Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024

Lab 6.6.4 Password Cracking using Rainbow Tables

From TestOut CompTIA Security+ Course

In this lab I will be analyzing a network interface to ascertain if a SYN Flood attack is occuring on “my network” as a hypothetical corporate IT Admin. I will be using Wireshark to listen in on packets and deduce a conclusion.

The scenario for this lab is as follows:

“A recent breach of a popular third-party service has exposed a password database. The security team is evaluating the risk of the exposed passwords for the company. The password hashes are saved in the root user's home directory, /root/captured_hashes.txt. You want to attempt to hack these passwords using a rainbow table. The password requirements for your company are as follows:

- **The password must be 12 or more characters in length.**
- **The password must include at least one uppercase and one lowercase letter.**
- **The password must have at least one of these special characters: !, ", #, $, %, &, _, ', *, or @.**
- **All passwords are encrypted using a hash algorithm of either md5 or sha1.**

In this lab, your task is to:

- **Create md5 and sha1 rainbow tables using rtgen.**
- **Sort the rainbow tables using the rtsortcommand.**
- **Crack the hashes using the rcrackcommand. You can run rcrack on an individual hash or the hash file (/root/captured_hashes.txt).**
- **Answer the questions.**

The type of charset that can be used to create a rainbow table is stored in the /usr/share/rainbowcrack/charset.txtfile. This file can be viewed using the cat command. “

Since we are working with Rainbow Tables we will need to generate some hashes. A Rainbow Table by definition is a table with varying hashes according to certain character sets.

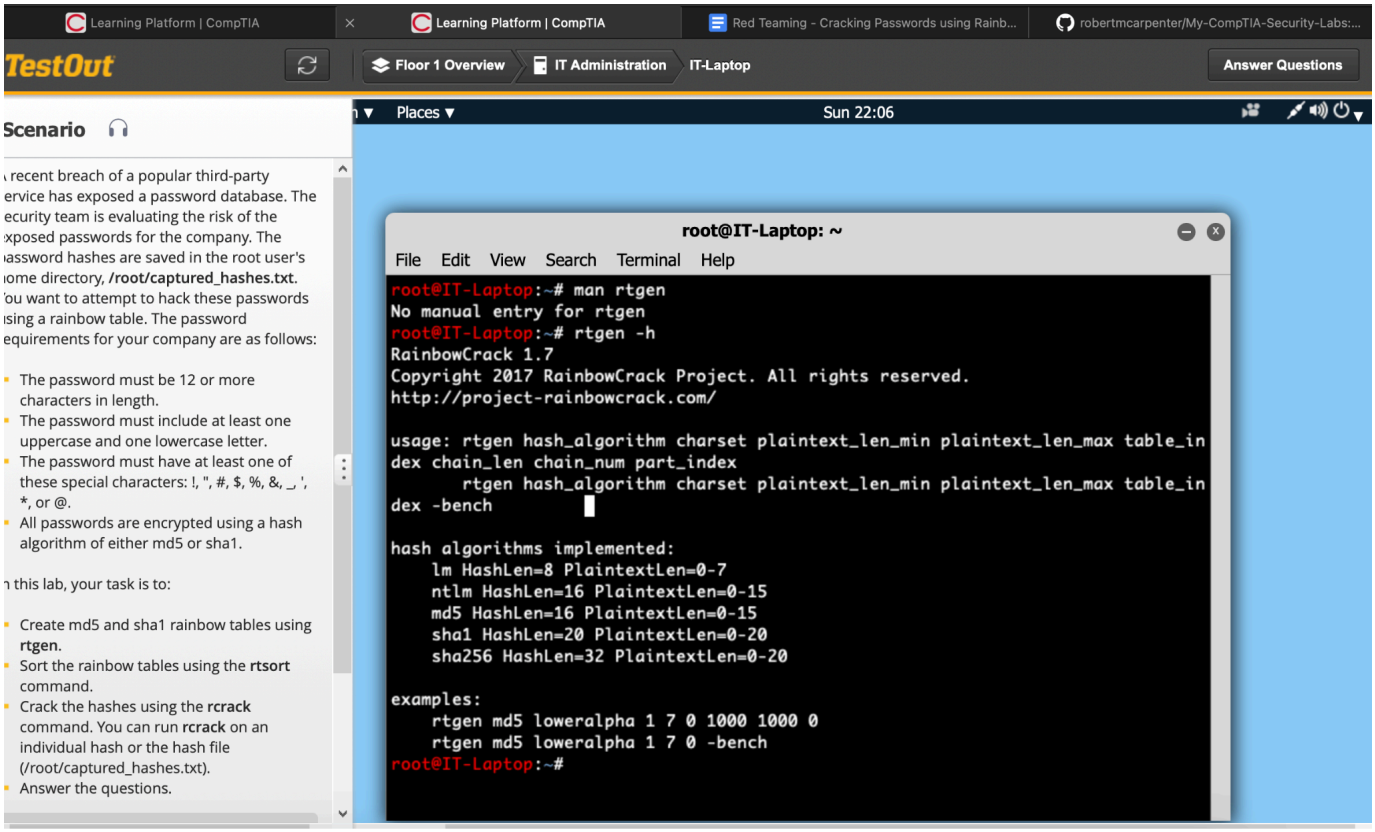
Our constraints in this lab is that the passwords are:

- **The password must be 12 or more characters in length.**
- **The password must include at least one uppercase and one lowercase letter.**
- **The password must have at least one of these special characters: !, ", #, $, %, &, _, ', *, or @.**
- **All passwords are encrypted using a hash algorithm of either md5 or sha1.**

Based off these constraints I will create a Rainbow Table around them.

Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024

The binary to do this on my Kali Linux box is **rtgen.** To see what this command can do I will issue the command: **rtgen -h**



As we can see , the syntax for inputting the **rtgen** command is listed above:

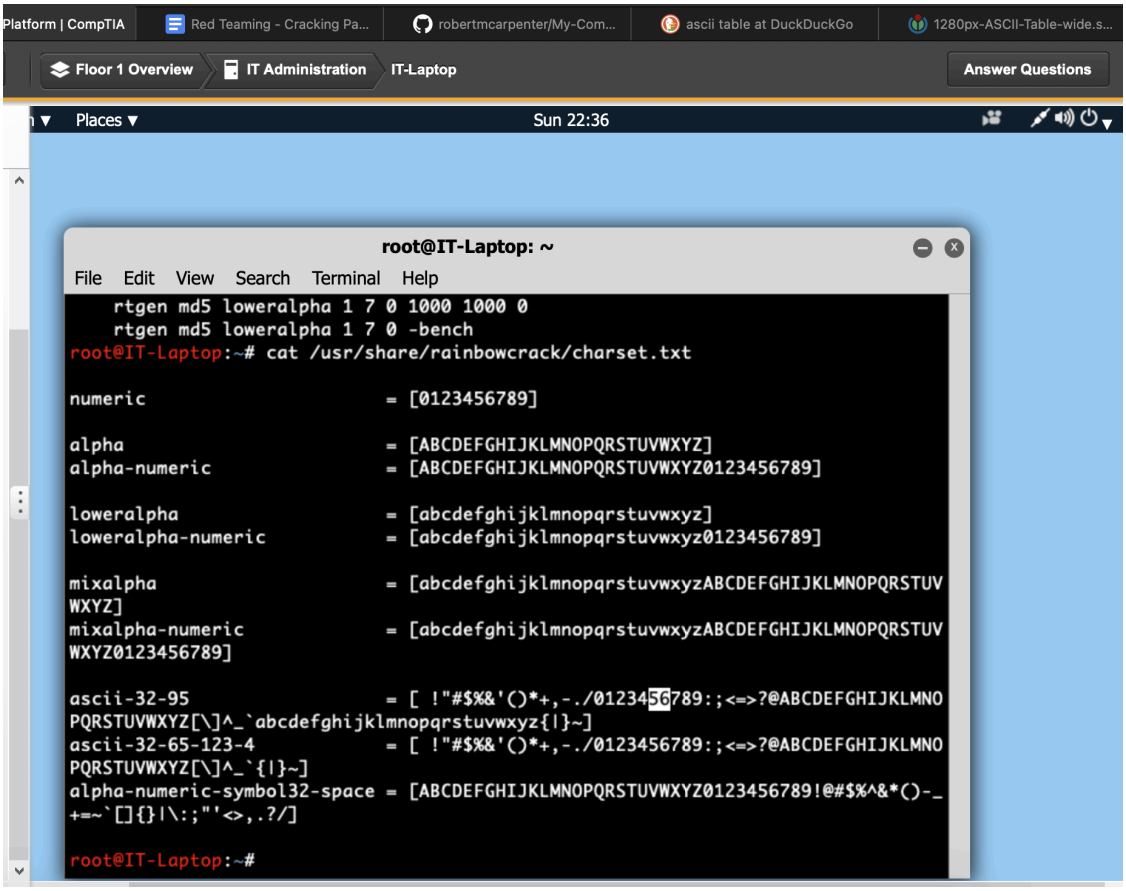
Rtgen [hash_algorithm] [charset] [plaintext_len_min] [plaintext_len_max] [table_index] [chain_len] …

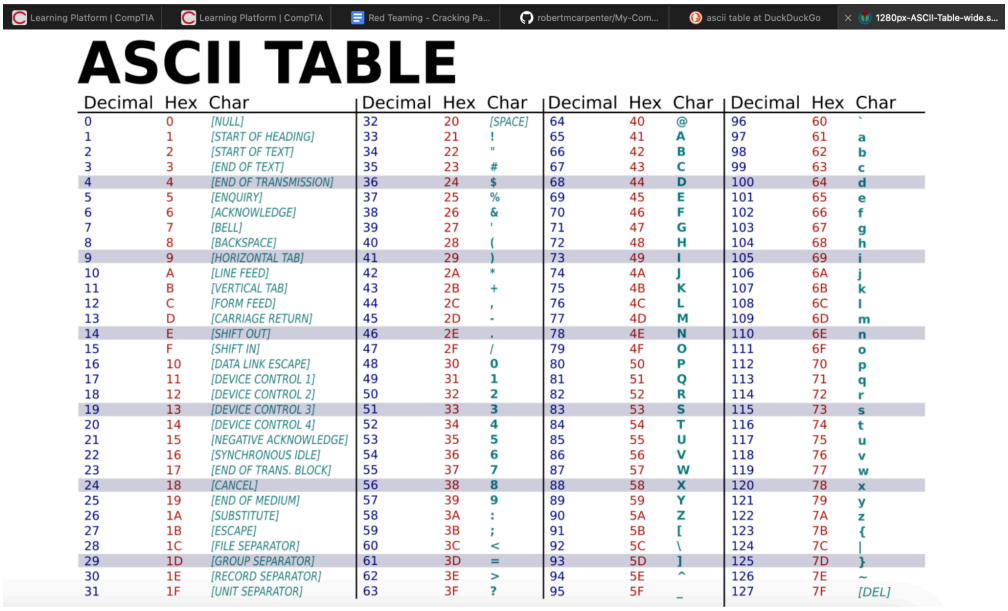
- **The password must be 12 or more characters in length.**
- **The password must include at least one uppercase and one lowercase letter.**
- **The password must have at least one of these special characters: !, ", #, $, %, &, _, ', *, or @.**
- **All passwords are encrypted using a hash algorithm of either md5 or sha1.**

