

---

# **COMS3000 – Information Security**

---

## **Assignment 3 – Report on The Security of the most Common Authentication Credential (Passwords)**

### **TEAM CRACKERS**

Jun Hong Marcus CHAN (4505737)

Jane PHOON (44591623)

Shao Qi TAN (44627755)

Zhi Heng TAN (45108635)

Sze Yan WONG (44592228)



School of Information Technology and Electrical Engineering

University of Queensland, Australia

Semester 2, 2018

## Table of Contents

1. Members.....	2
2. Solution .....	3
2.1 Tools.....	3
2.1.1 John the Ripper .....	3
2.1.2 Hashcat .....	4
2.2 Dictionary.....	5
2.3 Platform .....	6
3. Configurations.....	9
3.1 John the Ripper Configuration.....	9
3.1.1 “John the Ripper” Command lines .....	9
3.1.2 Guessing .....	10
3.1.3 John the Ripper Single Crack Mode.....	11
3.2 Hashcat Configuration.....	13
3.2.1 Dictionary Attack .....	14
3.2.2 Rule-based Attack .....	15
3.2.3 Hybrid Attack .....	16
4. Results.....	17
4.1 Details .....	17
4.2 Explanation .....	20
References .....	22
Appendices.....	23
Appendix A: More JTR Command Lines .....	23
Appendix B: Null Results from password cracking .....	24
Appendix C: Null Results from brute-force date cracking .....	26

# 1. Members

*Table 1: List of members and work done*

<b>Name</b>	<b>Work Done</b>
Jun Hong Marcus Chan (45057377)	Contributed to Single Crack Mode on John the Ripper Contributed to Dictionary Attack using “rockyou.txt” on Hashcat Contributed to Rule-based Attack using “leetspeak.rule” on Hashcat Contributed to Hybrid attack (Wordlist + mask) on Hashcat Contributed to Guessing Password on Johnny Contributed to documentation
Tan Zhi Heng (45108635)	Contributed to Single Crack Mode on John the Ripper Contributed to Dictionary Attack using “rockyou.txt” on Hashcat Contributed to Rule-based Attack on Hashcat Contributed to Wordlist + Rule-based Attack on Hashcat Contributed to documentation
Shao Qi Tan (44627755)	Contributed to Rule-based Attack on Hashcat Created custom wordlist for Rule-based Attack Contributed to Wordlist + Rule-based Attack on Hashcat Contributed to Dictionary Attack on Hashcat Contributed to documentation
Jane Phoon (44591623)	Contributed to Rule-based Attack on Hashcat Contributed to Brute-Force Attack + Mask for dates on Hashcat Created custom wordlist related to course for Rule-based Attack Contributed to documentation
Sze Yan Wong (44592228)	Contributed to Dictionary Attack on Hashcat Contributed to documentation

## 2. Solution

### 2.1 Tools

#### 2.1.1 John the Ripper

Johnny is a graphical user interface for John the Ripper that can be found in Kali Linux machine or installed on a Windows machine. Figures 2.1.1A and 2.1.1B displays the user interface of Johnny.



Figure 2.1.1A: Navigation Bar

72	<input checked="" type="checkbox"/>	s4391975		\$6\$o2lrebv2\$NAR/...	sha512crypt,crypt	548:548:s43919...
73	<input checked="" type="checkbox"/>	s4353334		\$6\$wcYGd2S0\$Gx2...	sha512crypt,crypt	549:549:s43533...
74	<input checked="" type="checkbox"/>	s4354198		\$6\$Sf7cPx2d\$I7h.iT...	sha512crypt,crypt	550:550:s43541...
75	<input checked="" type="checkbox"/>	s4391364		\$6\$3Co6Wieg\$2wb...	sha512crypt,crypt	551:551:s43913...
76	<input checked="" type="checkbox"/>	s4391349		\$6\$OmKTMZuC\$ET...	sha512crypt,crypt	552:552:s43913...
77	<input checked="" type="checkbox"/>	s4407495	12PASSWORD	\$6\$VUV0sjR\$bpgg9...	sha512crypt,crypt	553:553:s44074...
78	<input checked="" type="checkbox"/>	s4436755		\$6\$cn2PMFyf\$5iD9...	sha512crypt,crypt	554:554:s44367...
79	<input checked="" type="checkbox"/>	s4435548		\$6\$0MLnrC2v\$GIEF...	sha512crypt,crypt	555:555:s44355...
80	<input checked="" type="checkbox"/>	s4265235		\$6\$RMgfjeT2\$90/Pl...	sha512crypt,crypt	556:556:s42652...
81	<input checked="" type="checkbox"/>	s3047038		\$6\$TQKNazUs\$gm...	sha512crypt,crypt	557:557:s30470...
82	<input checked="" type="checkbox"/>	s4391499		\$6\$2b4lmHg5\$Qpx...	sha512crypt,crypt	558:558:s43914...
83	<input checked="" type="checkbox"/>	s4288377		\$6\$IW7jZfZM\$om5...	sha512crypt,crypt	559:559:s42883...
84	<input checked="" type="checkbox"/>	s4541624		\$6\$t/7jbxOg\$tnbx4/...	sha512crypt,crypt	560:560:s45416...
85	<input checked="" type="checkbox"/>	s4395021		\$6\$L3Xyy6XZ\$/ecH...	sha512crypt,crypt	561:561:s43950...
86	<input checked="" type="checkbox"/>	s4394947	jeff21	\$6\$y11Dqavc\$sgbf...	sha512crypt,crypt	562:562:s43949...
87	<input checked="" type="checkbox"/>	s4358038		\$6\$NhtJm4De\$Tk8...	sha512crypt,crypt	563:563:s43580...
88	<input checked="" type="checkbox"/>	s4437058		\$6\$C3TjGGs1\$VcD...	sha512crypt,crypt	564:564:s44370...
89	<input checked="" type="checkbox"/>	s4322603	Password1!	\$6\$9gle0TTi\$PMN...	sha512crypt,crypt	565:565:s43226...

