Laboratory Assignment in Computer Security EDA263/DIT641

IDENTIFICATION AND AUTHENTICATION REPORT.

Group 27
Mujurizi Coleb & Michael Abebaw

1. Introduction

This is a simple report for the laboratory and home assignment on user identification and authentication which refer to the process of verifying an identity claimed by or for a system entity. The report contains answers to the home assignment questions, conclusions made from both the authentication problem in the lab and the home assignment, and the appendix which contains a commented c programme code that was used to implement the solution in the lab. The problem in question was to implement a login program that mimics the traditional UNIX login, takes a username for identification and a salted password for verification. It uses a user database to restrict access of unauthorized users and is resistant to buffer overflow attacks and online password guesses. The solution also implements the password aging feature.

2. Answers to the home assignment questions.

Answers to question 3.1.1

Password aging is a technique of setting a certain period of time after which the user is prompted to create a new password. It is used by system administrators to defend against bad passwords in organizations.

The following methods among others exist to implement password ageing.
Administrator sets "password expiry warning messages" during login reminding users to change their passwords before expiry.

- If the passwords expire before they are changed, on the next log on after expire, a user is forced to first change the password before they are let in. A warning message like this is always set by the administrator *"WARNING: Your password has expired. You must change your password and login again"*.
- If the user account becomes inactive for a number of days after password expiry, the administrator can block the account. This will push the user to have his account activated again hence acquiring a new password.

Answers to question 3.1.2

Advantage of one time passwords:

- They are not vulnerable to replay attacks and eavesdropping.

Disadvantages:

- One time passwords are very hard for humans to memorize.
- They require additional technologies to work.
- In case the authentication server is compromised by the adversary, it can make way for the man-in-the-middle attack.

Answers to question 3.1.3

i.) *"Something the user knows"* is implemented through use of passwords, Personal Identification Numbers, or answers to a set of predefined questions.

Advantages:

- It is the least expensive authentication method to use.
- No need to carry additional hardware or devices.
- No need to install extra software.
- PINs and passwords can be changed at the user's convenience.

Disadvantages:

- An adversary may be able to guess passwords or stole them easily.
- A user may forget the password.
- It is weak and susceptible to various attacks.
- Its security depends on the user's ability to keep the passwords in secret.

Countermeasures:

- Preventing unauthorized access to password files.
- Use of intrusion detection measures to identify a compromise, and rapid reissuance of passwords in case the password file has been compromised.
- Use of hashing methods in encrypting passwords to make it harder for attackers to guess or crack them.

ii). *"Something the user has"* is implemented through use of smart cards, electronic keycards and physical keys.

Advantages:

- Users don't have to remember complex passwords.
- More secure to use than passwords.
- Attackers need to physically get possession of the authentication tool in order to get access which is not so easy.

Disadvantages:

- Additional costs in terms of replacement fees are incurred in case of losses.
- Users need to always carry the authentication tools/keys with them.
- An adversary may gain physical possession of the keycards and duplicate them.
- They require special readers.

Countermeasure:

- Users should jealously protect their possessions from external parties.

iii) *"Something the user is"* can be implemented through the use of fingerprints, facial characteristics, iris and retina pattern.

Advantages:

- A user doesn't need to carry any hardware nor remember complex passwords.
- It's the most secure authentication method and is very difficult and almost impossible to forge/falsify.

Disadvantages:

- It requires expensive equipment to input fingerprints, retina scans and face recognition.
- Causes stigma to the users who think the authentication procedures they go through may cause harm to their bodies.
- Change in the user's age or illness may cause change in face or temporal loss of friction ridge in fingerprint which can make identification difficult or impossible.

Countermeasure:

- User training should be done to educate the users and make them feel comfortable about their health when using the authentication systems.

Answers to question 3.1.4

a). A University

For an environment such as a university where there are many users accessing computer systems, passwords and PINs ("Something the user knows") is sufficient as they are cheap to deploy and easy to manage from administration point of view.

b). A Military Facility

"What the user is" authentication methods is recommended in this scenario. This is because a military facility is a very sensitive environment that needs maximum security measures which don't need to be compromised in any way. Also accesses are given to the right individual only. A two or more step measures can also be implemented in order to enhance the security measure. This type of authentication method is hard for adversaries to beat.

c). A Middle-sized Company

"Something the user has" authentication method is suitable for this since it would be affordable for every employee to have a keycard or smart card. It also ensures extra security than the security offered by use of passwords and PINS. Since the attacker needs to physically gain possession of the cards, it makes it hard to get around. If company's computer systems store critically important information that can't be risked to attack, additional biometric systems can then be deployed side by side.

d). A personal home computer

For a personal computer, passwords can be used for authentication since they are easy and cheap to create and there is not so much critical information to protect in personal home computer.

Answers to question 3.1.5

The system has to be authenticated against the user in order to avoid the "man in the middle" attacks. A user at one computer A needs to be sure that he/she is communicating with genuine computer B at the other end and the reverse should be true. A man in the middle attacker shouldn't be given any chance of spoofing the communication between the two hence authentication should be implemented in either way.

