Computer Security
EDA263
Laboration 1

David Göransson 900618-1177
Emil Johansson 850425-5574

1. Introduction

This assignment is divided into two parts, the first part consists of answering a number of questions regarding identification and authentication, and the second part consists of an implemented version of a simple login shell.
The answers for the questions can be found under section 2 and the code is found under appendix A. The last part of this report is conclusions, where we will discuss this assignment as well as the authentication problem in general.

David Göransson 900618-1177
Emil Johansson 850425-5574

2. Answers to questions

1. What is password ageing and what methods exist to implement it?

Password aging is the concept of keeping track of the age of the password. This could then be used to force users to change passwords on a regular basis. There are policies that can be configured in the operating system forcing a user to change password.

2. If you want to increase the security of the system you can use one-time passwords. What are the advantages? What are the disadvantages?

Advantages

+ Very Secure, improves defense against bruteforce (especially if salt is used)

Disadvantages

- Hard to implement
- Impractical (forces user to have external source of one-time passwords)

3. Authentication systems are often based on some knowledge shared by the computing system and the user. This knowledge can be of three types; something the user knows, has or is. For each of these types, answer the following questions:

a. How can the authentication be implemented?

User knows

By the system requesting a password for the desired account, which only the user should know.

User has

Most common is ID badges. NFC or RF chips. But also Public/Private key cryptography, the user has a private key. Or a device that generates a OTP, common in 2-factor authentication.

User is

Biometric authentication, such as fingerprint scanner, iris scanners or voice-recognition. Takes advantage of the unique physical traits humans have.

b. What advantages and disadvantages does the authentication method have with respect to e.g. user friendliness, cost of introduction and accuracy.

User knows

+ Easy to implement

David Göransson 900618-1177
Emil Johansson 850425-5574

+ Portable (User can log in anywhere as long as he/she has a keyboard and remember the password)
+ Easy to use, wide spread use.
- User needs to remember password (can be hard, especially if strong password)
- If lost, there is no security at all.
- Impractical, lazy users reuse their password, and if that is the case the password is not stronger than the weakest link among the places the password is used.

User has

+ Requires possession of the key or device to access account thus higher security.
- User needs to keep OTP-device or private key secure.

User is

+ User friendly, user does not need to remember password.
+ If well implemented, strong security. Also uncommon thus harder to attack.
- Hard to implement
- Usually expensive

c. How can we overcome the disadvantages?

User knows

One could use third party software, such as LastPass or 1Password. So the user is only required to remember a single password instead of several. The passwords generated by the software can then be much more secure, and will also differ on different sites. However this leads to a single point of failure, if someone get hold of your master password all your bases are lost.

There are no good way to overcome the problem with a compromised password, one can of course change the password upon discovering this. But the chances of discovering it is in general low.

User has

More portable solutions, such as OTP in your cellphone.

User is

This is more of a futuristic solution and there is yet no good solution for the public. E.g Apples touchID was cracked within a week after launch.

4. What authentication method would you recommend for use in computer systems in the following environments. Motivate your answer and state the assumptions you have made concerning the security needs in each environment.

a. A university

Type 1, user knows. It is easy to implement and a cheap solution which is wide spread. If a password is lost the user can go to the administration desk and get his password restored. There are probably no real sensitive information.

b. A military facility

Depends on how critical the facility is. Most common is probably only Type 2, in form of a ID card. However if it is a high security level facility a Combination of Type 1 and 2 would be suitable, possibly also use of Type3. (Depends on what technology they have)

c. A middle-sized company

Type 1, possibly in combination of Type 2. It depends on what the company is working with, Also restrictions to area might be used, e. g Type 2 to enter the company building and within the building the user can connect to the intranet by using a Type 1 authentication.

d. A personal home computer

Most likely a low threat level, a simple password should be enough.

5. In some systems you do not only implement authentication of the user against the system but also of the system against the user. What is the purpose of doing this?

Both parts identifying each other. E.g a user wants to connect to bank and thus we use a certificate, so the user knows he is connected to the correct entity and not a hacker.

6. A user is running a program containing the system call setuid(). Depending on who is running the program and the value of the argument to setuid(), different things will happen. Check the manual page for setuid(2) on the lab system (man -s2 setuid).

a. Study Figure 1. What are the values of the real user ID (ruid) and the effective user ID (euid) at i) and ii)?

i)
ruid = csec028
euid = csec028
ii)

David Göransson 900618-1177
Emil Johansson 850425-5574

ruid = csec028
euid = csec028

Table 1 illustrates six cases where root or a normal user account starts a normal program which uses setuid() with different arguments (user IDs). Fill in the third and fourth column in the table. The third column should state whether the system call will succeed or fail. The fourth column should state the user ID (ruid) of the program after the setuid() call.

Current User	setuid(UID)	success/failure	user ID after setuid()
root	0 (root)	success	0
root	20757 (csec069)	success	20757 (csec069)
root	20716 (csec028)	success	20716 (csec028)
csec028	0 (root)	failure	csec028
csec028	20757 (csec069)	failure	csec028
csec	20716 (csec028)	success	csec028

7. On Unix systems file access permissions on programs may be set to set-user-ID (these are often called SUID programs) as in the passwd program:

csec@legolas~ > ls -l /usr/bin/passwd

-rwsr-xr-x 1 root root 27888 Jul 26 09:22 /usr/bin/passwd

Study Figure 2 while answering the following questions:

a. What are the values of the real UID (ruid) and effective UID (euid) at i), when the user csec028 runs the SUID program /bin/prog?
ruid = csec028
euid = root

b. What is the purpose of using SUID on programs?
Allowing non-root users to do privileged actions without having the real privileges.