Figure 2.1.1B: Storing of password

The advantage of using Johnny in Kali is that the Graphical User Interface is intuitive compared to the others and it can be used for offline password cracking that has a neat way to present the cracked password as seen in Figure 2.1.1B above.

## 2.1.2 Hashcat

Hashcat is a much preferable offline password cracker for the team as it was updated recently on 7<sup>th</sup> October 2018 according to its official website. Since it was updated recently, it is much more up to date than John the Ripper. A screenshot of the GUI is as shown in Figure 2.1.2A.

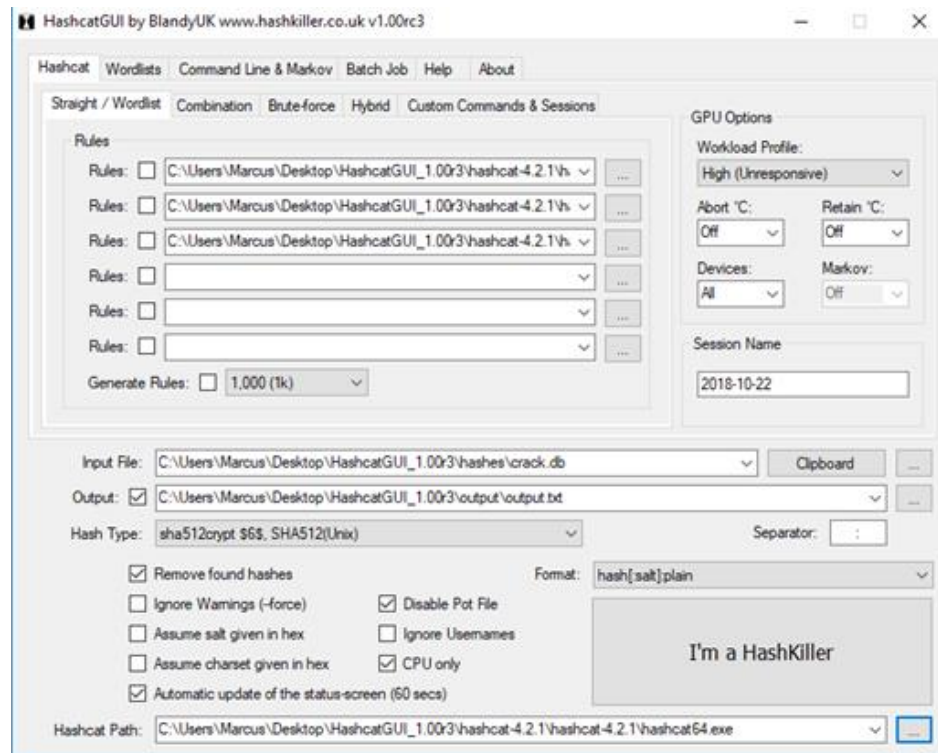


Figure 2.1.2A: HashcatGUI

There exist a third-party Graphical User Interface as well called HashcatGUI as seen in Figure 2.1.2A above which is a much simple way to run Hashcat instead of a command line interface.

The advantage of using a Graphic User Interface compared to using a command line interface is that it allows easy access in the modification of the command line base on specific clickable settings.

The disadvantage of Hashcat, however, is that the digest and salt have to be extracted separately for it to run smoothly.

## 2.2 Dictionary

There are many dictionaries available on the internet such as phpbb, phpbb with md5, Hotmail.txt and more but they are either too small or isn't clearly understood how they were built from or stolen, therefore, the dictionary word list that we have chosen is "rockyou.txt".

The advantage of "Rockyou.txt" is that it contains 14 million real passwords from the "RockYou!" social application hack.

However, the disadvantage is that the 14 million real passwords would not be able to match all 196 passwords in the given shadow9 file. Additionally, processing the high amount of passwords in the "Rockyou.txt" certainly requires an extended period to complete the course.

An additional dictionary that has been used is a custom wordlist that is thought up by the team members, such as username, "newpassword" and other possible passwords that could be thought of. Passwords that are more likely to be used by the students of COMS3000, such as using the students' name list and birthdays or using words that are related to the course provides a higher opportunity of guessing the right password. However, the possible combinations of words in the dictionary are limited which would not be able to fully utilise the parallel power of the device used for cracking.

## 2.3 Platform

The environment platforms that we used to attempt the password cracking are on a Windows machine, Macintosh and a Virtual Machine with all the specifications shown in Tables 2.4A to 2.4H.

*Table 2.4A: Device specifications 1*

<b>Marcus's PC</b>	
CPU	Intel® Core™ i5-4690K CPU @ 3.50Ghz (4 CPUs), 3.50Ghz
RAM	8.00GB
OS	Windows 10 Education 64-Bit

*Table 2.4B: Device specifications 2*

<b>Marcus's Laptop</b>	
CPU	Intel® Core™ i5-7200U CPU @ 2.50Ghz (4 CPUs), 2.70Ghz
RAM	8.00GB
OS	Windows 10 Home 64-Bit

*Table 2.4C: Device specifications 3*

<b>Marcus's VM</b>	
CPU	2 CPUs
RAM	2.00GB
OS	Kali Linux

*Table 2.4D: Device specifications 4*

<b>Shao Qi's Laptop</b>	
CPU	Intel® Core™ i7-8550U CPU @ 1.80GHz 1.99GHz
RAM	8.00GB
OS	Windows 10 Home 64-bit

*Table 2.4E: Device specifications 5*

<b>Shao Qi's Laptop 2</b>	
CPU	Intel(R) Core™ i7-6700HQ CPU @ 3.10Ghz (4 CPUs), 3.3Ghz
RAM	16.00GB
OS	MacOS Mojave v10.14

*Table 2.4F: Device specifications 6*

<b>Jane's Laptop</b>	
CPU	Intel® Core™ i5-6200U CPU @ 2.30GHz 2.40GHz
RAM	4.00GB
OS	Windows 10 Home 64-bit