I repasted the constraints again for quick reference. Looking at them in order I can tell that:

- **[plaintext_len_min] is 12:** Since minimum password length is 12
- [charset] is ascii-32-95: Because it's A-z , a-z, and special characters as well. The great thing about ASCII is that we can manually specify these

special characters using this standard. (See Ascii Table below). The Lab also gives us a hint that the available character sets are in **/usr/share/rainbowcrack/charset.txt**





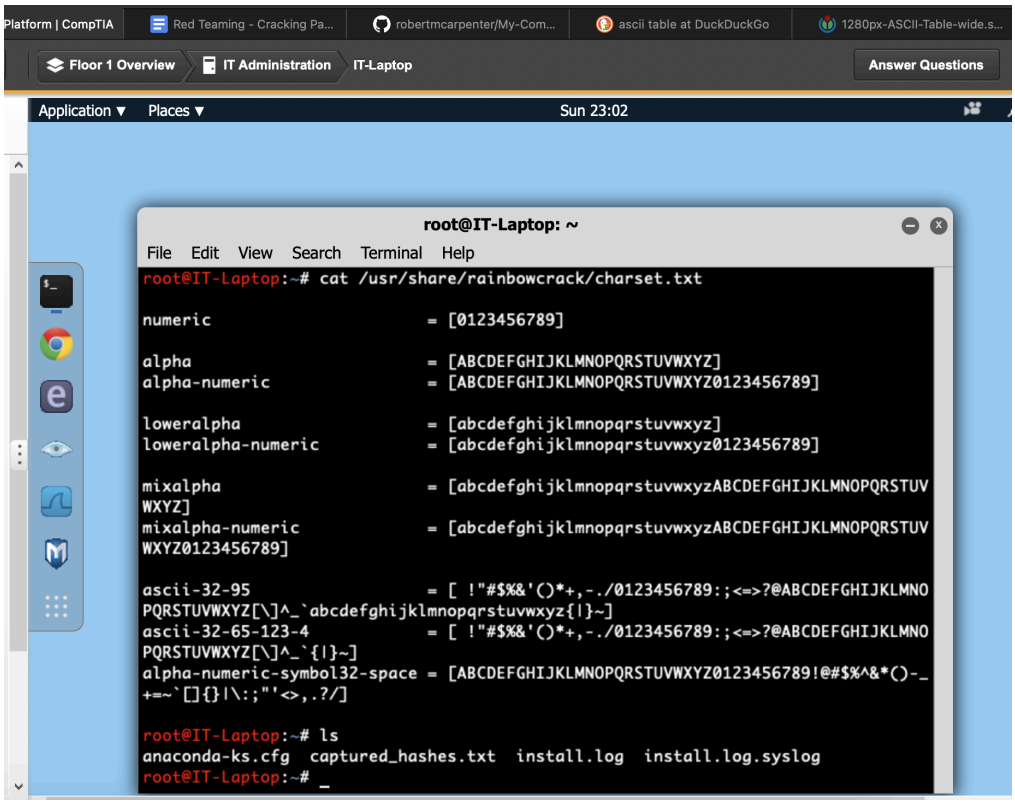
Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024

- [algorthim] is SHA1and md5 : Because we need to generate 2 Rainbow Tables, one sha1 and md5 per the lab's instructions.

To generate a sha1 rainbow table I will use:

rtgen sha1 ascii-32-95 1 20 0 1000 1000 0

Before, I will verify that Im in the correct directory I want to be, which is the same directory as the captured hashes from the data breach.

