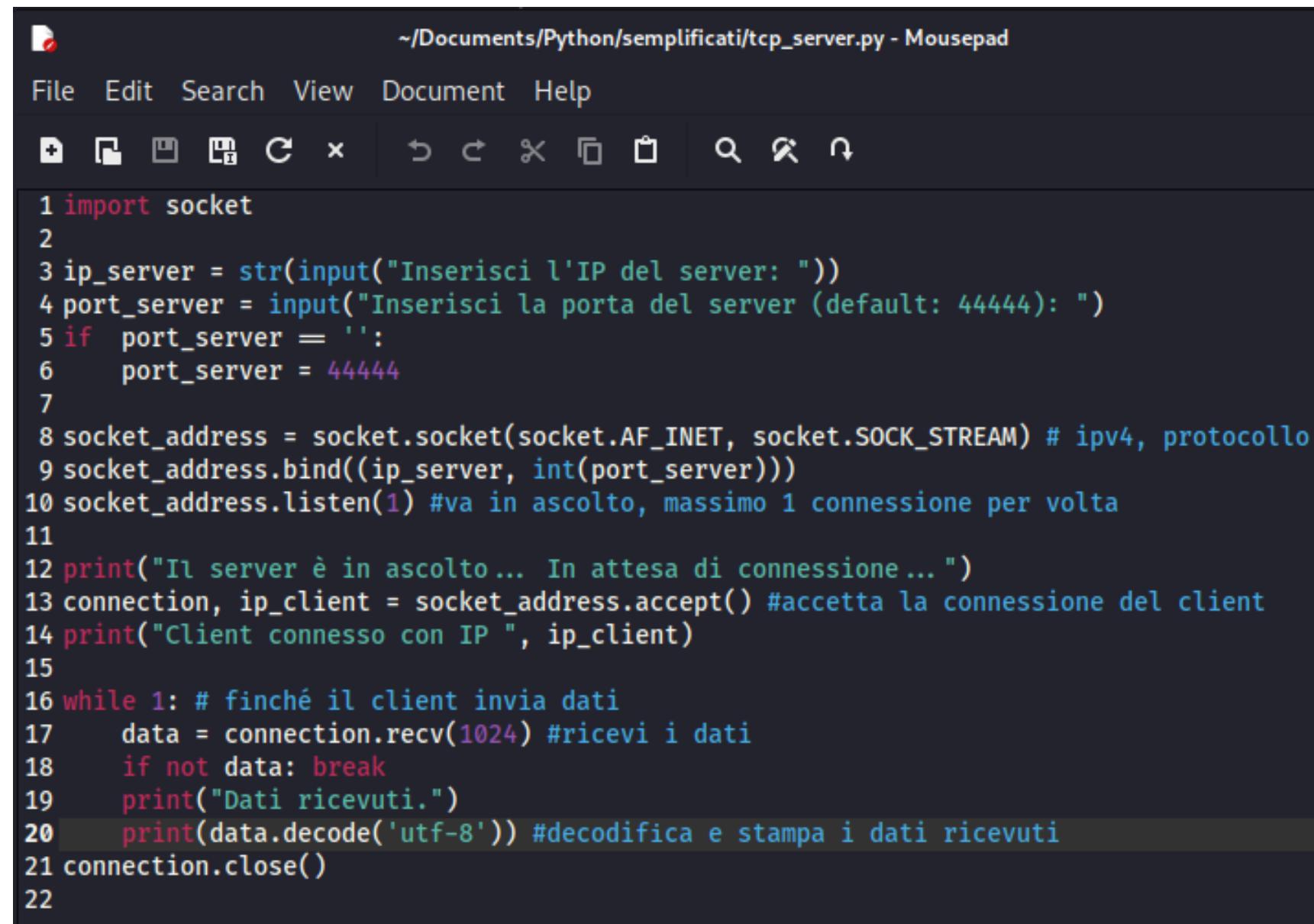
**Import socket:** importiamo il modulo socket che fornirà le funzioni necessarie per la comunicazione in rete

**Ip\_server:** l'utente inserirà l'IP del server con il quale si vuole comunicare, l'input verrà convertito in una stringa