David Göransson 900618-1177
Emil Johansson 850425-5574

8. Sometimes the setuid() (or perhaps more likely, seteuid(2)) function is used in programs where the set-user-ID bit is activated on the program's file access permissions. This is done for example in the /usr/bin/passwd program.

a. What are the values of ruid and euid at i) and ii) in the Figure 3?

i)
ruid = csec028
euid = root

ii)
ruid = csec028
euif = csec028

b. What is the purpose of doing this?

Allowing a user to temporarily have the privileges of root, to do a privileged task. If control is later given to the user, the privileges have been dropped and thus the user can not regain the root permissions.

David Göransson 900618-1177
Emil Johansson 850425-5574

3. Conclusions

We implemented a fairly simple solution for defending against multiple login-attempts. We set a limit to 1000 tries, and if that number is exceeded the account will be locked and could only be unlocked by an administrator. This will protect against brute-force attacks but could introduce another problem, more specifically denial of service attacks. An attacker could be able to use a fast script to create a large amount of login-attempts which would in the end make it impossible for a correct user to login. While discussing our implementation with the TA an extension to our solution was mentioned, which was to implement a small delay between an unsuccessful attempt to login and the possibility for another attempt. This would make it a lot slower for a script to try a number of passwords, and also harder to deny access for multiple users.

One interesting aspect with this assignment was to see how buffer overflow works in practise, and that it would also be a rather simple mistake to make in the code. It is also easy to understand that this would be a rather powerful tool to be used by an attacker, as they would be able to change a program to execute any desired code submitted by the attacker.

David Göransson 900618-1177
Emil Johansson 850425-5574

4. Appendix A

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdio_ext.h>
#include <string.h>
#include <signal.h>
#include <pwd.h>
#include <sys/types.h>
#include <crypt.h>
#include "pwent.h"

#define TRUE 1
#define FALSE 0
#define LENGTH 16
#define LOGIN_LIMIT 1000
#define AGE_LIMIT 10

int main(int argc, char *argv[]) {
  signal(SIGINT, SIG_IGN);

  mypwent *passwddata;

  char important[LENGTH] = "***IMPORTANT***";
  char user[LENGTH];
  char prompt[] = "password: ";
  char *user_pass;

  int i = 0;
  sighandler();

  while (TRUE) {
    /* check what important variable contains - do not remove, part of buffer overflow test */
    printf("Value of variable 'important' before input of login name: %s\n",
                            important);

    printf("login: ");
    fflush(NULL); /* Flush all  output buffers */
    __fpurge(stdin); /* Purge any data in stdin buffer */
    /* user fgets to avoid buffer overflow attacks */
    if (fgets(user, LENGTH, stdin) == NULL)
      exit(0);

    /* check to see if important variable is intact after input of login name - do not remove
*/
```

David Göransson 900618-1177
Emil Johansson 850425-5574

```
printf("Value of variable 'important' after input of login name: %*.*s\n",
                          LENGTH - 1, LENGTH - 1, important);

/* Make sure that line ends with \0 so we can search for username. */
for(i=0; i < LENGTH; i++){
  if(user[i] == '\n'){
    user[i] = '\0';
    break;
  }
}

user_pass = getpass(prompt);
/* Changed to use mygetpwnam instead of default mygetpwnam */
passwddata = mygetpwnam(user);

if (passwddata != NULL) {
  /* If login limit is reached, notify user and end login process */
  if(passwddata->pwfailed > LOGIN_LIMIT){
    printf("Your account has been suspended, contact the administrator\n");
    return 0;
  }
  /* Encrypt the password entered by the user and validate with the user data */
  user_pass = crypt(user_pass, passwddata->passwd_salt);
  if(!user_pass){
    /*Crypt failed, exit login shell */
    printf("Crypt failed, contact the administrator\n");
    return 0;
  }

  /* Compare the encrypted passwords */
  if (!strcmp(user_pass, passwddata->passwd)) {
    /* If there has been failed login attempts notify the user */
    if(passwddata->pwfailed != 0){
      printf(" *** WARNING! *** Failed attemps since last login: %d\n",
                  passwddata->pwfailed);
      passwddata->pwfailed = 0;
    }
    /* If password has reached its age notify the user */
    passwddata->pwage++;
    if(passwddata->pwage > AGE_LIMIT){
      printf(" *** WARNING! *** Your password is getting old, like Gandalf!\n");
    }

    /* Update the user login data (e.g pwfailed reset) */
    mysetpwent(user, passwddata);

    /* Lower the rights of the program to the user privilege level */
    if(setuid(passwddata->uid) == 0){
      printf(" You're in !\n");
      /* Start shell */
```

David Göransson 900618-1177
Emil Johansson 850425-5574

```
          execl("/bin/sh", "sh", NULL);
        }
        /* Setuid failed*/
        printf("Setuid failed, contact the administrator\n");
        return 0;
      } else {
        /* If bad password, increase pwfailed counter */
        passwddata->pwfailed++;
        mysetpwent(user, passwddata);
      }
    }
    /* Notify user about incorrect login information. */
    printf("Login Incorrect \n");
    /* Delay to prevent the evil power of bruteforce! */
    sleep(1);
  }
  return 0;
}
```

David Göransson 900618-1177
Emil Johansson 850425-5574