Answers to question 3.1.6

a) i. RUID =EUID = 20716 ii. EUID = 20716, RUID =20716

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

Answers to question 3.1.7

a) i. RUID for csec028 is 20716 ii. EUD for csec028 is 0

b) To temporarily escalate privileges of a user on a program. The user attains the privileges of the owner of the program.

Answers to question 3.1.8

a) i. EUID=0 , RUID=20716 ii. EUID=20716, RUID= 20716

b). Purpose is to cause any users or processes that run the file to have access to system resources as though they are the owners of the file. Setting the SUID bit for a program to the 'root' user should be discouraged because if the program is badly written and is manipulated through malicious input, it could cause a normal user to have access to all other users' programs since he will be accessing them as root. It exposes the system to attacks if an attacker gains access to a normal user privileges. He can then pass through the normal user program to do buffer overflow attacks on root which has its SUID bit set.

3. Conclusion

User authentication is an important step towards guarding a computer system from unauthorized access. The authentication procedure, however, can be vulnerable to different types of attacks that might occur at any of its stages. In this lab assignment, we dedicated ourselves at preventing some of those vulnerabilities such as the buffer overflow attacks, brute force attacks and password guessing.

Buffer overflow was prevented by implementing a function in our program that restricts the length of user input. We deployed a hashed password algorithm with a salt value to enhance the strength of users' passwords which makes it hard for the attacker to break.
Brute force attacks were prevented in our program by discouraging a user from making several consecutive failed login attempts. After a certain number of failed attempts, the system hangs for a few minutes before it can accept user input again. This helped to stop any kind of brute force/script attacks. The home assignment helped us to exhaustively understand the different authentication methods, their pros and cons and most importantly, the use of SUID command and the setuid( ) function call. Generally, this authentication problem and assignment challenged us as future systems administrators and presented us with better techniques of keeping our systems safe from unauthorized access and improving their security.

3. Appendix

```
/* $Header:
https://svn.ita.chalmers.se/repos/security/edu/course/computer_security/trunk/lab/login_linux/log
in_linux.c 585 2013-01-19 10:31:04Z pk@CHALMERS.SE $ */

/* gcc -Wall -g -o mylogin login.linux.c -lcrypt */

#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"
#include "pwent.c"

#define TRUE 1
#define FALSE 0
#define LENGTH 16
#define PWDAGE 10
#define WRGPWD 5

void sighandler() {
        // signal handling routines
        // SIG_ING ignores the signal
        signal(SIGTSTP,SIG_IGN); // to prevent Ctrl+Z
        signal(SIGQUIT,SIG_IGN); // to prevent Ctrl+'\'
        signal(SIGINT,SIG_IGN); // to prevent Ctrl+C
}

int main(int argc, char *argv[]) {

        mypwent *passwddata;

        char important[LENGTH] = "***IMPORTANT***";
```

```
        char user[LENGTH];
        char *c_pass;
        char prompt[] = "password: ";
        char *user_pass;

        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 */

                if (fgets(user,LENGTH,stdin) == NULL)
                        exit(0);

                user[strlen(user)-1]='\0';

                /* check to see if important variable is intact after input of login name - do not
remove */
                printf("Value of variable 'important' after input of login name: %*.*s\n",
                                LENGTH - 1, LENGTH - 1, important);

                user_pass = getpass(prompt);
                passwddata = mygetpwnam(user);

                // if the user data is found
                if (passwddata != NULL) {
                        // get the hash value of the given pwd
                        c_pass = crypt(user_pass, passwddata -> passwd_salt);
                        if(c_pass == NULL) {
                                printf("An error has occurred while hashing!\n");
                                continue;
                        }

                        // if login is successful
                        if (!strcmp(c_pass, passwddata -> passwd)) {

                                printf(" You're in ! \n");

                                // display failed login attempts
                                printf(" Number of failed login attempts: %d\n", passwddata-
>pwfailed);

                                // reset pwd failed variable
                                passwddata -> pwfailed = 0;

                                // increments age of the pwd
                                passwddata -> pwage += 1;

                                // prompt the user to change pwd
                                if (passwddata -> pwage >= PWDAGE)
                                        printf("It is now time to change your password!\n");

                                // saving the modified data
                                if (mysetpwent(user, passwddata)==-1) {
                                        printf("An error has occurred while saving!");
                                        continue;
```

```
                                }

                                // execute /bin/sh
                                if (setuid (passwddata->uid) == -1) {
                                        printf("setuid failed! \n");
                                        continue;
                                }
                                  else
                                        system("/bin/sh");

                        }
                        // if login is not successful
                        else {
                                // increments number of failed pwd
                                passwddata -> pwfailed += 1;

                                // protecting bruteforce attack
                                if (passwddata -> pwfailed >= WRGPWD) {
                                        printf("You entered wrong password too many times.\n Wait
for 5 minutes and try again!\n");
                                        sleep(300);  //After a number of failed attemps, the
systems suspends login for 5 minutes before it resumes again
                                }

                                // saving
                                if (mysetpwent(user, passwddata) == -1)
                                        printf("An error has occurred while saving!");
                        }
                }
                // user data not found
                else {
                        printf("The user is not found in the database! \n");
                        printf("Login Incorrect \n");
                }
        }
        return 0;
}
```