**Port\_server:** l'utente inserisce il numero della porta sulla quale il server ascolterà

**If:** è la condizione dove specifica che se l'utente non inserirà nulla, verrà inserita la porta 44444

# Scambio dati tramite TCP



The screenshot shows a code editor window titled 'tcp\_server.py - Mousepad'. The menu bar includes 'File', 'Edit', 'Search', 'View', 'Document', and 'Help'. Below the menu is a toolbar with various icons. The code itself is a Python script for a TCP server:

```
1 import socket
2
3 ip_server = str(input("Inserisci l'IP del server: "))
4 port_server = input("Inserisci la porta del server (default: 44444): ")
5 if port_server == '':
6     port_server = 44444
7
8 socket_address = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # ipv4, protocollo
9 socket_address.bind((ip_server, int(port_server)))
10 socket_address.listen(1) #va in ascolto, massimo 1 connessione per volta
11
12 print("Il server è in ascolto... In attesa di connessione...")
13 connection, ip_client = socket_address.accept() #accetta la connessione del client
14 print("Client connesso con IP ", ip_client)
15
16 while 1: # finché il client invia dati
17     data = connection.recv(1024) #ricevi i dati
18     if not data: break
19     print("Dati ricevuti.")
20     print(data.decode('utf-8')) #decodifica e stampa i dati ricevuti
21 connection.close()
22
```

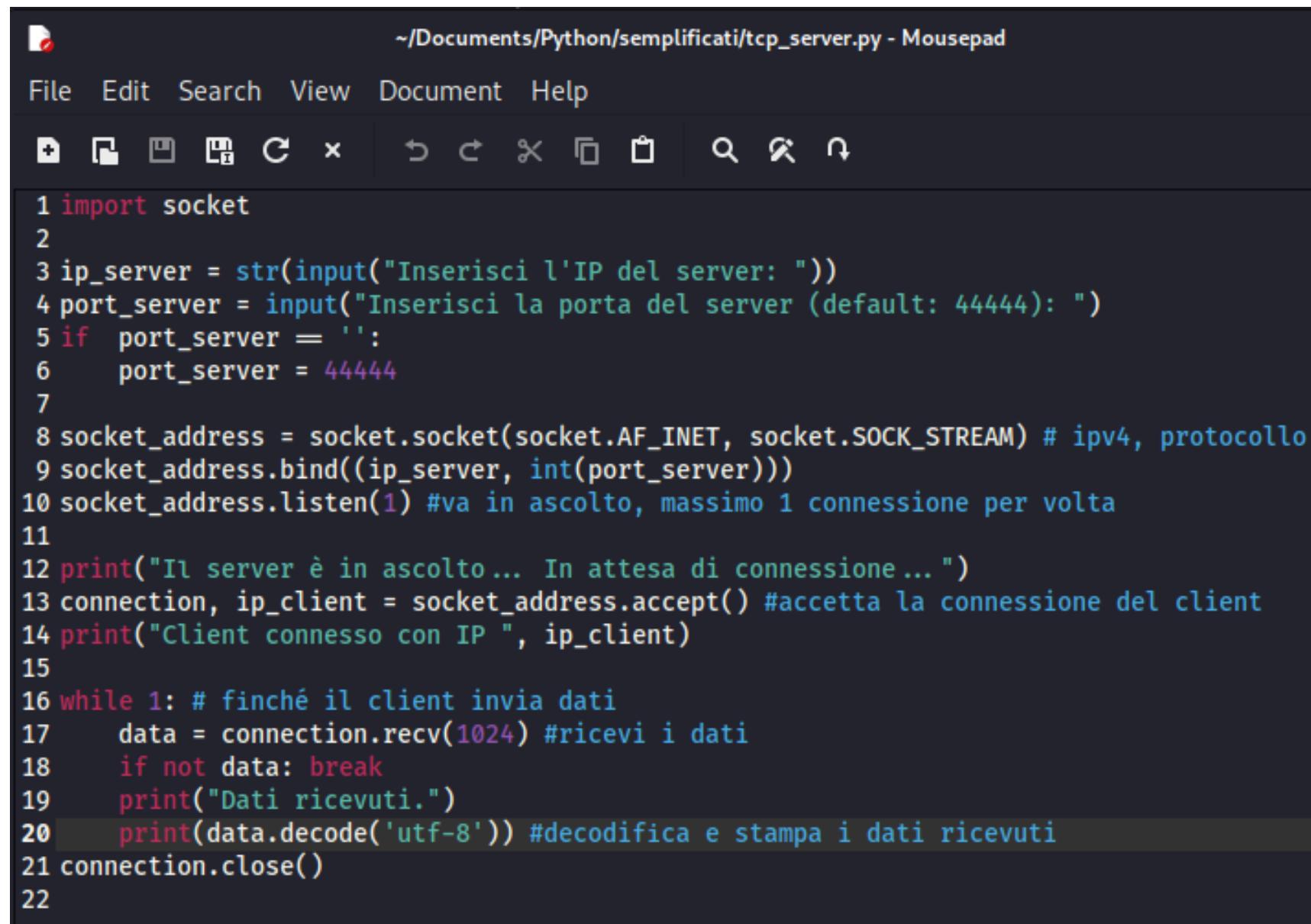
**Socket\_address:** crea un nuovo socket dove verrà specificato che verrà utilizzato il protocollo Ipv4 (AF-INET), e il protocollo TCP (SOCK-STREAM) che è orientato alla connessione