*Table 2.4G: Device specifications 7*

<b>Zhi Heng's Laptop</b>	
CPU	Intel® Core™ i7-7700U CPU @ 2.80GHz 2.81GHz
RAM	8.00 GB
OS	Windows 10 Home 64-bit

*Table 2.4H: Device specifications 8*

<b>Sze Yan's Laptop</b>	
CPU	Intel(R) Core™ i3-4005U CPU @ 1.70GHz 1.70Ghz
RAM	4.00GB
OS	Windows 10 Home 64-bit

## 3. Configurations

### 3.1 John the Ripper Configuration

#### 3.1.1 “John the Ripper” Command lines

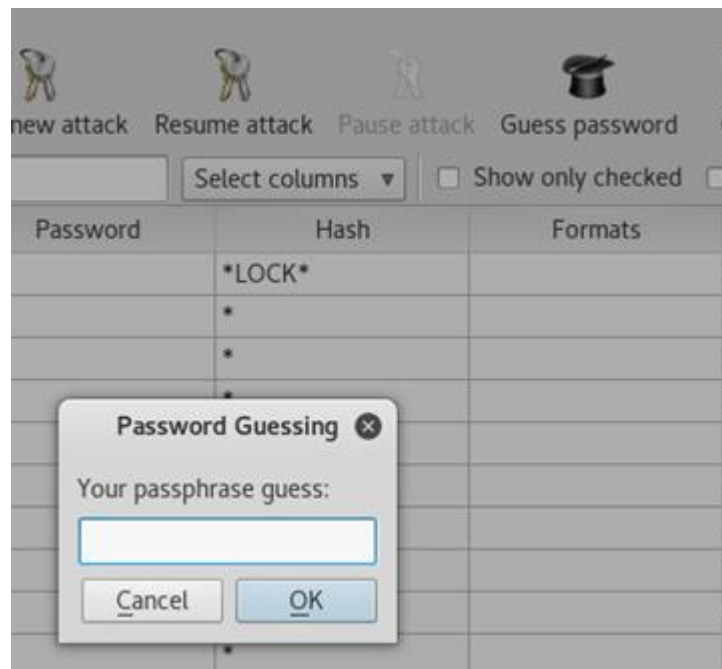
Table 3.1.1A shows the command lines that were used for John the Ripper password cracking [1].

*Table 3.1.1A: Command lines for JTR*

Commands	Description	Functions
--single	"single crack" mode	Rules from configuration file section
--wordlist=FILE --stdin	Wordlist mode compares words from a text file to the file given.	To enable wordlist mode
--rules	Word mangling rules from wordlist mode	[List.Rules: Wordlist].
--incremental[=MODE]	Section Mode used	(section [Incremental:MODE])..
--external=MODE	Filter out words	[List.External:MODE].
--stdout[=LENGTH]	Showing results of candidate's passwords	Show candidate passwords when a cracking mode is used except for single cracking mode.
--restore[=NAME]	Restore to the last session (provided that the session is created and named)	When a session is interrupted, it will reload that session
--session=NAME	Name a new session	Useful if multiple cracking sessions are running and the session that crashed or stopped could be continued.
--status[=NAME]	Show the status of a session	It displays the status of a running session (e.g. estimated time, temperature, etc.).

--show	Show passwords that are cracked	It outputs the passwords from the password files that has been mentioned. This is a way to have a quick check of what has been completed by the system to date.
--------	---------------------------------	---

### 3.1.2 Guessing



*Figure 3.1.2A: Guessing of password*

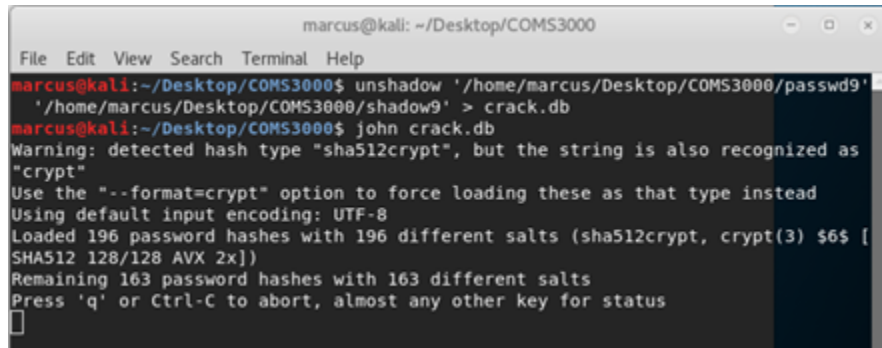
Through the use to Johnny, the graphical user interface of Johnny the Ripper, it has a feature called “Guess Password”. It tries to match the hashes against the user’s key in single guess password as seen in Figure 3.1.2A.

A total of 12 passwords were guessed such as “COMS3000” and “COMS3000/COMS7003.”

### 3.1.3 John the Ripper Single Crack Mode

*Table 3.1.3A: Steps for Single Crack Mode*

Step 1:	<code>unshadow [passwd9 Location] [shadow9 File Location] &gt; [Unshadow file name]</code>
Step 2:	<code>john [Unshadow file name]</code>



```
marcus@kali: ~/Desktop/COMS3000
File Edit View Search Terminal Help
marcus@kali:~/Desktop/COMS3000$ unshadow '/home/marcus/Desktop/COMS3000/passwd9'
'/home/marcus/Desktop/COMS3000/shadow9' > crack.db
marcus@kali:~/Desktop/COMS3000$ john crack.db
Warning: detected hash type "sha512crypt", but the string is also recognized as
"crypto"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 196 password hashes with 196 different salts (sha512crypt, crypt(3) $6$ [
SHA512 128/128 AVX 2x])
Remaining 163 password hashes with 163 different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
```

