



Buildweek 1

Testing di rete

By NexusMind group

Contents

01

Linee Generali

02

Verifica Verbi HTTP con risultati

03

Scansione delle porte con risultati

04

Report di valutazione e rischio sulla sicurezza

05

Conclusioni

1. Linee Generali

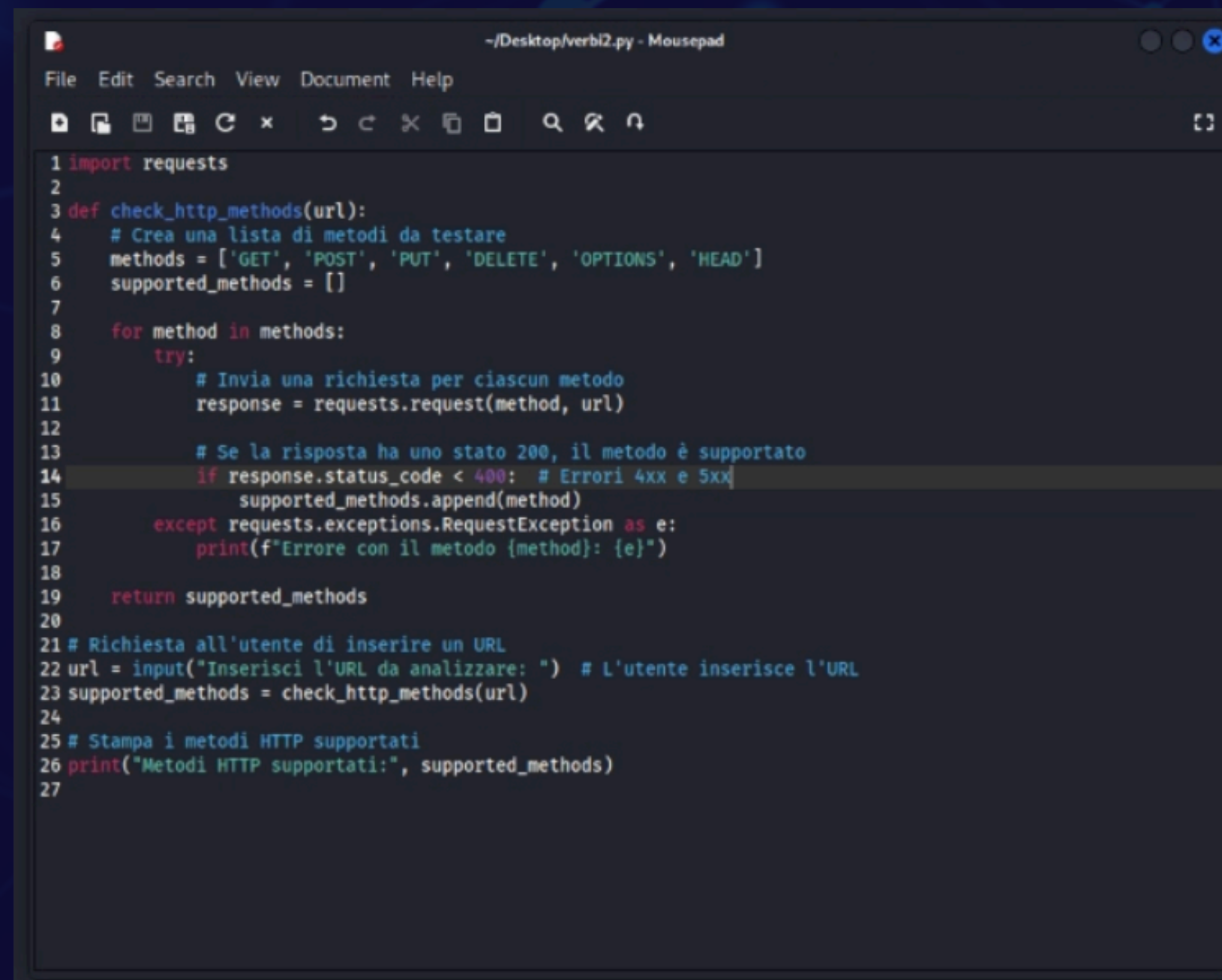
L'azienda Theta ci ha commissionato di testare la loro rete una volta implementata.

