University of Texas at San Antonio

Mastering Password Cracking: Tools and Techniques

Daniella Carbuccia

IS 3513: Information Assurance and Security

17 Nov. 2024

This lab explored different tools for password-cracking, such as Hashcat, John the Ripper, Ophcrack, and CrackStation, each bringing unique methods for handling password hashes. I worked with Windows and Linux hashes, experimenting with brute-force attacks, dictionary methods, and rainbow tables to understand the tools' strengths and challenges. This hands-on experience showed me how critical it is to correctly identify hash types for effective cracking. My journey began with setting up and installing each tool, leading me to discover the unique power of Hashcat.

To begin, I first set up the different tools we were going to explore in the lab. On all the tools we used, I made sure my package list was up to date by using the sudo apt update command. Then, I used sudo apt install to install the tools. For example, for Hashcat, I used sudo apt install hashcat. Similarly, I used sudo apt install John/Johnny to install John the Ripper command line and GUI. This command also worked for OphCrack as well, however, Ophcrack required an additional step of downloading rainbow tables. It was an easy setup process and there were no issues getting anything downloaded.

The first tool I tried out was Hashcat. Hashcat is a powerful tool known for its speed and versatility and can handle different hash types used on different systems such as Windows and Linux. It can support different attack modes such as dictionary, brute-force, mask and hybrid attacks. What sets it apart is its ability to use both CPUs and GPUs for processing which enhances its performance. According to an article from Splunk,

> The main difference between a CPU and a GPU lies in their approach to processing. CPUs are general-purpose workhorses, capable of handling a wide variety of tasks. On the other hand, GPUs specialize in parallel processing, which is exceptionally effective in handling graphics and tasks that can be parallelized. (Mitton).

In the context of password cracking, Hashcat uses GPUs parallel processing capabilities to accelerate the cracking process. The way it accomplishes this is by distributing tasks among many cores. This allows multiple password combinations to be tested simultaneously which speeds up attacks such as brute-force. This is especially effective for cracking large-scale or difficult password hashes quickly. When Hashcat leverages CPU and GPU power it can adapt to different hash types and attack scenarios which makes it a go-to tool for cracking hashes efficiently. When doing my lab, I attempted to use Hashcat for Windows and Linux password hashes. To do this for Windows, I had to first prepare the NTLM hashes. We downloaded two registry hives SAM, and SYSTEM and I changed my directory over to my downloads folder which contains the files using cd /home/danicarbuccia/Downloads. I was a little confused when using Hashcat because I kept getting errors, but I found out it was because I wasn't in the same directory of my SAM and SYSTEM registry hives. So, after I got into the correct file path, I extracted those NTLM hashes using the command samdump2 SYSTEM SAM. When you combine these two files, the tool extracts and decrypts the password hashes in a format that would allow Hashcrack to crack them. This part is essential because the password data in the SAM file is encrypted and can only be decrypted with SYSTEM file which has the decryption key. From here, I was able to use the Hashcat tool by using the command, hashcat -m 1000 -a 0 -o cracked_passwords.txt ntlm_hashes.txt /usr/share/wordlists/(wordlist txt file). With this result I was able to crack one password, but it was very fast. I also attempted to use Hashcat with Linux password hashes, but the process is a little different. The linux passwords I used were hashed using SHA-512 algorithm and they were stored in the shadow file we downloaded. I was able to tell what algorithm was being used because the hashes started with 6. There were a few algorithms, but I focused on the most common one. I had a few issues getting Hashcat to

understand the hashes I was providing so I had to do a bit of research and what I found was that I had to clean up the hashes in the Shadow file by using this command: awk -F':' '{print $2}' valid_hashes.txt | sed 's/:.*//' > cleaned_hashes.txt. The command awk extracted the hash and removed any colons that were trailing and any unnecessary data. After that, I used the command, hashcat -m 1800 -a 0 -o cracked_passwords.txt cleaned_hashes.txt /usr/share/wordlists/fasttrack.txt. This resulted in one cracked password, and it was fast because there were not many fields in the fasttrack.txt file. When I tried to run Hashcat using the rockyou.txt file, it was going to take an entire day to go through and compare so I had to consider other options.

The next tool I used specifically for Windows password hashes was Ophcrack. Ophcrack is a tool that is specifically designed to crack Windows passwords with the usage of rainbow tables. Rainbow tables have a precomputed set of hash values which makes it easier to find passwords with similar hashes. This is very effective for common passwords, but it isn't as effective for unique ones. According to an article from Achievable, "Ophcrack uses "rainbow tables" to guess and eventually find passwords. These virtual tables can contain billions of possible passwords that the program will check based on hashes to identify the correct password." (Iboshi). When I used this in my lab, we had to download the required rainbow tables from sourceforge and chose the Vista tables. I then tested out the GUI to make sure it words and I added my rainbow tables. From there I loaded the hashes and started the cracking process by clicking on the Crack button. This method yielded me several results and I cracked three passwords with it which really impressed me because I got the most results from this method and in the shortest amount of time. I liked this method the most just because of how fast and efficient

it was, however, if my passwords were less common, I am sure the results would vary significantly.

