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

A Hash is often used to safeguard passwords and other important data. As a penetration tester, you may need to find the cleartext value for several different hashes. The Hash library in Python allows you to build hash crackers according to your requirements quickly.

Hashlib is a powerful module that supports a wide range of algorithms.

```
┌──(root💀TryHackMe)-[/home/alper]
└─# python3
Python 3.9.1+ (default, Feb  5 2021, 13:46:56)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hashlib
>>> hashlib.algorithms_available
{'md5-sha1', 'sha3_256', 'sha384', 'shake_256', 'blake2b', 'sha512_224', 'md4', 'shake_128', 'ripemd160', 'sha3_224', 'md5', 'sha3_384', 'sha512_256', 'sha224', 'blake2s', 'w
hirlpool', 'sm3', 'sha512', 'sha1', 'sha256', 'sha3_512'}
>>>
```

Leaving aside some of the more exotic ones you will see in the list above, hashlib will support most of the commonly used hashing algorithms.

Hash Cracker

```python
import hashlib
import pyfiglet

ascii_banner = pyfiglet.figlet_format("TryHackMe \n Python 4 Pentesters \n HASH CRACKER for MD 5")
print(ascii_banner)

wordlist_location = str(input('Enter wordlist file location: '))
hash_input = str(input('Enter hash to be cracked: '))

with open(wordlist_location, 'r') as file:
    for line in file.readlines():
        hash_ob = hashlib.md5(line.strip().encode())
        hashed_pass = hash_ob.hexdigest()
        if hashed_pass == hash_input:
            print('Found cleartext password! ' + line.strip())
            exit(0)
```

This script will require two inputs: the location of the wordlist and the hash value.

As you probably know, hash values can not be cracked as they do not contain the cleartext value. Unlike encrypted values that can be "reversed" (e.g. decrypted), cleartext values for hashes can only be found starting with a list of potential cleartext values. A simplified process can be seen below;

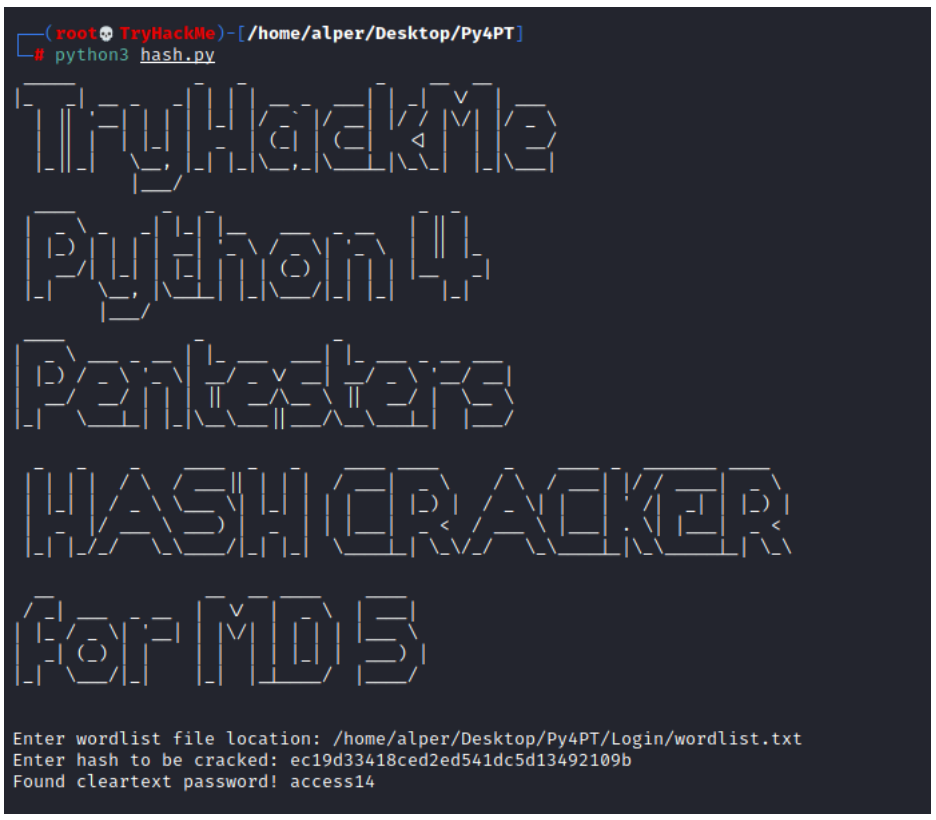
1. You retrieve the hash value "eccbc87e4b5ce2fe28308fd9f2a7baf3" from a database, which you suspect is the hash for a number between 1 and 5.
2. You create a file with possible cleartext values (numbers from 1 to 5)
3. You generate a list of hashes for values in the cleartext list (Hash values for numbers between 1 and 5)
4. You compare the generated hash with the hash value at hand (Matches hash value of the number 3)

Obviously, a more effective process can be designed, but the main principle will remain identical.

The script below follows an approach close to the one described above;

1. Asks for the location of a wordlist
2. Asks for the hash to be cracked
3. Reads values from the wordlist (one per line)
4. Converts cleartext values to MD5 hash values
5. Compares the generated MD5 hash value with the value entered by the user

Below: The MD5 cracking script, including the absolutely optional and tacky ASCII art banner.



Answer the questions below

What is the hash you found during directory enumeration?

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

What is the cleartext value of this hash?

Answer format: ******* Submit

Modify the script to work with SHA256 hashes.

No answer needed Completed

Using the modified script find the cleartext value for 5030c5bd002de8713fef5daebd597620f5e8bcea31c603dccdfcdf502a57cc60

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

Task 8 ○ Keyloggers

Task 9 ○ SSH Brute Forcing

Task 10 ○ Extra challenges