CSCI 5742 Cybersecurity Programming
HW1
Matt Sullivan

CSCI 5742 Cyber Lab 0, Password breaking/retrieval.

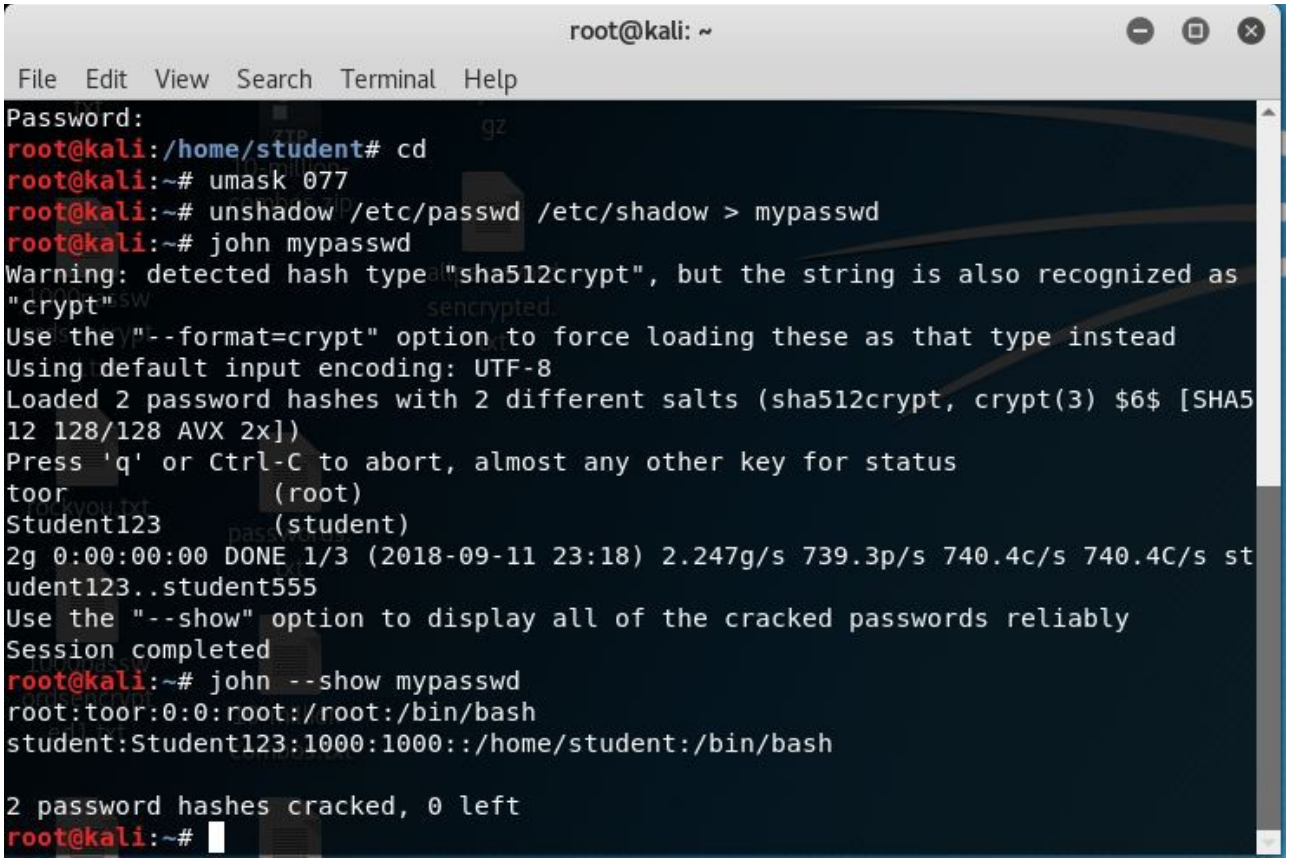
Name _____
Goals:

- Become familiar with John the Ripper
- Understand the various methods of password hacking
- Understand what constitutes a strong password, and why

Materials: Kali Linux Virtual Machine, internet connection

Part 1: Introduction to John the Ripper
In this part, we will become acquainted with John the Ripper. John is a comprehensive versatile password recovery program. It can recognize many types of encryption and inputs, and uses a variety of tactics to decode passwords. It is also very customizable. We are going to start by using the basic functionality of John the ripper to crack the sign in passwords on our virtual linux machine.

