MIECT: Security 2020-21

Practical Exercises: Pluggable Authentication Modules

November 24, 2020 Due date: no date

Changelog

- v1.0 - Initial version.

1 Introduction

The goal of these exercises is to explore the functionalities of PAM (Pluggable Authentication Modules) infrastructure present in current Linux distributions.

These exercises will also make use of the Portuguese Citizen Card as a way to authenticate users and to provide detailed logs.

Errors in PAM configuration files may block further access to the system. Beware! Keep at all times a session as root to be able to recover from mistakes!

2 Weak Passwords

By default, current Linux distributions do not verify the strength of the passwords in use. A weak password is a password which can be found through an intelligent search attack where a set of more likely passwords are tested (dictionary attack). The widespread availability of dictionaries with words and even well-known passwords, makes this attack very effective. Once the password is found, the security of the entire system is compromised as it can allow the execution of local exploits and access private data.

This policy can be changed so that the password modification procedure verifies the strength of the new passwords. In Linux these procedures are controlled using PAM (*Pluggable Authentication Modules*), which is also used to control several other authentication and access control aspects of any Linux system.

2.1 The pam_cracklib PAM module

Install `libpam-cracklib` using `apt-get`.

Take a look at the manual of `pam_cracklib` (`man pam_cracklib`) and understand the purpose of the module.

Create a test user named `beaker`[1] (with the `adduser beaker` command) and set a password for this

[1] Beaker is a character from The Muppet Show; he is a laboratory assistant and often a Guinea-pig for dangerous experiments.

Photo extracted from `https://en.wikipedia.org/wiki/Beaker_(Muppet)` user. Verify that you can login into its account (with the `su -l beaker` command).

The file `/etc/pam.d/common-password` stores the rules controlling what happens when users change passwords. This is a good candidate for enforcing strong passwords and is typically used for this purpose.

NOTE: as a general safety rule never close the editor where you change PAM configuration files before testing the effect of the modifications performed. Otherwise, you may not be able to revert your mistakes!

- Propose a reasonable policy for password security and enforce it. Consider password length, existence of symbols (e.g. `_,;:"'`~^+-*/=<>[]{}()!?@#$%&\`), lower and upper case characters and (decimal) digits.
- In order to better enhance the difficulty of finding the password, enforce a policy so that common passwords found in the system dictionary are forbidden. You can use the dictionary files available at `/usr/share/dict`.

Using the `beaker` user account, test the correct enforcement of the password safety model you have implemented.

In the end, please restore all PAM-related configurations! Otherwise you may be unable to log in into the system.

3 One-Time Passwords

One-Time Passwords (OTPs) can also be used for authenticating users. OTP systems can have different flavours; the oldest one computes uses a list of OTPs that users must print and carry with them. Other system use an application executed in a personal device for computing OTPs. In this guide we experiment the former.

As expected, PAM supports printed OTPs by means of a library and a definition in the configuration files. In Linux systems, it is required to install the packages `libpam-otpw` and `otpw-bin`.

If not available, get the source code file `otpw-1.5.tar.gz` from `https://www.cl.cam.ac.uk/~mgk25/download`, unpack it (with

```
tar xzvf otpw-1.5.tar.gz
```

install the package `libpam0g-dev`, run the

```
make
```

command to generate the necessary binaries and, finally, run

```
sudo make install
```

to install those binaries on the right directories. The `pam_otpw.so` library will be installed in the correct directory (`/lib/x86_64-linux-gnu/security`).

The next step is to add the `pam_otpw.so` module to the appropriate place in the PAM authentications stack. For the purpose of this laboratory guide, add it to the `common-auth` file, as an alternative authentication method relatively to the ones already existing:

```
auth    [success=2 default=ignore]      pam_unix.so nullok_secure
auth    [success=1 default=ignore]      pam_otpw.so
auth    requisite                       pam_deny.so
```

You should take in consideration that the order is important! Place `pam_otpw.so` after `pam_unix.so`. Note that with this configuration you can choose among the two types of authentication, Unix first, OTP second; if the Unix fails, OTP will be attempted. If they both fail, the authentication will fail.

