Lab 6

Password Cracking and Netcat

Note 1: Make sure that your Kali and Ubuntu VMs are running fine, and the network is set as "NatNetwork" (on Virtual Box).

1. Make your own dictionary for password cracking using crunch

   When we use the password cracking tools like *hydra* and *john-the-ripper*, we need to provide them with a dictionary. There exist ready-made ones, but we can create our own using *crunch*.

   The basic syntax for crunch is

   ```
   crunch [min len] [max len] [character set] [options]
(for displaying on the screen)
   ```

   ```
   crunch [min len] [max len] [character set] [options]
–o file (for outputting as a file)
   ```

   On Kali Terminal, type `crunch 3 3 abc` and run `crunch 3 3 abcd` (Did you get the idea how crunch works?) It will generate all possible words with repetitions (such as `bbb`) using characters `a`, `b` and `c`.

   ```
   $crunch 3 5 0123456789 –o numword.lst
   ```

   This will create all the possible words of length 3 to 5, all of which consist of numbers between `0` and `9`.

   The file size will be big – nearly 650 kB. If you want to use special characters, use backward slash. For example, \&, *, \% and etc.

   One of the useful options is `–t`. You can specify a pattern you're searching. Suppose that someone uses a password of eight characters and his birthday is `0829`. An attacker might want to try all the possible combinations ending with 0829. In this case, we can run

   ```
   $crunch 8 8 –t @@%^0829 –o birthday.txt
   ```

   Here,

   @ is a wildcard for lowercase alphabetical characters

   , is a wildcard for uppercase alphabetical characters

% is a wildcard for numeric and

^ is for special characters.

2. Cracking password using hydra online

   This exercise needs to access Metasploitable VM. Run it under the NatNetwork.

   First, create a user named "alice" in Metasploitable. Login to Metasploitable and type and run:

   ```
   $sudo useradd –m alice –G users –s /bin/bash
   ```

   Then, set a password for `alice`:

   ```
   $sudo passwd alice
   ```

   (Let us set up an easy password that consists of only five numbers. Even it may take quite a while to take find a five-digit password. So, choose a little bit short (and obvious) password for testing.)

   Then, go to Kali VM and create words list of 5-digit numbers using crunch. Can you do it using crunch command? Name your file `numword.txt`

   Now run hydra using the words list you have just created:

   ```
   $hydra –t 64 –l alice -P numword.txt –vV <Meta IP> ftp
   ```

   Have you found the password? (Note that 64 is a maximum number of concurrent connections to the target, and ftp is a protocol that hydra makes use of to perform brute-force.)

3. Cracking password using john-the-ripper

   First, create a user `steve` for testing on Kali:

   ```
   $ sudo useradd –m steve –G sudo –s /bin/bash
   ```

   Next, set password for victim on Kali:

   ```
   $ sudo passwd steve
   ```

   Combine entries of `/etc/passwd` and `/etc/shadow` by unshadowing:

   ```
   $ sudo unshadow /etc/passwd /etc/shadow > target_list
   ```

   Run John the Ripper using the password list provided by it:

   ```
   john –format=crypt –wordlist
   =/usr/share/john/password.lst target_list
   ```

Important!
Once the john-the-ripper has cracked the password, it will not do it again. It will save the cracked passwords. To view it, run

```
$john --show target_list
```

4. Calculating hashes using Python

   There are a few different ways to calculate hashes in Python. In this example, we will use the `hashlib` module (https://docs.python.org/3/library/hashlib.html) Create a file called `hashtest.py` that contains the following code to perform hashing.

```
import hashlib

hash = hashlib.sha1(b'abcdef').hexdigest()
print(hash)
```

   In the above code, `b'abcdef'` means the byte representation of the string `'abcdef'`. Converting the string to byte stream is necessary as the sha1 function in `hashlib` takes byte as input.
   Save the code as hash.py and compile using python3. What is the result?

   Note that in the `hashlib` module, `sha1()`, `sha224()`, `sha256()`, `sha384()`, `sha512()` and `md5()` are available. Try one or two such functions. (Note that these hash functions are SHA2 hash functions. To use SHA3, type `sha3_512()`, for example.)

5. Cracking hashes using Hashcat

   Hashcat is a popular tool to crack hashes. The command structure is as follows:

```
   $hashcat -a 0 -m [Hash Type] -o [Output File Path] --
potfile-disable [Hashes File      Path] [Dictionary File Path]
```

   The `-a` option is the attack mode. 0 means dictionary (or wordlist). 1 is a combination of dictionary files. (There are other cases such as specific patterns for dictionary words, but we do not use those cases in this lab.)

   It is worth noting that the potfile is a critical component of Hashcat that stores cracked hashes to prevent redundant cracking attempts. It is not recommended to disable it unless you have a specific need to do so: --potfile-disable.

   The hash types used for the option `-m` are the following:

#	Algorithm name
900	MD4
0	MD5
100	SHA1
1300	SHA2-224
1400	SHA2-256

Create a dictionary file of approximately 100 words using Crunch. Then create a file that contains a MD5 hash of one word contained in the dictionary file. Show that hashcat can output the pre-image of the hash. (In other words, show that hashcat can crack the MD5 hash.)

6. Extracting passwords using a Python program

   You have learned how to create a dictionary (password list). Suppose that you want to filter out passwords having a specific pattern from the existing dictionary. One useful technique is to use a regular expression filter.

   Your task is to write a python program to find possible passwords containing 0825 or 0827 using the regular expression (re) library (https://docs.python.org/2/library/re.html). You may want to refer to https://www.w3schools.com/python/python_regex.asp for a quicker reference.

   To create a password list, use crunch as follows.

   ```
   crunch 6 6 -t @@082% -o birthday2.txt
   ```

   We have written a Python code `match.py` that you can start with:

   ```
   import re #to use regular expression package

   dictionary = open("birthday2.txt","r") #to open password list
   file
   extracted2File = open("extracted_pwds.txt","w") #to write
   extracted passwords into a file
   regx = re.compile(" ") # you need to put your regular
   expression inside the quotation marks
   passwords = filter(regx.search, dictionary) #search the
   password using the regular expression provided

   for line in passwords:
           extracted2File.write(line)

   dictionary.close()
   extracted2File.close()
   ```