



Compagnia Theta

Progetto di Rete Compagnia Theta

Build week 1

Team ViperSec

Programma

- Progetto e Divisione di rete
- Struttura della Rete
- Programmazione (Port scanner)
- Verifica verbi HTTP
- Connessioni con socket di rete
- Risoluzione problemi sorti durante il progetto
- Dispositivi

Team ViperSec

PROGETTAZIONE RETE PER THETA

Team Leader
FLAVIO DI CROCE

CONTABILITA'
ANDREA CILLI

INFRASTUTTURA RETE
PIETRO QUINTO
FELIPE AMORIM SORIA
ANDREA CILLI

SCRITTURA CODICI DI SICUREZZA
CRISTIANO LANFRANCHI
GIANCLAUDIO CAPPELLETTI
CATERINA RUBINO

GRAPHIC DESIGNER
FLAVIO CROCE
VINCENZO CARACCIOLI

ViperSec

Nata nell'ombra delle minacce digitali, ViperSec è la squadra che agisce con la rapidità letale di una vipera.

Con una precisione quasi soprannaturale, si infiltra nei sistemi nemici, neutralizzando ogni attacco prima che si materializzi.

La loro leggenda narra di una forza implacabile che, con un solo morso digitale, trasforma vulnerabilità in punti di forza, proteggendo le informazioni più preziose come se fossero il loro unico tesoro.

Progetto

Siamo stati contattati dalla compagnia **Theta** per sviluppare un preventivo di spesa e un progetto di rete per la loro infrastruttura IT.

Suddiviso in 6 piani, con una divisione ideale di 20 postazioni di lavoro per piano.

Componenti aggiuntivi:

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

STEP

Analisi della richiesta

Creare un'infrastruttura di rete su sei piani secondo le esigenze dell'azienda

Progettazione della rete

Progettazione virtuale della rete su 6 livelli, con 20 dispositivi previsti per ciascun livello.

Sicurezza della rete

Messa in sicurezza della rete tramite divisione della stessa, firewall, NAS e IPS/IDS

Programmazione

Scrittura codici per port scanner e rilevamento verbi HTTP, risoluzione problematiche sorte

Consegna del preventivo

Consegna del preventivo di spesa e progetto di rete funzionale, sicura e facilmente gestibile.

1

2

3

4

5

PROGETTO E DIVISIONE DI RETE

PROGETTO INFRASTRUTTURA DI RETE



Scelta Subnetting + Vlan

Migliori Prestazioni

Se tutti i dispositivi fossero su un'unica rete, si creerebbero colli di bottiglia e rallentamenti.

Le Subnet e VLAN ottimizzano il traffico, evitando congestioni e migliorando l'efficienza.

Esempio: Amazon suddivide le sue reti per gestire milioni di richieste al secondo senza ritardi.

Maggiore Sicurezza

Separare i dispositivi riduce i rischi di attacchi informatici e accessi non autorizzati.

È possibile limitare l'accesso a dati sensibili e isolare sistemi critici.

Esempio: In ospedale, i dispositivi medici sono isolati dalla rete amministrativa per proteggerli da attacchi.

Scelta Subnetting + Vlan

Gestione e Scalabilità

Le reti devono essere facili da gestire e pronte per crescere con l'azienda.

Subnet e VLAN permettono di controllare ogni reparto separatamente, applicare aggiornamenti mirati e gestire priorità di rete.

Esempio: Le banche separano gli sportelli bancomat dai sistemi interni per sicurezza e affidabilità.

Conclusione

Le Subnet organizzano la rete a livello IP.

Le VLAN segmentano il traffico direttamente sugli switch.

Il risultato? Una rete più veloce, sicura e gestibile.

STRUTTURA DELLA RETE

Struttura della Rete

La rete aziendale è stata progettata con un'architettura **scalabile** e **sicura**, suddivisa in più piani per garantire un'organizzazione **efficiente**.

Ogni piano dispone di un numero significativo di dispositivi connessi, organizzati secondo una struttura gerarchica basata su **switch** di accesso e di distribuzione.