NOTE: do not close the editor! Just write the contents of file being edited. Otherwise, you may not be able to revert your mistakes!

After configuring PAM for using `pam_otpw.so` users can choose to use OTP by running the `otpw-gen` command. This will generate a file named `~/.otpw` containing some metadata, as well as a list of one-way hashes for the purpose of verifying the response to challenges.

Run the command in order to create the above referred file. You will need a password prefix. Choose any string and remember it! The prefix will be required for authentication purposes. A table will also be printed to the generator's standard output; this table provides the responses to the several challenges initiated by the PAM OTPW module. In a real exploitation scenario, this table would be printed on a sheet of paper or on a card. In this exercise you can save it on a file by redirecting the output of the generator.

To test if the system is working, execute as `root` the `login beaker` command. Provide an empty password for the first password prompt. You should be presented with a new password prompt carrying a number as challenge. The correct answer is composed by the password prefix and the corresponding entry in the previously referred table (including or not the space). If multiple numbers are provided, you must provide the answer to all challenges. Spaces are ignored.

4 Authentication using Smartcards

An example source code in C of a PAM module implementing local authentication with the Portuguese Citizen Card (CC) can be found at: `https://code.ua.pt/projects/ccpam`. We can get the source code using `git` and the following command:

```
git clone https://code.ua.pt/git/ccpam
```

Before you can compile the module, you should install the PAM library development files (`libpam0g-dev`) and the C language PKCS#11 development files (`libopencryptoki-dev`) using `apt-get`.

Compile the module by executing

```
make
```

and install it with

```
sudo make install
```

Note: The test cards fail the integrity verification. In order to use these cards with the PAM library, you must remove the call to the functions `PTEID_SetSODCAs` and `PTEID_SetSODChecking` in the `addCCuser.c` and `pam_PTEIDCC.c` files.

Edit (as root) the PAM configuration file `/etc/pam.d/common-auth` and a line the line using `pam_PTEIDCC.so` as follows:

```
auth    [success=3 default=ignore]      pam_unix.so nullok_secure
auth    [success=2 default=ignore]      pam_otpw.so
auth    [success=1 default=ignore]      pam_PTEIDCC.so /etc/CC/keys
auth    requisite                       pam_deny.so
```

as the third rule in the authentication stack.

NOTE: do not close the editor! Just write the contents of file being edited. Otherwise, you may not be able to revert your mistakes!

Using the `addCCuser` tool to bind a given CC to the test user (the default file is `/etc/CC/keys`).

Verify that you can use the CC to authenticate the test user.

Modify the library so that the actual name and BI of the user are logged to the `auth.log` file. You will have to consult the CC SDK documentation in order to determine the relevant PTEID API functions to use.

The following code may be helpful to log messages to `syslog`:

```c
#define PACKAGE "pam_PTEIDCC"

static void pam_cc_syslog(int priority, const char *format, ...) {
    va_list args;
    va_start(args, format);

    openlog(PACKAGE, LOG_CONS|LOG_PID, LOG_AUTHPRIV);
    vsyslog(priority, format, args);
    closelog();
    vfprintf(stderr, format, args);
}
```

Listing 1: Helper Function to write a string to syslog

```c
pam_cc_syslog(LOG_ALERT,"Name: %s Civil ID: %u\n", name, civil_id);
```

Listing 2: Example usage of the helper function

Once again, verify that this information is indeed present in the system authentication log file, `/var/log/auth.log`.

5 References

- Portuguese Citizen Card web site: `https://www.cartaodecidadao.pt`
- Linux PAM web site: `http://www.linux-pam.org`
- Linux PAM OTPW web site: `https://www.cl.cam.ac.uk/~mgk25/otpw.html`