**Socket\_address.bind:** associa l'indirizzo IP alla porta per verificare le connessioni in arrivo

**Socket\_address.listen (1):** il server resta in ascolto in attesa di connessioni, il numero 1 indica che il server accetterà massimo una connessione per volta

**Print:** stamperà un messaggio che indica che il server è pronto ad accettare connessioni

# Scambio dati tramite TCP



The screenshot shows a code editor window titled 'tcp\_server.py - Mousepad'. The menu bar includes 'File', 'Edit', 'Search', 'View', 'Document', and 'Help'. Below the menu is a toolbar with icons for new file, open file, save file, cut, copy, paste, find, and others. The code itself is a Python script for a TCP server:

```
1 import socket
2
3 ip_server = str(input("Inserisci l'IP del server: "))
4 port_server = input("Inserisci la porta del server (default: 44444): ")
5 if port_server == '':
6     port_server = 44444
7
8 socket_address = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # ipv4, protocollo
9 socket_address.bind((ip_server, int(port_server)))
10 socket_address.listen(1) #va in ascolto, massimo 1 connessione per volta
11
12 print("Il server è in ascolto... In attesa di connessione...")
13 connection, ip_client = socket_address.accept() #accetta la connessione del client
14 print("Client connesso con IP ", ip_client)
15
16 while 1: # finché il client invia dati
17     data = connection.recv(1024) #ricevi i dati
18     if not data: break
19     print("Dati ricevuti.")
20     print(data.decode('utf-8')) #decodifica e stampa i dati ricevuti
21 connection.close()
22
```

