CptS 427/527
Homework #3

Instructor: Adam Hahn
Due: 10/4/2019 at 11:59 pm

Assignment Setup
This assignment will need a Linux system (or virtual machine) with the appropriate libraries installed (`crypt.h`). It will also require the `passwd_cracker_shadow.c` code, the `shadow` and `passwd` files, and a dictionary words file (`words.txt`) from the class Gitlab page.

Assignment
This assignment will explore some of the basic cryptographic and authentication concepts discussed in class. Specifically, it will explore command-line tools for performing encryption, along with developing a C program to perform password cracking in Linux.

Part 1: General Questions

Question 1: Refer to the slide demonstrating how password entropy can be calculated. How many bits of entropy do the following password schemes have:

a) Only lower case English alphabet characters, length 10.

b) Upper case and lower case English alphabet, numbers (0-9), and only the following special characters @#$%&, length 8 characters.

c) A random 6 digit (0-9) pin number.

d) If a human was planning to create a password with the same entropy as **b)**, how many characters would they need? Hint: use the NIST-based scheme for estimating human-based entropy as defined in class.

Question 2: Explain the benefit of salts to protect password hashes? What addition security do they provide and why?

Question 3: Assume you have two password hash files that both have the same `administrator` account and password (assume both use the same hash function). However, the file from `SystemA` uses 8 bit salts and the file from `SystemB` uses 32 bit salts. Which hash is easier to crack using assuming both brute force and rainbow tables? Why?

Question 4: Review the following challenge-response based authentication scheme. Assume k_{pu} is the claimants (C) public key (and is known to the verifier, V) and k_{pr} is their private key. C_{id} is the claimant's identifier (e.g., username). The Verifier first submits the C_{id} to the claimant, and the claimant concatenates that value with a signed hash, which verified by the Verifier using the public key.

V → C: C_{id}
C → V: C_id || $\{H(C\_id)\}k_{pr}$

What vulnerability exists in this scheme? How could an attacker manipulate it to gain system access?

Part 2: Cracking Linux Password Hashes

This section will explore how password hashes are stored on the Linux operating system. It will walk you through the password files and demonstrate tools to crack password hashes.

A. Take a look at the `password` and `shadow` files, (which can be found in `/etc/passwd` and `/etc/shadow` in a real system). The actual password hashes are stored in the `/etc/shadow`, while `/etc/passwd` stores more general account information. Also, read the following `man` pages "`man shadow`" and "`man 3 crypt`" to better understand the `/etc/shadow` file.

Question 5: What hash algorithm is used to create the password hash for users `user1-user5`? List and identify both the salt and hash for users `user1-user5`.

Question 6: Expand the program `passwd_cracker.c` on Github to implement a password cracker which will determine the passwords associated with users `user1-user5`. The program currently reads the shadow file (`shadow`) and a dictionary file (`words.txt`), parses the password file, and then enumerates through the passwords. You'll need to complete the following:

- Run the program with `sudo`, which is needed to read `/etc/shadow`
- Store the user id, hash, and salt into arrays so that we can use them later when checking against each word within the dictionary file.
- Enumerate through the dictionary and compare the hash of that word with the shadow hash value. To do this, you'll need to use the crypt function (from the `crypt.h` library). Here's an example:

```
hash = crypt(word, finalsalt);
```

- Check to see if a password is matched. If so, print out the resulting userid and password
- You can compile the program using the following command:

```
$ gcc -o passwd_cracker passwd_cracker.c -lcrypt
```

Submit your code

Question 7: Run your program for at least 30 minutes. What passwords were recovered for what users?