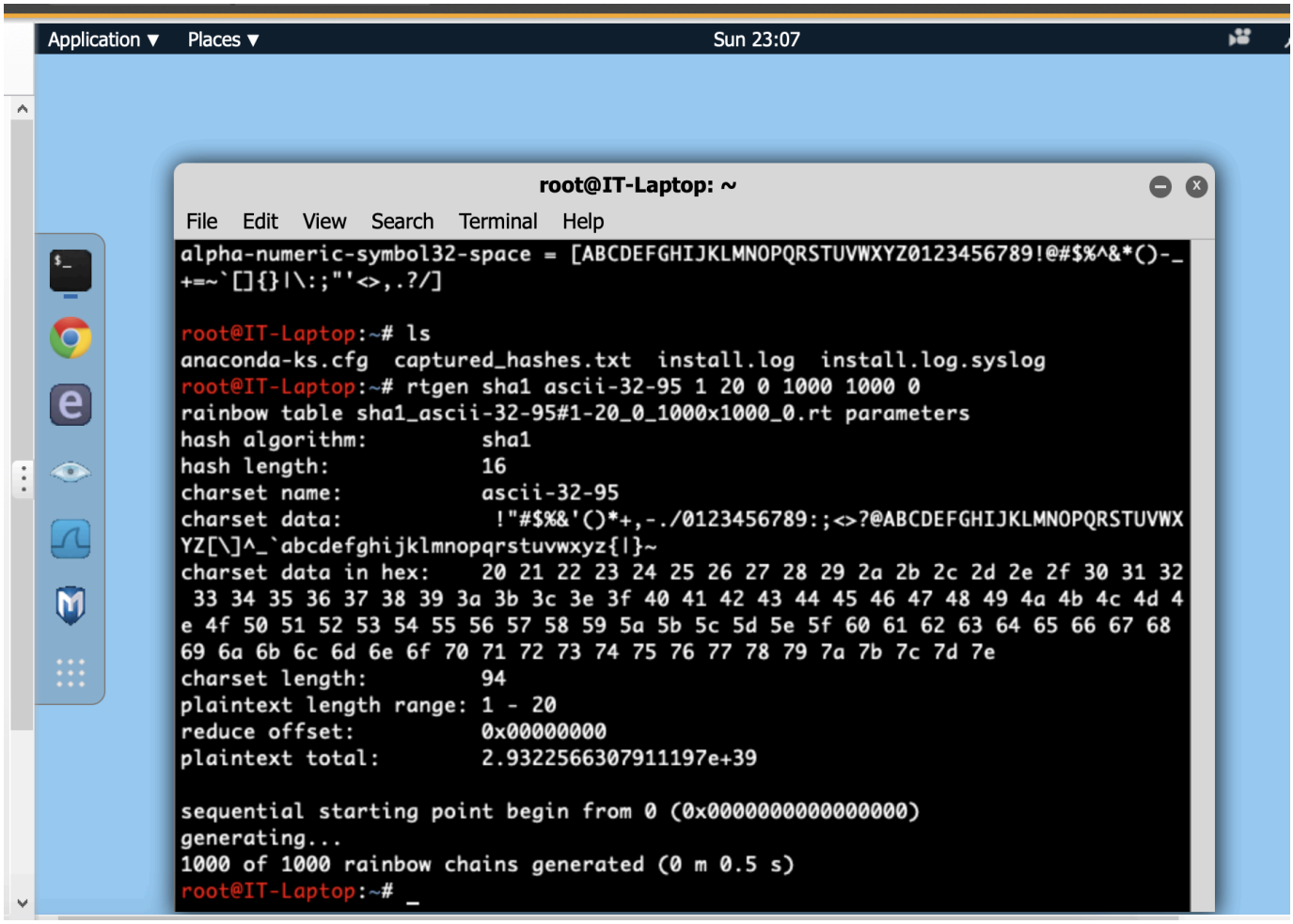
- 

Indeed I am! I will generate the table sha1 and md5 tables with these two commands:

rtgen sha1 ascii-32-95 1 20 0 1000 1000 0

rtgen md5 ascii-32-95 1 20 0 1000 1000 0

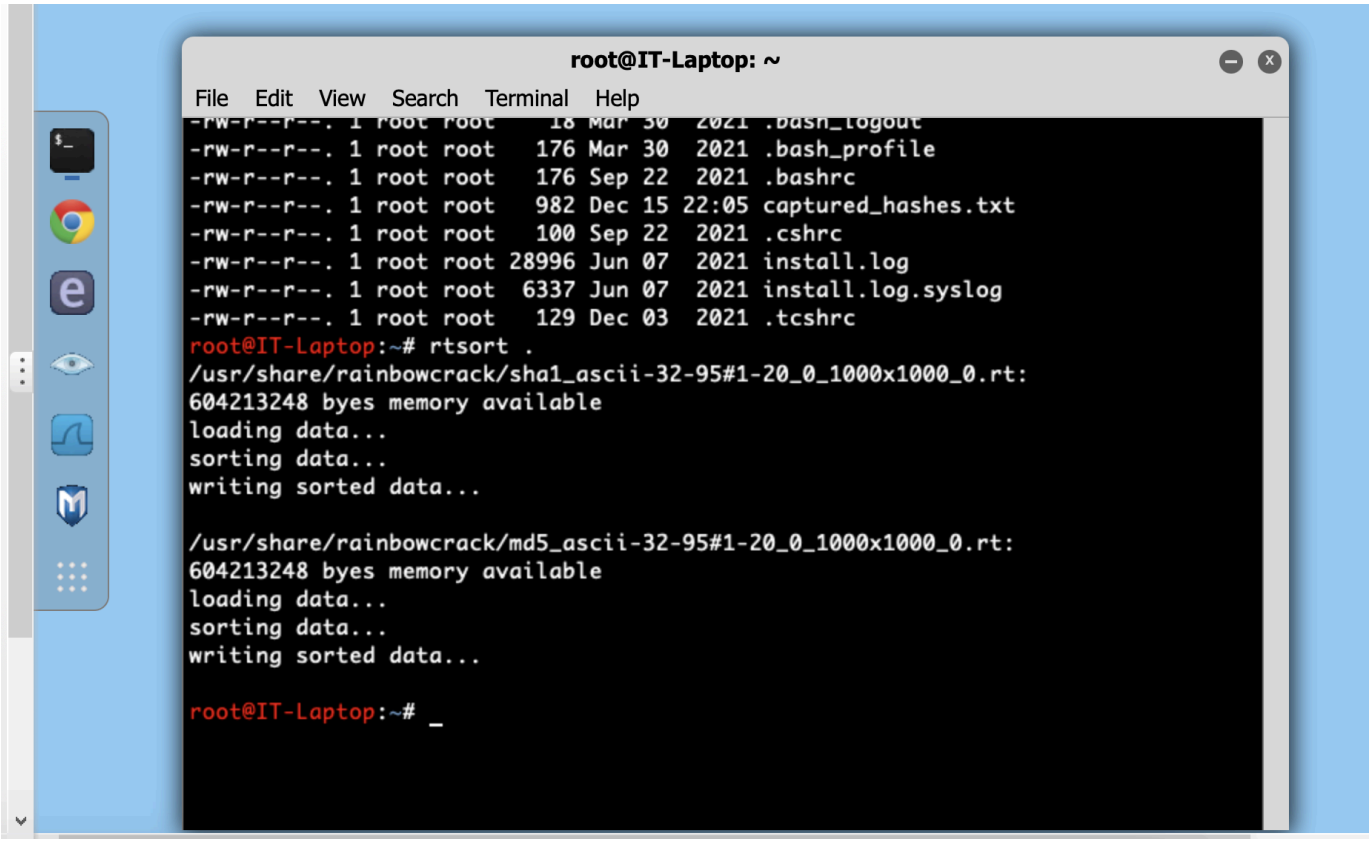
Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024