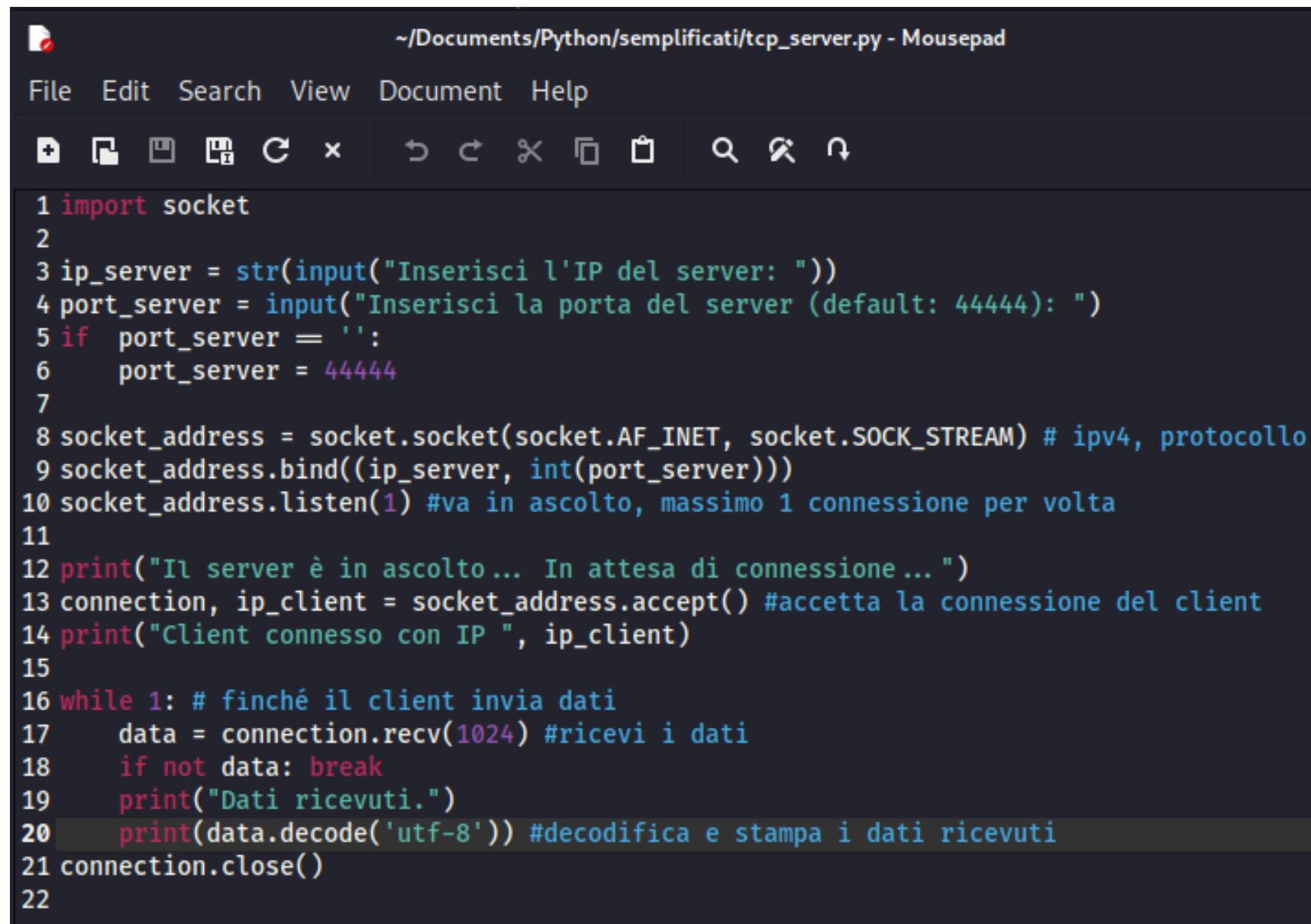
**Connection, IP client:** è il nuovo socket contenente l'indirizzo IP del client che si sarà collegato

**Print:** stampa l'indirizzo IP del client connesso

**While 1:** è il ciclo che continuerà finché ci saranno dati da ricevere dal client, mentre `connection.recv(1024)` vuol dire che riceverà fino a 1024 byte di dati dal client. 1 equivale a True.

**If not data break:** è la condizione dove, nel caso non siano stati ricevuti i dati, il ciclo terminerà

# Scambio dati tramite TCP



The screenshot shows a code editor window titled "tcp\_server.py - Mousepad". The menu bar includes File, Edit, Search, View, Document, and Help. The toolbar contains icons for new file, open, save, cut, copy, paste, find, and others. The code itself is a Python script for a TCP server:

```
1 import socket
2
3 ip_server = str(input("Inserisci l'IP del server: "))
4 port_server = input("Inserisci la porta del server (default: 44444): ")
5 if port_server == '':
6     port_server = 44444
7
8 socket_address = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # ipv4, protocollo
9 socket_address.bind((ip_server, int(port_server)))
10 socket_address.listen(1) #va in ascolto, massimo 1 connessione per volta
11
12 print("Il server è in ascolto... In attesa di connessione...")
13 connection, ip_client = socket_address.accept() #accetta la connessione del client
14 print("Client connesso con IP ", ip_client)
15
16 while 1: # finché il client invia dati
17     data = connection.recv(1024) #ricevi i dati
18     if not data: break
19     print("Dati ricevuti.")
20     print(data.decode('utf-8')) #decodifica e stampa i dati ricevuti
21 connection.close()
22
```

**Print (dati ricevuti):** stamperà un messaggio che indicherà se i dati sono stati effettivamente ricevuti

**Print (data decode):** decodifica i dati ricevuti utilizzando la codifica (UTF\_8) e la stampa

**Connection.close:** chiude il socket di connessione con il client

# Blocco del traffico con PFSense

# Obiettivo del firewall

Il firewall è un dispositivo o un software progettato per **proteggere una rete interna da minacce esterne**, regolando il traffico di rete in entrata e in uscita.

**Esamina il traffico di rete**, spacchettando e ri-impacchettando i pacchetti e decide se consentire o bloccare il passaggio dei dati in base a regole predefinite.