- First, we are going to turn on our Linux virtual machine. You can use the default settings, but something to note is that if the files are not on your machine already, you should make sure that the Network Adapter to NAT (so we can sign on to the internet to download the files)
- Log on to the Kali linux as **student** with password **Student123**.
- Open the command prompt.
- Type `john` into the console, if John the Ripper is installed on the machine (as it will be with our virtual machine) this screen gives you the version, syntax, and some options
- At the prompt enter root mode by typing **su** with the password **toor**
- In the console, type `umask 077`
- **Why do we have to enter the umask command?**
  Otherwise we do not have permission to read, write, or execute the password file. We must have these permission to access the file
- Now type in `unshadow /etc/passwd /etc/shadow > mypasswd`
- This combines our password and shadow into one file
- At the console, type: `john mypasswd`
- Wait for a few seconds, and you will see the Session completed message
- It should look like this:



- To see a list of the cracked passwords, type `john --show mypasswd`
- This will display the list of passwords in the format <username>:<password> It will also tell you how many passwords have been cracked and how many are remaining.
- Also note that John keeps a record of all cracked passwords. If there are no remaining passwords to crack, John will not run on the file.
- Type `exit` to return to student mode.
- Now lets try a password of our own. In the console type `cd Desktop`
- Create a `hash.txt` file in the directory you will be working in (I suggest the desktop for ease of transferring files)
- Open Firefox
- Go to http://www.miraclesalad.com/webtools/md5.php
- Type in a password of your choosing (but keep it less than 6 characters)
- Copy and paste the returned hash into hash.txt
- In the console, type `john --format=raw-md5 hash.txt`
- If the program takes more than a few seconds, it may take up to 10 minutes to correctly decrypt the password. We will explain why in a minute. Also, the longer your password is, it will a significantly longer time take to decrypt

➢ **Why does it take so much longer to decrypt a longer password?**
Increasing the number of characters in a set is a process that grows at 2^n

- Great work! Now we will take a deeper look at how John works.

Part 2: Background and Preparation

When you call the command John, the program is actually running three different modes sequentially on your passwords. First, the Single search is used. Using the username as a basis, the search uses all of it's string mutating tools to see if the password is some variant on the username. Next, for the Wordlist search, the program uses a list of words and compares the words on the list to see if any of them match the passwords. There is a default wordlist, but it is small and we'll be replacing it with one of our own. Also note that there are by default no mutations performed on the words in the wordlist, though you can specify if you want them to apply. Thirdly and finally, there is the Incremental step. In this step John tries every possible combination (default is any printable character, but it can be narrowed if you have narrower search parameters). This of course is much slower, and (depending on the password strength) may not find an answer in a reasonable amount of time. We will examine each of these in turn, but first, we have some preparation to do.

- **From the description, what type of attack is a Wordlist search?**
  A dictionary attack
- **Why would the default on the Wordlist NOT mutate the words of the list?**
  A dictionary attack is a volume attack (many words on the list), if you combined the large number of words with the large number of variations, it would be an incredibly slow process

- Cracking one or two passwords is cool, but we really want to see what John can do. For this, we'll need a much larger set. We'll download one with 10 million actual username and password combinations, and we can take any number of those that we'd like.
- Open Firefox and go to `https://mega.nz/#!SdYnkJRJ!HmD04LH8Gk8JtlNG6O2NnF2yH9qWJPWtSXbLU2ZR9Q8`
- When Firefox prompts you, select "Save as", then open the containing folder through Firefox
- Move from your download folder to the desktop for convenience (can use the GUI)
- Unfortunately, we need to clean up the file a bit before John can go through it.
- First of all, we need to unzip the file.
- Type `unzip 10-million-combos.txt`
- Unfortunately, there are 2 utc-8 errors in the file, we'll have to clean them up.
- In your console, type `iconv -c -f 'UTF-8' -t 'UTF-8//IGNORE' 10-million-combos.txt | tr -d '\0' >10-million-combos-clean.txt`
- Type `nano 10-million-combos-clean.txt`
- It will take a minute to load, but it is much faster than other word processors
- Using `Ctrl+W`, search for `markcgilberteternity`, there will be two of them, and you'll notice that the second one doesn't have a : between the username and the password. Add the : or delete the line.
- In the same way, search for `sailer1216` and either add a colon or delete the line.
- Using pycharm or whatever editor you wish, copy and paste the two files on the next page into two different programs (I suggest pullpasswords.py and md5encrypter.py)
- Run them both in the same directory that your downloaded password list is in. Run pullpasswords first and md5encrypter second.
- Note: The programs as written will produce a list of ~10000 usernames and passwords