Success! I've generated the sha1 table. Now let's generate the md5 table.

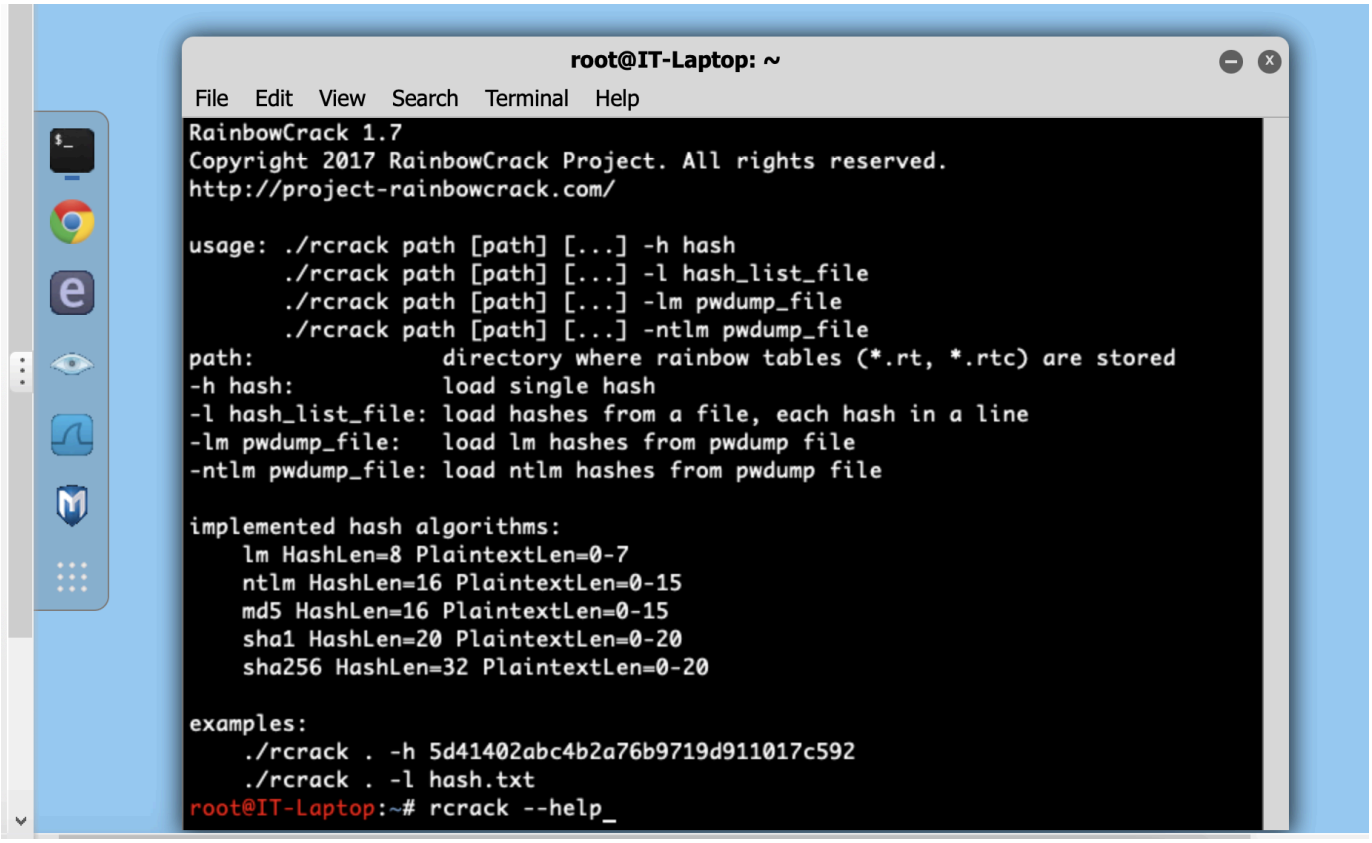
After generating both tables I need sort them. The default directory/path to rainbow tables is **/usr/share/rainbowcrack.** In order to sort the table I need to use the **rtsort** binary as part of the rainbowcrack package.

Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024



Notice how **rtsort** has sorted both the md5 and sha1 tables. To execute the actual crack using the rainbow tables I need to use **the binary "rcrack -l [path/to/hashfile]"** To check the options and flags for the **rcrack** command I can use **rcrack –help.**

Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024

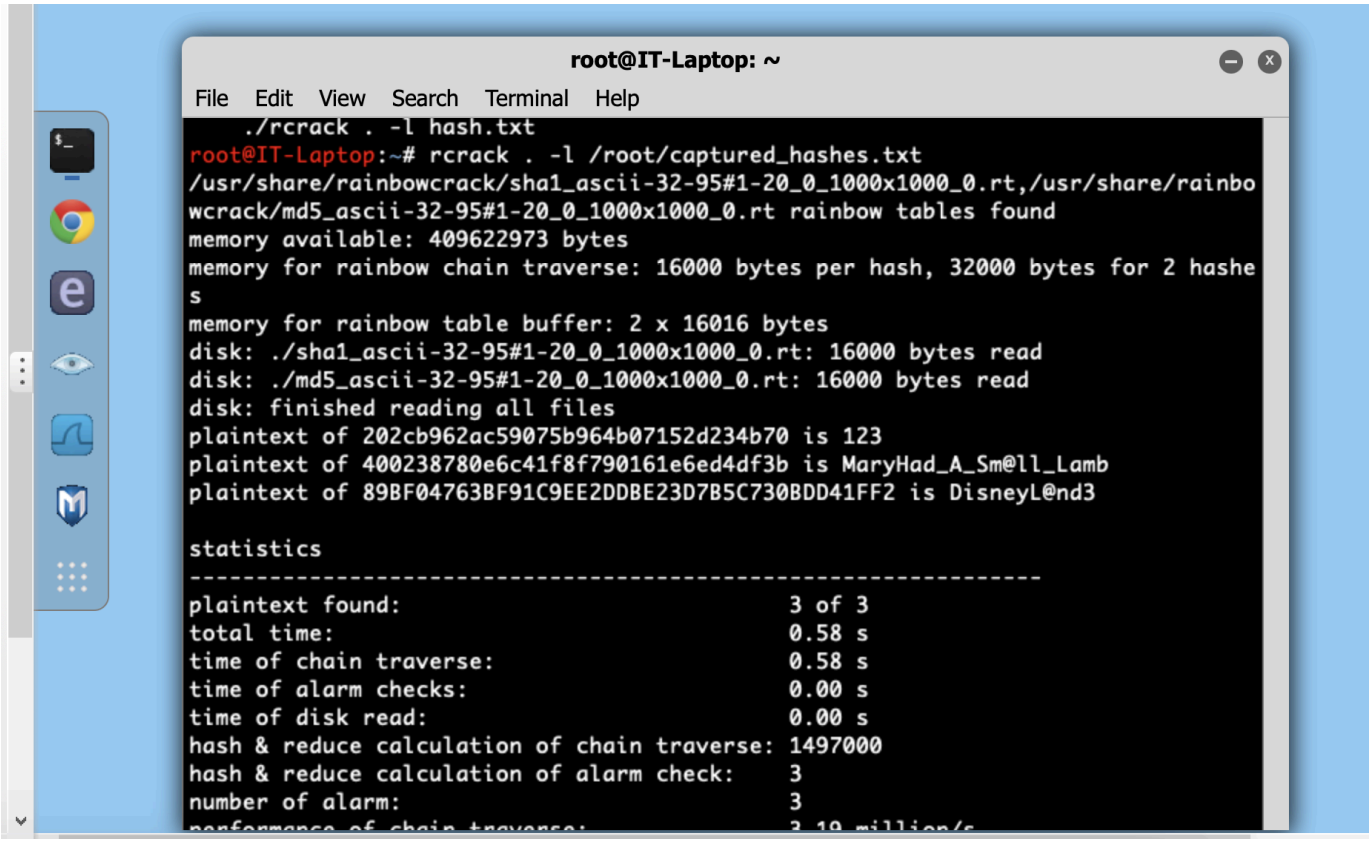


I'll need to use -l which is a hash list file. I can feed in the **captured_hashes.txt** to the command to use the Rainbow tables on them.

To crack the hashes I will issue:

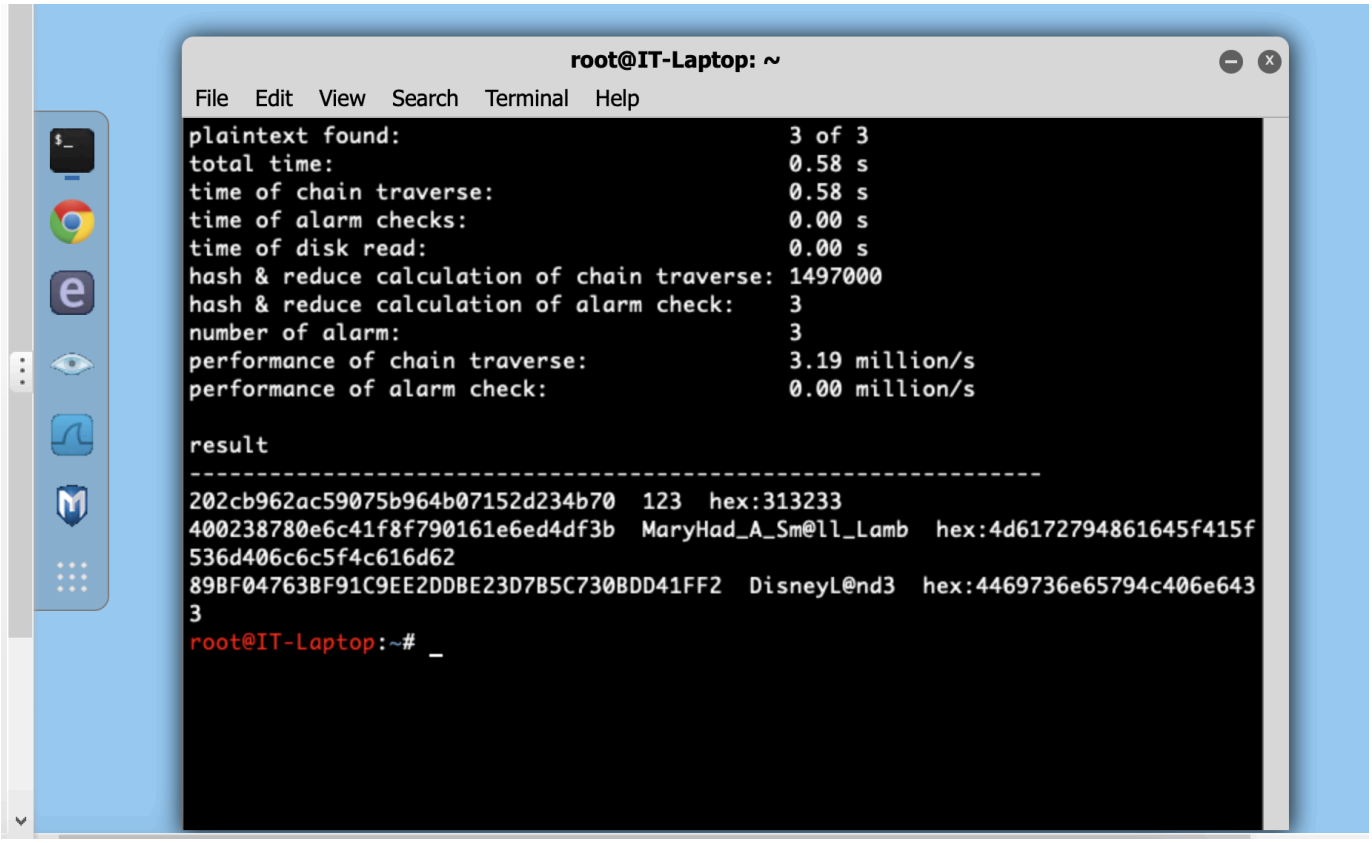
rcrack . -l /root/captured_hashes.txt

Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024



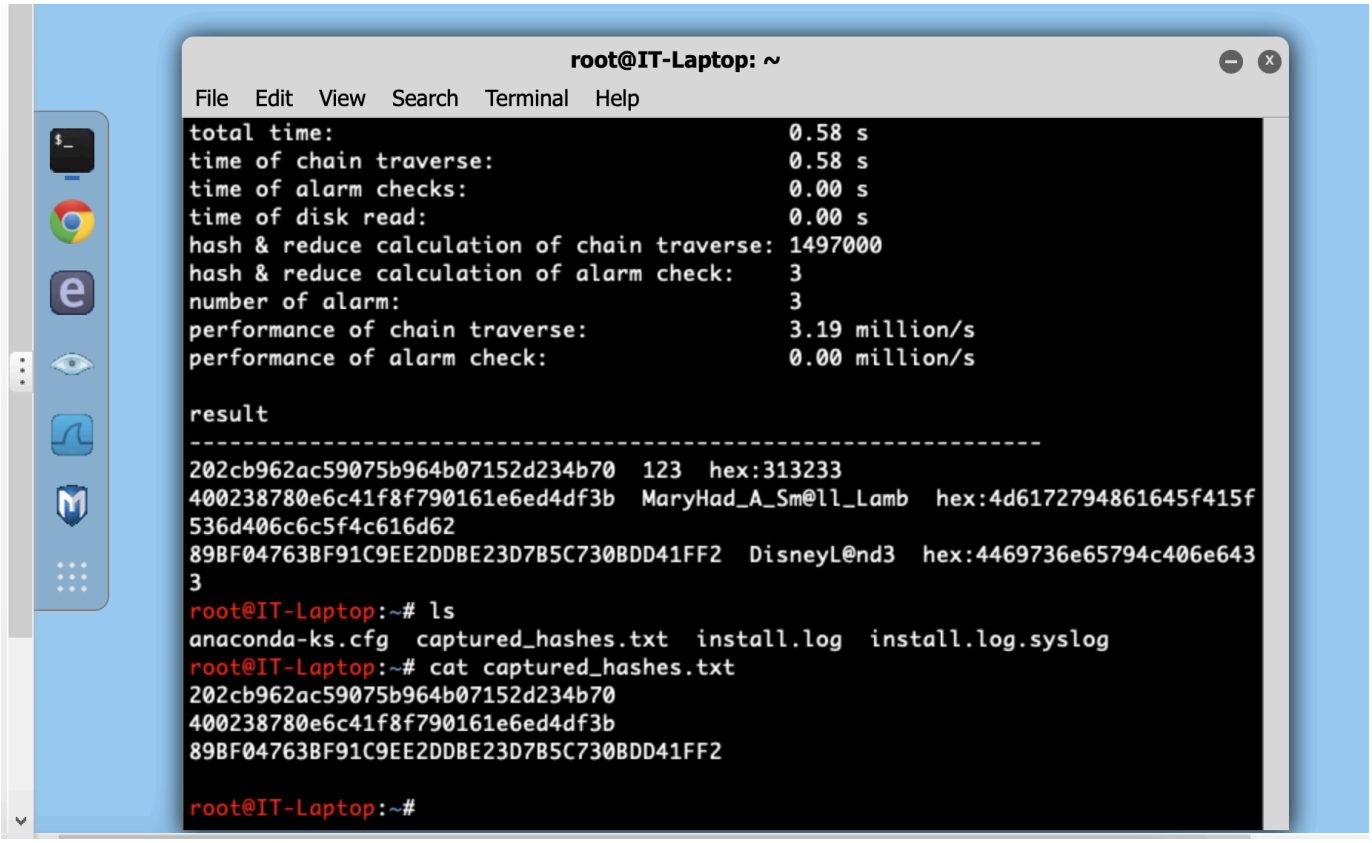
Looks like we've found some matches! I'll screenshot the second half of this output because it's quite a lot.

Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024



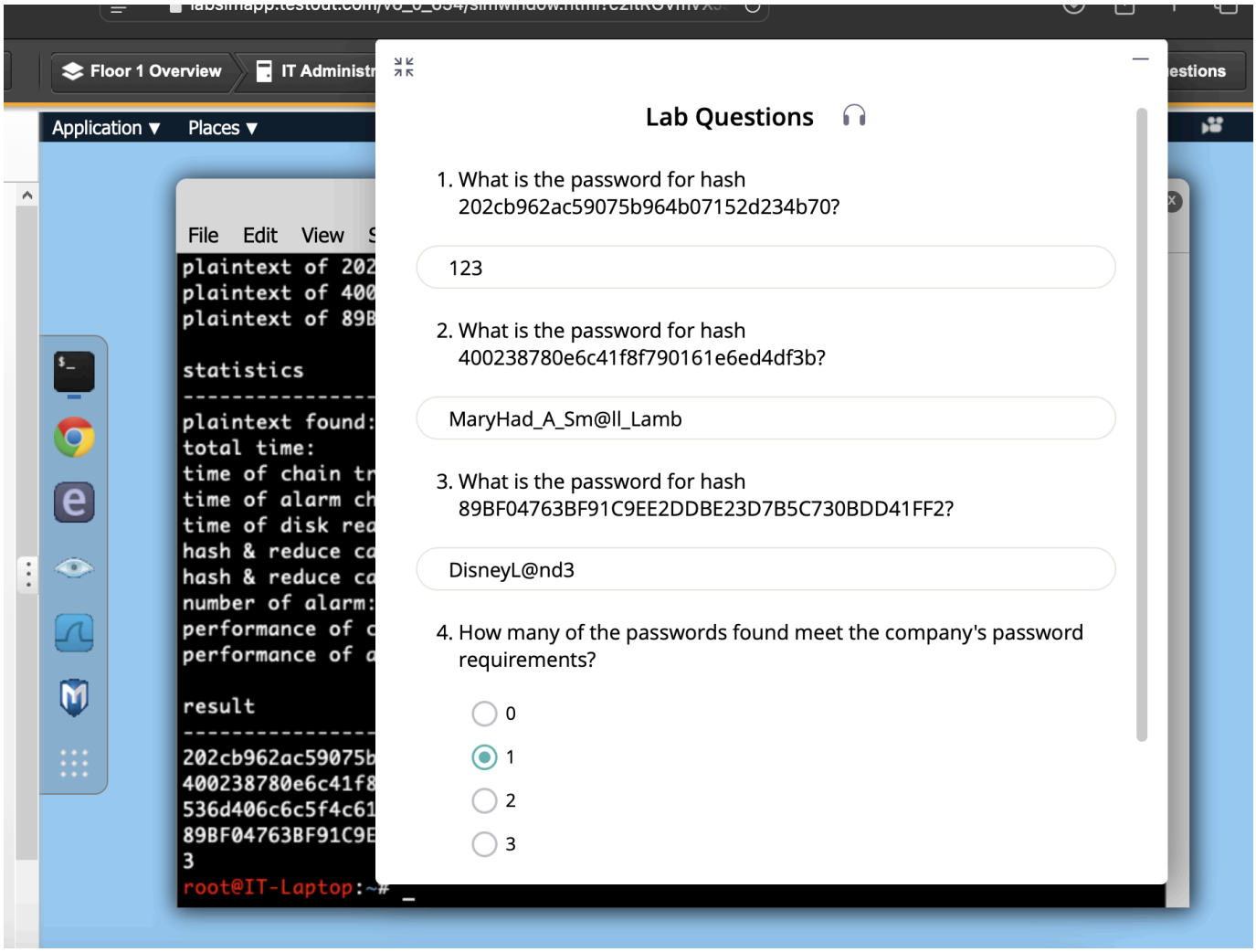
BINGO!!!! We've successfully cracked the hashes. Let's verify that we've cracked all the hashes by printing out the captured_hashes.txt

Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024



We indeed cracked all of them. Let's answer the questions now:

Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024



Only one of the passwords matches the company's password policy. This now concludes this lab.

Robert Carpenter
github.com/robertmcarpenter
Sat December 14th 2024