Homework 1
CS 468: Network Security

September 1, 2023

Getting Started

The homework assignment is due at **September 15, 2023 at 11:59pm local time**. For this assignment, you are required to write your solution in Python programming language, version 3.5 or later. Files required for this assignment are provided along in `hw1_files.zip`. The goal of this assignment is to learn how to break ciphers and authentication systems. There are a total of two tasks in this homework assignment. The first task focuses on dictionary attacks and password cracking. Task 2, which is a continuation of task 1 focuses on breaking substitution ciphers. Make sure you read the complete handout before starting.

Task 1: Password Cracking With Dictionary Attacks (15 points)

A dictionary attack is a method to break authentication systems based on the assumption that users are likely to use common words (present in a dictionary) as passwords for accounts. One way to perform such an attack is actively trying to login with multiple password attempts until one guesses the correct password. This however, is infeasible as only a few incorrect attempts will trigger a firewall or an intrusion detection system and block the malicious IP address from making further login attempts. Alternatively, if the attacker somehow gets access to hashes of the users' passwords, they can perform an offline dictionary attack by computing hashes for a large range of dictionary words and then compare them to the actual password hashes. In this case, if two hashes match, the attacker will know the password and can directly authenticate in a single attempt.

Your task in this assignment is to perform an offline dictionary attack to crack passwords of user accounts on a linux server. Linux systems store the details of user accounts and their password hashes in the file: `/etc/shadow`. A simplified version of the `shadow` file from a linux server has been provided in `hw1_files.zip`. While there are many online tools that you can use, in this task you are required to write your own password cracker. To assist you, we have also provided `dictionary.txt` which you can use as a starting point.

The `shadow` file contains a total of 7 users. For this first task, you will crack passwords for `user1` through `user6` (we will come back to `user7` in task 2). Note that password hashes contained in the `shadow` file are of varying levels of difficulty and this will require you to diversify your cracking approach.

- Different passwords have been hashed using different hashing algorithms, `md5, sha1, sha256` etc.
- One password has been hashed along with a SALT.
- `user3` has enabled a built in additional security feature, in which their password was encoded with a caesar cipher before it was hashed.
- One user has their password in leet speak.

Note that while the simpler passwords will be in the dictionary, the ones which are encoded as ciphertext, or are in leet speak will require you to apply a transformation to words in your dictionary. Eg:

- The password `helloworld` encoded with a caesar cipher of shift 5 will be first converted into ciphertext `mjqqtbtwqi`, and then hashed.

- In leet speak, a user may have their password as `l37m31nn0w`, which is standard english is `letmeinnow`.

For this task, you can assume that all passwords are alphanumeric ranging from 5-12 characters in length (this also includes any leet speak transformations). You may assume a SALT to be of only numeric value, in the range (0-9) and 5 digits in length. The SALT is appended at the end of a password before it is hashed eg: `hash(passwordSALT)`.

What to Submit?

You are required to submit the following files:

1. `password_cracker.py`: This is main solution code. Make sure your code compiles and is well commented.
2. `passwords.txt`: Contains all usernames (user[1-7]) and their corresponding passwords that you cracked.

   Autograder format: The passwords will be graded with the autograder and you must submit it with the correct format to receive full credit. Each line must have both username and password separated with a colon:
   `user1:password1`
   `user2:password2`
   ...
3. `explanation_1.txt`: Describe in a paragraph or two your approach for this task. Additionally, report the **total execution time** required for your functions to crack the passwords for users 1-6. You can list the total time taken for each user, specifying the duration in seconds, minutes, or hours. Also mention the approaches you followed and any other hurdles/difficulties you faced.

Task 2: Breaking Substitution Ciphers (10 points)

For this task you will be cracking the password for `user7`. `User7` is smart and decides to create a custom security feature in which the password is encoded using a substitution cipher before it is hashed. `User7` also encrypts all his files and documents in his directory using the same substitution cipher and mapping. The file `encrypted.txt` in `hw1_files.zip` is a piece of encrypted text belonging to `user7` which we have managed to obtain.

Your goal in this task is to use the encrypted text to find the mapping of the substitution cipher used by `user7`, and then apply that same mapping to the passwords in `dictionary.txt` to figure out their password by performing a standard dictionary attack.

What to Submit?

For this task you are required to submit the following files:

1. `analysis.py`: This is the code the performs an analysis on the encrypted text and prints out the mapping, and is well commented.
2. `plaintext.txt`: The plaintext corresponding to the ciphertext in `encrypted.txt`.

   Autograder format: The plaintext will be graded with the autograder and you must submit it with the correct format to receive full credit. The string comparison is case insensitive, however, the punctuation must be correct.
3. `explanation.txt`: Describe in a paragraph or two about your approach to break the encryption. Also list any online resources you may have used.

Submission

To submit, create a zip archive of the 6 files, named `hw1.zip` and upload it to Gradescope.

Good Luck!