

Build Week I

Progetto di Rete per la compagnia Theta

Specifiche del Progetto

Data la traccia:

Progetto di Rete per la Compagnia Theta

Specifiche del Progetto

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)

Progetto di Rete per la Compagnia Theta

Rete Interna Aziendale

- **Switch per ogni piano:** Collegare i 20 computer di ciascun piano a uno switch dedicato.
- **Router:** Collegare tutti gli switch dei vari piani a un router centrale.
- **Firewall:** Posizionare il firewall perimetrale tra il router interno e la connessione a Internet.
- **NAS:** Collegare il NAS allo switch al piano terra (vicino al router) per garantire l'accesso ai dati da parte di tutti i computer aziendali.
- **IDS/IPS:** Implementare 3 IDS/IPS nel perimetro interno per monitorare il traffico di rete e prevenire intrusioni.

Rete Esterna (Internet)

- **Connessione a Internet:** Collegare il firewall perimetrale a Internet.
- **Web Server:** Posizionare il web server (DVWA di Metasploitable) nella zona demilitarizzata (DMZ) tra il firewall e la connessione a Internet, garantendo così un accesso sicuro dall'esterno .
- Se avete bisogno di un altro firewall potete comprarlo e montarlo.

Testing della Rete

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

1. **Verifica dei Verbi HTTP:** Scriveremo 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.

Report Finale

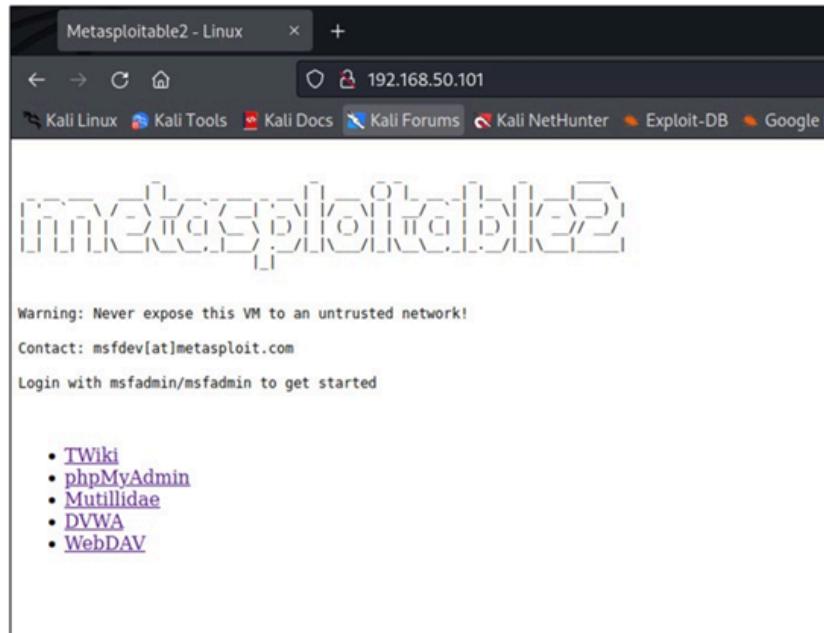
Alla conclusione dei test, redigeremo un report dettagliato che includerà:

- **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.

Questo approccio garantirà che l'infrastruttura di rete della compagnia Theta sia ben progettata, sicura e pronta per operare in modo efficiente .

Linee guida per la risoluzione del progetto:

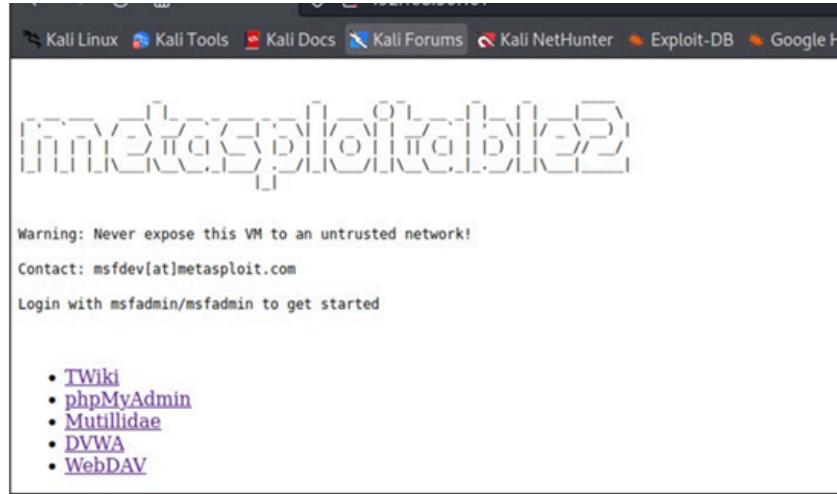
- Il Web Server di Theta verrà simulato dalla macchina Metasploitable
- Per la scansione dei servizi, non è possibile utilizzare i tool già pronti. Lo studente dovrà dunque scrivere in linguaggio Python un scanner che accetti in input un IP ed un range di porte da scansionare (la programmazione fa parte dell'esercizio). L'output del programma dovrà essere una lista delle porte con relativo stato (aperta/chiusa).
- La macchina Metasploitable espone un servizio web sulla porta 80, potete controllarlo da interfaccia grafica digitando l'indirizzo della vostra Metasploitable nella barra del browser da Kali (assicuratevi che ci sia comunicazione tra la macchina Kali e Metasploitable prima, Kali deve essere in grado di «pingare» la macchina Metasploitable).



Linee guida per la risoluzione del progetto:

Cliccate su phpMyAdmin e controllate quali verbi HTTP sono abilitati utilizzando un programma in Python anche in questo caso scritto da voi (la programmazione del tool fa parte dell'esercizio).

Il tool dovrà prendere input un path e dare in output come risultato una lista dei metodi abilitati su quel determinato path.



Bonus:

1. Eseguire il subnetting per scegliere la subnet più appropriata per la rete .
2. Creare un programma in python che catturi il socket di rete.

Struttura dell'edificio e della rete

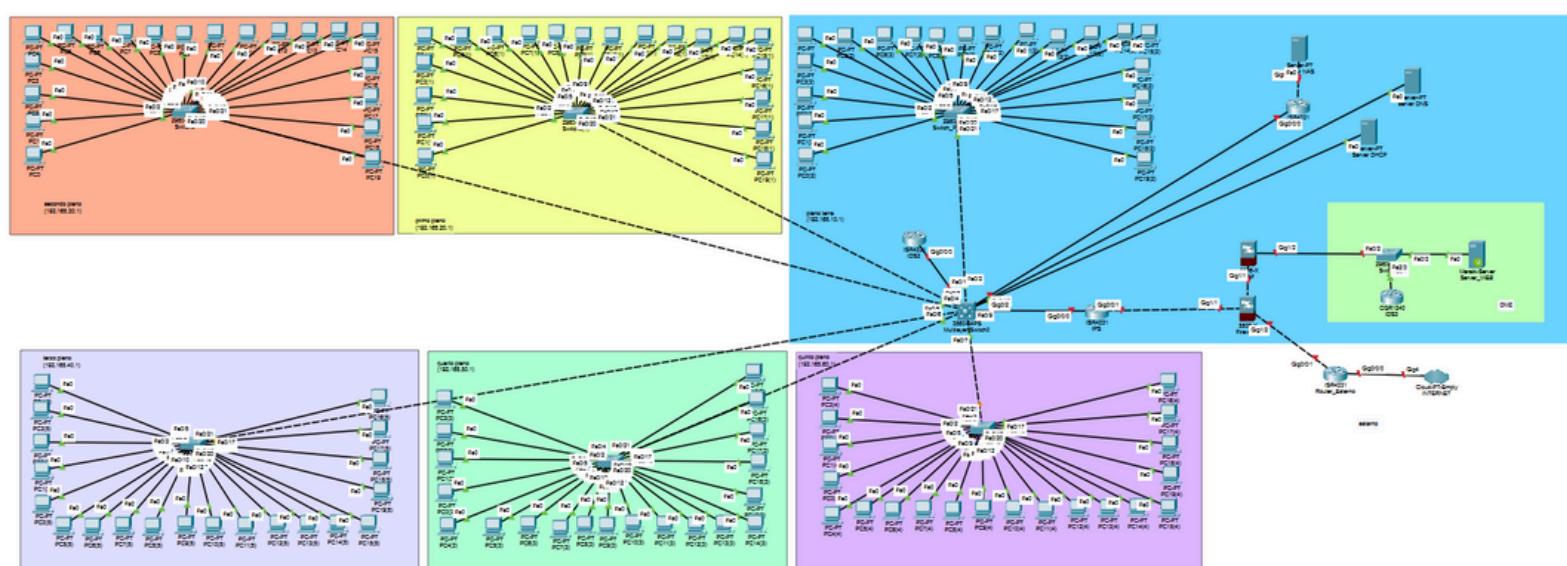
Come proposto dalla traccia, andremo a progettare una rete per la Compagnia “Theta”, costituita da un palazzo di 6 piani, dove in ognuno di essi troveremo 20 PC (120 in totale), collegati tramite uno switch che a sua volta sarà collegato al Router centrale.

