Mobile Devices Forensic- Lab3

**MSc in Computing**

richard dunne

2016

Table of Contents

[Introduction 2](#_Toc443853545)

[Evaluation 2](#_Toc443853546)

[Cracking Tools 2](#_Toc443853547)

[Password Cracking 3](#_Toc443853548)

[1. Dictionary Attack 3](#_Toc443853549)

[2. Mask/Character Set Attack 3](#_Toc443853550)

[3. Bruteforce 3](#_Toc443853551)

[Choosing the method 3](#_Toc443853552)

[Stages of the crack 4](#_Toc443853553)

[Results 4](#_Toc443853554)

[Other programs 5](#_Toc443853555)

[Appendix 6](#_Toc443853556)

[References 8](#_Toc443853557)

# Introduction

*Mobile Device Forensics involves the collection of forensically sound evidence (or admissible evidence that can be presented to the court of law). Using the program you developed in lab 2, find the hash code of the following two files MDF-SF1.docx and MDF-SF2.docx. Since these two files are password protected, write a Python program to break the password. After breaking the password find the hash code again. Compare the values of hash code before and after breaking passwords. List and use two more password management tools for breaking password. Compare the hash codes with those generated by your program. Make sure that you are program is documented. It is important that your program is uploaded in GitHub as well or any other web based repository.*

# Evaluation

The initial stage of any problem solving is to analyse the task. The two documents that must be cracked are Microsoft Office documents. Office 2007-2010 documents are Microsoft file type.

According to the Microsoft whitepaper (microsoft, 2016), the following specifications are defined:

* Key derivation is performed using 50,000 iterations[ of SHA-1 (Leblanc, 2016)
* Uses a 16-byte (128-bit) random salt
* AES is the block cipher used to encrypt the document.
* By default, 128-bit key are used. There is a registry tweak to change this to 256-bit.

Prior to commencing, the task request was to find a hash of the two documents. MD5 Hash of both documents before any cracking is:

* MDF-SF1.docx : dd06b1c03f4ad67c99d4f371f4001fa4
* MDF-SF2.docx : 69a6db156d051c48e655d5a95cff84b4

# Cracking Tools

There are many open source and paid applications that claim to crack office documents, some of which are:

* Ophcrack:

http://ophcrack.sourceforge.net/

* Cain and able

http://www.oxid.it/cain.html

* John the ripper

http://www.openwall.com/john/

* Advanced Office Password Recovery

<https://www.elcomsoft.com/aopr.html>

* Hascat

Http://hashcat.net

# Password Cracking

There are three main way to crack a password, as described below.

### Dictionary Attack

Dictionary attacks are just what they sound like; the dictionary is used to find a password. Hackers basically have very large text files that include millions of generic passwords, such as password, iloveyou, 12345, admin, or 123546789.

### Mask/Character Set Attack

If a hacker cannot guess a password from a dictionary of known passwords, their next option will be to use some general rules to try a lot of combinations of specified characters. This means that instead of trying a list of passwords, a hacker would specify a list of characters to try.

For example, if the hacker knew the password was [just numbers](http://www.quickanddirtytips.com/education/math), they could instruct their program to only try number combinations as passwords. From here, the program would try every combination of numbers until it cracked the password. Hackers can specify numerous other settings, like minimum and maximum length, how many times to repeat a specific character in a row, and many more. This decreases the amount of work the program would need to do.

### Bruteforce

The third option may be to "bruteforce" your password. A bruteforce tries every character combination until it gets the password. Generally, this type of attack is impractical, though--as anything over 10 characters would take millions of years to figure out!

# Choosing the method

The first method is to use a Directory attack. As we have been advised, the password combination is simple, made up of between 4 and 5 characters. A Directory attack needs a wordlist of as many passwords as possible. There are many wordlists available online, mostly from data breaches which have occurred in companies over the last few years (Walker, 2016).

The user can also create their own list for a Character Set Attack using a Linux tool crunch. This will populate a list with whatever parameters requested. Below is an example of how to create a list with minimum word size of 4 and maximum word size of 5 and use the character set of all the alphabet and numbers 0-9. An output file will be created called mix-wordlist.txt.

*/usr/bin/crunch 3 3 -f /usr/share/crunch/charset.lst mixalpha-numeric-all-space -o mix\_wordlist.txt*

I have chosen to use the rockyou list (BOWES, 2016).

This list is the result of a hack on rockyou.com. It had a data breached which resulted in the exposure of over 32 Million user accounts. To compound the severity of the security breach, it was found that RockYou are storing all user account data in plain text in their database, exposing all that information to attackers (Cubrilovic, 2016).

# Stages of the crack

There are two main stages of cracking the file.

Stage one is using office2john paython program: Pass your file into office2john a python program; it extract the hashes. One example is:

*365-2013-password.docx:$office$\*2013\*100000\*256\*16\*d4fc9302eedabf9872b24ca700a5258b\*7c9554d582520747ec3e872f109a7026\*1af5b5024f00e35eaf5fd8148b410b57e7451a32898acaf14275a8c119c3a4fd*

Stage two uses the Hash and world list, and inputs this hash and other options into john.

./john --wordlist=rockyou.lst doc.hash

### Results

The two passwords were cracked over a few hours.

The two passwords are:

MDF-S1.docx: 4321

MDF-SF2.docx: 1a2b3

MD5 Hash of both documents after cracking:

MDF-SF1.docx : dd06b1c03f4ad67c99d4f371f4001fa4

MDF-SF2.docx : 69a6db156d051c48e655d5a95cff84b4

# Other programs

* **Advanced Office Password**

I used advanced office password recovery tool in windows.