PFsense è un firewall di tipo software, è gratuito e Open Source e lo utilizzeremo per filtrare in modo statico (ovvero inserendo noi manualmente una regola con parametri statici, ovvero IP e porta da bloccare).

# Obiettivo del firewall

In questo esercizio, configureremo una regola di firewall su pfSense per **bloccare il traffico HTTP** tra due macchine virtuali, Kali e Metasploitable, utilizzando VirtualBox.

Il web server (Metasploitable) utilizza il **protocollo HTTP** e ospita la sua pagina web.

Nella regola del firewall, andremo a **bloccare la porta 80**, che è la porta del protocollo HTTP.

# Configurazione VirtualBox

## Configurazione delle reti su VirtualBox:

- Aggiungi **due schede di rete interne** a pfSense, assegnando ad ognuna di esse una rete differente.
  - Esempio: Rete interna LAN1, Rete interna LAN2.
- Assegna una rete interna a Kali e una Metasploitable per evitare che il traffico passi direttamente tra le due macchine. Le due macchine dovranno poi interfacciarsi alle due reti interne differenti.
  - Esempio: LAN1 per Metasploitable e LAN2 per Kali.

# Configurazione pfSense

Tramite l'interfaccia web di pfSense:

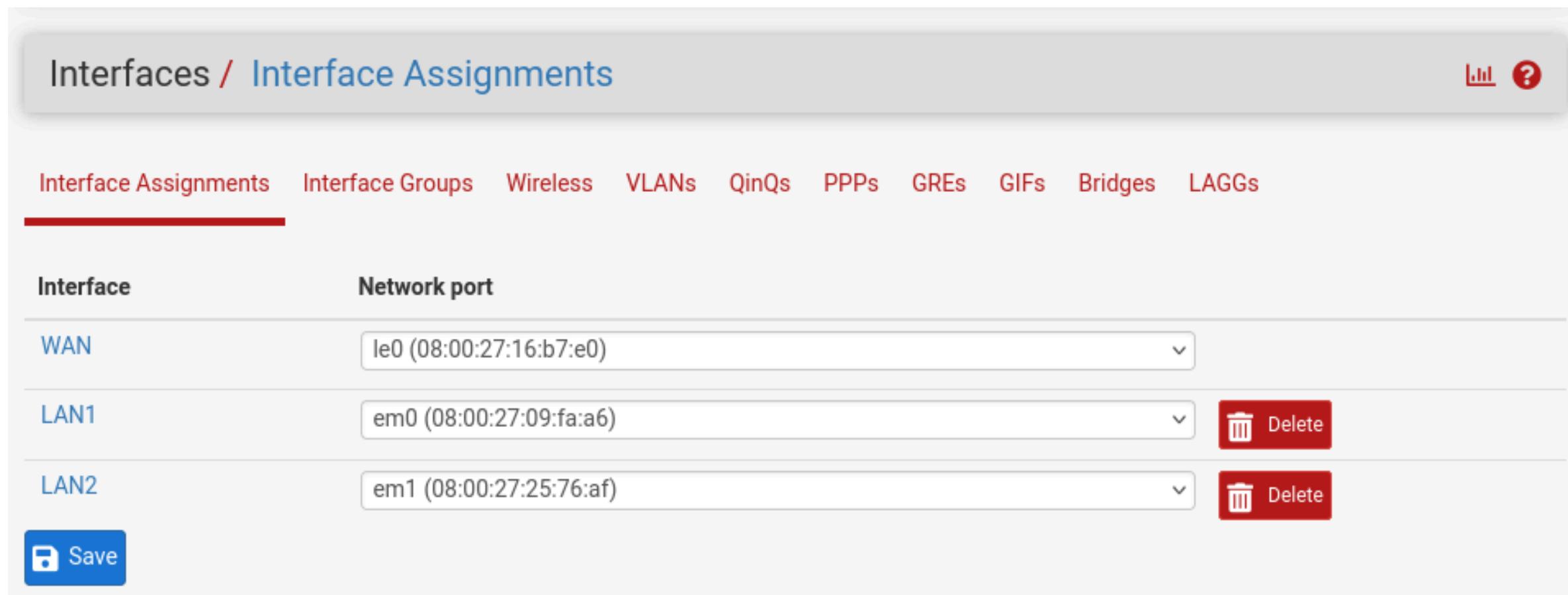
- Vai su System > Interfaces e aggiungi una **seconda interfaccia di rete LAN**.
  - Sempre su System > Interfaces, assegna le interfacce di rete alle relative schede di rete (ad esempio: LAN1 con indirizzo MAC 08:00:27:09:FA:A6. Il MAC lo vedi dalle impostazioni della scheda di rete di VirtualBox)