Partendo da questo presupposto, ci siamo immaginati la struttura della rete in questo modo: troveremo il Router centrale al piano terra che instraderà i dati verso gli switch dei vari piani, fino ad arrivare ai PC.

Al piano terra abbiamo deciso di posizionare gran parte della nostra componente hardware, più nello specifico, nella nostra **DMZ** (zona demilitarizzata) abbiamo inserito:

- Il *Web Server* (garantendo così un accesso sicuro dall'esterno).
- Un dispositivo *IDS* per monitorare il traffico.
- Uno *switch* per aver il tutto collegato correttamente.

Quando abbiamo iniziato a pensare all'architettura della rete, la componente sicurezza ha avuto la priorità, a dimostrazione di ciò, abbiamo predisposto e seguito fin da subito e dalla base le prime norme e implementazioni per avere una protezione della rete concreta ed affidabile: come noteremo dall'immagine che riporterò di seguito, al piano terra abbiamo un Firewall, un WAF, un IPS e due IDS. IL Firewall è fondamentale per coprire la zona perimetrale dell'edificio, un IDS è posizionato nella DMZ e controlla il traffico di rete attraverso la tecnica del port mirroring mentre il secondo IDS è invece posizionato all'interno della rete aziendale. L'IPS è posizionato tra il Firewall e lo switch centrale.



La rete è composta da 120 PC suddivisi su 6 diversi piani, con 20 PC per piano e collegati ad un server DHCP dedicato per assegnare dinamicamente gli indirizzi IP ai dispositivi connessi, oltre a fornire una default gateway per ciascun PC. Questa

default gateway consente ai dispositivi di ogni piano di raggiungere il dispositivo che opera a livello 3 del modello ISO/OSI(switch centrale) e di comunicare con i dispositivi situati sugli altri piani.

Per collegare tutti i piani e ottimizzare la segmentazione della rete, sono stati utilizzati 6 switch di accesso, ciascuno posizionato su un piano. Gli switch sono connessi a uno switch centrale, che funge da punto di aggregazione della rete. Tale topologia garantisce una rete stabile, scalabile e facilmente gestibile.

Per migliorare l'efficienza della rete e isolare il traffico tra i piani, abbiamo implementato 6 VLAN (Virtual LAN), una per ogni piano. La segmentazione tramite VLAN permette di:

Isolare il traffico locale per ridurre le collisioni e migliorare le prestazioni.

Garantire maggiore sicurezza, isolando il traffico di ciascun piano.

Semplificare la gestione della rete.

Le VLAN sono state configurate nel seguente modo:

VLAN 10: Piano terra (IP range: 192.168.10.0/24)

VLAN 20: Primo piano (IP range: 192.168.20.0/24)

VLAN 30: Secondo piano (IP range: 192.168.30.0/24)

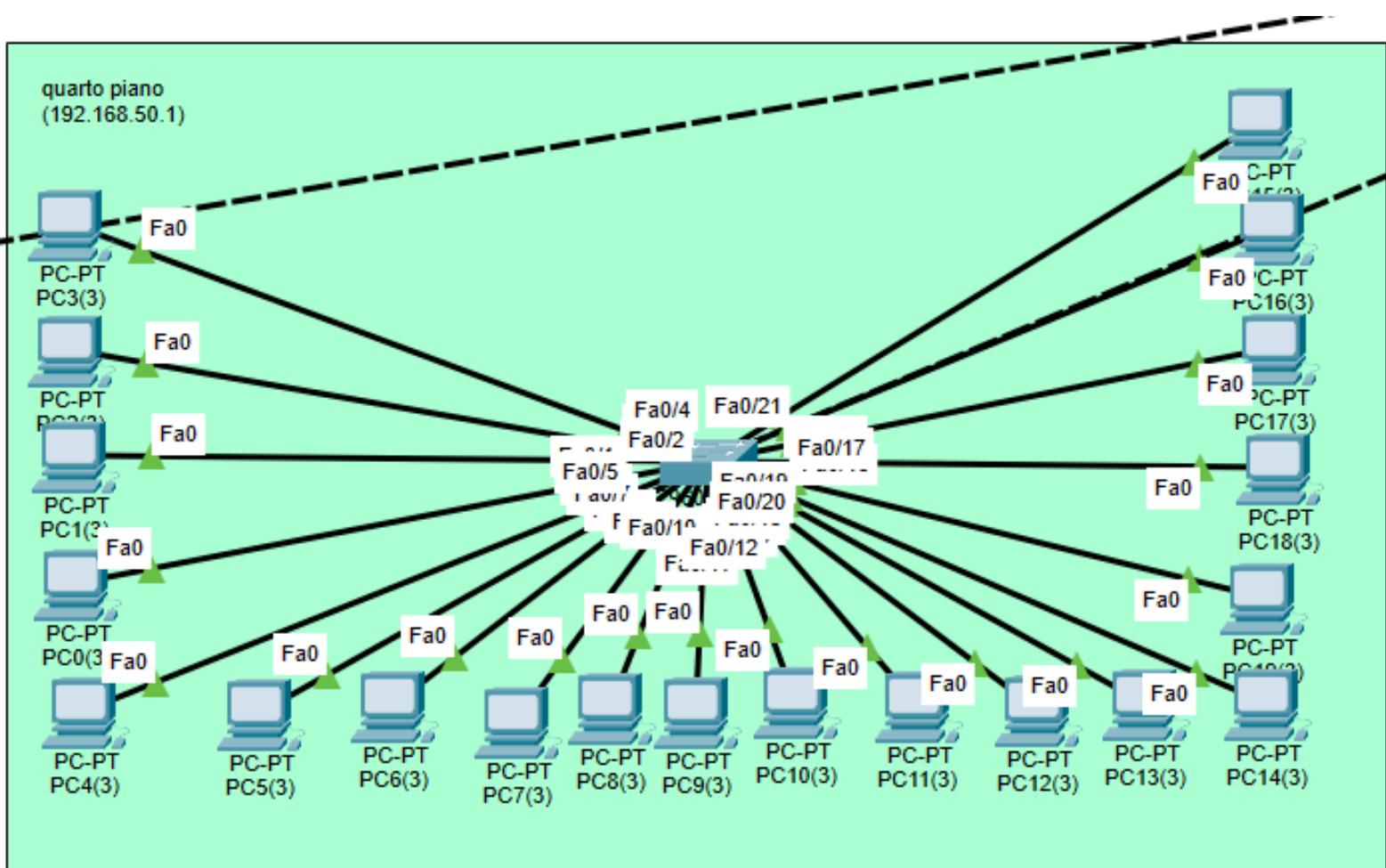
VLAN 40: Terzo piano (IP range: 192.168.40.0/24)

VLAN 50: Quarto piano (IP range: 192.168.50.0/24)

VLAN 60: Quinto piano (IP range: 192.168.60.0/24)

Per far comunicare le VLAN tra loro e con il router centrale, si potrebbe implementare il metodo Router on a Stick. Questo metodo prevede che il router abbia una singola interfaccia fisica (collegata allo switch centrale) su cui vengono configurate sotto interfacce virtuali, ciascuna dedicata a una VLAN specifica. Nel nostro caso c'è lo switch di livello 3 (multilayer) che funge da Router on a Stick .