*Figure 3.1.3A Single Crack Mode on John the Ripper in kali*

John the Ripper single crack mode is used as the first step in cracking attempts with the methods displayed in Figure 3.1.1.3A and Figure 3.1.1.3A. It uses information such as the username and password. Using the username as duffer5 and running Single crack mode on John the Ripper, it will use the string “duffer5” and test it against the hash to see whether the password matches. [2]

Six passwords were obtained by using Single Crack mode. Out of the six passwords, five can be found using a dictionary attack as well. The reason John The Ripper single crack mode found six is that it applies a vast set of mangling rules. Besides, passwords that are guessed will be tried against all loaded password hashes just in case other users in the system (or in this case: course) uses the same password. [2]

Figure 3.1.3B shows the successfully cracked passwords through the Single Crack mode.

```
Command Prompt
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Eugene_PC>cd C:\Program Files\JohnTheRipper\john180j1w\run

C:\Program Files\JohnTheRipper\john180j1w\run>john --session=eugene shadow9
1 [main] john 5436 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
the public mailing list cygwin@cygwin.com
Warning: hash encoding string length 98, type id $6
appears to be unsupported on this system; will not load such hashes.
Loaded 196 password hashes with 196 different salts (sha512crypt, crypt(3) $6$ [SHA512 32/32 OpenSSL])
Remaining 192 password hashes with 192 different salts
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:06:06 72.74% 1/3 (ETA: 00:28:12) 0g/s 997.3p/s 997.3c/s 997.3C/s 9999966..s9999968
0g 0:00:13:02 0.60% 2/3 (ETA: 2018-10-15 12:23) 0g/s 713.4p/s 988.7c/s 988.7C/s rangers..88888888
qazwsx (s4397013)
!@#$$%^ (s4511723)
Password1 (duffer3)
Marvin (s4370752)
hunter2 (s4390551)
asdzxc (s4510863)
6g 0:09:29:14 3/3 0.000175g/s 21.81p/s 1046c/s 1046C/s antite..albina
6g 0:10:49:24 3/3 0.000153g/s 19.81p/s 1047c/s 1047C/s abubar..arcura
6g 0:10:53:22 3/3 0.000153g/s 19.72p/s 1047c/s 1047C/s bubrix..burley
6g 0:10:53:25 3/3 0.000153g/s 19.72p/s 1047c/s 1047C/s bubrix..burley
Use the "--show" option to display all of the cracked passwords reliably
Session aborted

C:\Program Files\JohnTheRipper\john180j1w\run>john --show
1 [main] john 1684 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
the public mailing list cygwin@cygwin.com
Password files required, but none specified
```

*Figure 3.1.3B Single Crack Mode on John the Ripper in Windows*

## 3.2 Hashcat Configuration

Table 3.2A displays the Hashcat configuration used for the attacks [3].

*Table 3.2A: Hashcat Command Lines*

<b>Configurations</b>	<b>Description</b>
-a 0 -a 6	-a with a value of '0' designated as a dictionary attack  -a with a value of '6' designated as a hybrid attack (wordlist + appended mask)
--session=[SESSION NAME]	Name of session name to continue cracking password after shutting down the machine
-m 1800	-m with a value of '1800' designated the type of hash that we are cracking which is sha512crypt.
-w 3	-w with a value '3' designated the type of Workload Profile which is set to High. Using a high workload carries the advantage of increasing the speed to crack passwords, but it will heavily burden the CPU which causes the inability to run other programmes.
-D 1	Use CPU
--status	Enable automatic update of the status screen
--status-timer=60	Update status screen for every 60 seconds to determine the status of the process
--potfile-disable	Do not store the hash of the crack password
--remove	Enable remove of cracked hashes from the overall hash file crack to reduce computation
-p :	-p with a value of ':' designated to add separator for the output file which will be in the form of Hash:Password

-o "/example.txt"	To specify the output file location
-r "/example.rule"	To specify the rule file location

### 3.2.1 Dictionary Attack

The configuration used for dictionary attacks are shown below, and Figure 3.2.1A displays the status of a dictionary attack.

hashcat64.exe -a 0 --session=[SESSION NAME] -m 1800 -w 3 -D 1 --status --status-timer=60 --potfile-disable --remove -p : -o "[OUTPUT FILE LOCATION]" --outfile-format=3 "[HASH FILE LOCATION]" "[WORDLIST FILE LOCATION]"

```

Session.....: 2018-10-23
Status.....: Running
Hash.Type.....: sha512crypt $6$, SHA512 (Unix)
Hash.Target....: C:\Users\Marcus\Desktop\HashcatGUI_1.00r3\hashes\crack.db
Time.Started...: Tue Oct 23 00:44:35 2018 (1 sec)
Time.Estimated...: Thu Oct 25 17:10:36 2018 (2 days, 16 hours)
Guess.Base.....: File (C:\Users\Marcus\Desktop\HashcatGUI_1.00r3\dict\rockyou.txt)
Guess.Queue....: 1/1 (100.00%)
Speed.Dev.#1....: 12121 H/s (51.33ms) @ Accel:128 Loops:64 Thr:32 Vec:1
Recovered.....: 0/196 (0.00%) Digests, 0/196 (0.00%) Salts
Progress.....: 0/2811499264 (0.00%)
Rejected.....: 0/0 (0.00%)
Restore.Point...: 0/14344384 (0.00%)
Candidates.#1...: 123456 -> soydivina
HWMon.Dev.#1....: Temp: 54c Fan: 0% Util: 95% Core:1328MHz Mem:3004MHz Bus:16
[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => _

```

Figure 3.2.1A: Dictionary Attack

Unlike brute-force attack, dictionary attack uses a predetermined list of words that are commonly or possibly used as a password for logins. With this, the time taken to complete the execution has greatly reduced as the trials are limited to the words in the list. However, it is unlikely to have all possible passwords in the dictionary, hence, many of them could be missed, even with one-character difference.

A total of 19 passwords were found using a dictionary attack.