Struttura della Rete

Gli elementi principali della rete includono:

- **Switch di accesso** per collegare i dispositivi finali.
- **Switch di distribuzione** per aggregare il traffico e inoltrarlo ai router.
- **Quattro router**, configurati per garantire ridondanza e bilanciamento del traffico.
- **Firewall** perimetrale, posizionato tra la rete aziendale e la WAN per la protezione dagli attacchi esterni.
- **DMZ (Zona Demilitarizzata)**, contenente un server **HTTP** e un sistema **IPS (Intrusion Prevention System)** per garantire sicurezza avanzata.
- **NAS (Intrusion Detection System)** offre archiviazione centralizzata in rete per backup, condivisione e gestione dati.
- **L'IDS (Intrusion Detection System)** monitora il traffico di rete per rilevare attività sospette o attacchi informatici.

Scelta di Switch, Firewall e Router

Switch

Sono stati scelti **switch Layer 2** per il livello di accesso, poiché consentono una connessione diretta ai dispositivi degli utenti finali. Per il livello di distribuzione e core, sono stati impiegati **switch Layer 3**, che permettono la gestione del traffico tra i piani e l'instradamento verso i router.

Questa separazione permette di:

- **Ottimizzare il traffico**, riducendo il dominio di broadcast.
- **Migliorare le prestazioni**, evitando congestioni di rete.
- **Garantire scalabilità**, facilitando future espansioni della rete.

Scelta di Switch, Firewall e Router

Posizionamento degli Switch

- Gli **switch** di accesso connettono i dispositivi locali di ogni piano, migliorando la gestione del traffico locale.
- Gli **switch** di distribuzione aggregano il traffico degli switch di accesso e lo inviano ai router per l'instradamento.

Scelta di Switch, Firewall e Router

Firewall Perimetrale

Il firewall implementato è un **Cisco 5506-X**, posizionato tra la rete aziendale e la **WAN**. Il suo scopo principale è:

- Controllare il traffico in entrata e in uscita.
- Filtrare il traffico in base a policy di sicurezza aziendali.
- Impedire accessi non autorizzati alle risorse interne.

Scelta di Switch, Firewall e Router

Motivo del posizionamento:

- Il **firewall** è collocato tra la rete interna e la WAN per proteggere l'azienda da attacchi esterni.
- Si trova prima della **DMZ** per garantire che solo il traffico controllato possa accedere ai servizi esposti, come il **server HTTP**.

Configurazione dei Router e Backup

I quattro router presenti nella rete sono stati configurati per garantire:

- **Instradamento** efficiente tra le **VLAN** e le **subnet**.
- **Ridondanza**, con uno schema di **failover** per evitare interruzioni in caso di guasto di un router.
- **Bilanciamento** del carico, per distribuire equamente il traffico di rete.

Configurazione dei Router e Backup

Configurazione del backup dei router

Per garantire continuità operativa, sono stati implementati protocolli avanzati:

- **OSPF (Open Shortest Path First)** o **EIGRP** per un routing dinamico che si adatta ai cambiamenti della rete.
- **HSRP (Hot Standby Router Protocol)** o **VRRP (Virtual Router Redundancy Protocol)** per la ridondanza:
 - Un router è attivo come principale.
 - Gli altri due router sono in standby e subentrano automaticamente in caso di guasto.
 - Il quarto router è necessario per filtrare il traffico in entrata
- **Bilanciamento del carico**, configurando costi di routing diversi per distribuire il traffico in modo equo.

Funzionamento della DMZ e IPS

La DMZ (Zona Demilitarizzata) è un'area della rete separata dalla rete interna e progettata per ospitare servizi accessibili dall'esterno, riducendo il rischio di compromissione della rete interna.

- **Il server HTTP** si trova nella DMZ per gestire richieste web senza esporre la rete interna.
- **L'IPS (Intrusion Prevention System)** monitora il traffico per identificare e bloccare eventuali attacchi prima che possano compromettere i server.

Funzionamento della DMZ e IPS

Motivo della separazione:

- La **DMZ** limita l'accesso ai server pubblici, evitando che un attacco a questi servizi possa propagarsi alla rete interna.
- L'**IPS** analizza il traffico e interviene automaticamente per bloccare le minacce.

NAS e IDS

IDS (Intrusion Detection System)

Oltre all'**IPS** nella **DMZ**, è stato implementato un sistema **IDS** per monitorare il traffico interno e rilevare attività sospette.

L'**IDS** analizza i pacchetti di rete in tempo reale, identificando eventuali minacce e generando **alert** per gli amministratori di rete. Il suo posizionamento strategico tra gli switch di distribuzione e il firewall permette di rilevare attacchi interni e tentativi di accesso non autorizzati.

NAS e IDS

NAS (Network-Attached Storage)

È stato implementato un sistema **NAS** per la gestione e l'archiviazione centralizzata dei dati aziendali.