Esempio predisposizione di un singolo piano:



VLAN

The screenshot shows the 'VLAN Configuration' page of the Switch2 web interface. On the left, a vertical navigation menu lists several categories: GLOBAL, Settings, Algorithm Settings, SWITCHING, VLAN Database (which is selected and highlighted in blue), INTERFACE, FastEthernet0/1, FastEthernet0/2, FastEthernet0/3, FastEthernet0/4, FastEthernet0/5, FastEthernet0/6, and FastEthernet0/7. The main right-hand panel is titled 'VLAN Configuration' and contains fields for 'VLAN Number' and 'VLAN Name'. Below these fields is a table listing existing VLANs:

VLAN No	VLAN Name
10	piano-terra
20	primo-piano
30	secondo-piano
40	terzo-piano
50	quarto-piano
60	quinto-piano

At the bottom right of the table are 'Add' and 'Re' buttons.

Posizionamento IDS/IPS

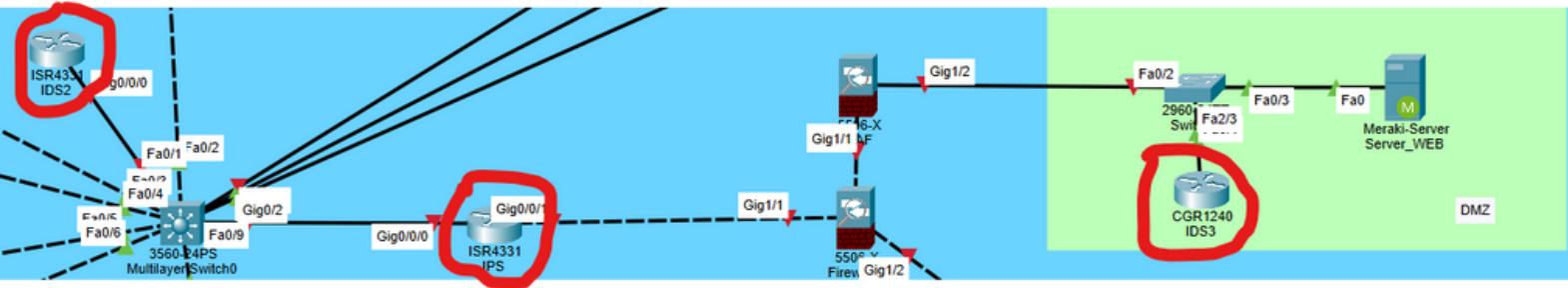
Riprendendo il discorso lato sicurezza, abbiamo optato per l'implementazione di un IPS e due IDS.

L'IDS ha il compito di monitorare il traffico di rete e segnalare eventuali comportamenti anomali o potenziali attacchi, ma non blocca il traffico. Per questo motivo, va posizionato in punti strategici della rete dove può analizzare una grande quantità di traffico.

L'IPS non solo rileva le minacce ma ha anche il compito di bloccare il traffico dannoso. Per questo motivo, va posizionato

in linea lungo il flusso del traffico, in modo da poter interrompere le connessioni sospette.

A dimostrazione di quanto appena enunciato abbiamo infine deciso di posizionare i dispositivi nel seguente modo:



Come possiamo constatare dall'immagine abbiamo usato la tecnica del “Port Mirroring” per usufruire al massimo dell’efficienza dell’IDS, mi spiego meglio:

Il port mirroring è una tecnica di rete che consente di duplicare il traffico di rete in entrata e/o in uscita da una o più porte di uno switch a un'altra porta specifica. Questo permette di monitorare e analizzare il traffico senza interrompere il normale flusso di dati.

Abbiamo quindi adottato questa tecnica per alimentare sistemi di rilevamento e prevenzione delle intrusioni. L’IDS, collegato allo switch in questo modo, non darà nessun problema di latenza, nel momento in cui lo switch farà il duplicato del pacchetto e lo manderà anche all’IDS avremo un controllo migliore e il traffico di rete rimarrà il medesimo.

A differenza dell’IDS, l’ISP ha maggior efficienza se viene utilizzato in serie, per questo motivo abbiamo deciso di metterlo tra lo Switch centrale e il Firewall, così da avere maggiore

controllo ed eventualmente blocco di minacce che potrebbero insediarsi nella nostra rete.

SCHEMA SEMPLIFICATO DELLA RETE



Preventivo di spesa della Rete

Articolo	Modello	Quantità	Costo	Riserva	Totale
Switch	Cisco WS-C2960S-48FPS-L	8	145	1	1305
Switch Centrale	Cisco C9200-24P-A	1	899	0	899
Router Centrale	Cisco ISR4331-V/K9	1	400	0	400
Routers	Cisco ISR4331-SEC/K9	5	150	1	900
Firewall	Cisco FPR1010-ASA-K9	2	585	1	1755
NAS	Synology DS1823xs	1	2206	0	2206
Server Web	Server HPE ProLiant DL380 Gen10	1	2.975	0	2.975
Gruppo di continuità	Eaton 9PX2200IRTPB	1	1932	0	1932
					12372

Il preventivo di spesa ammonta a 12.372,00€.

Come si puo' notare dalla tabella abbiamo inserito:

- 8 Switch da 145,00€ l'uno (tenendone 1 di riserva). Possiamo trovarli a questo link: <https://it-planet.com/it/p/cisco-ws-c2960s-48fps-l-1597.html?c=140>
- 1 Switch centrale da 899,00€ . Possiamo trovarlo a questo link: <https://it-planet.com/it/p/cisco-c9200-24p-a-216241.html?c=2884>
- 1 router centrale da 400,00€. Possiamo trovarlo a questo link: <https://it-planet.com/it/p/cisco-isr4331-v/k9-856.html?c=2929>
- 5 dispositivi tra Router, ISP, IDS da 150,00€ (tenendone 1 di riserva). Possiamo trovarli a questo link: <https://it-planet.com/it/p/cisco-isr4331-sec/k9-855.html?c=2929>
- 2 Firewall da 585,00€ (tenendone 1 di riserva). Possiamo trovarli a questo link: <https://it-planet.com/it/p/cisco-fpr1010-asa-k9-382230.html?number=7830630000>
- 1 NAS da 2206,00€. Possiamo trovarlo a questo link: <https://www.amazon.it/Synology-DS1823xs-Hot-Swap-Processor-frequency/dp/B0BWJR33RQ>
- 1 Server Web da 2975,00€. Possiamo trovarlo a questo link: <https://buy.hpe.com/it/it/compute/rack-servers/proliant->

[dl300-servers/proliant-dl380-server/server-hpe-proliant-dl380-gen10/p/1010026818](https://www.dreamcompute.com/servers/dl300-servers/proliant-dl380-server/server-hpe-proliant-dl380-gen10/p/1010026818)