### 3.2.2 Rule-based Attack

The configuration used for rule-based attacks is shown as below and Figure 3.2.2A displays the status of a rule-based attack.

hashcat64.exe -a 0 --session=[SESSION NAME] -m 1800 -w 3 -D 1 --status --status-timer=60 --potfile-disable --remove -p : -o "[OUTPUT FILE LOCATION]" --outfile-format=3 -r "[RULE FILE LOCATION]" "[HASH FILE LOCATION]" "[WORDLIST FILE LOCATION]"

```
Session.....: 2018-10-23
Status.....: Running
Hash.Type.....: sha512crypt $6$, SHA512 (Unix)
Hash.Target.....: C:\Users\Marcus\Desktop\HashcatGUI_1.00r3\HashcatGUI_1.00r3\hashes\crack.db
Time.Started.....: Tue Oct 23 00:45:54 2018 (20 secs)
Time.Estimated.....: Mon Dec 03 00:37:47 2018 (40 days, 23 hours)
Guess.Base.....: File (C:\Users\Marcus\Desktop\HashcatGUI_1.00r3\HashcatGUI_1.00r3\dict\rockyou.txt)
Guess.Mod.....: Rules (C:\Users\Marcus\Desktop\HashcatGUI_1.00r3\hashcat-4.2.1\hashcat-4.2.1\rules\leetspeak.rule)
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 13494 H/s (49.59ms) @ Accel:128 Loops:64 Thr:32 Vec:1
Recovered.....: 0/196 (0.00%) Digests, 0/196 (0.00%) Salts
Progress.....: 212992/47795487488 (0.00%)
Rejected.....: 0/212992 (0.00%)
Restore.Point.....: 0/14344384 (0.00%)
Candidates.#1.....: 123456 -> soydivina
Mon.Dev.#1.....: Temp: 63c Fan: 41% Util:100% Core:1328MHz Mem:3004MHz Bus:16

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit =>
```

Figure 3.2.2A Rule-based Attack

For rule-based attack, a rule file and a wordlist were used together for a single attack. Several different types of rules were used for each attack such as the best64 rule and the leetspeak rule.

Leetspeak rule changes the candidate word, e.g. Password as P@ssword, Pa55w0rd and other sorts of combination. Leetspeak exploits the commonality of using symbols or numbers to replace letters that are visually similar to the symbols, which is still easy to remember compared to other combinations of letters and symbols. Passwords are required to be remembered hence it is essential to apply the leetspeak rule to perform the rule-based attack.

The Best64 rule mostly appends random numbers or characters to the end or insert them into the candidate word. Besides, some of the rules rotate or delete parts of the candidate word. Some examples would be the password to password1, password123 and passw. This rule is used because it has a high percentage (32.42%) of successful cracks with a minimum amount of candidates generated from the rule compared to other rule files. It contributes a higher efficiency, which is advantageous to run on a CPU especially with a limited amount of time provided. [4]

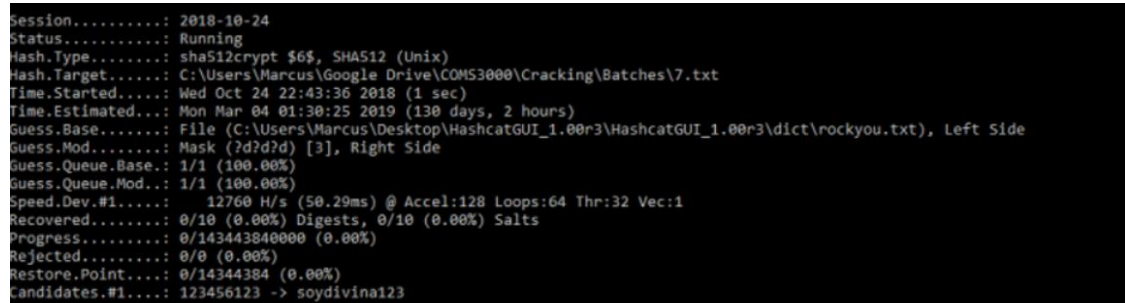
A total of 2 passwords were found using a Rule-based attack.



### 3.2.3 Hybrid Attack

The configuration used for the hybrid attack is specified as below, and Figure 3.1.2.3A displays the status of a hybrid attack.

```
hashcat64.exe -a 6 --session=[SESSION NAME] -m 1800 -w 3 -D 1 --status --status-timer=60 --  
potfile-disable --remove -p : -o "[OUTPUT FILE LOCATION]" --outfile-format=3 "[HASH  
FILE LOCATION]" "[WORDLIST FILE LOCATION]" ?d?d?d
```



```
Session.....: 2018-10-24  
Status.....: Running  
Hash.Type.....: sha512crypt $6$, SHA512 (Unix)  
Hash.Target.....: C:\Users\Marcus\Google Drive\COM53000\Cracking\Batches\7.txt  
Time.Started.....: Wed Oct 24 22:43:36 2018 (1 sec)  
Time.Estimated...: Mon Mar 04 01:30:25 2019 (130 days, 2 hours)  
Guess.Base.....: File (C:\Users\Marcus\Desktop\HashcatGUI_1.00r3\HashcatGUI_1.00r3\dict\rockyou.txt), Left Side  
Guess.Mod.....: Mask (?d?d?d) [3], Right Side  
Guess.Queue.Base.: 1/1 (100.00%)  
Guess.Queue.Mod..: 1/1 (100.00%)  
Speed.Dev.#1.....: 12760 H/s (50.29ms) @ Accel:128 Loops:64 Thr:32 Vec:1  
Recovered.....: 0/10 (0.00%) Digests, 0/10 (0.00%) Salts  
Progress.....: 0/143443840000 (0.00%)  
Rejected.....: 0/0 (0.00%)  
Restore.Point....: 0/14344384 (0.00%)  
Candidates.#1....: 123456123 -> soydivina123
```

*Figure 3.2.3A Hybrid Attack*

A hybrid attack is an attack that utilises a dictionary, brute force and mangling rules. In this era, most people would use this way of password cracking. One of the simplest formulae for passwords today consists of 1 dictionary word, uppercase first letter and add two numbers at the end. If a special character is required, add an exclamation point or full stop behind. [5]

A total of 1 password were found using a Hybrid attack (wordlist + mask).