Il **NAS** permette agli utenti di accedere e condividere file in modo sicuro all'interno della rete, garantendo backup automatici e ridondanza dei dati.

Il **NAS** è posizionato all'interno della rete interna, protetto dal firewall e accessibile solo da utenti autorizzati. Inoltre, utilizza protocolli di sicurezza come **SMB** e **NFS** per garantire una comunicazione sicura tra i dispositivi.

Conclusione

La progettazione di questa rete garantisce:

- **Sicurezza avanzata**, grazie al firewall, alla **DMZ** e all'**IPS**.
- **Prestazioni ottimizzate**, con una suddivisione efficiente della rete in livelli.
- **Affidabilità e continuità operativa**, grazie ai router configurati in modalità backup.
- **Scalabilità**, permettendo future espansioni senza compromettere le prestazioni.
- L'architettura implementata assicura un'infrastruttura di rete solida, protetta e capace di supportare le esigenze aziendali nel lungo termine.

PROGRAMMAZIONE

Programmazione: Python

Per testare le vulnerabilità dell'azienda abbiamo effettuato:

- la verifica dei verbi **HTTP**;
- la scansione delle **porte**.

Specifichiamo nel dettaglio cosa sono effettivamente i dati presi in oggetto:

- I verbi **HTTP** sono “comandi” che il protocollo **HTTP** definisce per specificare le azioni che un client (es: un browser) può eseguire su una risorsa (es: una pagina web).
- I **protocolli di rete** sono un insieme di regole che permettono ai dispositivi di comunicare in rete; sono linguaggi standardizzati di comunicazione tra dispositivi.

Ogni protocollo è assegnato (di default) ad una porta specifica sulla rete. Per effettuare i test sopra elencati, abbiamo scritto e sviluppato i rispettivi programmi in Python

PORT-SCANNER (scansione delle porte di rete)

```
import socket

target = input("Inserisci indirizzo IP da scansionare: ")
portrange = input("Inserisci il range delle porte (es. 5-200)")

input_corretto=False
while input_corretto==False:
    onlyOpen = input("Vuoi solo porte aperte? (y/n): ").lower().strip()
    if onlyOpen == 'y' or onlyOpen == 'n':
        input_corretto=True

lowport = int(portrange.split("-")[0])
highport = int(portrange.split("-")[1])

print("Scannerizzando: ",target, "da porta ",lowport, "a porta", highport)
```

Obiettivo: testare un server per capire quali porte sono in ascolto (aperte) sul dispositivo preso in esame. Questo ci consente di rilevare quali sono le porte vulnerabili (es: non crittografate) per implementare meccanismi di difesa efficaci.

- **Importazione libreria SOCKET** per creare un canale di comunicazione verso altri dispositivi.
- **Inserimento INPUT** di TARGET e PORTRANGE: per chiedere all'utente rispettivamente Indirizzo IP da scansionare e il range delle porte prese in esame.
- **Ciclo WHILE**: l'utente può scegliere se visualizzare l'elenco completo delle porte o soltanto le porte aperte.
- **Inserimento variabili LOWPORT e HIGHPORT** per definire gli estremi del range.

PORT-SCANNER (scansione delle porte di rete)

```
for port in range(lowport,highport):
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    status=s.connect_ex((target,port))

    if(onlyOpen == "y"):

        if status==0:
            print("**** Port: ",port," - OPEN ****")
    else:
        if status==0:
            print("**** Port: ",port," - OPEN ****")
        else:
            print("Port: ",port," - closed")

    s.close()
```

- **ciclo FOR**: permette di far scorrere le porte e, per ognuna di esse, tentare una connessione.
 - **s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)** : funzione che permette di creare e memorizzare nella variabile S il socket, utilizzando indirizzi di tipo IPv4 (AF_INET) e connessioni con protocollo TCP (SOCK_STREAM).
 - **status=s.connect_ex((target,port))** : tentativo di connessione e memorizzazione del risultato nella variabile STATUS.
-
- **Inserimento IF-ELSE**: controllo condizione ONLYOPEN==Y, che permette di stampare a schermo solo le porte aperte o l'elenco completo delle porte.

Le porte aperte vengono stampate solo se STATUS==0 (connessione andata a buon fine).

VERIFICA VERBI HTTP

VERIFICA VERBI HTTP

Con questo programma andiamo ad effettuare una richiesta al web server per sapere quali metodi HTTP vengono accettati.