- 1 gruppo di continuità per i server da 1932,00€. Possiamo trovarlo a questo link: <https://it-planet.com/it/p/eaton-9px2200irtbp-208377.html>

TESTING DELLA RETE

Verifica dei verbi HTTP

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

Riporto di seguito lo screenshot del codice:

```
GNU nano 8.1
import requests

def invio_richiesta(metodo, url, data=None):
    try:
        risposta = requests.request(metodo, url, json=data, timeout=5)
        print(f'{metodo.upper()} {url} - Status Code: {risposta.status_code}')
        return risposta
    except requests.exceptions.ConnectionError:
        print(f'{metodo.upper()} {url} - Connessione rifiutata. Verifica che il server sia attivo.')
    except requests.exceptions.Timeout:
        print(f'{metodo.upper()} {url} - Timeout. Il server non ha risposto in tempo.')
    except requests.exceptions.RequestException as e:
        print(f'{metodo.upper()} {url} - Errore durante la richiesta: {e}')
    return None

Home
descrizione_codice = {
    200: "OK",
    201: "Created",
    202: "Accepted",
    204: "No Content",
    301: "Moved Permanently",
    302: "Found",
    400: "Bad Request",
    401: "Unauthorized",
    403: "Forbidden",
    404: "Not Found",
    405: "Method Not Allowed",
    500: "Internal Server Error",
    502: "Bad Gateway",
    503: "Service Unavailable"
}

def inserisci_url():
    while True:
        url = input("Inserisci l'URL (es. www.esempio.com): ").strip()
        if not url.startswith(('http://', 'https://')):
            url = 'http://' + url

        try:
            if requests.head(url, allow_redirects=True, timeout=1).ok:
                print(f"URL valido e raggiungibile: {url}")
                return url
            else:
                print("URL non raggiungibile. Riprovare.")
        except requests.RequestException:
            print("Errore nel raggiungere l'URL. Riprovare.")
```

```

url= inserisci_url()

data = {
    'chiave': 'valore',
}

print("Inizio delle richieste HTTP\n")

get_response = invio_richiesta('get', url)
post_response = invio_richiesta('post', url, data=data)
put_response = invio_richiesta('put', url, data=data)
delete_response = invio_richiesta('delete', url)

print("\nRiepilogo delle risposte:")
risposte = {
    'GET': get_response,
    'POST': post_response,
    'PUT': put_response,
    'DELETE': delete_response
}

for metodo, risposta in risposte.items():
    if risposta is not None:
        code = risposta.status_code
        descrizione = descrizione_codice.get(code, risposta.reason)
    else:
        code = 'N/A'
        descrizione = 'N/A'
    print(f"{metodo} - Status Code: {code} ({descrizione})")

```

Come già spiegato sopra, questo codice servirà per ricevere come risposta, dall'URL che andremo ad inserire, i verbi HTTP che abbiamo scritto (GET, POST, PUT DELETE) e per vederne il loro stato. Grazie alla modalità con la quale è stato scritto, siamo riusciti a guidare l'utente nella digitazione dell'URL in modo tale che se dovesse sbagliare gli verrà notificato qual è l'errore e gli verrà ripresentata l'occasione di poter digitare nuovamente l'URL voluto. Abbiamo anche specificato per ogni codice di risposta il corrispettivo status preciso, ovvero, per esempio: "202" = "Accepted", "302" = "Found" e così via.

Riporto di seguito il risultato che ci darà il programma una volta messo in esecuzione:

```
(kali㉿kali)-[~/Desktop/EserciziPython]
$ python verbiaggiornato.py
Inserisci l'URL (es. www.esempio.com): 192.168.60.150/phpMyAdmin
URL valido e raggiungibile: http://192.168.60.150/phpMyAdmin
Inizio delle richieste HTTP

GET http://192.168.60.150/phpMyAdmin - Status Code: 200
POST http://192.168.60.150/phpMyAdmin - Status Code: 200
PUT http://192.168.60.150/phpMyAdmin - Status Code: 200
DELETE http://192.168.60.150/phpMyAdmin - Status Code: 200
```

Riepilogo delle risposte:

```
GET - Status Code: 200 (OK)
POST - Status Code: 200 (OK)
PUT - Status Code: 200 (OK)
DELETE - Status Code: 200 (OK)
```

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.

Riporto screenshot del codice scritto di seguito:

```
GNU nano 8.1
import socket
import re
import pandas as pd
import subprocess
import platform

def ottieni_ip_locali():
    ip_list = []
    hostname = socket.gethostname()
    try:
        ip_list = socket.gethostbyname_ex(hostname)[2]
    except Exception as e:
        print(f"Errore nell'ottenere gli IP locali: {e}")
    return ip_list

def verifica_raggiungibilita(ip):
    param = "-n" if platform.system().lower() == "windows" else "-c"
    comando = ["ping", param, "1", ip]
    try:
        esito = subprocess.run(comando, stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)
        return esito.returncode == 0
    except Exception as e:
        print(f"Errore durante la verifica dell'IP: {e}")
        return False

def portscanner_range(ip, porta_iniziale, porta_finale):
    risultato_scanner = {}
    for porta in range(porta_iniziale, porta_finale + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(0.5)
        risultato = sock.connect_ex((ip, porta))
        if risultato == 0:
            risultato_scanner[porta] = "Aperta"
        else:
            risultato_scanner[porta] = "Chiusa"
        sock.close()
    return risultato_scanner

def portscanner_lista(ip, porte):
    risultato_scanner = {}
    for porta in porte:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(0.5)
        risultato = sock.connect_ex((ip, porta))
        if risultato == 0:
            risultato_scanner[porta] = "Aperta"
        else:
            risultato_scanner[porta] = "Chiusa"
        sock.close()
```

```
GNU nano 8.1
    Trasporti
    sock.close()
    return risultato_scanner

def validazione_ip(ip):
    ottetti = ip.split(".")
    if len(ottetti) != 4:
        return False
    for ottetto in ottetti:
        if not ottetto.isdigit() or not 0 <= int(ottetto) <= 255:
            return False
    return True

def scelta_ip():
    ip_locali = ottieni_ip_locali()
    print("Indirizzi IP locali disponibili:")
    for i, ip in enumerate(ip_locali, 1):
        print(f"{i}. {ip}")

    while True:
        ip = input("\nInserisci l'indirizzo IP da scansionare: ").strip()
        if validazione_ip(ip):
            if verifica_raggiungibilita(ip):
                print("Hai inserito un IP valido e raggiungibile.")
                return ip
            else:
                print("L'indirizzo IP inserito non è raggiungibile.")
        else:
            print("L'indirizzo IP non è valido. Assicurati di usare il formato xxx.xxx.xxx.xxx.")

def controllo_porte(porta):
    if porta.isdigit():
        porta = int(porta)
        if porta <= 65535:
            return porta
        else:
            print("Il valore massimo di una porta è 65535.")
    else:
        print("Inserisci una porta valida!")

def scelta_porta_iniziale():
    porta_iniziale = input("Inserisci la porta di inizio: ").strip()
    porta_iniziale = controllo_porte(porta_iniziale)
    if porta_iniziale:
        return porta_iniziale
    return scelta_porta_iniziale()

def scelta_porta_finale(porta_iniziale):
    porta_finale = input("Inserisci la porta di fine: ").strip()
```

```

GNU nano 8.1
scanporte.py *

def scelta_porta_finale(porta_iniziale):
    porta_finale = input("Inserisci la porta di fine: ").strip()
    porta_finale = controllo_porte(porta_finale)
    if porta_finale:
        if porta_finale >= porta_iniziale:
            return porta_finale
        else:
            File System print("La porta di inizio non può essere maggiore della porta di fine!")
    return scelta_porta_finale(porta_iniziale)

def scelta_porte():
    lista_porte = input("Inserire le porte che si vogliono controllare lasciando uno spazio tra ogni porta: ").strip()
    lista_porte = re.sub(r"\d+", "", lista_porte)
    lista_porte = lista_porte.split(sep=" ")
    for i in range(len(lista_porte)):
        lista_porte[i] = controllo_porte(lista_porte[i])
        if not lista_porte[i]:
            print(f"Eliminata {i+1}° porta inserita perché non valida.")

    lista_porte = [porta for porta in lista_porte if porta is not None]
    lista_porte.sort()
    return lista_porte

ip = scelta_ip()
while True:
    scelta_funzionamento = input("Digitare 1 se si vuole controllare per un range di porte\nDigitare 2 per controllare per una lista di porte: ").strip()
    if scelta_funzionamento in ["1", "2"]:
        break
    print("Inserire un valore valido! (1 o 2)")

if scelta_funzionamento == "1":
    porta_iniziale = scelta_porta_iniziale()
    porta_finale = scelta_porta_finale(porta_iniziale)
    porte_risultato = portscanner_range(ip, porta_iniziale, porta_finale)
    print(f"\nPorte nel range {porta_iniziale} - {porta_finale}:")

elif scelta_funzionamento == "2":
    lista_porte = scelta_porte()
    porte_risultato = portscanner_lista(ip, lista_porte)
    print(f"Porte nella lista {lista_porte}:")

if porte_risultato:
    df = pd.DataFrame(list(porte_risultato.items()), columns=["Porta", "Stato"])
    print("\nLista Porte:\n")
    print(df.to_string(index=False))
else:
    print("Non sono state inserite porte valide!")