The next tool I attempted to use was John the Ripper. According to an article from Varonis, “JtR supports several common encryption technologies out-of-the-box for UNIX and Windows-based systems. (ed. Mac is UNIX based). JtR autodetects the encryption on the hashed data and compares it against a large plain-text file that contains popular passwords, hashing each password, and then stopping it when it finds a match. Simple.” (Buckbee). This feature showcases jtR’s ability to quickly match hashes which makes it a versatile option in cracking different types of password hashes. However, combining this with custom wordlists built through OSINT can further improve success rates by creating more targeted guesses. This is particularly useful when standard dictionaries do not yield results, and personalized lists can reduce the time and effort required to find matches. However, the hash type needs to be properly identified before using this tool or even for a tool like Hashcat because it ensures the right cracking algorithm is applied. If the hash type isn’t specified it could lead to inaccurate results, or it won’t be able to crack the passwords at all. To ensure ease of accessibility, jtR offers a GUI that is known as Johnny. I primarily used Johnny as it is easier on the eyes, but when I used it with Linux password hashes, I only got one result. The way I used it was by unshadowing the shadow.txt file using the command sudo unshadow /etc/passwd /etc/shadow > hashes.txt. This combined the passwd and shadow files which allowed root access since the shadow file requires certain privileges. This did leave my file a little strange when I opened it so I had to change the format of the hashes like I did in the Hashcat portion. I then typed Johnny in the terminal and once the GUI popped up, I opened the cracked_passwords.txt file in it and chose my attack as wordlist. I searched for the rockyou.txt word list and ran it against that file, but I only got one

password from it that I already retrieved from another method, so this tool was not my favorite to use. I think if the text file were short, it would be better but when I tried the command-line version, it planned to take a whole day to complete, and I didn't have time for it. So, I had to attempt other options to get the results I wanted.

Finally, I landed on the CrackStation tool. Unlike the other tools we used, CrackStation is known for its convenience and how simple it is. It can crack simpler hashes due to an extensive database of precomputed hashes. This is especially helpful when identifying weak passwords or identifying known vulnerabilities in password security, however, there are some privacy concerns because password-data is being submitted to a third party. Overall, it is best used for simpler cases and should be combined with other tools in cases of more complex passwords. When I used this tool, I had a few issues with formatting and it couldn't read my hashes, but after some research, I realized I had to remove the colons and information at the end, and I was able to find one password using the tool. It was the easiest method I found but it couldn't crack every password I had so it wasn't the best. Despite the limitations, it was a quick solution for the simpler passwords I had.

In conclusion, this lab gave me hands-on experience with different password-cracking tools like Hashcat, John the Ripper, Ophcrack, and CrackStation. Each tool had its own strengths and ways of tackling password hashes, showing how important it is to pick the right tool and method for each job. I saw how identifying hash types correctly and using custom wordlists could make a big difference in cracking success. But it's also important to think about the ethical and legal side of things. Using these tools should always be done with proper permission and for the right reasons to stay ethical and avoid legal trouble. This lab really highlighted how important it is to use these tools responsibly in cybersecurity.

Works Cited

Buckbee, Michael. “How to Use John the Ripper: Tips and Tutorials.” *Varonis*, Varonis, 21 Dec. 2022, www.varonis.com/blog/john-the-ripper.

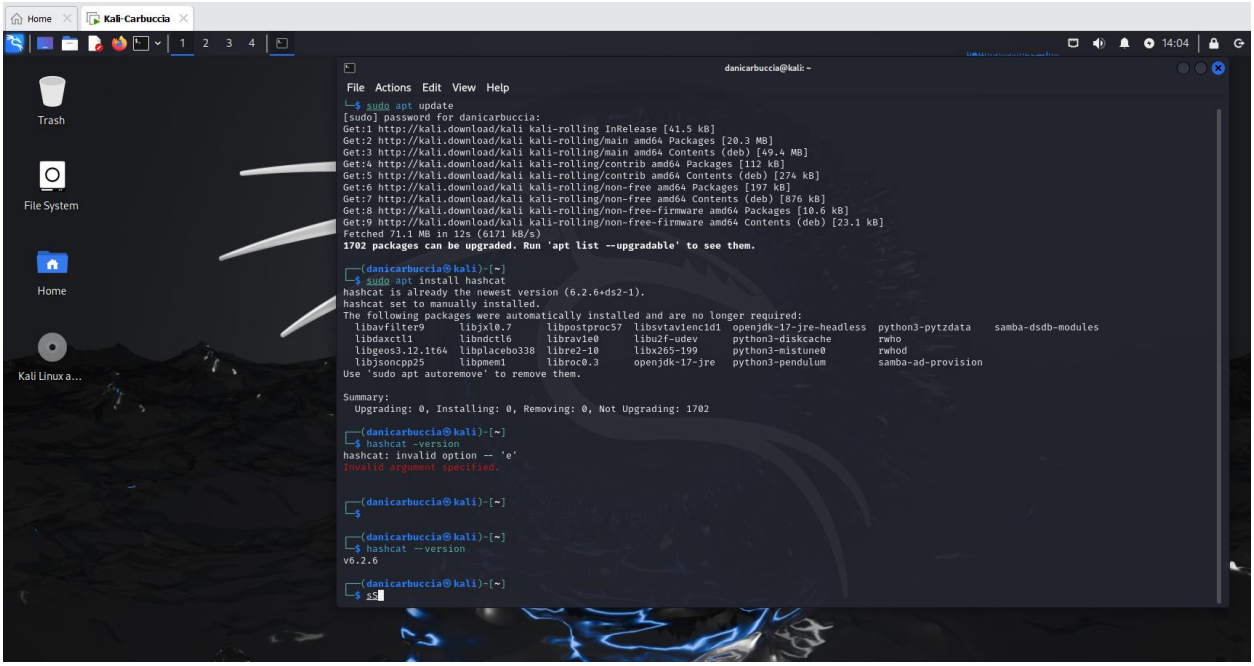
Iboshi, Ben. “Forgot Your Windows Password? Try Ophcrack.” *Achievable Test Prep*, 15 Aug. 2024, blog.achievable.me/tech/forgot-your-windows-password-try-ophcrack/.

Mitton, Leanne. “CPUs vs Gpus: Comparing Compute Power.” *Splunk*, 26 Mar. 2024, www.splunk.com/en_us/blog/learn/cpu-vs-gpu.html.