Di seguito riportiamo i metodi (verbi) principali:

- **GET**: utilizzato per richiedere una risorsa al server.
- **PUT**: utilizzato per aggiornare una risorsa esistente o crearne una nuova.
- **POST**: utilizzato solo per creare una nuova risorsa. In alcuni casi viene utilizzato anche per avviare un'operazione.
- **DELETE**: utilizzato per eliminare una risorsa.
- **OPTIONS**: utilizzato per chiedere le opzioni di comunicazione per la risorsa indicata (per esempio il server può rispondere quali altri metodi sono applicabili alla risorsa).
- **HEAD**: ha la stessa funzione del GET, ma richiede solo i metadati (cioè l'intestazione HTTP).
- **TRACE**: utilizzato normalmente per il debug.

Per risorse si intende qualsiasi tipo di informazione richiesta (es: testo, pagina web, immagine, ecc...).

VERIFICA VERBI HTTP

```
import requests

host = ("http://" + input("ip host: "))
porta = input("porta (default 80):")

if porta == "":
    porta=80

url = host + ":" + str(porta)

risposta = requests.options(url)

try:
    risposta = requests.options(url, timeout=5)

    metodi_http = risposta.headers.get("Allow")
    if metodi_http:
        print("\nMetodi HTTP disponibili:", metodi_http)
    else:
        print("\nNessun metodo HTTP attivo.")

except requests.RequestException as e:
    print("\nErrore nella richiesta:", e)
```

Questa è la prima versione del codice utilizzato per rilevare i metodi HTTP visibili e accettati.

Eseguendolo una prima volta, il programma non rilevava alcun metodo HTTP attivo.

Abbiamo perciò effettuato con un tentativo da terminale, utilizzando il seguente comando:

"curl -i -X OPTIONS http://192.168.40.101:80/".

In questo caso il programma rispondeva con **STATUS 200**, che equivale all'accettazione del metodo (OPTIONS).

VERIFICA VERBI HTTP

Abbiamo fatto un ulteriore prova su **Burpsuite**, (programma utilizzato per intercettare pacchetti durante la comunicazione tra due dispositivi) riscontrando che alla richiesta OPTIONS di alcune pagine (phpMyAdmin, DVWA), il programma ci reindirizzava automaticamente all'index e restituiva come risposta STATUS 200: **in sintesi il programma ha ignorato il metodo pur restituendolo come accettato.**

Analizzando la problematica, abbiamo provato ad aggirare l'ostacolo: abbiamo ispezionato la pagina phpMyAdmin, cercando un path(percors) più in profondità, per by-passare l'index.

In conclusione abbiamo deciso di importare nel codice **Beautifulsoup**, libreria che semplifica l'estrazione di informazioni dalle pagine web.

VERIFICA VERBI HTTP

Di seguito il codice revisionato:

```
> import requests ...
>
> def get_html_soup(link): ...
>
> def riempi_scelte(link): ...
> def stampa_menu(scelte): ...
>
> def try_connection(url): ...
>
> def try_bypass(url): ...

home_meta=(f"http://{input("Inserisci l'ip del dispositivo da verificare: ")}")
scelte=riempi_scelte(home_meta)

url = stampa_menu(scelte)
try_connection(url)
```

Questo codice ha il compito di **analizzare una pagina web per trovare tutti i collegamenti** (link) presenti.

Successivamente, **permette all'utente di selezionare uno di questi collegamenti per inviare una richiesta** HTTP di tipo OPTIONS all'URL scelto.

Infine, **tenta di rilevare i metodi HTTP supportati dall'URL.**

Trattandosi di un codice abbastanza elaborato e lungo, abbiamo deciso di utilizzare più funzioni, in modo da rendere il risultato finale più ordinato e comprensibile.

Andiamo adesso a spiegare nel dettaglio ogni funzione:

VERIFICA VERBI HTTP

```
import requests
from bs4 import BeautifulSoup

def get_html_soup(link):
    html=requests.get(link)
    soup = BeautifulSoup(html.content, "html.parser")
    return soup

def riempi_scelte(link):
    soup=get_html_soup(link)
    links=soup.find_all('a', href=True)
    scelte= {}
    for i in links:
        scelte.update({i.text.lower() : i['href']})
    return scelte

def stampa_menu(scelte):
    count=1
    for s in scelte:
        print(f"{count}. {s}")
        count +=1
    key = input("Scegli il nome del percorso: ").lower().strip()
    scelta = scelte[key]
    host = home_meta + scelta
    print(host)

    url = host
    return url
```

- **get_html_soup(link)**: Questa funzione prende il link fornito, ne estrae il contenuto HTML e lo analizza con BeautifulSoup.
- **riempi_scelte(link)**: Questa funzione utilizza l'HTML analizzato per trovare tutti i collegamenti (tag <a>) nella pagina, memorizza il testo come chiavi, e gli attributi "href" come valori in un dizionario.
- **stampa_menu(scelte)**: Questa funzione stampa un menu numerato dei collegamenti raccolti. Chiede all'utente di sceglierne uno per nome e costruisce l'URL completo per il collegamento selezionato.