I test sono stati i seguenti:

1. Verifica dei verbi HTTP per inviare richieste "HTTP" al web server e verificare le risposte. Abbiamo, inoltre, documentato le risposte ricevute dal Web server per ogni verbo.
2. Scansione delle porte sui dispositivi di rete per verificare la sicurezza e l'accessibilità delle varie porte in comunicazione. Abbiamo, inoltre, redatto una lista completa delle porte aperte e chiuse sui vari dispositivi con relative raccomandazioni e rischi sulla sicurezza.

2.1 Verifica dei verbi HTTP con risultati

Per la verifica dei verbi HTTP (come GET, POST, PUT, DELETE, ecc.) abbiamo utilizzato lo script di destra creato in ambiente python, il quale ci ha mostrato i metodi presenti a una determinata URL. Utilizza la libreria requests per inviare richieste HTTP e permette di controllare la risposta del server per determinare se un metodo è accettato o meno.



```
1 import requests
2
3 def check_http_methods(url):
4     # Crea una lista di metodi da testare
5     methods = ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS', 'HEAD']
6     supported_methods = []
7
8     for method in methods:
9         try:
10             # Invia una richiesta per ciascun metodo
11             response = requests.request(method, url)
12
13             # Se la risposta ha uno stato 200, il metodo è supportato
14             if response.status_code < 400: # Errori 4xx e 5xx
15                 supported_methods.append(method)
16         except requests.exceptions.RequestException as e:
17             print(f"Errore con il metodo {method}: {e}")
18
19     return supported_methods
20
21 # Richiesta all'utente di inserire un URL
22 url = input("Inserisci l'URL da analizzare: ") # L'utente inserisce l'URL
23 supported_methods = check_http_methods(url)
24
25 # Stampa i metodi HTTP supportati
26 print("Metodi HTTP supportati:", supported_methods)
27
```


2.2 Verifica dei verbi HTTP con risultati

Come risultato dei test effettuati abbiamo ricevuto i seguenti metodi dal web server: GET, POST, PUT, DELETE, OPTIONS e HEAD. La documentazione delle risposte ricevute dai server durante l'invio di richieste HTTP è fondamentale per comprendere il comportamento del server in relazione ai metodi supportati. Conoscere e analizzare le risposte aiuta a identificare le funzionalità disponibili e le possibili vulnerabilità, garantendo così una maggiore sicurezza nelle applicazioni web.