Interfaces / Interface Assignments

Interface Assignments    Interface Groups    Wireless    VLANs    QinQs    PPPs    GREs    GIFs    Bridges    LAGGs

Interface	Network port	
WAN	le0 (08:00:27:16:b7:e0)	
LAN1	em0 (08:00:27:09:fa:a6)	<span>Delete</span>
LAN2	em1 (08:00:27:25:76:af)	<span>Delete</span>

Save



# Configurazione pfSense

- Configura il gateway tra LAN1 e LAN2:
  - Vai su System > Routing e imposta i **gateway** per entrambe le reti, assegnando a ciascuna rete il proprio indirizzo IP gateway.
  - Ora Metasploitable e Kali potranno comunicare tra loro. Senza il gateway, essendo le due macchine su due reti diverse, non potrebbero comunicare.
  - Il gateway ci permette di instradare il traffico tra reti diverse.

System / Routing / Gateways

Gateways Static Routes Gateway Groups

Gateways							
	Name	Default	Interface	Gateway	Monitor IP	Description	Actions
<input type="checkbox"/>	<input checked="" type="radio"/> LAN1	Default (IPv4)	LAN1	192.168.1.155	192.168.1.155	kali	
<input type="checkbox"/>	<input checked="" type="radio"/> LAN2		LAN2	192.168.2.155	192.168.2.155	metasploitable	

Save Add

# Configurazione pfSense

- Configura il **server DHCP** sia per LAN1 che per LAN2:
  - Vai su Services > DHCP server e abilita il server DHCP per entrambe le LAN
  - Imposta un pool di indirizzi IP per entrambe le LAN (es. 192.168.1.50–100, 192.168.2.50–100)
  - Ora Metasploitable e Kali otterranno automaticamente la configurazione di rete da pfSense (ip, subnet mask, gateway)
  - Entrambe le macchine, a loro volta, saranno impostate su DHCP

The screenshot shows two parts of the pfSense DHCP configuration interface:

**General DHCP Options:**

- WAN   LAN1   LAN2
- General DHCP Options
- DHCP Backend: ISC DHCP
- Enable:  Enable DHCP server on LAN1 interface

**Primary Address Pool:**

Primary Address Pool	
Subnet	192.168.1.0/24
Subnet Range	192.168.1.1 - 192.168.1.254
Address Pool Range	192.168.1.50
	From
	To
	192.168.1.100

# Configurazione regola del firewall

Tramite l'interfaccia web di pfSense:

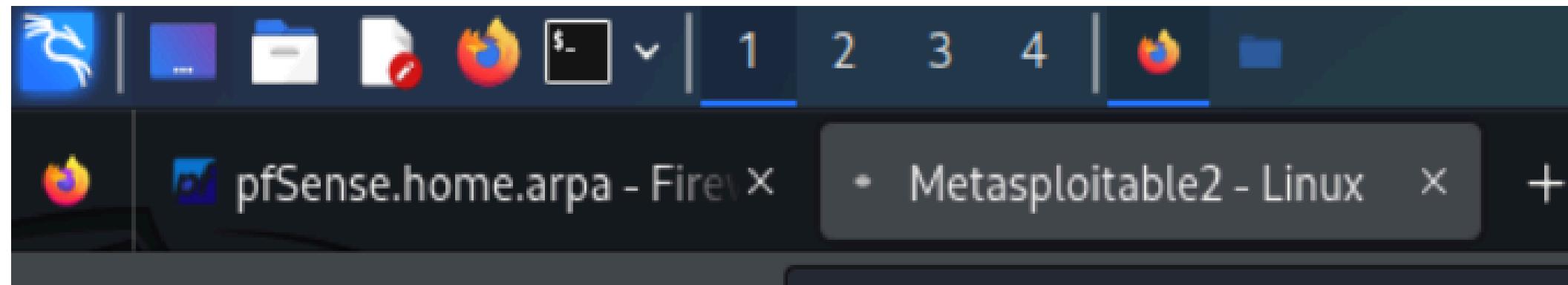
- Creazione della regola firewall:
  - Vai su Firewall > Rules > LAN nell'interfaccia di pfSense.
  - Crea una regola che blocchi il traffico TCP in ingresso e in uscita tra Kali e Metasploitable
    - Inserisci gli indirizzi IP delle due macchine
    - Seleziona protocollo HTTP per bloccare questo protocollo specifico verso il dispositivo di destinazione
  - Applica la regola e verifica che Kali non possa più connettersi a Metasploitable
    - Prova a riaccedere a Metasploitable tramite browser inserendo l'IP della macchina.

