Introduction to Computer Security Module – G6077

-

Password cracking lab

The following lab uses ​**johntheripper**​, ​**crunch** , ​**office2john.py**​ and **Hashcat** in a kali environment. These tools used in varieties of way. In this lab, we will just look at the cracking a password.

# Getting Started

1) You should have access to GUI-Kali on AWS

2) In GUI- Kali, open the browser and logged into your module Canvas to download a

folder called PasswordFiles.zip

3) In GUI-Kali, take the folder from the Downloads to Desktop, cut/paste works fine.

4) Use these command in terminal

Make sure that you are in the Desktop folder

#mkdir PasswordFiles

# unzip PasswordFiles.zip –d PasswordFiles

5) You will not be able to open the word files as they are password protected.

**office2john.py**​ is a utility that will extract the hashed password from an office file. There are many other variations of this which enable you to apply the technique to login passwords, archives, pdfs etc

6) Open john terminal and use the following to go to the working dir:

We’ll now go through a simple crack on a word doc (tooeasy.doc) with a poor password (short and well known)

To extract the hash from a word doc run the command:

## # python office2john.py tooeasy.doc > tooeasy.txt

## // are you running command from correct directory, if not utilise the knowledge gained in first week and cd to correct directory.

If the above does not work, it could be that office2john is not installed. You will need to run:

**office2john.py**​ is a utility that will extract the hashed password from an office file. There are many other variations, which enable you to apply technique to login passwords, archives, pdfs etc

# wget https://raw.githubusercontent.com/magnumripper/JohnTheRipper/bleeding-jumbo/run/office2john.py

After installation, if you need to reboot the Kali, just simply use reboot, Kali will reboot and will ask for the login again.

This will create a file “tooeasy.txt” that can then be put through ​**john**​ the ripper to crack the password by running:

# john --wordlist= /usr/share/wordlists/nmap.lst tooeasy.txt

This should only take a few seconds as this is a ridiculously simple password.

Once you have cracked a password this is stored, so to view the password again use the command:

## #john --show tooeasy.txt

To start with, we make john use a default word list of common passwords stored in nmap.lst (this password was in the list!) so it is found quickly.

Once cracked then the password is stored in a file ​**~/.john/john.pot**​ so they can be retrieved using the —show option. If you clear this file you can start again as once the cracked password has been stored then you can’t run the crack again as it will already have the password.

# Giving it a go

Repeat the above steps for the file ​**stilltooeasy.doc**​ (this has a longer password but still in the known list) and file ​**anothereasyone.docx**​.

Next attempt the file ​**pass5** ​- this has a much longer password that isn't in the list of known ones, but is still a dictionary word so we can use a different wordlist (​**dictionary.txt**​). This file is provided on the Canvas.

Extract the hashed password as before using ​**office2john**​ and then use the following command to crack the password as before except you are using the dictionary instead of the default known password file.

## john --wordlist=dictionary.txt pass5.txt

This will take a while due to it starting with ‘s’ (approx 15 minutes)

The file ​**pass6** ​is another dictionary word that is quicker as it starts with c (approx 3 minutes)

Try the same process on pass8. This should take approx 2 mins.

John works through the wordlist sequentially.

Cracking with Hashcat

We can begin by displaying the help menu (**--help**) for Hashcat. This will provide us with a wealth of information including usage options, hash modes, and other features. There is a ton of information here, so I won't show the output, but you should dive into it if you really want to know Hashcat.

~# hashcat --help

From the output, we're just interested in the MS Office hash modes. Near the bottom of the help menu, we will find the MS Office mode options and their corresponding numbers. We know from our hash that this is an Office 2013 file, so locate its number ID of **9600**.

Now we can set the rest of our options using the following command.

~# hashcat -a 0 -m 9600 --username -o cracked.txt hash.txt /usr/share/wordlists/nmap.lst

* The -a flag sets the attack type as the default straight mode of 0.
* The -m flag specifies the mode we want to use, which we just found.
* The --username option ignores any usernames in the hash file.
* We can specify the output file as cracked.txt with the -o flag.
* And finally, we can pass in hash.txt which contains the hash, and set a word list just like we did earlier.

Hashcat will then begin cracking.

hashcat (v5.1.0) starting...

\* Device #2: Not a native Intel OpenCL runtime. Expect massive speed loss.

You can use --force to override, but do not report related errors.

OpenCL Platform #1: Intel(R) Corporation

========================================

\* Device #1: Intel(R) Core(TM) i5 CPU M 480 @ 2.67GHz, 934/3736 MB allocatable, 4MCU

...