- The 10-million-combos-clean.txt has a tab between username and password instead of the : required by john. Our first program replaces the tab with a :.

```
# pullPasswords, replaces the “\t” between the columns with required ':'
file_object=open("10-million-combos-clean.txt", "r")
newFileObject=open("passwords.txt", "w+")

for line in file_object:
   line=line.replace('\t',':')
   newFileObject.write(line)

file_object.close()
newFileObject.close()
```

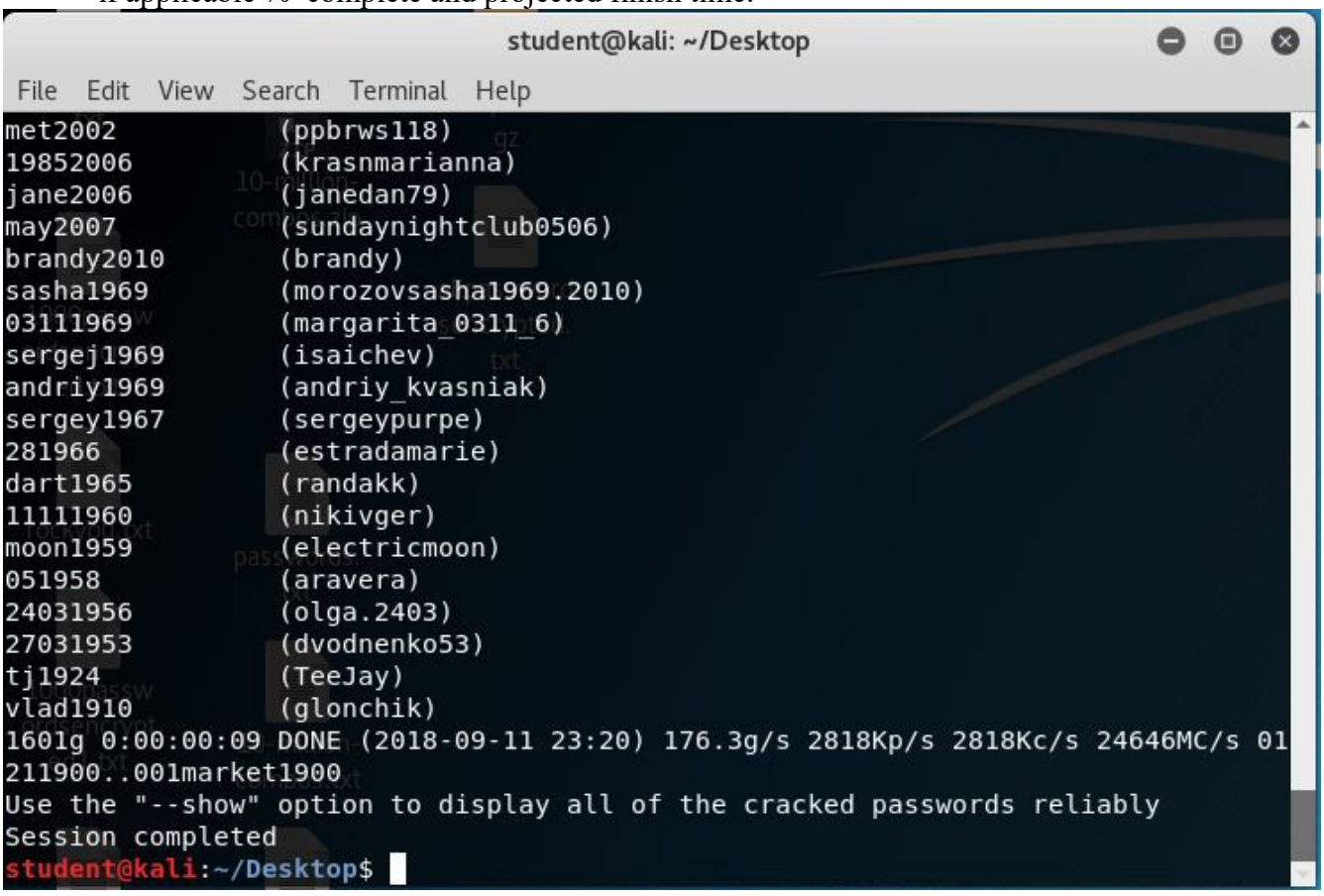
- Our second program uses the included python libraries to hash the passwords using md5.

```
#md5 encrypter
import hashlib
workFile=open("passwords.txt", "r")
newFile=open("passwordsencrypted.txt" , "w+")

def md5hash(string1):
   m=hashlib.md5()
   m.update(string1.encode('utf-8'))
   return m.hexdigest()

encryptList=[]
index =0

for line in workFile:
   if (index>=1000) and (index%1000==0):          //Note, this is where we scale the size of the list
      workLine=line.split(":")
      password=workLine[1].strip('\n')
      add=""
      add=add+(workLine[0])+(":")+(md5hash(password))
      encryptList.append(add)
      index +=1
   else:
      index +=1

for item in encryptList:
   newFile.write(item)
   newFile.write('\n')

workFile.close()
newFile.close()
```