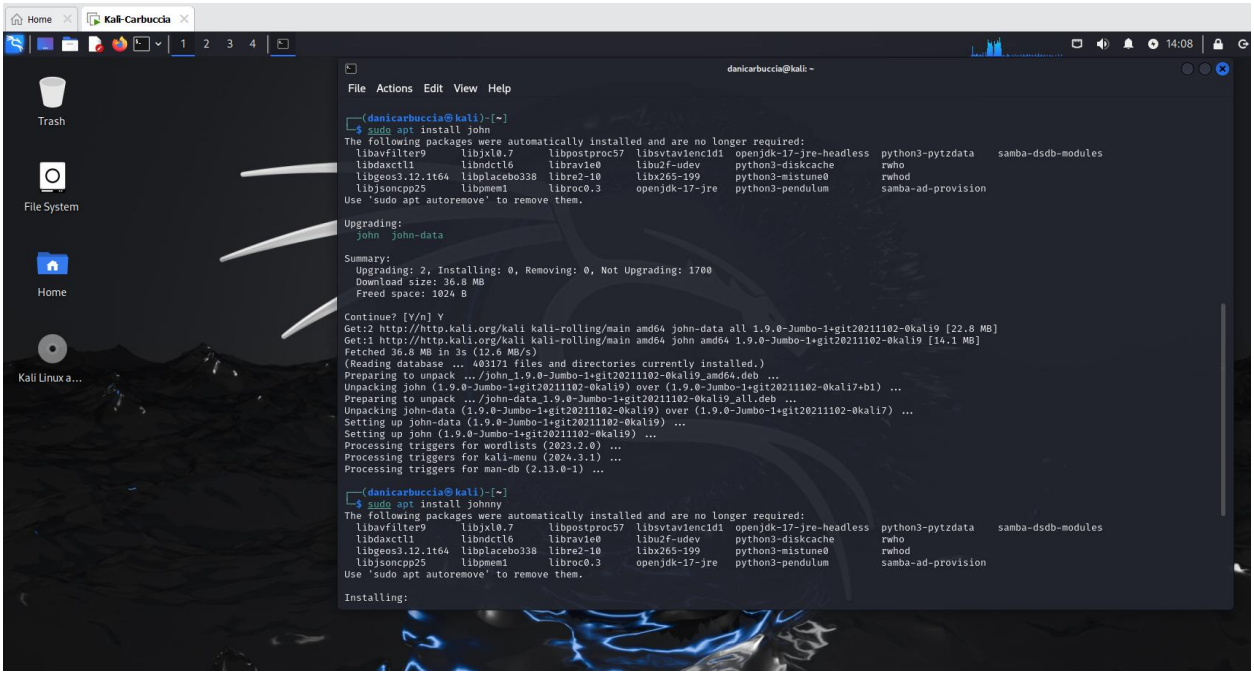
Oechslin, and Tissieres. “Ophcrack Files.” Sourceforge.

Sharma, Himanshu. “Kali Linux - An Ethical Hacker’s Cookbook.” *O’Reilly Online Learning*, Packt Publishing, www.oreilly.com/library/view/kali-linux/9781787121829/69d66516-5f6b-4a8a-85b7-cd56857cfc69.xhtml. Accessed 18 Nov. 2024.

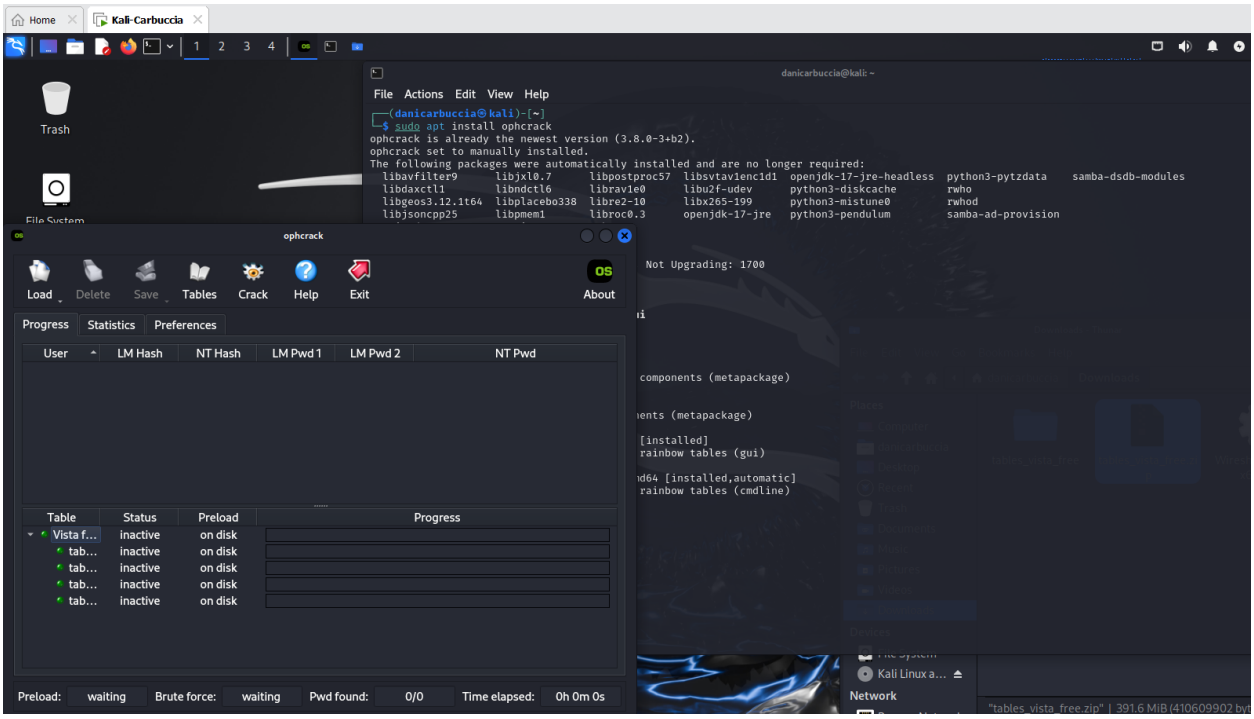
Installation of Hashcat on Kali Linux:



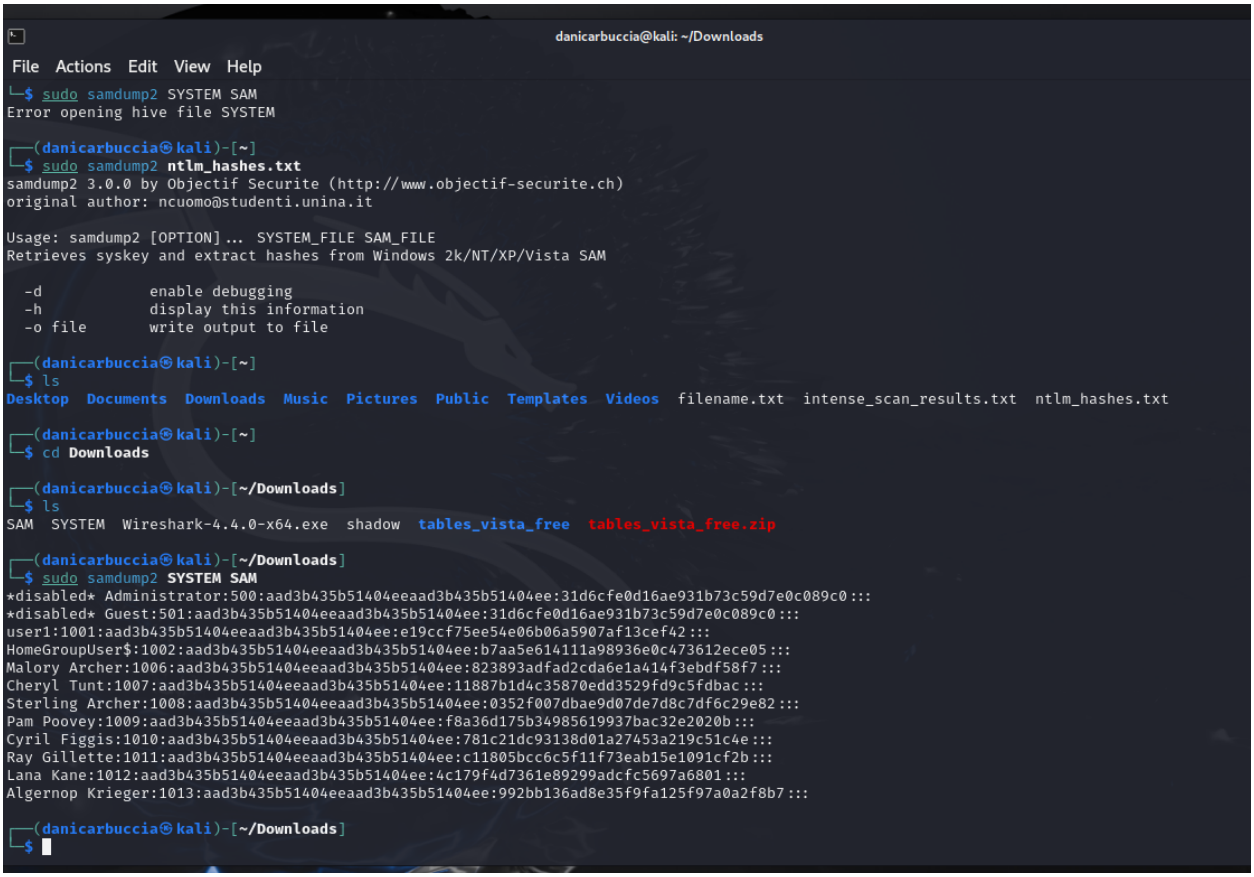
Installation of John and Johnny:



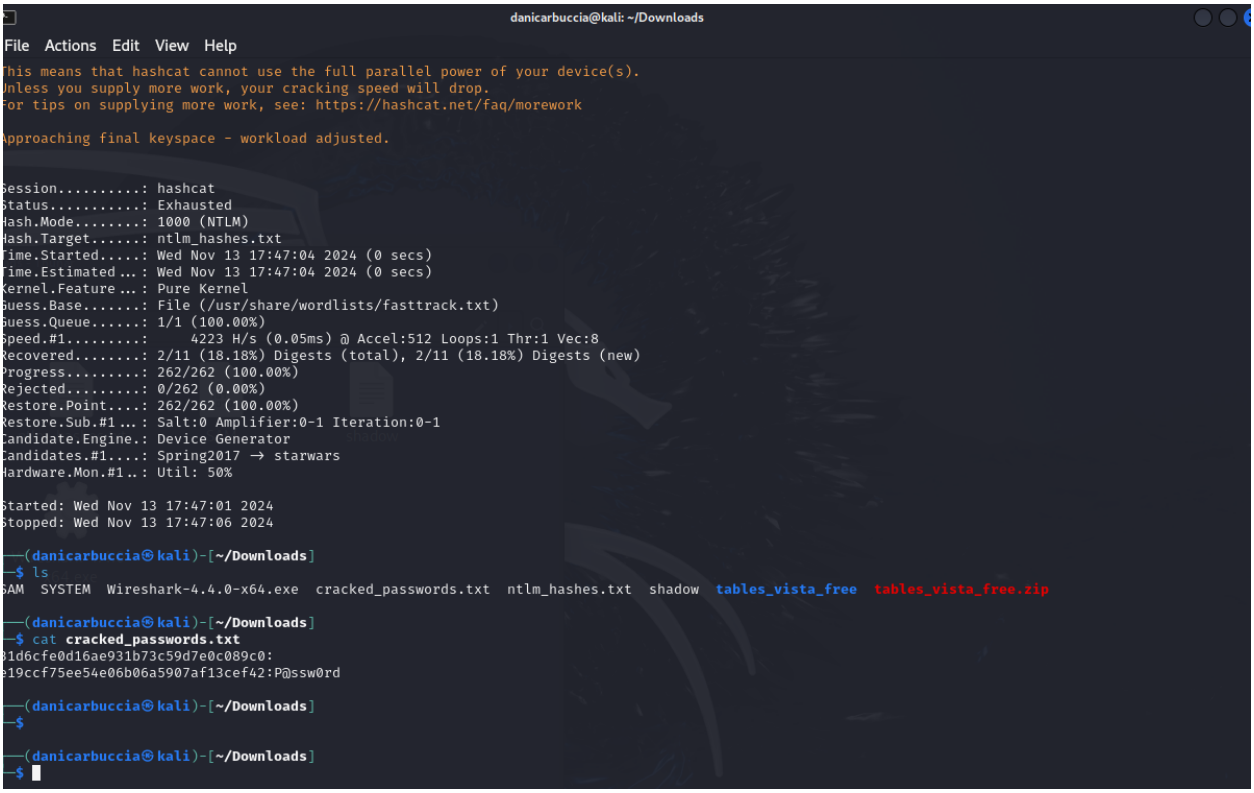
Rainbow tables in ophcrack GUI:



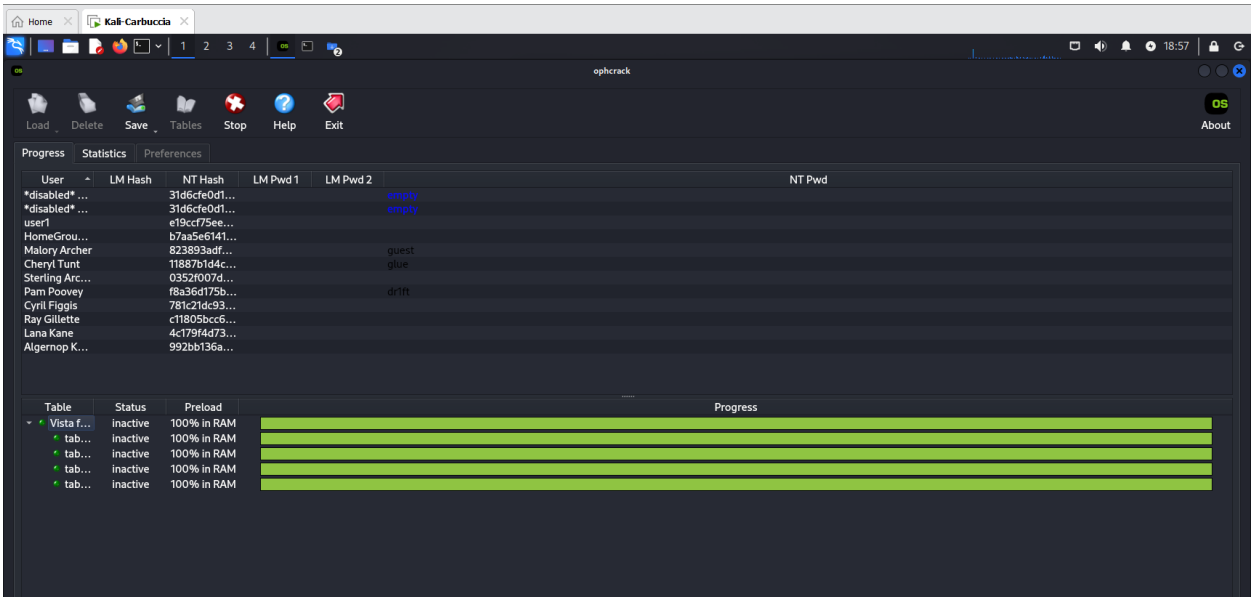
Extracted Sam and System registry hives:



Results from cracking Windows hashes with hashcat and opened cracked_passwords.txt:

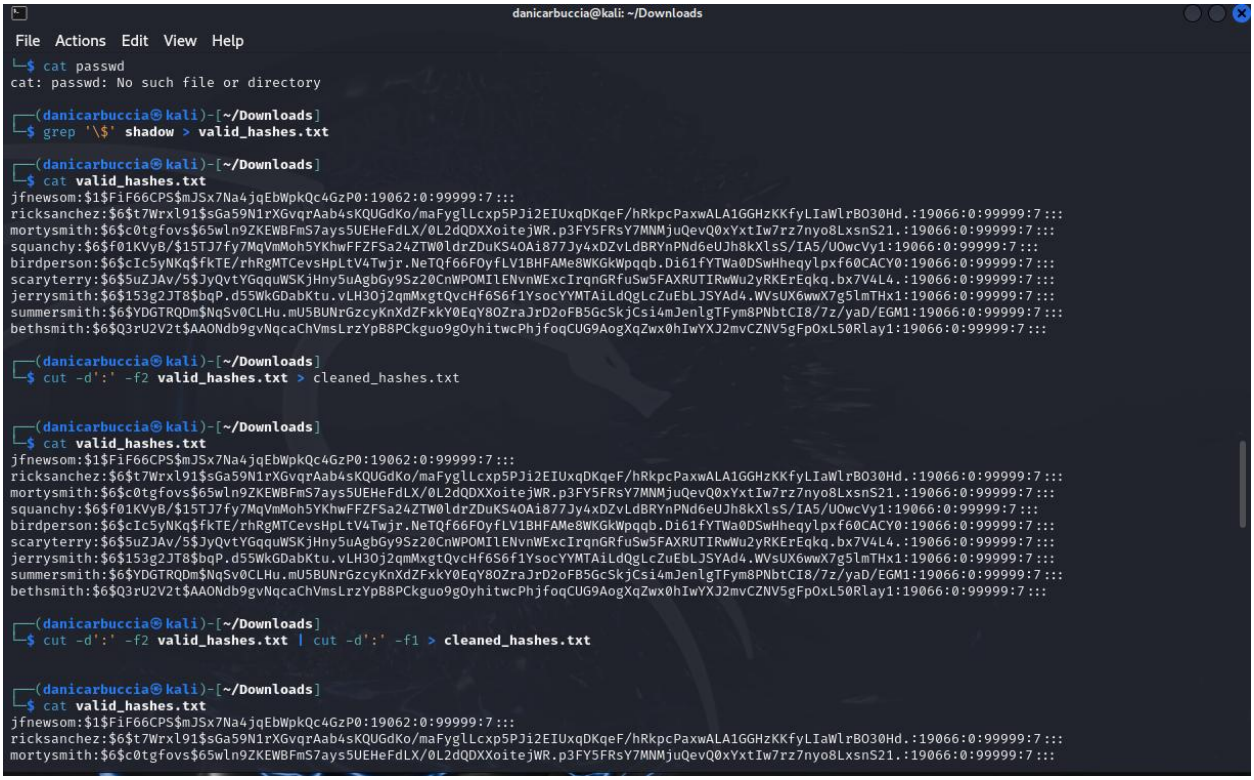


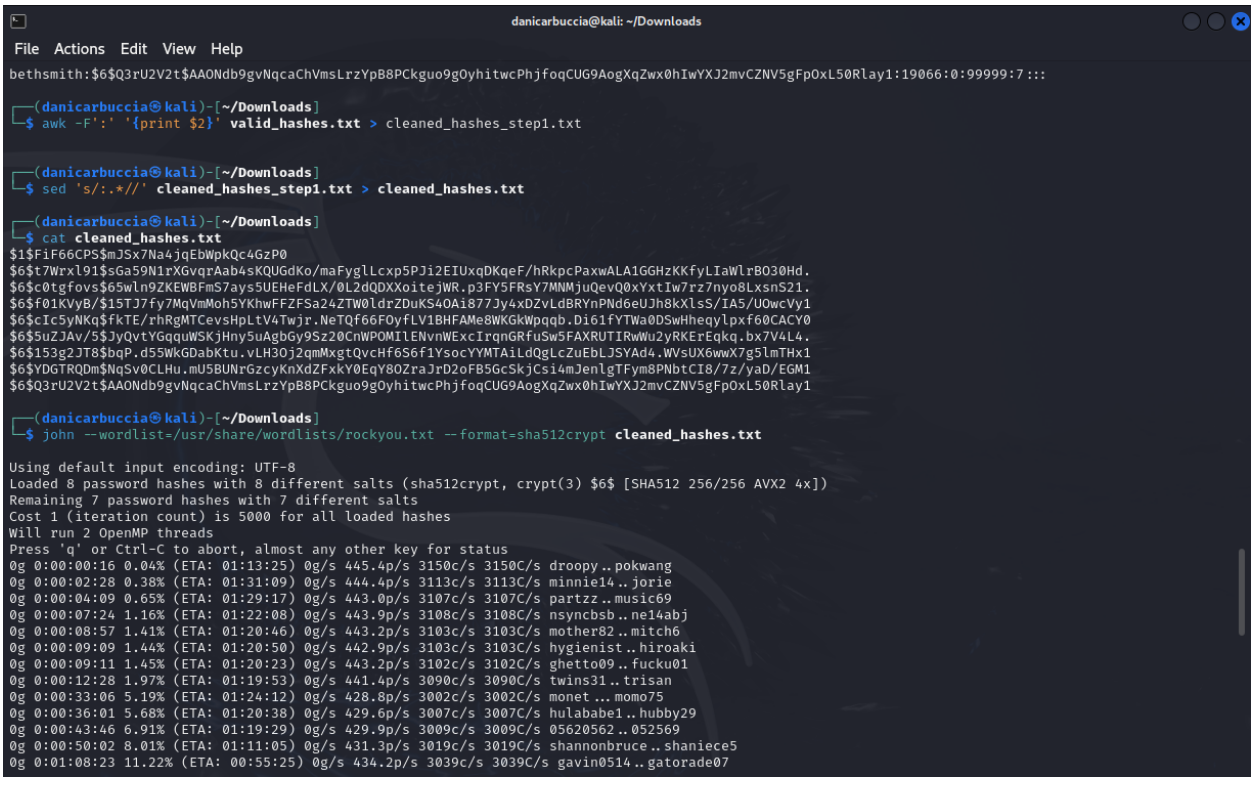
Windows password hashes cracked with ophcrack:



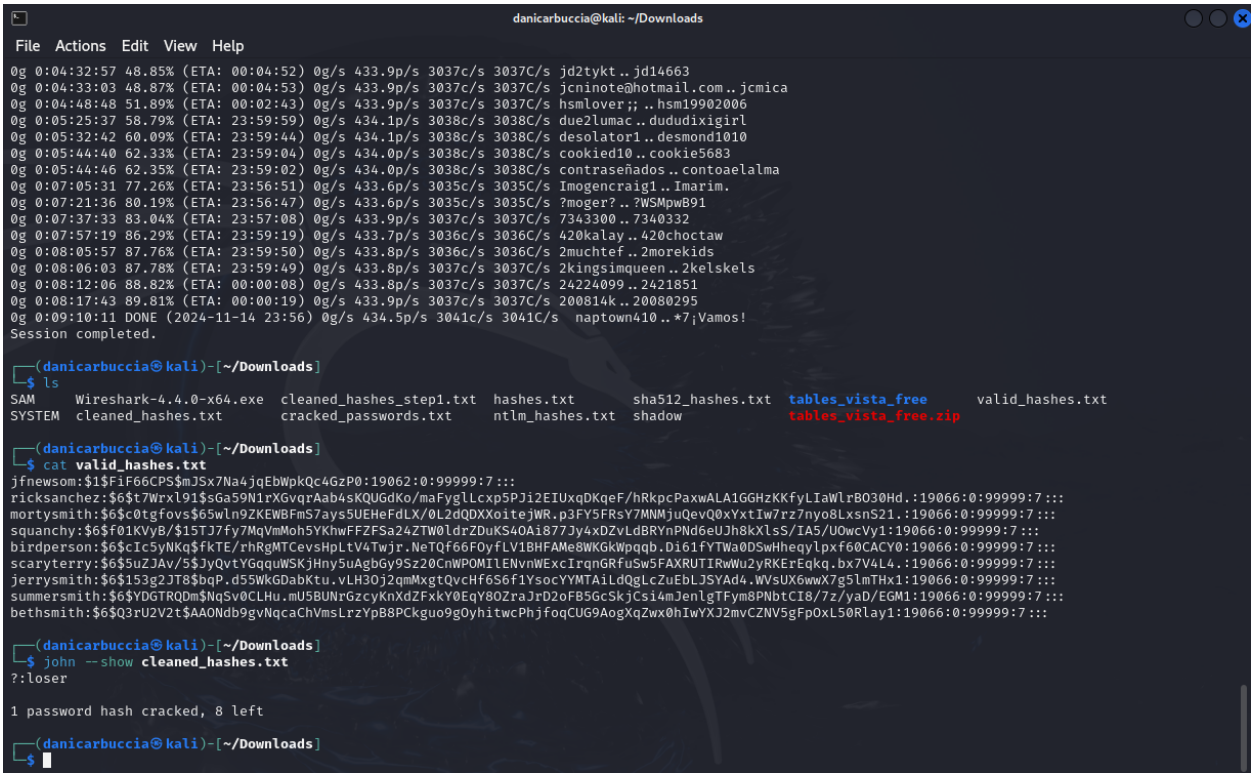
Linux password cracked using hashcrack and with the fasttrack.txt wordlist:

Had to make the hashes more readable for John the Ripper because it has issues reading the provided format:


danicarbuccia@kali: ~/Downloads
File Actions Edit View Help
└─$ cat passwd
cat: passwd: No such file or directory
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ grep '\$' shadow > valid_hashes.txt
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ cat valid_hashes.txt
jfnewsom:1FiF66CPS$mJSx7Na4jqEbWpkQc4GzP0:19062:0:99999:7:::
ricksanchez:6t7Wrxl91$sGa59N1rXGvqrAab4sKQUGdKo/maFyglLcxp5PJi2EIUxqDKqeF/hRkpcPaxwALA1GGHzKKfyLIaWlrBO30Hd.:19066:0:99999:7:::
mortysmith:6c0tgfovs$65wln9ZKEWBFmS7ays5UEHeFdLX/0L2dQDXXoitejWR.p3FY5FRsY7MNMjuQevQ0xYxtIw7rz7nyo8LxsnS21.:19066:0:99999:7:::
squanchy:6f01KVyB/$15TJ7fy7MqVmMoh5YKhwFFZFSa24ZTW0ldrZDuKS4OAi877Jy4xDZvLdBRYnPNd6eUJh8kXlsS/IA5/UOwcVy1:19066:0:99999:7:::
birdperson:6cIc5yNKq$fkTE/rhRgMTCevsHpLtV4Twjr.NeTQf66FOyfLV1BHFAMe8WKGkWpqqb.Di61fYTWa0DSwHheqylpxf60CACY0:19066:0:99999:7:::
scaryterry:$6$5uZJAv/5$JyQvtYGqquWSKjHny5uAgbGy9Sz20CnWPOMIlENvnWExcIrqnGRfuSw5FAXRUTIRwWu2yRKErEqkq.bx7V4L4.:19066:0:99999:7:::
jerrysmith:$6$153g2JT8$bqP.d55WkGDabKtu.vLH3Oj2qmMxgtQvcHf6S6f1YsocYYMTAiLdQgLcZuEbLJSYAd4.WVsUX6wwX7g5lmTHx1:19066:0:99999:7:::
summersmith:6YDGTRQDm$NqSv0CLHu.mU5BUNrGzcyKnXdZFxkY0EqY8OZraJrD2oFB5GcSkjCsi4mJenlgTFym8PNbtCI8/7z/yaD/EGM1:19066:0:99999:7:::
bethsmith:6Q3rU2V2t$AAONdb9gvNqcaChVmsLrzYpB8PCkguo9gOyhitwcPhjfoqCUG9AogXqZwx0hIwYXJ2mvCZNV5gFpOxL50Rlay1:19066:0:99999:7:::
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ cut -d':' -f2 valid_hashes.txt > cleaned_hashes.txt
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ cat valid_hashes.txt
jfnewsom:1FiF66CPS$mJSx7Na4jqEbWpkQc4GzP0:19062:0:99999:7:::
ricksanchez:6t7Wrxl91$sGa59N1rXGvqrAab4sKQUGdKo/maFyglLcxp5PJi2EIUxqDKqeF/hRkpcPaxwALA1GGHzKKfyLIaWlrBO30Hd.:19066:0:99999:7:::
mortysmith:6c0tgfovs$65wln9ZKEWBFmS7ays5UEHeFdLX/0L2dQDXXoitejWR.p3FY5FRsY7MNMjuQevQ0xYxtIw7rz7nyo8LxsnS21.:19066:0:99999:7:::
squanchy:6f01KVyB/$15TJ7fy7MqVmMoh5YKhwFFZFSa24ZTW0ldrZDuKS4OAi877Jy4xDZvLdBRYnPNd6eUJh8kXlsS/IA5/UOwcVy1:19066:0:99999:7:::
birdperson:6cIc5yNKq$fkTE/rhRgMTCevsHpLtV4Twjr.NeTQf66FOyfLV1BHFAMe8WKGkWpqqb.Di61fYTWa0DSwHheqylpxf60CACY0:19066:0:99999:7:::
scaryterry:$6$5uZJAv/5$JyQvtYGqquWSKjHny5uAgbGy9Sz20CnWPOMIlENvnWExcIrqnGRfuSw5FAXRUTIRwWu2yRKErEqkq.bx7V4L4.:19066:0:99999:7:::
jerrysmith:$6$153g2JT8$bqP.d55WkGDabKtu.vLH3Oj2qmMxgtQvcHf6S6f1YsocYYMTAiLdQgLcZuEbLJSYAd4.WVsUX6wwX7g5lmTHx1:19066:0:99999:7:::
summersmith:6YDGTRQDm$NqSv0CLHu.mU5BUNrGzcyKnXdZFxkY0EqY8OZraJrD2oFB5GcSkjCsi4mJenlgTFym8PNbtCI8/7z/yaD/EGM1:19066:0:99999:7:::
bethsmith:6Q3rU2V2t$AAONdb9gvNqcaChVmsLrzYpB8PCkguo9gOyhitwcPhjfoqCUG9AogXqZwx0hIwYXJ2mvCZNV5gFpOxL50Rlay1:19066:0:99999:7:::
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ cut -d':' -f2 valid_hashes.txt | cut -d':' -f1 > cleaned_hashes.txt
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ cat valid_hashes.txt
jfnewsom:1FiF66CPS$mJSx7Na4jqEbWpkQc4GzP0:19062:0:99999:7:::
ricksanchez:6t7Wrxl91$sGa59N1rXGvqrAab4sKQUGdKo/maFyglLcxp5PJi2EIUxqDKqeF/hRkpcPaxwALA1GGHzKKfyLIaWlrBO30Hd.:19066:0:99999:7:::
mortysmith:6c0tgfovs$65wln9ZKEWBFmS7ays5UEHeFdLX/0L2dQDXXoitejWR.p3FY5FRsY7MNMjuQevQ0xYxtIw7rz7nyo8LxsnS21.:19066:0:99999:7:::