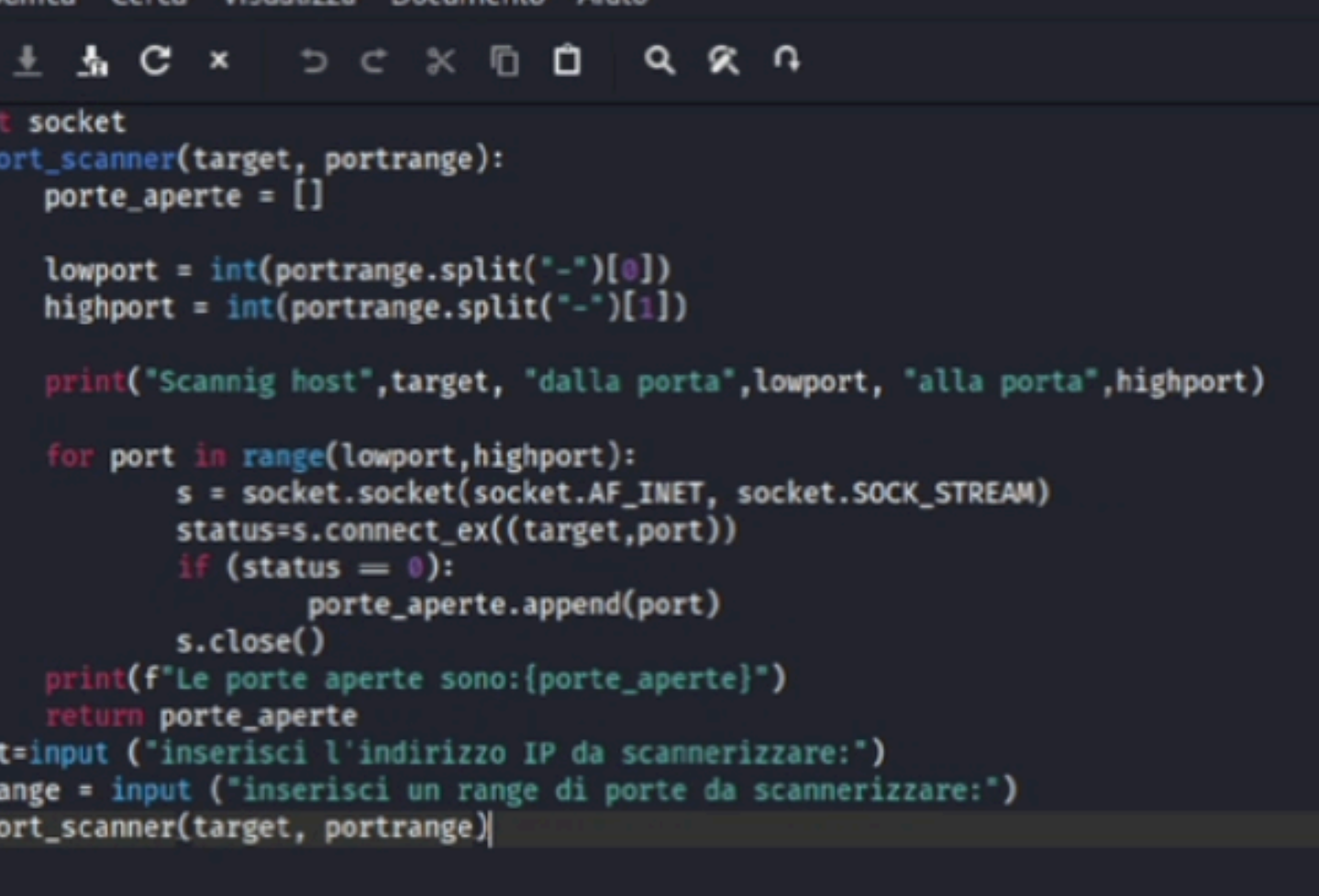
```
kali@kali: ~/Desktop
File Actions Edit View Help

(kali@kali)-[~/Desktop]
$ python verbi2.py
Inserisci l'URL da analizzare: http://192.168.50.101/phpMyAdmin/
Metodi HTTP supportati: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS', 'HEAD']

(kali@kali)-[~/Desktop]
$
```

3.1 Scansione delle porte con risultati

Per la scansione delle porte abbiamo creato un programma in ambiente python che rilevi tutte le porte che sono aperte entro un range definito.



The screenshot shows a Windows Notepad application window titled "~\Scrivania\port_scanner.py - Mousepad". The menu bar includes "File", "Modifica", "Cerca", "Visualizza", "Documento", and "Aiuto". The toolbar contains standard editing icons. The Python code is as follows:

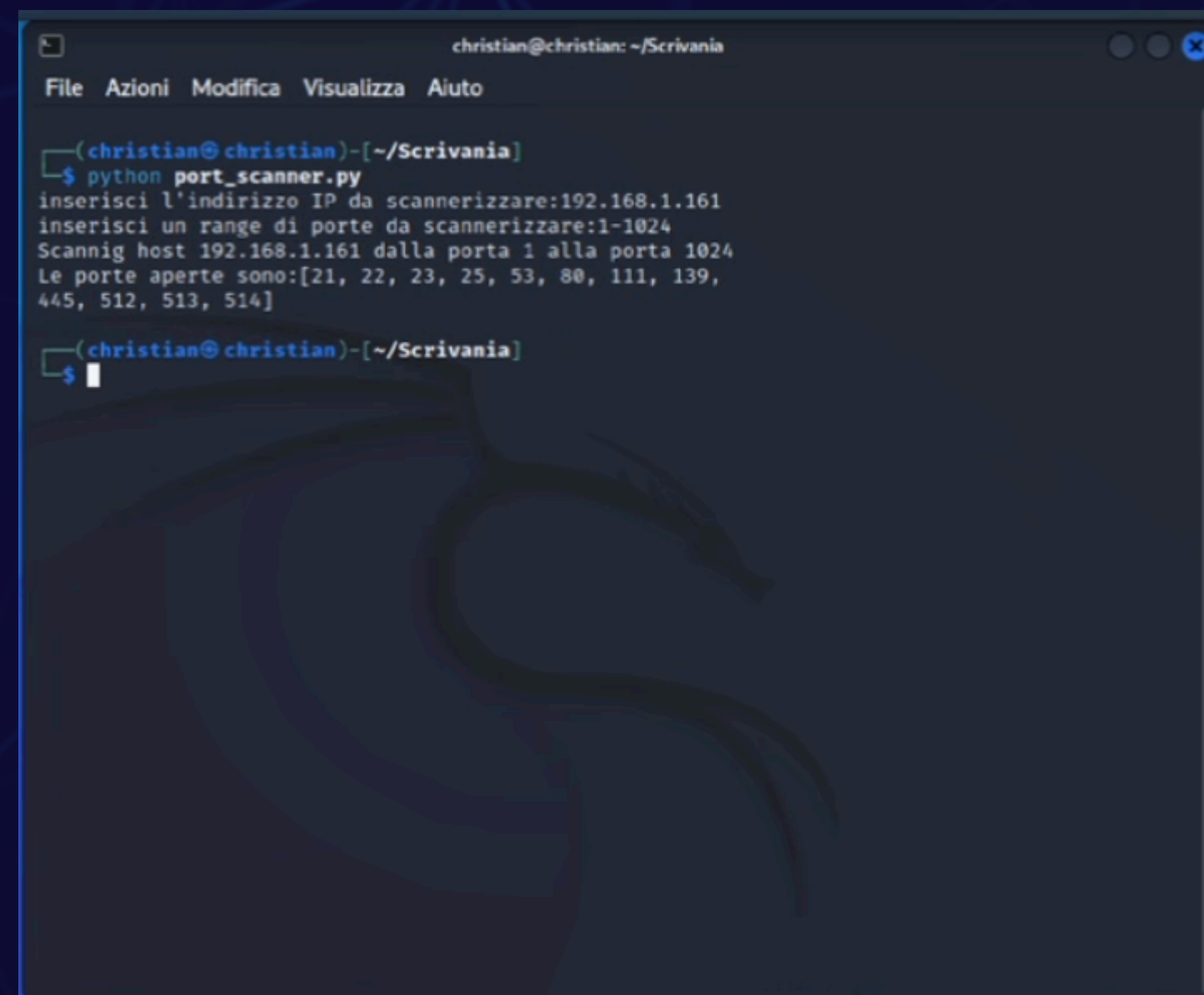
```

1 import socket
2 def port_scanner(target, portrange):
3     porte_aperte = []
4
5     lowport = int(portrange.split("-")[0])
6     highport = int(portrange.split("-")[1])
7
8     print("Scannig host",target, "dalla porta",lowport, "alla porta",highport)
9
10    for port in range(lowport,highport):
11        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12        status=s.connect_ex((target,port))
13        if (status == 0):
14            porte_aperte.append(port)
15        s.close()
16    print(f"Le porte aperte sono:{porte_aperte}")
17    return porte_aperte
18 target=input ("inserisci l'indirizzo IP da scannerizzare:")
19 portrange = input ("inserisci un range di porte da scannerizzare:")
20 c = port_scanner(target, portrange)
21
22