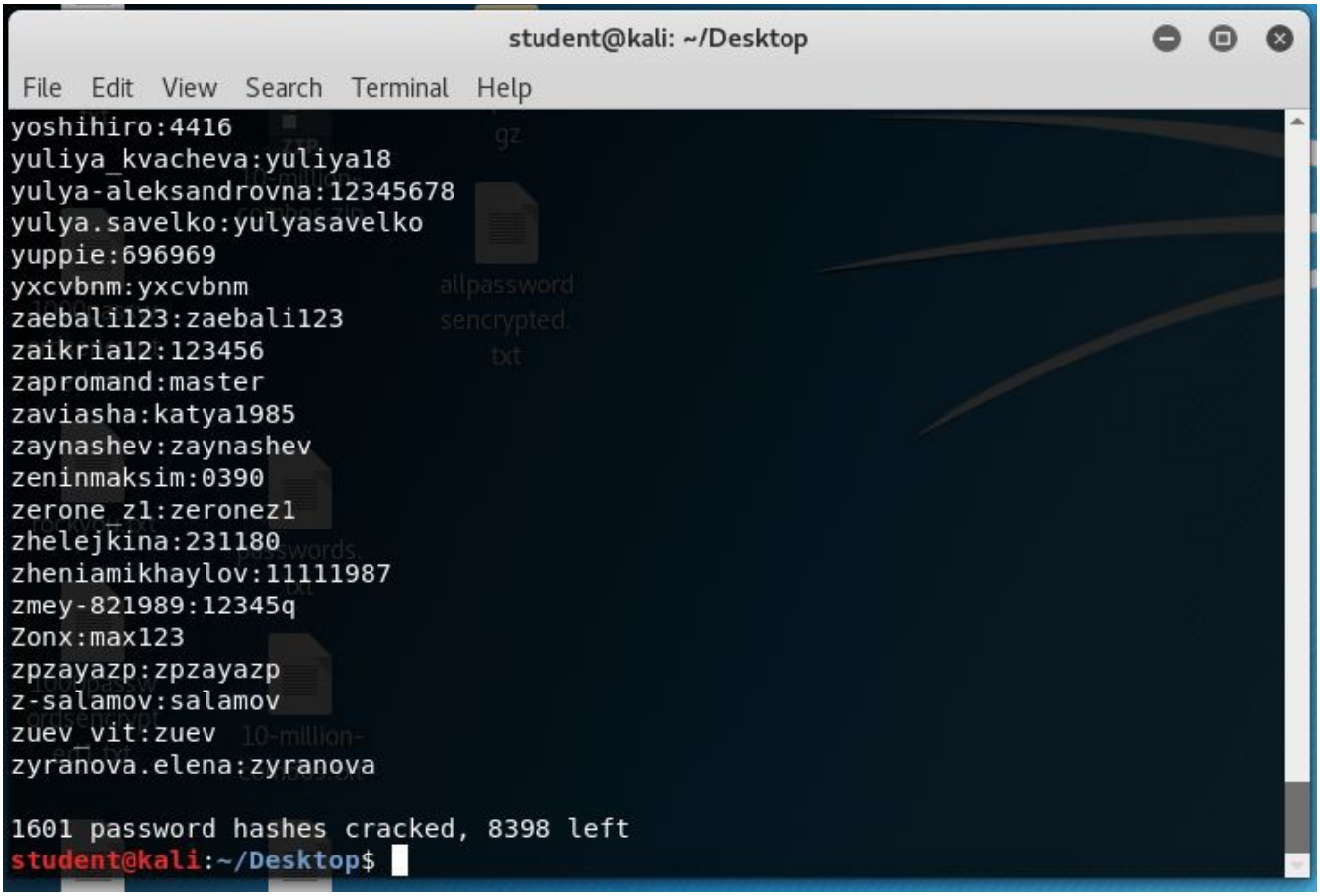
- We now have our list of passwords in an acceptable format for John to analyze.
- Next, we will acquire a larger wordlist than the tiny default one
- Kali linux comes with several wordlists that we can use, but we'll be going with the rockyou.txt
- If you're still in the desktop directory, type in `cp /usr/share/wordlists/rockyou.txt.gz ./`
- This will create a copy to the Desktop
- Now type in gunzip `rockyou.txt.gz` The rockyou.txt wordlist is now available.

