CIS4367.01 **Brandon Thompson**
Prof. Elibol Due Date: 2/5/20

Homework 5

PROBLEM #3.1:

Explain the suitability or unsuitability of the following passwords:

a qwerty

b Einstein

c wysiwyg (for ”what you see is what you get”)

d drowssap

e KVK919

f Florida

g *laptop_admin#

h crzyp@ss

SOLUTION:

a All lowercase letters and they are consecutive on the keyboard. Unsuitable.

b No numbers, and password is a name, and thus is easily guessable. Unsuitable.

c Hard to guess phrase, more than 6 characters. Suitable.

d is the phrase ’password’ backwards. Unsuitable.

e No special characters, fewer than 8 characters. Unsuitable.

f No special characters or numbers, password is a name and this is easily guessable. Unsuitable.

g Despite special characters in the front and back the majority is a common phrase that would be guessed. Unsuitable.

h More than 6 characters, includes special character. Suitable.

PROBLEM #3.2:

An early attempt to force users to use less predictable passwords involved computer-supplied passwords. The passwords were eight characters long and were taken from the character set consisting of lowercase letters and digits. They were generated by a pseudorandom number generator with 2^{15} possible starting values. Using the technology of the time, the time required to search through all character strings of length 8 from a 36-character alphabet was 112 years. Unfortunately, this is not a true reflection of the actual security of the system. Explain the problem.

SOLUTION:
The number of possible strings that can be made from using 36 character with a length of 8 characters is $36^8 \approx 2^{41}$, but a pseudorandom number generator only has 2^{15} possible starting values. So you only need to check 2^{15} passwords.

PROBLEM #3.3:

Assume that passwords are selected from four-character combinations of 26 alphabetic characters. Assume that an adversary is able to attempt passwords at a rate of one per second.

a Assuming no feedback to the adversary until each attempt has been completed, what is the expected time to discover the correct password?

b Assuming feedback to the adversary flagging an error as each incorrect character is entered, what is the expected time to discover the correct password?

SOLUTION:

a Number of possible passwords $26^4 = 456976$, worst case check.

  Average amount of attempts is half the worst case. $\frac{26^4}{2} = 228488$ attempts, or 228488 seconds.
  Number of seconds in a day is $60 \times 60 \times 24 = 86400$
  $\frac{228488}{86400} = 2.644$ days or 63.456 hours to find the correct password.

b If user gets feedback with each character then the number of checks is $26 \times 4 = 104$ for the worst case. Then the average case is $\frac{104}{2} = 52$ checks or 52 seconds to find the correct password.

PROBLEM #3.6:

Assume that passwords are limited to the use of the 95 printable ASCII characters and that all passwords are 10 characters in length. Assume a password cracker with an encryption rate of 6.4 million encryptions per second. How long will it take to test exhaustively all possible passwords on a UNIX system?

SOLUTION:
The number of possible password combinations from 95 characters with a length of 10 is $95^{10} \approx 6 \times 10^{19}$. The encryption rate is 6.4 million per second, or 6×10^{6} passwords per second.
Then the total time to find all passwords on a UNIX system is:

$$\frac{6 \times 10^{19} \text{ passwords}}{6 \times 10^{6} \text{ passwords/second}} = 9.4 \times 10^{12} \text{ seconds.}$$

Seconds in a year is : $60 \times 60 \times 24 \times 365 = 31536000$ seconds $9.4 \times 10^{12} \times \frac{1}{31536000} =$ 298072 years to test all possible passwords.

PROBLEM #3.7:

Because of the known risks of the UNIX password system, the SunOS-4.0 documentation recommends that the password file be removed and replaced with a publicly readable file called /etc/publickey. An entry in the file for user A consists of a user's identifier ID_A, the user's public key, PU_a, and the corresponding private key PR_a.

This private key is encrypted using DES with a key derived from the user's login password P_a. When A logs in, the system decrypts $E(P_a, PR_a)$ to obtain PR_a.

a The system then verifies that P_a was correctly supplied. How?

b How can an opponent attack this system?

SOLUTION:
The private and public key are inverse's, to validate P_a, the value of PR_a can be verified by encrypting the public key with an arbitrary number X. Decrypt the encrypted value with PR_a.

$$X = D\left(PR_a, E\left(PU_a, X\right)\right).$$

PROBLEM #3.8:

It was stated that the inclusion of the salt in the UNIX password scheme increases the difficulty of guessing by a factor of 4096. But the salt is stored in plaintext in the same entry as the corresponding ciphertext password. Therefore, those two characters are known to the attacker and need not be guessed. Why is it asserted that the salt increases security?

SOLUTION:
Salt serves three purposes:

- Prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, they receive different salt values. So the hashed value of the passwords will be different.
- Increases difficulty of offline dictionary attacks. IF salt length is b bits, the number of possible passwords is increased by 2^b. Making it harder to guess a password.
- Makes it almost impossible to find out whether a person with passwords on two or more systems has used the same password on all of them.

PROBLEM #3.9:

Assuming that you have successfully answered the preceding problem and understand the significance of the salt, here is another question. Wouldn't it be possible to thwart completely all password crackers by dramatically increasing the salt size to, say, 24 or 48 bits?

SOLUTION:
No, protecting from password crackers depends on the user population. The password is generated with the original user entered password, a random salt, and a hash function. Increasing the salt size will stop multiple users from having the same salt value. If multiple users have the same salt value then the attacker needs to perform multiple encryptions per user password.

PROBLEM #3.11:

For the biometric authentication protocols illustrated in Figure 3.12, note that the biometric capture device is authenticated in the case of a static biometric but not authenticated for a dynamic biometric. Explain why authentication is useful in the case of a stable biometric but not needed in the case of a dynamic biometric.

fig3_12.jpg

SOLUTION:
In dynamic biometric authentication, a random number r, and a random sequence x are used as a challenge. Because of the randomness in these numbers there is no need to have another layer of authentication.