```

Abbiamo strutturato anche questo codice in modo tale che l'utente sia il più possibile guidato e che il codice stesso faccia quanti più controlli possibili in merito al come l'utente potrà andare ad immettere l'IP.

Più nello specifico, abbiamo inserito dei controlli per verificare gli IP locali, se l'IP è raggiungibile e se verrà scritto in modo corretto (formato da 4 ottetti di cui ognuno il valore massimo è 255).

Il codice è pensato per poter procedere in due modi: possiamo scegliere se andare a verificare le porte all'interno di un range (opzione 1) oppure se andare a fare un controllo solamente su determinate porte, quindi tramite una lista (opzione 2).

Ogni volta che dovesse presentarsi un errore, quindi per esempio se il valore digitato è superiore al limite massimo delle porte, o se viene scritto in modo errato, il nostro codice “stamperà” un errore a schermo per indicare all’utente che non l’ha digitato in modo corretto e quindi di tentare nuovamente.

Tramite “pandas” che abbiamo importato, ci presenterà la lista delle porte selezionate con il relativo stato all’interno di una matrice.

Riporto di seguito screenshot del risultato del programma una volta messo in esecuzione, per entrambe le opzioni disponibili:

```
(kali㉿kali)-[~/Desktop/EserciziPython]
$ python scanportultimaversione.py
INSERISCI L'INDIRIZZO IP DA SCANSIONARE: 192.168.60.150
Hai inserito un Ip Valido.
Digitare 1 Se Si Vuole Controllare Per Un Range Di Porte
Digitare 2 Per Controllare Per Una Lista Di Porte
1
INSERISCI LA PORTA DI INIZIO: 80
INSERISCI LA PORTA DI FINE: 100
Porte Nel Range 80 - 100:
Lista Porte:
Porta Stato
 80 Aperta
 81 Chiusa
 82 Chiusa
 83 Chiusa
 84 Chiusa
 85 Chiusa
 86 Chiusa
 87 Chiusa
 88 Chiusa
 89 Chiusa
 90 Chiusa
 91 Chiusa
 92 Chiusa
 93 Chiusa
 94 Chiusa
 95 Chiusa
 96 Chiusa
 97 Chiusa
 98 Chiusa
 99 Chiusa
100 Chiusa
```

```
(kali㉿kali)-[~/Desktop/EserciziPython]
$ python scanportesultimaversione.py
INSERISCI L'INDIRIZZO IP DA SCANSIONARE: 192.168.60.150
Hai inserito un Ip Valido.
Digitare 1 Se Si Vuole Controllare Per Un Range Di Porte
Digitare 2 Per Controllare Per Una Lista Di Porte
2
Inserire Le Porte Che Si Vogliono Controllare Lasciando Uno Spazio Tra Ogni Porta:
80 22 23 25 443
Porte Nella Lista[22, 23, 25, 80, 443]:
Lista Porte:

Porta  Stato
 22 Aperta
 23 Aperta
 25 Aperta
 80 Aperta
443 Chiusa
```

Raccomandazioni di sicurezza sulle porte

La presenza di porte aperte su un sistema può esporre a rischi di sicurezza, soprattutto se i servizi associati non sono configurati correttamente o presentano vulnerabilità note.

Di seguito espongo una panoramica delle porte elencate e le raccomandazioni di sicurezza per ciascuna:

```
Lista Porte:

Porta  Stato
 21 Aperta
 22 Aperta
 23 Aperta
 25 Aperta
 53 Aperta
 80 Aperta
111 Aperta
139 Aperta
445 Aperta
512 Aperta
513 Aperta
514 Aperta
```