Part 3: Hashing it out with John

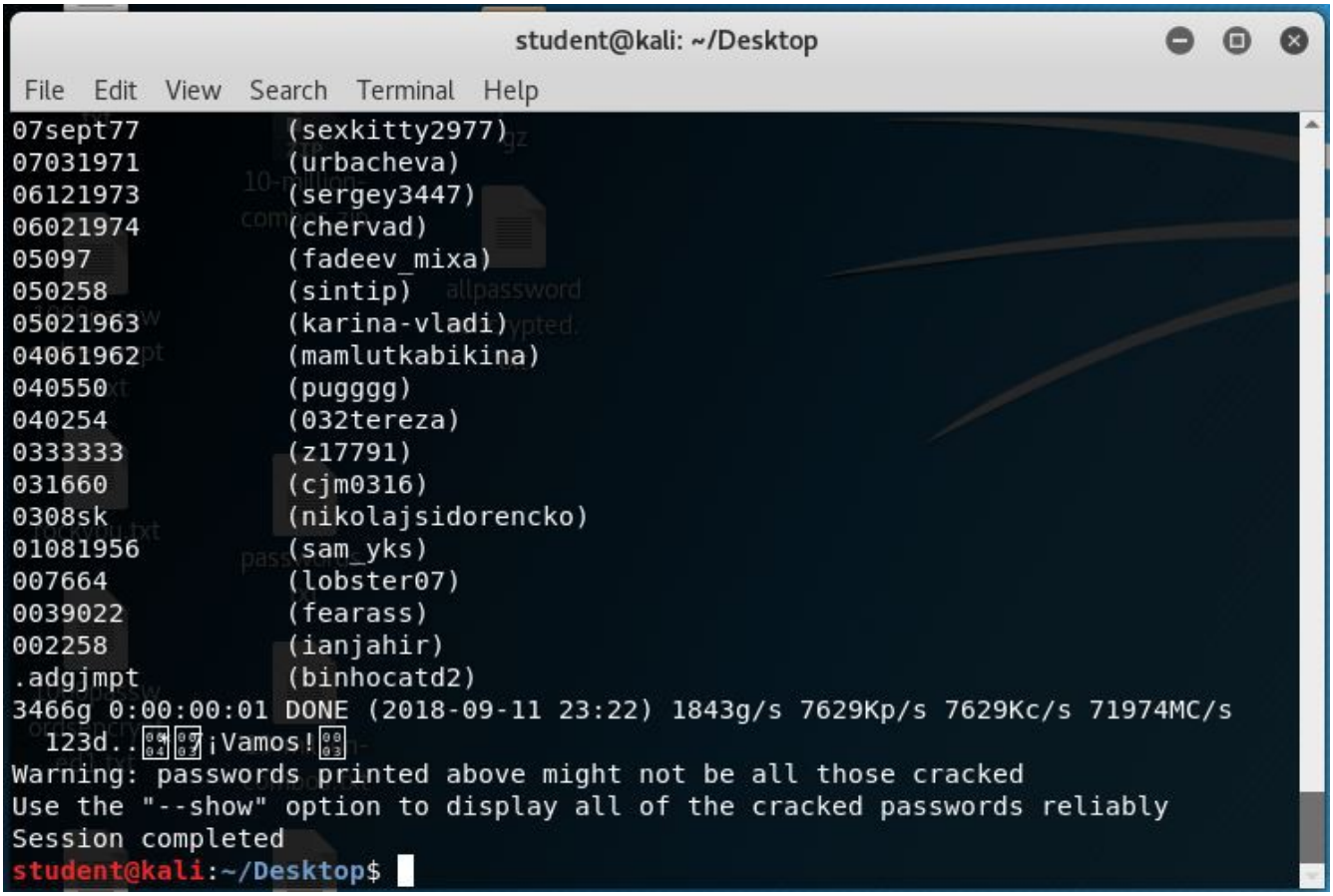
- Alright, now we just need to roll up our sleeves and get cracking. While we could pass the entire list through to John and have it be completely hands off, we want to see how each of the tests works.
- In your console type `john --single --format=raw-md5 passwordsencrypted.txt`
- John will immediately start checking the usernames versus the passwords, also mutating the username for variation.
- This is the first opportunity we have to get a mid process status update from John. Press any key other than q or ctrl+c to get a report on john's speed, hashes being tried, the execution time, and if applicable % complete and projected finish time.