# Configurazione regola del firewall

Se tutto è andato a buon fine, quando la regola è applicata sarà **impossibile raggiungere l'interfaccia web** di Metasploitable.

Il browser proverà a caricare all'infinito la pagina di Metasploitable, ma non ci riuscirà mai.

Quando la regola è disabilitata, invece, la comunicazione tra Kali e Metasploitable sarà nuovamente possibile.



# Preventivo e dispositivi scelti

# Preventivo e dispositivi scelti

Qui il preventivo completo: [VISUALIZZA IL PREVENTIVO](#)

Di seguito, la spiegazione del perché abbiamo scelto questi dispositivi per l'azienda Thetha.

**Firewall:** Il Palo Alto Networks PA-3220 è un firewall progettato per offrire prestazioni elevate e sicurezza avanzata in ambienti di rete complessi.

È in grado di gestire grandi volumi di traffico, garantendo al contempo una bassa latenza.



# Dispositivi scelti

**Switch:** Cisco Catalyst 9200 è uno switch di rete ad alte prestazioni progettati per rispondere alle esigenze di imprese moderne e data center.

Questi dispositivi offrono una combinazione di velocità, sicurezza e flessibilità, rendendoli ideali per creare reti affidabili e scalabili.



**NAS:** Il SanDisk Professional 24TB è un'unità di storage ad alte prestazioni progettata specificamente per professionisti e aziende che necessitano di grandi capacità di archiviazione.

Questo dispositivo offre una combinazione di velocità, affidabilità e capacità, rendendolo ideale per gestire file di grandi dimensioni come video 4K, immagini ad alta risoluzione e altri dati multimediali.



# Dispositivi scelti

**Router:** Cisco Meraki MX85 è un router di rete di fascia alta, progettato specificamente per soddisfare le esigenze delle medie e grandi imprese.

Offre un'elevata capacità di elaborazione e una bassa latenza, ideale per gestire un elevata quantità di traffico.



**IDS/IPS:** Il Cisco Firepower 1010 è un sistema di rilevamento e prevenzione delle intrusioni ad alte prestazioni progettato per proteggere le reti da una vasta gamma di minacce informatiche.



# Dispositivi scelti

**Computer:** Il Lenovo IdeaCentre 3 è un desktop compatto e potente, progettato per offrire prestazioni affidabili e un design elegante.

Con un processore AMD Ryzen 5 e 16GB di RAM, questo PC è una macchina affidabile in grado di gestire tante attività in multitasking.



**Cuffie con microfono:** Il Sennheiser PC 3.2 Chat è un auricolare progettato specificamente per offrire una comunicazione chiara e nitida durante le chiamate vocali e le videoconferenze.

Grazie alla sua tecnologia di cancellazione del rumore, questo microfono è in grado di ridurre significativamente i rumori di fondo, garantendo una qualità audio eccellente.



# Dispositivi scelti

**Mouse:** Logitech G203 è un mouse silenzioso e resistente ai liquidi, garantendo il funzionamento anche in caso di eventuali sversamenti accidentali.



**Tastiera:** Corsair K55 CORE RGB è una tastiera a membrana che offre un ottimo rapporto qualità-prezzo, unendo prestazioni affidabili, silenziosità e una migliore lettura dei tasti anche in scarse condizioni di luce.



# Dispositivi scelti

**Monitor:** AOC 27B2H è dotato di un pannello IPS Full HD da 27" con colori realistici. Le connessioni VGA e HDMI offrono la massima flessibilità.

Inoltre, è dotato delle tecnologie anti-luce blu e anti-sfarfallio per il benessere degli occhi.



**Web server:** Cisco UCS C220 M7 offre alte grandi prestazioni grazie ai processori Intel Xeon di 5<sup>a</sup> generazione.

È noto per la sua efficienza, rendendolo adatto per gestire servizi web, database e carichi di lavoro aziendali.