VERIFICA VERBI HTTP

```
def try_connection(url):
    try:

        risposta = requests.options(url, timeout=5)

        metodi_http = risposta.headers.get("Allow")
        if metodi_http:
            print("\nMetodi HTTP disponibili:", metodi_http)
        else:
            try_bypass(url)
    except requests.RequestException as e:
        print("\nErrore nella richiesta:", e)

def try_bypass(url):
    print("\nNessun metodo HTTP attivo o protezione attiva.")
    print("\nTentativo di bypass:")
    soup=get_html_soup(url)
    percorso_im=soup.img['src']
    url = url + percorso_im[2:]
    print("\nTentando sul l'url: ",url)
    risposta = requests.options(url, timeout=5)
    metodi_http = risposta.headers.get("Allow")
    if metodi_http:
        print("\nTentativo di bypass avvenuto con successo!")
        print("\nMetodi HTTP disponibili:", metodi_http)
    else:
        print("\nTentativo di bypass fallito o nessun metodo HTTP permesso")
```

- **try_connection(url):** Questa funzione prova a inviare una richiesta OPTIONS all'URL dato. Se riesce, stampa i metodi HTTP disponibili. In caso contrario, tenta un bypass.
- **try_bypass(url):** Questa funzione tenta di accedere all'immagine all'URL dato e riprova la richiesta OPTIONS.

ANALISI DELLE VULNERABILITÀ

Analisi delle vulnerabilità

Dalle analisi precedentemente svolte sono emerse le seguenti possibili problematiche:

- **Verbi HTTP** (phpmyadmin, twiki, mutillidae, dvwa):
 - **PUT, POST, DELETE** (si consiglia di proteggere con autenticazione)
 - **TRACE, OPTIONS** (si consiglia di disabilitare)
- **Verbi HTTP** (webdav):
 - Valgono le stesse considerazioni precedentemente elencate
 - (si consiglia di disabilitare tutti gli altri metodi attivi (**PROPFIND, PROPPATCH, COPY, MOVE, LOCK, UNLOCK**)

Analisi delle vulnerabilità

Risultato scansione porte di rete:

- **FTP e HTTP**: si consiglia la sostituzione con protocolli più sicuri, che garantiscono la crittografia dei dati (**FTPS - HTTPS**)
- Utilizzare la **porta 22 (SSH)** al posto della **porta 23 (TELNET)**
- Si consiglia di non utilizzare **NETBIOS** e **SAMBA**, in quanto trasmettono i dati in chiaro
- Si consiglia, in ogni caso, di disabilitare tutte le porte non utilizzate e spostare le porte indispensabili per l'azienda (es. se l'azienda utilizza il protocollo **TELNET**, possiamo spostarlo sulla **porta 4444**, anziché tenerlo sulla **porta 23**, di default)

CONNESSIONI CON SOCKET TCP

CONNESSIONI CON SOCKET TCP

Spieghiamo brevemente il funzionamento di questo programma:

```
import socket

ip_server = str(input("Inserisci l'IP del server: "))
port_server = input("Inserisci la porta del server: ")

socket_address = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket_address.bind((ip_server, int(port_server)))
socket_address.listen(1)

print("Il server e' in ascolto")
connection, ip_client = socket_address.accept()
print("Client connesso con IP", ip_client)

while 1:
    data = connection.recv(1024)
    if not data: break
    print("Dati ricevuti")
    print(data.decode('utf-8'))
connection.close()
```

- **Importazione della libreria socket:** fornisce le funzionalità necessarie per creare un server e stabilire connessioni con i client.
- **ip_server = str(input("Inserisci l'IP del server: "))** **port_server = input("Inserisci la porta del server: ")**: l'utente è chiamato a inserire manualmente l'IP del server e la porta su cui questo deve ascoltare le connessioni.
- **socket_address = socket.socket(socket.AF_INET, socket.SOCK_STREAM)**: creazione di un socket di tipo AF_INET (protocollo IPv4) e di tipo SOCK_STREAM (socket TCP) utilizzato per connessioni affidabili e orientate alla connessione. Stabilisce il tipo di comunicazione che il server utilizzerà.
- **socket_address.bind((ip_server, int(port_server)))**: il server collega il socket all'indirizzo IP e alla porta, specificati in precedenza.