danicarbuccia@kali: ~/Downloads
File Actions Edit View Help
bethsmith:6Q3rU2V2t$AAONdb9gvNqcaChVmsLrzYpB8PCkguo9gOyhitwcPhjfoqCUG9AogXqZwx0hIwYXJ2mvCZNV5gFpOxL50Rlay1:19066:0:99999:7:::
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ awk -F':' '{print $2}' valid_hashes.txt > cleaned_hashes_step1.txt
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ sed 's/:.*//' cleaned_hashes_step1.txt > cleaned_hashes.txt
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ cat cleaned_hashes.txt
1FiF66CPS$mJSx7Na4jqEbWpkQc4GzP0
6t7Wrxl91$sGa59N1rXGvqrAab4sKQUGdKo/maFyglLcxp5PJi2EIUxqDKqeF/hRkpcPaxwALA1GGHzKKfyLIaWlrBO30Hd.
6c0tgfovs$65wln9ZKEWBFmS7ays5UEHeFdLX/0L2dQDXXoitejWR.p3FY5FRsY7MNMjuQevQ0xYxtIw7rz7nyo8LxsnS21.
6f01KVyB/$15TJ7fy7MqVmMoh5YKhwFFZFSa24ZTW0ldrZDuKS4OAi877Jy4xDZvLdBRYnPNd6eUJh8kXlsS/IA5/UOwcVy1
6cIc5yNKq$fkTE/rhRgMTCevsHpLtV4Twjr.NeTQf66FOyfLV1BHFAMe8WKGkWpqqb.Di61fYTWa0DSwHheqylpxf60CACY0
$6$5uZJAv/5$JyQvtYGqquWSKjHny5uAgbGy9Sz20CnWPOMIlENvnWExcIrqnGRfuSw5FAXRUTIRwWu2yRKErEqkq.bx7V4L4.
$6$153g2JT8$bqP.d55WkGDabKtu.vLH3Oj2qmMxgtQvcHf6S6f1YsocYYMTAiLdQgLcZuEbLJSYAd4.WVsUX6wwX7g5lmTHx1
6YDGTRQDm$NqSv0CLHu.mU5BUNrGzcyKnXdZFxkY0EqY8OZraJrD2oFB5GcSkjCsi4mJenlgTFym8PNbtCI8/7z/yaD/EGM1
6Q3rU2V2t$AAONdb9gvNqcaChVmsLrzYpB8PCkguo9gOyhitwcPhjfoqCUG9AogXqZwx0hIwYXJ2mvCZNV5gFpOxL50Rlay1
┌──(danicarbuccia㉿kali)-[~/Downloads]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=sha512crypt cleaned_hashes.txt
Using default input encoding: UTF-8
Loaded 8 password hashes with 8 different salts (sha512crypt, crypt(3) 6 [SHA512 256/256 AVX2 4x])
Remaining 7 password hashes with 7 different salts
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:16 0.04% (ETA: 01:13:25) 0g/s 445.4p/s 3150c/s 3150C/s droopy..pokwang
0g 0:00:02:28 0.38% (ETA: 01:31:09) 0g/s 444.4p/s 3113c/s 3113C/s minnie14..jorie
0g 0:00:04:09 0.65% (ETA: 01:29:17) 0g/s 443.0p/s 3107c/s 3107C/s partzz..music69
0g 0:00:07:24 1.16% (ETA: 01:22:08) 0g/s 443.9p/s 3108c/s 3108C/s nsyncbsb..ne14abj
0g 0:00:08:57 1.41% (ETA: 01:20:46) 0g/s 443.2p/s 3103c/s 3103C/s mother82..mitch6
0g 0:00:09:09 1.44% (ETA: 01:20:50) 0g/s 442.9p/s 3103c/s 3103C/s hygienist..hiroaki
0g 0:00:09:11 1.45% (ETA: 01:20:23) 0g/s 443.2p/s 3102c/s 3102C/s ghetto09..fucku01
0g 0:00:12:28 1.97% (ETA: 01:19:53) 0g/s 441.4p/s 3090c/s 3090C/s twins31..trisan
0g 0:00:33:06 5.19% (ETA: 01:24:12) 0g/s 428.8p/s 3002c/s 3002C/s monet ... momo75
0g 0:00:36:01 5.68% (ETA: 01:20:38) 0g/s 429.6p/s 3007c/s 3007C/s hulababe1..hubby29
0g 0:00:43:46 6.91% (ETA: 01:19:29) 0g/s 429.9p/s 3009c/s 3009C/s 05620562..052569
0g 0:00:50:02 8.01% (ETA: 01:11:05) 0g/s 431.3p/s 3019c/s 3019C/s shannonbruce..shaniece5
0g 0:01:08:23 11.22% (ETA: 00:55:25) 0g/s 434.2p/s 3039c/s 3039C/s gavin0514..gatorade07

Attempted to use John the Ripper against the rockyou.txt wordlist but only got the same password:



Password for Pam Poovey found using CrackStation:

Enter up to 20 non-salted hashes, one per line:

f8a36d175b34985619937bac32e2020b

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
f8a36d175b34985619937bac32e2020b	NTLM	dr1ft

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.