After some time has passed, the status will show as cracked, and we are ready to view the password.

Session..........: hashcat

Status...........: Cracked

Hash.Type........: MS Office 2007

Hash.Target......: $office$\*2007\*20\*128\*16\*a7c7a4eadc2d90fb22c073c6324...2b6870

Time.Started.....: Tue Feb 5 15:08:00 2019 (4 secs)

Time.Estimated...: Tue Feb 5 15:08:04 2019 (0 secs)

Guess.Base.......: File (/usr/share/wordlists/nmap.lst)

Guess.Queue......: 1/1 (100.00%)

Speed.#1.........: 610 H/s (8.51ms) @ Accel:512 Loops:128 Thr:1 Vec:4

Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts

Progress.........: 2048/5084 (40.28%)

Rejected.........: 0/2048 (0.00%)

Restore.Point....: 0/5084 (0.00%)

Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:49920-50000

Candidates.#1....: #!comment: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IMPORTANT NMAP LICENSE TERMS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -> Princess

Started: Tue Feb 5 15:07:50 2019

Stopped: Tue Feb 5 15:08:05 2019

Simply **cat** out the specified output file, and it will show the hash with the plaintext password tacked on the end.

~# cat cracked.txt

$office$\*2007\*20\*128\*16\*a7c7a4eadc2d90fb22c073c6324b6b49\*abc5f80409f5f96f97e184e44aacd0b7\*930b0c48a7eb5e13a57af4f3030b48e9402b6870:apple

Success! Now we know two methods of cracking the hash after extracting it from a password-protected Microsoft Office file with office2john.

# Wordlists

Custom wordlists can be created using the **crunch** ​ program so you can target a crack using​ any relevant info you know.

**crunch <min-len> <max-len> [<charset string>] [options]**

eg

## crunch 1 3 > 1-3char.lst

will create a list of every combination of lowercase a-z passwords (default)

can use charsets from a file:

**/usr/share/rainbowcrack/charset.txt** - contains all characters split into different categories​

eg

## crunch 1 3 -f /usr/share/rainbowcrack/charset.txt numeric > 1-3all.lst

This will create 3 character passwords from a pool of 1234567890

Use **man crunch**​ for more details​

You could create a list of pets names, cars, anagrams of DOB, based on postcodes.

To crack the file pass9 we think the password is the users DOB.

Use crunch to create all possible passwords based on a users DOB ‘18th March 1975’

We can try the 6 digit DOB of 180375 - if it doesn't open the doc then perhaps it's an anagram of this?

## crunch 6 6 013578 -o dob.lst john --wordlist=dob.lst pass9.txt

(approx 4mins)

This creates a wordlist of every combination of the characters/numbers given. As we think this is the DOB we can skip all combinations 1-5 characters and just use the length of 6.

If it doesn't work then could be based on full year (1975 instead of just 75), so build a new wordlist:

## crunch 8 8 0135789 dob2.lst

Remember:

The more characters => more combinations => longer to crack

# Wilcards

We can also use wildcards (any character in a set location).

For example, shoulder surfing may get you part of a password. Eg you're certain of the first 3 characters (jef) and pretty sure that it was 7 in total and the other 4 could be anything lower case.

The file to test this on is ​**wildcard1.docx**

We will substitute any combination of lower case letters as the last 4 characters

## crunch 7 7 abcdefghijklmnopqrstuvwxyz -t jef@@@@ -o newlist.lst john --wordlist=newlist.lst wildcard1.txt​ (approx 5mins)

If lower case does not work then expand the combination to include upper case/digits/other characters.

# Expansion

Try creating a few word docs with different passwords and getting someone else to try and crack.

Not too large or complicated though as you may want to go home at some point.

If we then try a file with a more difficult password it will try the known passwords first before dropping to ASCII incremental/brute force method - very time consuming

The file ​**toodifficult.docx** ​was ran overnight and wasn't cracked.

Create a 5 char wordlist using all characters and attempt to crack the file ​**pass7.docx**

# Final Task

In the lab, we dealt with the MS Word files only. PDFs are another important type of files that are used a lot in organisations. Passwords are used to protect PDFs. You are provided with a PDF file, PasswordProtectedFile. You will need to crack the password for this file.

Working in the security industry is quite different from other areas of the computer science. Working as a security expert, you will know solutions on a top of your head for a number of problems. However, there will be situations where you will need to search and find a solution. This task will test your abilities on such situations.