CONNESSIONI CON SOCKET TCP

Spieghiamo brevemente il funzionamento di questo programma:

```
import socket

ip_server = str(input("Inserisci l'IP del server: "))
port_server = int(input("Inserisci la porta del server: "))

socket_address = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket_address.bind((ip_server,int(port_server)))
socket_address.listen(1)

print("Il server e' in ascolto")
connection, ip_client = socket_address.accept()
print("Client connesso con IP",ip_client)

while 1:
    data = connection.recv(1024)
    if not data: break
    print("Dati ricevuti")
    print(data.decode('utf-8'))
connection.close()
```

- **socket_address.listen(1)**: il server viene impostato in modalità ascolto, il che significa che attende richieste di connessione da parte di client. Il numero 1 tra parentesi indica che il server accetterà una connessione alla volta.

Quando una connessione viene stabilita, la funzione accept() restituisce:

- connection, che rappresenta il canale di comunicazione con il client.
- ip_client, che contiene l'indirizzo IP del client connesso.

CONNESSIONI CON SOCKET TCP

Esempio pratico del funzionamento:

```
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Code/python]
$ python connessione_socket.py
Inserisci l'IP del server: 0.0.0.0
Inserisci la porta del server: 5000
Il server e' in ascolto
Client connesso con IP ('127.0.0.1', 38122)
Dati ricevuti
Ciao ti ho fregato

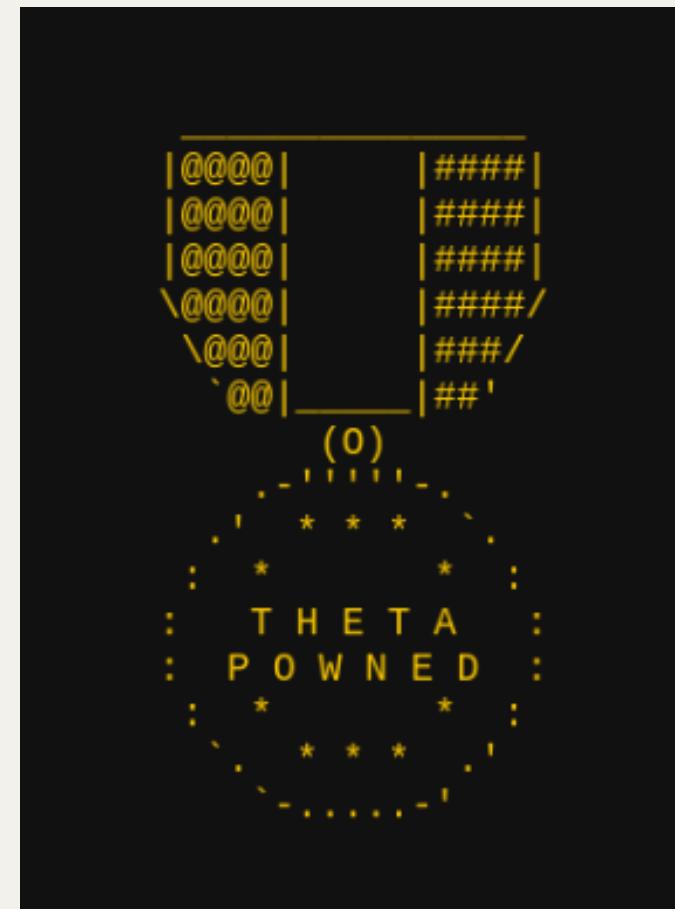
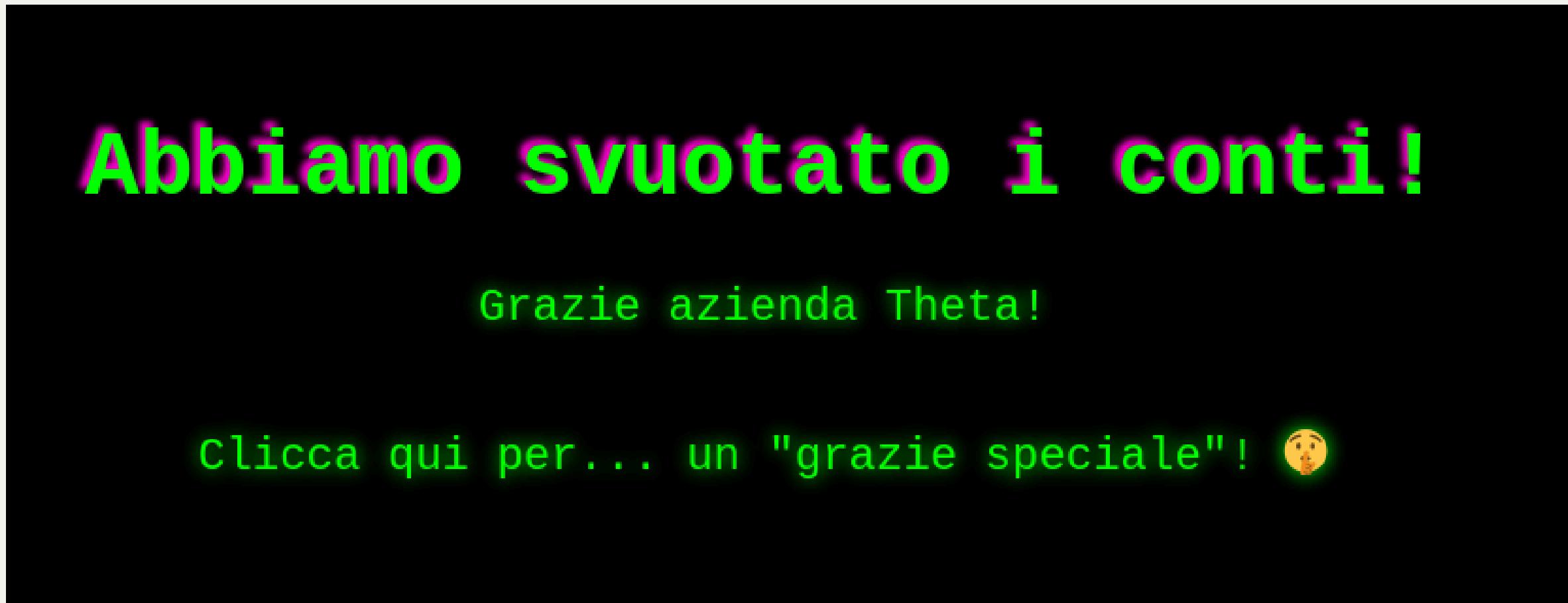
(kali㉿kali)-[~/Desktop/Code/python]
$ netcat 0.0.0.0 5000
Ciao ti ho fregato