- After this is completed, we can see how many passwords this method cracked by typing in `john --show --format=raw-md5 passwordsencrypted.txt`



- Please notice that with these (and with the custom password you ran in part 1) we have added "`--format=raw-md5`" as a parameter. This is to specify how the passwords are hashed.
- ➢ **Why might we need to specify, as John is usually able to detect what format the hash is?**
  Because the type of md5 is ambiguous, it resembles other hash types so closely that it is impossible for John to definitively state one way or the other what type it is
- Now we will test the wordlist. In your console: type `john --wordlist=rockyou.txt --format=raw-md5 passwordsencrypted.txt`
- Type `john --show --format=raw-md5 passwordsencrypted.txt`



- Finally, we will end with an Incremental search
- Type `john --incremental --format=raw-md5 passwordsencrypted.txt`
- As you will see, john starts out getting a good number of decoded passwords, but eventually starts slowing down

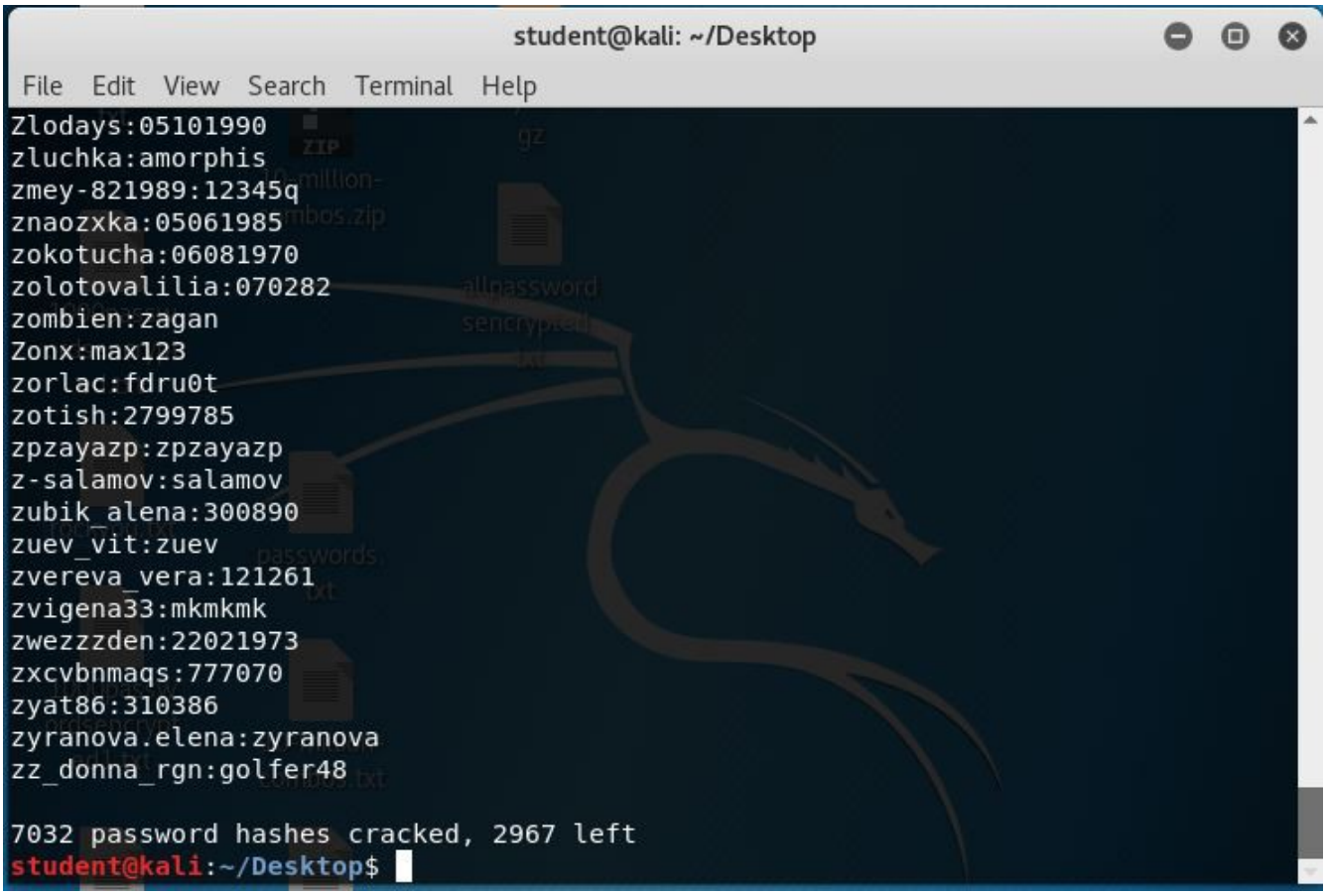
➢ **What is a possible reason for the slowdown?**
John is optimized so that keys that are more likely to be hits are tried first, also, the more passwords there are left the more likely it is to get a hit

