CPS 633
Lab 1 – Part 5 Word Problems

Alessandro Profenna
Tolaz Hewa
Steven Dong

1Q) Supposed 'hashed' passwords are salted using 4-bit salt values. Discuss the effect of using these salts in terms of computational and storage requirement of the password cracker described in section 3.

1A) If passwords are salted using 4-bit salt values, this allows for a much more secure database of login information. One of the primary reasons is because adding any additional values to a password exponentially increases the number of possible permutations of strings, which would use significant computational resources. This would work against the password cracker in Question 3, which searches through a database already filled with hundreds of thousands of entries. Also, since salt values are an additional security feature to hashing, there is a much lower chance of a rainbow table finding a match.

2Q) Passwords in this lab are allowed to be up to 12 character long. What happens if someone picks a null password, i.e. a password containing zero characters? Most people select passwords that are 8 characters or less. What is the impact of this on the security of the system?

2A) In our lab, we ensured that a password containing zero characters would not be allowed, since that would result in almost instant access to an account with only the user ID. In the situation where a password is 8 characters or less, the chances of a person or computer discovering the password increases. This is because a password of low number of strings has a much shorter list of possible permutations than a password with longer strings. Therefore, the security of a system with relaxed password rules is much less secure.

3Q) Suppose the encryption algorithm, E, used to produce password hashes is weak. For example, the output of the encryption algorithm used in this lab is simply a linear combination of the input bits. How can an attacker use this knowledge to attack the system? What are some of the property of a good cryptographic hash function that can be used to address this weakness?

3A) A weak encryption algorithm allows attackers to easily figure out how it works and use that knowledge to recreate its functionality. A simpler encryption algorithm will result in fewer attempts by the attacker to get it right. It is even possible that an attacker could create a reverse hash function. Once created, discovering the plaintext of a password would be quick and easy. Essentially, in order to prevent this, a good cryptographic hash function should be hard to invert. Ensuring that no hash key can be found does this. It is also important for the hash function to be designed so that each plaintext password results in a unique hash. It is possible that sometimes two passwords can have the same hash, which means that the function is weak.

4Q) To harden the security of password schemes, many systems will only allow only a limited number of unsuccessful attempts, as required in Part 1. However, the number of attempts are typically considered within a fixed period of time. What should be a typical time period? Discuss pros and cons of making this time shorter or longer.

4A) Figuring out the most effective time restrictions on locked password attempts can be quite subjective. It is a balance between protecting the system from attackers and trying not to inconvenience authentic users who happen to forget their password. Depending on the type of system, a locking of anywhere from an hour to a day seems reasonable. A database with sensitive information will typically use a longer lock-out and possibly notify the user via email if too many unsuccessful attempts are made. The benefit of this is that in the case of a brute-force attack by someone with malicious intent, the information is protected. The downside is that the actual user is unable to access his/her account during this time.