```

RISOLUZIONE PROBLEMI SORTI DURANTE IL PROGETTO

Analisi file sospetto

Il 19 febbraio 2025, l'azienda **Theta** ha segnalato la presenza di un file sospetto. Questo file conteneva immagini, in cui erano stati occultati dati protetti da password ed era costituito da **enigmi sequenziali**. Una volta superati tutti gli step siamo arrivati a conoscenza di un possibile **attacco** ai conti dell'azienda.



Tecniche principali utilizzate per la decodifica: **Steghide, Conversioni in Base64**

DISPOSITIVI

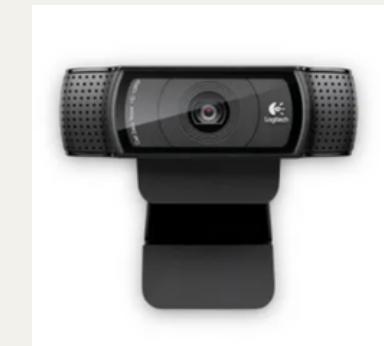
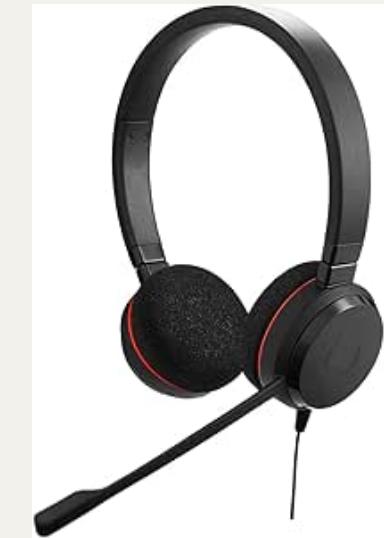
Postazione di lavoro