## 4. Results

### 4.1 Details

Table 4.1A displays all the passwords cracked by the members of this team with methods specified.

*Table 4.1A: ResultSet on Cracked Password*

No	Username	Password	Summary
1	duffer5	duffer5	Johnny/John the Ripper “Single Crack Mode.”
2	duffer	password	Dictionary attack using “Rockyou.txt.”
3	duffer2	Password	
4	duffer3	Password1	
5	duffer4	password123	
6	duffer6	Duffer	
7	s4397013	qazwsx	
8	s4370752	Marvin	
9	s4407495	12PASSWORD	
10	s4394947	jeff21	
11	s4322603	Password1!	
12	s4390551	hunter2	
13	s4495817	1q2w3e4r5t	

14	s4386246	PASSWORDCHANGED	
15	s4511723	!@#\$\$%^	
16	s4433395	AmberRose1996	
17	s4435400	ianhayes	
18	s4459222	alphaomega3	
19	s4504604	mist26	
20	s4510863	asdzxc	
21	s4315128	COMS3000/7003	Guessing of password
22	s4541010	COMS3000/7003	
23	s4384569	COMS3000/7003	
24	s4490071	COMS3000/7003	
25	s4496087	COMS3000/7003	
26	s4500036	COMS3000/7003	
27	s4488954	COMS3000/7003	
28	s4528082	COMS3000/7003	
29	s4495508	COMS3000/7003	
30	s4417496	COMS3000/7003	
31	s4354619	COMS3000/7003	

32	s4425286	COMS3000	
33	s4511454	rocketma11	Rule-Based Attack using “Rockyou.txt” and a set of rule base on “leetspeak.rule”
34	s4432736	goodpassword1	Rule-based Attack using custom wordlist and set a rule base on "Best64.rule"
35	s4431293	merlin505	Hybrid attack using “Rockyou.txt” and a mask using “?d?d?d”

## 4.2 Explanation

There are different types of password such as a bad, good, better and best password [6].

A bad password is passwords to avoid such as digits (e.g. 0000 - 9999), the word “password”(e.g. password, password12, or any variant), English dictionary words (e.g. kitchen, chocolate or any variant), names(e.g. John, Jane, or any variant), personal information (email address, phone number, date of birth) and keyboard pattern (qwerty, asdf, abc123). An example of a bad password according to Table 4.1A are from numbers 1 - 20 such that password are easily obtained through a dictionary of common passwords and password from 21 - 32 are a bad password as it is the password that generally obtains through guessing.

Good passwords are password a combination of 10 or more uppercase character (A-Z), lowercase character (a-z), a number (0-9) and symbol (such as !, @, #). For example, the candidate password could be like “M4rcus\$ChAnz20” as it fit the requirement above to be a good password. An example of a good password according to Table 4.1A is from numbers 33 – 35 can be classified as a good password as the password are not found using a dictionary and requires more effort to be crack such as the use of a rule-based attack or a hybrid attack.

A better password is the password that uses passphrases, which are easier to remember, but difficult to guess. A password creator can create a password by taking the first initials of phrase for example, “COMS3000 is such an interesting course” as “C3iSaIc”. As such it is difficult to do a crack the password. There is no example from Table 4.1A as a password that uses passphrases takes a lot more effort to be crack. Those password that is not found in the shadow9 file is categorised better passwords.

Best password is the strongest password that can be created using a password manager. A password manager is a software that generates a complex and unique password for multiple accounts. The only thing you have to remember is the password to the password manager. A good password manager would enhance its security by including a 2-step verification process.

Table 4.2A summarises the reasons for each cracked passwords having low security.

*Table 4.2A: Weak Passwords Explanation*

No	Reason
1	<ul style="list-style-type: none"><li>• A Single Crack method was used</li><li>• The password is identical to its Username (hackers usually use Single Crack Mode first and match it with the Username)</li><li>• Contains a low number of characters</li><li>• Fills with only lowercase letters</li><li>• No usage of symbols</li><li>• Use of words from a dictionary</li><li>• The password has a low entropy</li></ul>
2-20	<ul style="list-style-type: none"><li>• A dictionary attack was used</li><li>• Use of words that can be found in the dictionary</li></ul>

	<ul style="list-style-type: none"> <li>Commonly used and predictable passwords</li> <li>Passwords 7,13,15 and 20 are merely keyboard patterns which are often used and easily guessed</li> <li>Passwords 2 and 17 have no combination of cases, letters and symbols that provide a higher entropy</li> </ul>
21-32	<ul style="list-style-type: none"> <li>Guessing of Password was used</li> <li>Using passwords related to the system/ course (e.g. using Facebook as a password for Facebook)</li> <li>“COMS3000/7003” is a default password</li> <li>In this case “COMS3000/7003”, “COMS3000” where the candidates to be chosen for guessing of a password.</li> </ul>
33	<ul style="list-style-type: none"> <li>A rule-based attack was used</li> <li>Words from dictionary combine with a rule such as leetspeak.</li> <li>Leetspeak rule converts characters such as A to ‘^,’ ‘@’ etc. which is commonly used to ‘enhance’ a password’s security while maintaining the ease of remembering the password.</li> <li>In this case, Rocketmail was found in the dictionary; leetspeak rule changes it to Rocketma11.</li> </ul>
34	<ul style="list-style-type: none"> <li>A rule-based attack was used</li> <li>Contain two words from a dictionary</li> <li>No Uppercase letter used</li> <li>Commonly used words “good” and “password”, would be easily cracked with a combination attack together with a rule</li> <li>Appending only a number ‘1’ at the end of the phrase, which is a commonly used rule for a rule-based attack</li> </ul>
35	<ul style="list-style-type: none"> <li>A hybrid attack was used</li> <li>Contain 1-word match in the dictionary</li> <li>The password is too short</li> <li>A mask for 000 - 999 was used for wordlist + mask to have three numbers appended behind the wordlist. E.g (xxx505). This is because appending three digits behind characters is one of the most common password patterns found in a study [5].</li> </ul>