RACCOMANDAZIONI

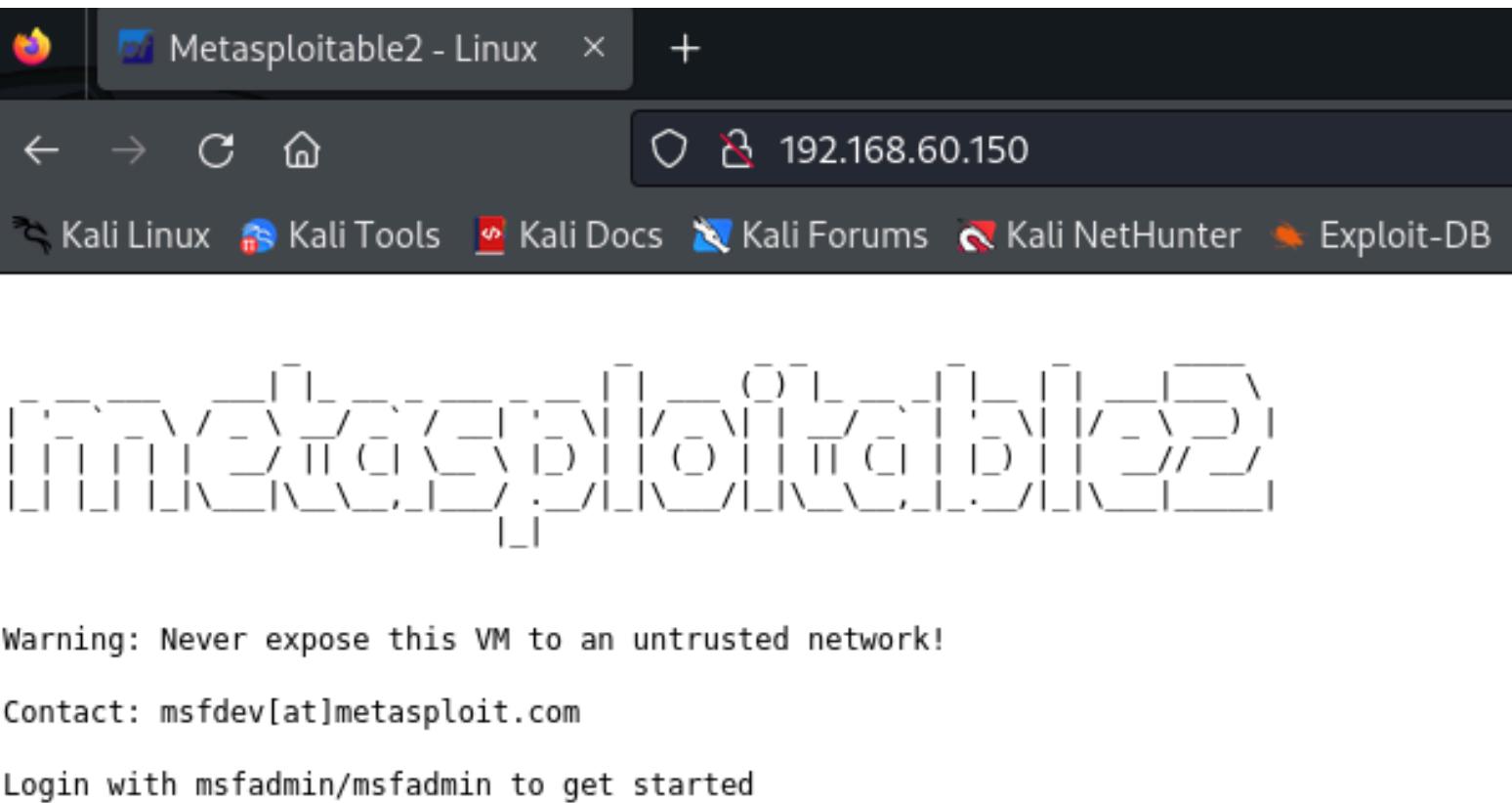
PORTE	FUNZIONAMENTO	RISCHI	RACCOMANDAZIONI
21 (FTP)	Usata per il trasferimento di file tra client e server	Trasferisce dati, incluse le credenziali, in chiaro, vulnerabile a intercettazioni (sniffing) e attacchi MITM (man-in-the-middle)	Usare SFTP o FTPS per una comunicazione crittografata
23 (TELNET)	Protocollo per connessioni remote su linea di comando	Comunicazioni in chiaro, vulnerabile a intercettazioni e attacchi di brute force	Sostituire Telnet con SSH, che offre crittografia sicura
25 (SMTP)	Utilizzata per l'invio di email	Può essere sfruttata per spam o spoofing (falsificazione dell'identità del mittente)	Configurare l'autenticazione e limitare l'accesso ai soli sistemi autorizzati
53 (DNS)	Risolve nomi di dominio in indirizzi IP	Vulnerabile ad attacchi di spoofing o amplificazione DDoS	Implementare DNSSEC e limitare richieste da fonti non attendibili
80 (HTTP)	Protocollo per il traffico web non crittografato	Comunicazioni in chiaro, vulnerabile a intercettazioni	Usare HTTPS (porta 443) per crittografia tramite TLS
111 (RPC bind)	Gestisce chiamate a procedure remote (Remote Procedure Call)	Possibile punto di ingresso per attacchi, inclusi DDoS e privilege escalation	Disabilitare se non necessario o filtrare l'accesso con un firewall
139 (NetBIOS)	Usata per condivisione di file e stampanti in reti Windows	Vulnerabile a attacchi di enumerazione e brute force	Disabilitare se non utilizzata o limitare l'accesso tramite firewall
445 (SMB)	Protocollo per la condivisione di file in rete	Frequentemente sfruttata in attacchi ransomware	Usare SMBv3 con crittografia e applicare patch di sicurezza
512 (RExec)	Permette l'esecuzione remota di comandi	Trasmette credenziali in chiaro, altamente insicura	Disabilitare e utilizzare protocolli sicuri come SSH
513 (RLOGIN)	Consente l'accesso remoto a sistemi Unix/Linux	Trasmissione in chiaro e vulnerabile a spoofing e attacchi MITM	Disabilitare e utilizzare protocolli sicuri come SSH
514 (RSH)	Consente l'esecuzione remota di comandi senza autenticazione avanzata	Comunicazioni in chiaro e altamente vulnerabile a intercettazioni	Disabilitare e utilizzare protocolli sicuri come SSH

Servizio web macchina Metasploitable

Come richiesto dalla traccia:

“La macchina Metasploitable espone un servizio web sulla porta 80, potete controllarlo da interfaccia grafica digitando l'indirizzo della vostra Metasploitable nella barra del browser da Kali (assicuratevi che ci sia una comunicazione tra la macchina Kali e Metasploitable prima, Kali deve essere in grado di «pingare» la macchina Metasploitable)”.

Nelle immagini che inseriremo successivamente potremo constatare come la macchina “Kali” riuscirà a comunicare correttamente attraverso il comando “ping” con la macchina Metasploitable, e nella seconda vedremo come visualizzare correttamente il servizio web sulla porta 80 tramite interfaccia grafica.



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

```
(kali㉿kali)-[~]
$ ping 192.168.60.150
PING 192.168.60.150 (192.168.60.150) 56(84) bytes of data.
64 bytes from 192.168.60.150: icmp_seq=1 ttl=63 time=1.38 ms
64 bytes from 192.168.60.150: icmp_seq=2 ttl=63 time=0.892 ms
64 bytes from 192.168.60.150: icmp_seq=3 ttl=63 time=2.21 ms
64 bytes from 192.168.60.150: icmp_seq=4 ttl=63 time=1.82 ms
64 bytes from 192.168.60.150: icmp_seq=5 ttl=63 time=2.17 ms
64 bytes from 192.168.60.150: icmp_seq=6 ttl=63 time=0.966 ms
64 bytes from 192.168.60.150: icmp_seq=7 ttl=63 time=1.21 ms
64 bytes from 192.168.60.150: icmp_seq=8 ttl=63 time=1.25 ms
64 bytes from 192.168.60.150: icmp_seq=9 ttl=63 time=1.50 ms
64 bytes from 192.168.60.150: icmp_seq=10 ttl=63 time=1.58 ms
^C
Contact: msfdev[at]metasploit.com
--- 192.168.60.150 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9022ms
rtt min/avg/max/mdev = 0.892/1.495/2.206/0.432 ms
```

Verbi HTTP abilitati su “phpMyAdmin”

Un altro metodo, oltre a quello utilizzato precedentemente (che prevedeva l’uso del programma in Python), per vedere quali verbi HTTP sono abilitati su “phpMyAdmin” è quello di andare sulla schermata “Privileges” attraverso il servizio web della macchina Metasploitable.

Possiamo fare riferimento alla prima immagine delle due precedentemente inserite. Cliccando su “phpMyAdmin” ci ritroveremo davanti a questa schermata:

phpMyAdmin

Server: localhost

Databases SQL Status VariablesCharsets Engines Privileges Processes Export Import

Actions

- Change password
- Log out

MySQL localhost

Create new database Collation Create MySQL connection collation: Collation

Interface

Language: English Theme / Style: Original Custom color: Reset Font size: 82%

MySQL

- Server: Localhost via UNIX socket
- Server version: 5.0.51a-3ubuntu5
- Protocol version: 10
- User: debian-sys-maint@localhost
- MySQL charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.2.8 (Ubuntu) DAV/2
- MySQL client version: 5.0.51a
- PHP extension: mysql

phpMyAdmin

- Version information: 3.1.1
- Documentation
- Wiki
- Official Homepage
- [ChangeLog] [Subversion] [Lists]

Andando a cliccare sulla voce “Privileges” in alto, potremo visualizzare quali sono i verbi HTTP abilitati, come dimostrato dallo screenshot di seguito:

192.168.60.150/phpMyAdmin/index.php?token=8a9e5af4d79011d29cf623bb07f2317d

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Server: localhost

Databases SQL Status VariablesCharsets Engines Privileges Processes Export Import