- **Computer:** LENOVO IDEACENTER, PC 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.
- **Monitor:** S31C Flat, 24", progettato per garantire un'ottima qualità dell'immagine e un'elevata affidabilità, ideale per ambienti professionali e uffici. Pensato per chi necessita di un display affidabile per la produttività, con colori accurati e un design ergonomico che favorisce il comfort durante l'uso prolungato.
- **Mouse e Tastiera:** Kit Wired HP 225, la tastiera cablata USB è progettata per un uso prolungato, con tasti durevoli, design elegante e prova di schizzi. Mouse semplice da configurare e utilizzare; non è necessario installare alcun software; grazie al cavo USB, puoi collegare il cavo e utilizzare il mouse del computer



Postazione di lavoro

- **Cuffie:** JEBRA EVOLVE, cuffie professionali progettate per migliorare la produttività e la qualità delle comunicazioni in ambienti di lavoro, specialmente per chi utilizza frequentemente chiamate VoIP e videoconferenze, ideali per professionisti che lavorano in ambienti dinamici e hanno bisogno di cuffie affidabili per comunicazioni chiare e senza distrazioni.
- **Webcam:** LOGITECH C920, una delle webcam più popolari e affidabili per videoconferenze, streaming e creazione di contenuti. È apprezzata per la sua qualità video, l'audio integrato e la compatibilità con molte piattaforme.



Dispositivi di rete

- **Switch:** SWITCH CISCO 2960, switch di livello enterprise progettato per reti aziendali, offrendo affidabilità, sicurezza e facilità di gestione. È ideale per reti LAN di imprese, uffici e ambienti industriali.
- **Router:** ROUTER CISCO MERAKI MX85, 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.



Dispositivi di rete

- **IDS/IPS:** Il Cisco Firepower 1140 è un dispositivo progettato per fornire una protezione robusta per la rete che offre funzionalità avanzate di Intrusion Detection System (IDS) e Intrusion Prevention System (IPS). Come IDS, monitora il traffico di rete per identificare attività sospette o potenzialmente dannose, generando avvisi quando vengono rilevati eventi anomali. Come IPS, il dispositivo non solo rileva le minacce, ma può anche intervenire attivamente per bloccare o mitigare gli attacchi in tempo reale, proteggendo così la rete da potenziali compromissioni.
- **Firewall:** Il Cisco ASA 5506-X è un firewall di nuova generazione (NGFW) progettato per garantire sicurezza avanzata a piccole e medie imprese o filiali aziendali. Fa parte della serie Cisco ASA (Adaptive Security Appliance) e integra il sistema FirePOWER per una protezione completa contro le minacce informatiche. Soluzione affidabile e potente per aziende che necessitano di protezione avanzata della rete, con funzionalità di firewall, IPS e VPN, il tutto in un dispositivo compatto e facile da gestire.



Dispositivi di rete

- **NAS:** NAS Synology DiskStation® DS1823xs+, 8 alloggiamenti per unità HDD, JBOD SATA multi-corsia 6Gb/s, Supporta QNAP NAS, Windows® e PC Ubuntu®, I vassoi bloccabili impediscono la rimozione accidentale del disco rigido. Il cavo di interconnessione SFF-8088 (o SFF-8644) supporta quattro canali SATA da 6 Gb/s per cavo (24 Gb/s in totale) e garantisce velocità di trasferimento dati più elevate rispetto a eSATA e USB.
- **WEBSERVER:** CISCO UCS C220 M7, server rack di ultima generazione progettato per offrire alte prestazioni, scalabilità e flessibilità in ambienti data center, cloud e AI/ML. Fa parte della famiglia Cisco Unified Computing System (UCS), ottimizzata per la virtualizzazione, il cloud computing e i carichi di lavoro aziendali più esigenti. soluzione affidabile per aziende che cercano prestazioni di alto livello e gestione centralizzata per i loro carichi di lavoro IT più critici.



Manodopera

Manodopera operai ed elettricisti:

Per l'allestimento della rete presso la sede della compagnia Theta ci siamo rivolti all'azienda Devitalia Telecomunicazioni, certificata ISO 9001 e ISO 27001. L'azienda segue inoltre gli standard ISO/IEC 11801 per l'allestimento della rete, garantendo un'infrastruttura conforme alle migliori pratiche internazionali in termini di cablaggio strutturato e prestazioni di rete.



Clicca qui per visualizzare il preventivo