# References

- [1] Openwall, "John the Ripper's command line syntax," Openwall, 21 January 2016. [Online]. Available: <https://www.openwall.com/john/doc/OPTIONS.shtml>. [Accessed 24 October 2018].
- [2] Openwall, "John the Ripper's cracking modes," Openwall, 29 May 2013. [Online]. Available: <https://www.openwall.com/john/doc/MODES.shtml>. [Accessed 24 October 2018].
- [3] Hashcat, "Hashcat - Options," Hashcat, [Online]. Available: <https://hashcat.net/wiki/doku.php?id=hashcat>. [Accessed 21 October 2018].
- [4] Not So Secure, "One Rule to Rule Them All," Not So Secure, 1 June 2017. [Online]. Available: <https://www.ntsossecure.com/one-rule-to-rule-them-all/>. [Accessed October 24 2018].
- [5] K. Komando, "How to create a strong password," 20 May 2015. [Online]. Available: <https://www.usatoday.com/story/tech/columnist/komando/2015/05/15/strong-passwords/27240877/>. [Accessed 24 October 2018].
- [6] Harvard Information Security, "Use Strong Passwords," Harvard Information Security, [Online]. Available: <https://security.harvard.edu/use-strong-passwords>. [Accessed 24 October 2018].
- [7] M. Dobinson, "Information Security - Password Cracking," [Online]. Available: [https://learn.uq.edu.au/bbcswebdav/pid-3914378-dt-content-rid-17130646\\_1/courses/COMS3000S\\_6860\\_60706/Matt\\_Dobinson\\_Passwords.pdf](https://learn.uq.edu.au/bbcswebdav/pid-3914378-dt-content-rid-17130646_1/courses/COMS3000S_6860_60706/Matt_Dobinson_Passwords.pdf). [Accessed 24 October 2018].

# Appendices

## Appendix A: More JTR Command Lines

Command	Description	Function
--make-charset=FILE	Charset is written to overwrite a FILE	-
--test[=TIME]	Tests and benchmarks are run as TIME seconds each	Tests are hashing algorithms and benchmarks them.
--users=[-]LOGIN UID[,..]	Tells the system not to load specific users	The selection of accounts for cracking.
--groups=[-]GID[,..]	Load users that are or not from a specific group.	John is told to load or not to load information for accounts in a specific group.
--shells=[-]SHELL[,..]	Load users with or without shells	Useful to show accounts with a valid shell or not load accounts with bad shells.
--salts=[-]N	Load salts with or without a certain amount of password.	To have better performance in some cases and to be faster. For example, some salts are faster.
--save-memory=LEVEL	Memory saving enabled at Level 1 to 3	This option is beneficial if you do not have enough memory on your PC and saves a lot but might slow down the process and performance.
--node=MIN[-MAX]/TOTAL	Node's number range / TOTAL count	-



--fork=N	fork N processes	-
--format=NAME	force hash type NAME	-

(Table A.1: Additional JTR Command lines)

## Appendix B: Null Results from password cracking

```
C:\Windows\System32\cmd.exe

* Update your OpenCL runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

Session.....: 2018-10-15
Status.....: Exhausted
Hash.Type.....: sha512crypt $6$, SHA512 (Unix)
Hash.Target.....: C:\Users\Eugene_PC\Desktop\3000\hashes\1.txt
Time.Started.....: Mon Oct 15 00:51:47 2018 (6 hours, 33 mins)
Time.Estimated...: Mon Oct 15 07:25:06 2018 (0 secs)
Guess.Base.....: File (C:\Users\Eugene_PC\Desktop\3000\Rockyou\rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 6078 H/s (19.16ms) @ Accel:256 Loops:64 Thr:32 Vec:1
Recovered.....: 0/10 (0.00%) Digests, 0/10 (0.00%) Salts
Progress.....: 143443840/143443840 (100.00%)
Rejected.....: 0/143443840 (0.00%)
Restore.Point....: 14344384/14344384 (100.00%)
Candidates.#1....: $HEX[2321676f7468] -> $HEX[042a0337c2a156616d6f732103]
HWMon.Dev.#1.....: Temp: 50c Util: 0% Core:1670MHz Mem:3504MHz Bus:8

Started: Mon Oct 15 00:51:36 2018
Stopped: Mon Oct 15 07:25:08 2018
```

Figure B.1: Dictionary Attack with rockyou.txt

Figure B.1 shows that part of the hashes extracted from Shadow9 with Rockyou.txt but no results.

```

Approaching final keyspace - workload adjusted.

Session.....: 2018-10-15
Status.....: Exhausted
Hash.Type.....: sha512crypt $6$, SHA512 (Unix)
Hash.Target.....: C:\Users\Eugene_PC\Desktop\3000\hashes\2.txt
Time.Started.....: Mon Oct 15 22:35:01 2018 (6 hours, 33 mins)
Time.Estimated...: Tue Oct 16 05:08:23 2018 (0 secs)
Guess.Base.....: File (C:\Users\Eugene_PC\Desktop\3000\Rockyou\rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 6078 H/s (19.04ms) @ Accel:256 Loops:64 Thr:32 Vec:1
Recovered.....: 0/10 (0.00%) Digests, 0/10 (0.00%) Salts
Progress.....: 143443840/143443840 (100.00%)
Rejected.....: 0/143443840 (0.00%)
Restore.Point....: 14344384/14344384 (100.00%)
Candidates.#1....: $HEX[2321676f7468] -> $HEX[042a0337c2a156616d6f732103]
HWMon.Dev.#1.....: Temp: 54c Util: 85% Core:1670MHz Mem:3504MHz Bus:8

Started: Mon Oct 15 22:34:49 2018
Stopped: Tue Oct 16 05:08:25 2018