User overview

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
[Show all]																									
User	Host	Password	Global privileges ¹																				Grant		
Any	%	--	USAGE																				No		
debian-sys-maint		No	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, REPLICATION SLAVE, REPLICATION CLIENT, EXECUTE																				Yes		
guest	%	No	ALL PRIVILEGES																				Yes		
root	%	No	ALL PRIVILEGES																				Yes		

Check All / Uncheck All

SVILUPPO DEL PRIMO BONUS PYTHON:

Creare un programma in Python che catturi il socket di rete

Di seguito andremo a spiegare il codice che abbiamo scritto per catturare il socket di rete, innanzitutto presentiamo come è stato scritto il codice nelle immagini riportate successivamente:

```
GNU nano 8.1
import socket
import subprocess
import platform

def ottieni_ip_dispositivo():
    ip_list = []
    try:
        hostname = socket.gethostname()
        ip_list = socket.gethostbyname_ex(hostname)[2]
    except Exception as e:
        print(f"Errore nell'ottenere gli IP: {e}")
    return ip_list

def verifica_raggiungibilita(ip):
    param = "-n" if platform.system().lower() == "windows" else "-c"
    comando = ["ping", param, "1", ip]
    try:
        esito = subprocess.run(comando, stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)
        return esito.returncode == 0
    except Exception as e:
        print(f"Errore durante la verifica dell'IP: {e}")
    return False

def validazione_ip(ip):
    ottetti = ip.split(".")
    if len(ottetti) != 4:
        return False
    for ottetto in ottetti:
        if not ottetto.isdigit() or not 0 <= int(ottetto) <= 255:
            return False
    return True

def scelta_ip():
    ip_disponibili = ottieni_ip_dispositivo()
    print("Indirizzi IP disponibili:")
    for ip in ip_disponibili:
        print(f"- {ip}")

    while True:
        ip = input("\nInserisci l'indirizzo IP da utilizzare: ").strip()
        if validazione_ip(ip):
            if verifica_raggiungibilita(ip):
                print("Hai inserito un IP valido e raggiungibile.")
                return ip
            else:
                print("L'indirizzo IP inserito non è raggiungibile.")
        else:
            print("IP non valido o non raggiungibile.")
```

```

GNU nano 8.1
Trash  if verifica_raggiungibilita(ip):
        print("Hai inserito un IP valido e raggiungibile.")
        return ip
    else:
        print("L'indirizzo IP inserito non è raggiungibile.")
else:
    print("L'indirizzo IP non è valido. Assicurati di usare il formato xxx.xxx.xxx.xxx.")

File System
def controllo_porte(porta):
    if porta.isdigit():
        porta = int(porta)
        if porta <= 65535:
            return porta
        else:
            print("Il valore massimo di una porta è 65535.")
    else:
        print("Inserisci una porta valida!")

def scelta_porta():
    while True:
        porta = input("Inserisci la porta da utilizzare: ").strip()
        porta = controllo_porte(porta)
        if porta:
            return porta

Esercizi
def avvio_socket(ip, porta):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((ip, porta))
    s.listen(1)
    print("Server avviato! In attesa di connessioni ...")

    connection, address = s.accept()
    print("Client connesso con indirizzo:", address)

    while True:
        data = connection.recv(1024)
        if not data:
            break
        print(data.decode('utf-8'))

    connection.close()

ip = scelta_ip()
porta = scelta_porta()
avvio_socket(ip, porta)

```

All'interno di questo programma Python abbiamo inserito nuovamente la parte di codice di controllo delle porte che abbiamo già mostrato in quello precedente, che, in combinazione alla scelta di interfaccia disponibile e rilevamento dell'IP locale presente ci darà maggiore controllo sui pacchetti che andremo a registrare.

Inoltre, abbiamo inserito un'ulteriore scelta sia sulla quantità di pacchetti che andremo a registrare, andando a chiedere direttamente all'utente quanti ne vorrà includere, sia sul filtro del/dei Protocolli (TCP/UDP/ICMP) che interessano all'utente.

Esattamente come i precedenti programmi abbiamo deciso di avere un maggiore controllo, pertanto, sono presenti porzioni di codice dove notificheremo alla persona che andrà ad utilizzare il programma eventuali errori di digitazione o errori logici (per esempio, qualora dovesse inserire un numero non valido, comparirà a schermo la motivazione per cui dovrà inserirlo nuovamente, con spiegazione dedicata a questo caso che abbiamo contemplato e che faremo comparire appunto a schermo).

Mettendo in esecuzione il programma, quello che andremo a visualizzare, sarà questo:

```
(kali㉿kali)-[~]
└─$ cd Desktop/EserciziPython
Invalid port name '127.0.1.1'.
(kali㉿kali)-[~/Desktop/EserciziPython]
└─$ python socketbonus1.py
Indirizzi IP disponibili:
- 127.0.1.1...
Connected to 127.0.1.1.
Inserisci l'indirizzo IP da utilizzare: 127.0.1.1
Hai inserito un IP valido e raggiungibile. di funzionamento del programma Python.
Inserisci la porta da utilizzare: 80
Server avviato! In attesa di connessioni ...
Client connesso con indirizzo: ('127.0.0.1', 59516)
Ciao Epicode! Ecco la nostra dimostrazione di funzionamento del programma Python.
```

```
(kali㉿kali)-[~]nus1.py
└─$ telnet 127.0.1.1 80
Trying 127.0.1.1...
Connected to 127.0.1.1.
Escapes character is ^].da utilizzare: 127.0.1.1
Ciao Epicode! Ecco la nostra dimostrazione di funzionamento del programma Python.
Inserisci la porta da utilizzare: 80
Server avviato! In attesa di connessioni ...
```

Utilizzando il nostro programma, assieme al comando telnet + IP della nostra macchina Metasploitable + la porta dedicata avremo il risultato sopra riportato, che è quello che ci aspettavamo.

SVILUPPO SECONDO BONUS PYTHON

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

Data la traccia:

“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.

Il codice che siamo andati a scrivere è il seguente:

```

GNU nano 8.1                                         traffico.py *
from scapy.all import sniff, wrpcap, get_if_list
from scapy.layers.inet import IP, TCP, UDP, ICMP

def gestore_pacchetti(pacchetto):
    if IP in pacchetto:
        ip_sorgente = pacchetto[IP].src
        ip_destinazione = pacchetto[IP].dst
    File System
        if TCP in pacchetto:
            protocollo = "TCP"
            porta_sorgente = pacchetto[TCP].sport
            porta_destinazione = pacchetto[TCP].dport
    Home
        elif UDP in pacchetto:
            protocollo = "UDP"
            porta_sorgente = pacchetto[UDP].sport
            porta_destinazione = pacchetto[UDP].dport
        elif ICMP in pacchetto:
            protocollo = "ICMP"
            porta_sorgente = None
            porta_destinazione = None
        else:
            protocollo = "Altro"
            porta_sorgente = None
            porta_destinazione = None
    EserciziPyt
    print(f"Protocollo: {protocollo}, IP Sorgente: {ip_sorgente}, IP Destinazione: {ip_destinazione}, "
          f"Porta Sorgente: {porta_sorgente}, Porta Destinazione: {porta_destinazione}")

def cattura_pacchetti(interfaccia, filtro_protocollo, nome_file, numero_pacchetti=0):
    if filtro_protocollo in ["tcp", "udp", "icmp"]:
        print(f"Avvio Cattura Pacchetti Su Interfaccia: {interfaccia} Con Filtro: {filtro_protocollo}")
    else:
        filtro_protocollo = None
        print(f"Avvio Cattura Pacchetti Su Interfaccia: {interfaccia} Senza Filtri:")

    pacchetti_catturati = sniff(iface=interfaccia, filter=filtro_protocollo, prn=gestore_pacchetti, count=numero_pacchetti)

    if not nome_file.endswith(".pcap"):
        nome_file = nome_file + ".pcap"
    wrpcap(nome_file, pacchetti_catturati)
    print(f"Packetti catturati salvati in: {nome_file}")

interfacce = get_if_list()
print("Interfacce Disponibili:")
for interfaccia in interfacce:

    interfacce = get_if_list()
    print("Interfacce Disponibili:")
    for interfaccia in interfacce:
        print(f"- {interfaccia}")
    while True:
        interfaccia = input("Inserisci l'interfaccia di rete: ").strip()
        if interfaccia in interfacce:
            break
        print("Inserire Un Interfaccia Disponibile")

    filtro_protocollo = input("Inserisci il protocollo da filtrare (TCP/UDP/ICMP).\nQualunque Altro Valore Non Avvierà Nessun Filtro: ").strip().lower()
    nome_file = input("Inserisci il nome del file di output: ").strip()

    while True:
        numero_pacchetti = input("Inserisci il numero di pacchetti da catturare (0 per cattura illimitata): ")
        if numero_pacchetti.isdigit():
            numero_pacchetti = int(numero_pacchetti)
            if numero_pacchetti >= 0:
                break
            else:
                print("Inserire Un Numero Maggiore o Uguale a 0")
        else:
            print("Inserire Un Numero")

    cattura_pacchetti(interfaccia, filtro_protocollo, nome_file, numero_pacchetti)

