Exercises in Data Security: Sheet 1

Robin Sharp

Autumn 2005

These exercises are largely taken from Chapter 18 of Stallings' book "Cryptography and Network Security", 3rd. edition.

Exercise A: Assume that passwords are selected from 4-character combinations of 26 characters from the English alphabet. Assume further that an adversary is able to try out passwords at a rate of one per second.

1. Assuming that the system gives no feedback to the adversary until each attempt has been *completed*, what is the expected time to discover the correct password?
2. Assuming the system gives a beep every time an incorrect character is typed in, what is the expected time to discover the correct password?

Exercise B: A phonetic password generator picks two 3-letter segments randomly to make up each 6-letter password. The form of each segment is CVC (consonant, vowel, consonant), where $V = \{a, e, i, o, u\}$, and $C = A - V$. You may assume the the alphabet A consists of all the letters of the English alphabet.

1. What is the total password population?
2. What is the probability of an adversary guessing a password correctly?

Exercise C: Assume that passwords are limited to the use of the 95 printable ASCII characters and that all passwords are 10 characters long. Assume further that you have access to a password cracker which can perform encryptions at the rate of 6.4 million password encryptions per second. How long will it take to test exhaustively all possible passwords on a UNIX system where you have access to the encrypted form of the password via the file `/etc/passwd`?

Exercise D: The inclusion of a 12-bit *salt* in the Unix password scheme is said to increase the difficulty of guessing the password by a factor $2^{12} = 4096$. But the salt is stored in plaintext form in the password file in the same entry as the encrypted password. So the two characters used to derive the salt do not have to be guessed. Why is it claimed that use of the salt increases password security?

Exercise E: The SunOS-4.0 documentation recommends that the ordinary Unix password file be removed an replaced by a file `/etc/publickey`. An entry in the file for user A consists of the user's identifier id_A and two encryption keys: A's public key a and private key a^{-1}. To keep the private key secret, it is encrypted using a symmetric encryption algorithm[1], $\mathcal{E}$, with a third key, k, which is derived from the user's login password p_A, so that what is stored in the file is in fact $\mathcal{E}_k(a^{-1})$.

When the user logs in, the system decrypts this stored value to retrieve a^{-1} and checks that the password was in fact correct.

1. How can this be done?
2. Could an intruder attack the system and find A's password? If so, how?

[1] A symmetric algorithm is one which uses the same key for encryption and decryption.