```

*Figure B.2: Dictionary Attack with rockyou.txt*

The second part of the hashes extracted from Shadow9 with Rockyou.txt but no results obtained as displayed in Figure B.2.

```

Session.....: 2018-10-24
Status.....: Exhausted
Hash.Type.....: sha512crypt $6$, SHA512 (Unix)
Hash.Target.....: C:\Users\Eugene_PC\Desktop\3000\hashes\3.txt
Time.Started.....: Wed Oct 24 01:24:15 2018 (9 hours, 17 mins)
Time.Estimated...: Wed Oct 24 10:41:24 2018 (0 secs)
Guess.Base.....: File (C:\Users\Eugene_PC\Desktop\3000\Rockyou\rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 5999 H/s (19.39ms) @ Accel:256 Loops:64 Thr:32 Vec:1
Recovered.....: 0/14 (0.00%) Digests, 0/14 (0.00%) Salts
Progress.....: 200821376/200821376 (100.00%)
Rejected.....: 0/200821376 (0.00%)
Restore.Point....: 14344384/14344384 (100.00%)
Candidates.#1....: $HEX[2321676f7468] -> $HEX[042a0337c2a156616d6f732103]
HWMon.Dev.#1.....: Temp: 58c Util: 82% Core:1645MHz Mem:3504MHz Bus:8

Started: Wed Oct 24 01:24:03 2018
Stopped: Wed Oct 24 10:41:25 2018

```

*Figure B.3: Dictionary Attack with rockyou.txt*

As shown in Figure B.3, the third part of the hashes extracted from Shadow9 with Rockyou.txt and but no results obtained.

## Appendix C: Null Results from brute-force date cracking

Brute-force attack with date combinations of 6 digits and eight digits were performed in the format of dd-mm-yy and dd-mm-yyyy. The mask file used contains the following: 012,8901,?1?d?1?d?2?d and the hashcat configuration used is displayed as Figure C.1 below.

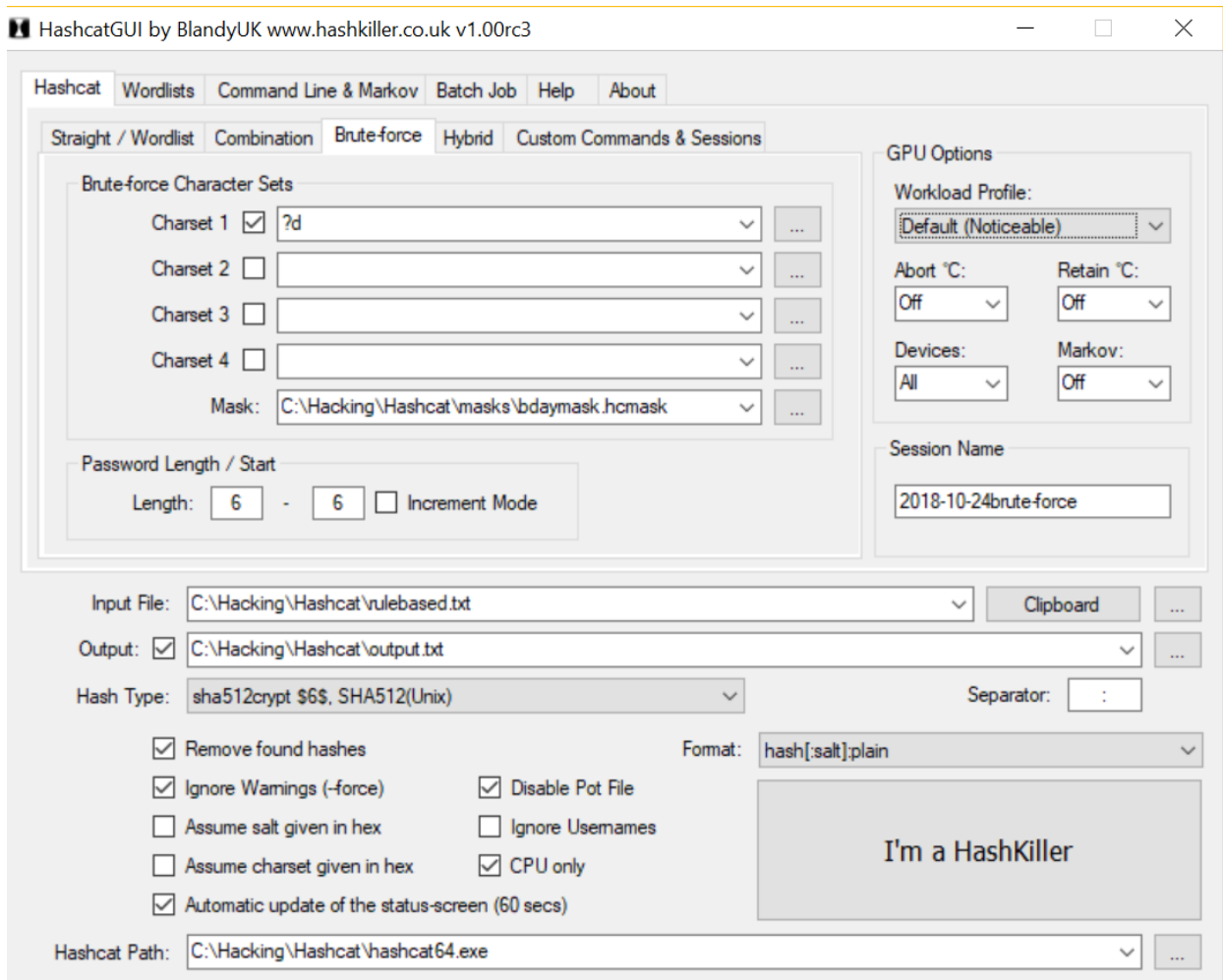


Figure C.1: Configurations of Brute-Force Attack on Hashcat GUI

However, this method did not manage to crack any passwords from the shadow file.