```

Grazie a questo programma, fin dall'inizio avremo il rilevamento e inserimento dell'interfaccia disponibile come in figura in basso:

```
(kali㉿kali)-[~/Desktop/EserciziPython]
$ python traffico.py
```

Interfacce Disponibili:

- lo
- eth0

Inserisci l'interfaccia di rete:

Una volta inserito, abbiamo pensato di poter scegliere direttamente all'utente secondo quale protocollo andare a filtrare i pacchetti (lasciando l'opzione con cui, se non verrà scelto nessun filtro, non verrà filtrato in nessun modo):

```
(kali㉿kali)-[~/Desktop/EserciziPython]
$ python traffico.py
Interfacce Disponibili:
- lo
- eth0
Inserisci l'interfaccia di rete: eth0
Inserisci il protocollo da filtrare (TCP/UDP/ICMP).
Qualunque Altro Valore Non Avvierà Nessun Filtro: █
```

In questo caso noi andremo semplicemente ad immettere una lettera qualsiasi per far in modo che l'intercettazione dei pacchetti non sia filtrata ed andare così a verificare che possa registrarli tutti. Come accennato precedentemente avremo anche la possibilità di scegliere quanti pacchetti prendere e come chiamare il file che andremo a salvare con all'interno i pacchetti registrati in modo da poterlo utilizzare ed analizzare in futuro tramite per esempio un software come Wireshark:

```
(kali㉿kali)-[~/Desktop/EserciziPython]
$ python traffico.py
Interfacce Disponibili:
- lo
- eth0
Inserisci l'interfaccia di rete: eth0
Inserisci il protocollo da filtrare (TCP/UDP/ICMP).
Qualunque Altro Valore Non Avvierà Nessun Filtro: g
Inserisci il nome del file di output: TestPresentazione1
Inserisci il numero di pacchetti da catturare (0 per cattura illimitata): 100
```

Procedendo su un altro terminale con i comandi di “ping” e “nmap” potremo avere questo risultato:

```
(kali㉿kali)-[~/Desktop/EserciziPython] tl=63 time=1.51 ms
$ sudo python traffico.py: icmp_seq=4 ttl=63 time=0.987 ms
[sudo] password for kali: 192.168.60.150: icmp_seq=5 ttl=63 time=1.57 ms
Interfacce Disponibili: lo 192.168.60.150: icmp_seq=6 ttl=63 time=0.995 ms
- lo test from 192.168.60.150: icmp_seq=7 ttl=63 time=1.05 ms
- eth0 test from 192.168.60.150: icmp_seq=8 ttl=63 time=1.93 ms
Inserisci l'interfaccia di rete: eth0
Inserisci il protocollo da filtrare (TCP/UDP/ICMP). n=1.02 ms
Qualunque Altro Valore Non Avvierà Nessun Filtro: g
Inserisci il nome del file di output: TestPresentazione1
Inserisci il numero di pacchetti da catturare (0 per cattura illimitata): 100
Avvio Cattura Pacchetti Su Interfaccia: eth0 Senza Filtri: 0023ms
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: None, Porta Destinazione: None
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: None, Porta Destinazione: None
Protocollo: TCP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: 43096, Porta Destinazione: 80
Protocollo: TCP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: 49558, Porta Destinazione: 443
Protocollo: TCP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: 443, Porta Destinazione: 49558
Protocollo: TCP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: 80, Porta Destinazione: 43096
Protocollo: TCP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: 43096, Porta Destinazione: 80
Protocollo: TCP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: 43096, Porta Destinazione: 80
Protocollo: UDP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.50.1, Porta Sorgente: 43961, Porta Destinazione: 53
Protocollo: UDP, IP Sorgente: 192.168.50.1, IP Destinazione: 192.168.50.151, Porta Sorgente: 53, Porta Destinazione: 43961
Protocollo: TCP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: 43112, Porta Destinazione: 80
Protocollo: TCP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.50.151, Porta Sorgente: 80, Porta Destinazione: 43112
Protocollo: TCP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: 43112, Porta Destinazione: 80
Protocollo: TCP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150, Porta Sorgente: 43112, Porta Destinazione: 80
```

```
(kali㉿kali)-[~] $ ping 192.168.60.150
PING 192.168.60.150 (192.168.60.150) 56(84) bytes of data.
64 bytes from 192.168.60.150: icmp_seq=1 ttl=63 time=5.15 ms
64 bytes from 192.168.60.150: icmp_seq=2 ttl=63 time=1.31 ms
64 bytes from 192.168.60.150: icmp_seq=3 ttl=63 time=1.51 ms
64 bytes from 192.168.60.150: icmp_seq=4 ttl=63 time=0.987 ms
64 bytes from 192.168.60.150: icmp_seq=5 ttl=63 time=1.57 ms
64 bytes from 192.168.60.150: icmp_seq=6 ttl=63 time=0.995 ms
64 bytes from 192.168.60.150: icmp_seq=7 ttl=63 time=1.05 ms
64 bytes from 192.168.60.150: icmp_seq=8 ttl=63 time=1.93 ms
64 bytes from 192.168.60.150: icmp_seq=9 ttl=63 time=1.13 ms
64 bytes from 192.168.60.150: icmp_seq=10 ttl=63 time=1.02 ms
64 bytes from 192.168.60.150: icmp_seq=11 ttl=63 time=0.959 ms
^Cscrivi il nome del file di output: TestPresentazione1
— 192.168.60.150 ping statistics —
11 packets transmitted, 11 received, 0% packet loss, time 10023ms
rtt min/avg/max/mdev = 0.959/1.601/5.150/1.160 ms Destinazione: 192.168.60.150
Protocollo: ICMP, IP Sorgente: 192.168.60.150, IP Destinazione: 192.168.60.150
(kali㉿kali)-[~] $ nmap -p 80 192.168.60.150
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-20 05:44 EST
Nmap scan report for 192.168.60.150
Host is up (0.0012s latency).
PORT      STATE SERVICE
80/tcp     open  http
IP        Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150
Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds
Protocollo: ICMP, IP Sorgente: 192.168.50.151, IP Destinazione: 192.168.60.150
```

TITOLI DI CODA

Progetto ideato, sviluppato, personalizzato e completato da:

- Alberto Pino
- Alberto Greco
- Matteo Bosio
- Michele Mollea

- Valerio Brugioni
- Andrea Emanuelli
- Simone Bova
- Gabriel El Gamal
- Giovanni Sammaritano
- Alex Forti