The results were the same. As I knew the passwords were simple, I could set the tool to use a limited set of characters which speeds the recovery immensely. The hash of the files as excepted did not change:

MDF-SF1.docx : dd06b1c03f4ad67c99d4f371f4001fa4

MDF-SF2.docx : 69a6db156d051c48e655d5a95cff84b4

* **Hashcat**

This is another Linux tool which like john need the hashes already exported for it to work I again used the office2john.py tool . The command is:

cudaHashcat64.exe -a 0 -m 9400 --username -o found.txt hash.txt pass.txt

MDF-SF1.docx : dd06b1c03f4ad67c99d4f371f4001fa4

MDF-SF2.docx : 69a6db156d051c48e655d5a95cff84b4

Conclusion

I did not have the skills to write a program to do this. Whilst researching the process, I found it over complicated, with the use of multiple programs and lists. I decided to automate the process. I wrote a Linux script. It will ask the user for the office file name and which word list you want to use. It will then pass the details to office2john and output the results to a file, then input this with outer variables into the john application. The cracking process will then start automatically.

I have included a small word list, but it only has ~3500 most common password.

The script is called office2john2john.sh and is available as complete package at github:

<https://github.com/rdunne/lab3>

For the script to work one must change the right on offcie2john2john.sh

*chmod 744 offcie2john2john.shR*

Run it by calling the script *./offcie2john2john.sh*

# Appendix

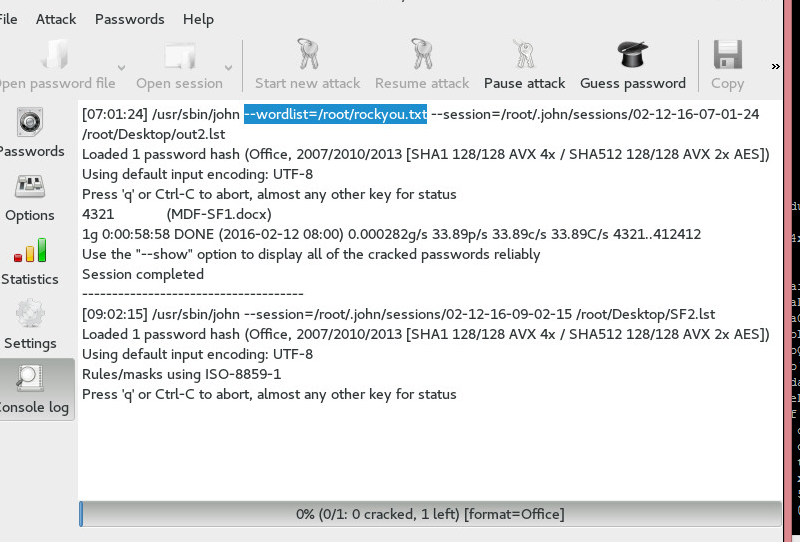


Figure 1: using john to crack file MDF1

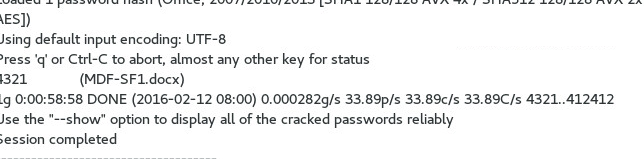


Figure 2: first hash cracked

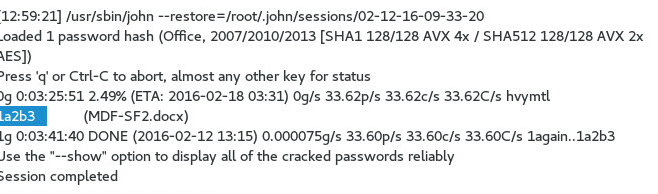


Figure 3: second hash cracked

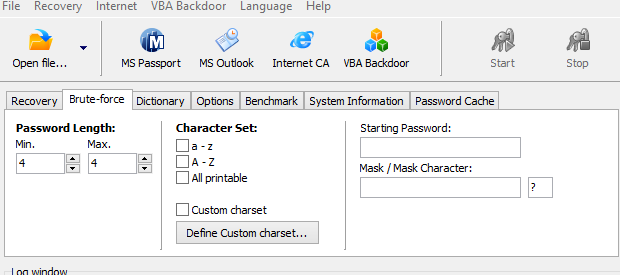


Figure 4: office recovery tool set for a 4X4 attack

Office2john2john.sh

*#!/bin/bash*

*echo -n "Enter your office document[ENTER]: "*

*read doc*

*echo -n "Enter your wordlist, if none just leave blank[ENTER]: "*

*read list*

*echo "You entered:$doc"*

*python office2john.py $doc > $doc.hash*

*echo " your hash is now stored in $doc.hash"*

*./john --wordlist=$list --session=`date +%Y%m%d\_%H%M` $doc.hash*

Figure 5: office2john2john script

# References

BOWES, R. (2016, 2 19). *skullsecurity*. Retrieved from skullsecurity: http://downloads.skullsecurity.org/passwords/rockyou.txt.bz2

Cubrilovic, N. (2016, 2 19). *RockYou Hack: From Bad To Worse*. Retrieved from techcrunch.com: http://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/

Leblanc, D. (2016, 2 15). *blogs.msdn.com*. Retrieved from blogs.msdn.com: http://blogs.msdn.com/b/david\_leblanc/archive/2008/12/05/office-crypto-kdf-details.aspx

microsoft. (2016, 2 15). *technet*. Retrieved from microsoft: http://technet.microsoft.com/en-us/library/cc179080.aspx

Walker, D. (2016, 2 10). *scmagazine.com*. Retrieved from research-helps-companies-determine-if-theyve-suffered-data-leaks: http://www.scmagazine.com/research-helps-companies-determine-if-theyve-suffered-data-leaks/article/380063/