```


3.2 Scansione delle porte con risultati

Questo Script esegue una scansione delle porte su un host specifico (in questo caso 192.168.1.161) verificando quali porte TCP all'interno di un intervallo definito dall'utente sono aperte. La scansione avviene cercando di aprire una connessione su ogni porta e, se la connessione ha successo, la porta viene considerata aperta e aggiunta alla lista dei risultati.



```
christian@christian: ~/Scrivania
File Azioni Modifica Visualizza Aiuto

(christian@christian)-[~/Scrivania]
$ python port_scanner.py
inserisci l'indirizzo IP da scannerizzare:192.168.1.161
inserisci un range di porte da scannerizzare:1-1024
Scannig host 192.168.1.161 dalla porta 1 alla porta 1024
Le porte aperte sono:[21, 22, 23, 25, 53, 80, 111, 139,
445, 512, 513, 514]

(christian@christian)-[~/Scrivania]
$
```

4. Report di valutazione e rischio sulla sicurezza

L'obiettivo di questa valutazione è identificare le porte aperte su un host specifico e valutarne i rischi associati per la sicurezza della rete aziendale. Il rilevamento di porte aperte può indicare la presenza di servizi attivi che potrebbero, tuttavia, essere vulnerabili a vari attacchi, come accessi non autorizzati, sfruttamento di vulnerabilità software o malware.

Porta 21 (FTP): Trasmette informazioni in chiaro e lo rende vulnerabile a attacchi man-in-the-middle. Raccomandiamo di disabilitare FTP e utilizzare FTPS per una trasmissione sicura.

Porta 22 (SSH): Protocollo sicuro tuttavia raccomandiamo l'utilizzo di chiavi SSH al posto delle password e limitazione degli accessi tramite firewall.

Porta 23 (Telnet): Trasmette anch'esso in chiaro. Raccomandiamo di disabilitarlo e usare SSH.

Porta 25 (SMTP): Utilizzata per l'invio di email. Potrebbe essere sfruttata per attacchi di spam o relay non autorizzato. Raccomandiamo di configurarlo per prevenire il relay aperto.

Porta 53 (DNS): Se il DNS è vulnerabile potrebbe essere sfruttato per attacchi di avvelenamento della cache. Raccomandiamo di limitare le Query DNS a sorgenti fidate.

Porta 80 (HTTP): Trasmissione del traffico in chiaro. Raccomandiamo l'utilizzo del protocollo HTTPS per crittografare il traffico.

Porta 111 (RPC): Può essere sfruttato per attacchi remoti o per raccogliere informazioni di sistema. Raccomandiamo di disabilitarlo se non necessario o proteggerlo tramite firewall.

Porta 139 e 445 (Netbios e SMB): Sono porte che sono comunemente obiettivo di attacchi di rete come il Ransomware o l'accesso non autorizzato ai file. Raccomandiamo di disabilitarli se non necessari e utilizzare SMBv3 per una maggiore sicurezza e assicurarsi di utilizzare patch recenti.

Porte 512, 513, 514 (rlogin, rsh): Questi servizi trasmettono servizi in chiaro e sono obsoleti. Raccomandiamo di disabilitarli e utilizzare protocolli moderni e sicuri come SSH.

La presenza di numerose porte aperte, molte delle quali utilizzano protocolli non sicuri o obsoleti, rappresenta un rischio significativo per la sicurezza della rete. È fondamentale prendere misure immediate per disabilitare i servizi obsoleti e non sicuri, limitare l'accesso alle porte essenziali e garantire che tutti i servizi esposti siano configurati correttamente e aggiornati.

5. Conclusioni

Ringraziamo l'azienda Theta per averci dato la possibilità di testare la loro rete tuttavia vogliamo nuovamente ricordare l'importanza di una rete sicura e di una buona manutenzione per evitare di incorrere in problemi gravi e irreparabili.

Cordialmente,
NexusMind Group