➢ **What would be a possible way to increase speed if we know something about the password?**
If the password only used certain characters, we could specify what family of characters so that john didn't use the default "All printable characters" for the incremental search

- As is stated in the documentation, John (specifically when he hits incremental mode) **is not necessarily supposed to complete processing** (unless both the number of passwords is small and the number of characters is low), because if a password is good enough, it may be far too long of a process to brute force it. The designers recommend stopping it when the rate of new passwords is no longer acceptable to the user. Since all passwords cracked are saved, there is no penalty interrupting the test. In fact, John saves its status so it can be resumed if desired.

➢ **Why are the tests run in this order? What would be the result of doing the tests out of order?**
They are run in this order to maximize number of passwords cracked to time spent. The result of running them out of order would be fewer passwords in the same amount of time (not as big of a deal if you flip flop single and wordlist, HUGE difference if you start with incremental)

- Depending on how long you let John run for, you can easily crack 7000+ out of the 10,000 using just the tools built into Kali in less than an hour.
- As a test of the speed/benefit ratio of mutable wordlist type `john --wordlist=rockyou.txt --format=raw-md5 --rules=jumbo passwordencrypted.txt`
- In this case, --rules specifies rules, and "jumbo" means "all of the rules"
- **What is the expected completion time of this set?**
  Depends on the size of the passwordencrypted.txt, but with 10,000 and rockyou.txt as a wordlist, it's about 12-14 hours
- John is also one of the most popular password hacking toolsets around. While it may not be the fastest, it features a fairly comprehensive package with great customization options. I highly recommend looking at the documentation at https://www.openwall.com/john/doc/
- After all your cracking is done, in less than 30 minutes of actual processing time, John has managed to crack >70% of passwords in a given file. If you show John again, your output should look something like this:



Part 4: Customizing John (OPTIONAL)

- I would like to close out this lab by giving you an opportunity to try out the different options for a wordlist
- Since we have all of the previously cracked passwords stored in john.pot, we will have to delete john.pot unless we want to generate a completely new list (by modifying md5-encrypter)

- Type `cd` into the terminal to return to the home directory
- Type `find . -name john*` There should be 3 files found, including `./.john/john.pot`
- Type `rm ./.john/john.pot` Now we have a clean slate again!
- Go to https://www.openwall.com/john/doc/, scroll down and look up the Options, Config, Examples, and Rules pages. Here is all the documentation you need to customize John to your heart's content.
- These rules can be used to expand the range (by mutating the wordlist), while other options (especially the incremental option for different sets of characters) are used to increase speed if you know something about the password.

➢ **Since password cracking is a very strenuous task, John has been designed to use multiple threads to maximize efficiency. What other improvements (on the hardware side) would help increase speed for these tasks?**
Using the computer's GPU in addition to its CPU to run the calculations. HashCat is another Kali tool that leverages the power of the GPU to crack passwords.

➢ **In your own words, please explain what you've learned in this lab. What lessons have you taken away that could be useful in the future?**
John is a very powerful tool, with large customization options.