

Tiffany Garcia

**Github Repo Link:** [https://github.com/tiffany6139/pass478crackingsynthetic\\_project.git](https://github.com/tiffany6139/pass478crackingsynthetic_project.git)

In this password cracking project, I developed a simulation of an attack in which the attacker would gain access from a user who was victim to a data breach attack. This experiment consists of having 2 docker containers named UbuntuUser1 and UbuntuAttacker within a network. The utilization of 3 python files were also developed to complete the attack. This project can be divided into 2 parts the password efficiency where I test the ability to hash 1000 passwords that will then be used by the attacker to simulate a dictionary attack. Then the next part is where I actually perform a demonstration of a root access attempt that's successful.

This project also has the following assumptions based on what ended up working completely. The tcpdump requires root access therefore I did have to know the password previously to get the pcap files. Another assumption is that the attack will have a range in which the user's password might be and using that will have 4 found hashes to simulate the attack. Also, due to hashcat's short keyspace I had to limit the total hashes to feasibility.

In the first part, I hashed all 1000 passwords using MD5, SHA1, and SALT with SHA256. This portion had no errors and was able to develop 2 files for each encryption method where one file had 4 hashes and the other had the rest of the hashes. This was done to find the amount of time it takes to decrypt the attacks in the attacker container. Once completed, I opened a terminal, to run the commands in README to complete a simulation of each hash decryption. To simulate part 2, I commented out the list of all 6 commands but can easily be uncommented to run the simulation. This portion shows hashcat's decrypted hashes and all of the commands were between 10-12 seconds. The only one that was not cracked was the SALT with SHA256 and that took about 30 seconds before it exhausted.

In the second part, I simulated the attack by knowing the four decrypted hashes from UbuntuAttacker terminal. For this attack I ran 2 terminals for UbuntuUser1. The first terminal was to run the tcpdump that would generate a pcap file once the process was finished. The issue with this was that I was looking for incorrect and correct attempts in the pcap but it wasn't there, instead information only appeared after the correct attempt. However, I developed an interactive script that shows after the sudo is ran "Success!" appears. This terminal is left running while another one is opened. In this terminal, I ran the bash of UbuntuUser1 and then ran the python file attackerscript.py to run an interactive script where the user is able to add the passwords found in UbuntuAttacker and then use those passwords to find the correct one.

I also proposed that I would be able to limit the number of attempts of sudo but that is already a built-in feature and the recommended method for securing a docker container is to just remove the use of sudo. This project completes all of the original proposed features such as using commonly used passwords to develop password hashes which would then be cracked by the

docker container using hashcat. There are still some errors that would've been fixed like the pcap file properly showing everything or the sudo being able to be modified.