ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Факультет безопасности информационных технологий

Кафедра проектирования и безопасности компьютерных систем

Дисциплина:

«Операционные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

Выполнили:

Студент группы N3248

Назаров М.В

Проверил:

Савков Сергей Витальевич

Санкт-

Петербург 2022г.

PAM-модуль

Усиленный вариант

1. Придумать и написать свой PAM-модуль (сложная авторизация действий)

Идея для авторизации**:**

Идея заключается в том, что вместо обычного ввода пароля, который несложно брутфорсится, мы можем использовать умножение матрицы пользователя(значение которое значет только пользователь) на рандомно генерирующуюся строку из поля целых по модулю простого числа(желательно большого для усложнения обращения), таким образом, мы должны лишь проверить правильно ли пользователь посчитал значение произведения секретной матрицы на рандомный вектор-столбец по модулю простого числа.

Код PAM-модуля:

```
#include <security/pam_modules.h>
#include <stdarg.h>
#include <time.h>

#define PAM_SM_AUTH
#define MAX_V 41

PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags, int argc,
const
char **argv) {
    unsigned int ctrl;
    int retval;
    const char *name, *p;
    char *right;
    time_t mytime;
    mytime = time(0);
    srandom(mytime);
    int random_vector[4];
    random_vector[0] = random() % MAX_V + 1;
    random_vector[1] = random() % MAX_V + 1;
    random_vector[2] = random() % MAX_V + 1;
    random_vector[3] = random() % MAX_V + 1;
    retval = pam_get_user(pamh, &name, "Enter your login: ");
    {
        struct pam_conv *conv;
        struct pam_msg *pmsg[4], msg[4];
        struct pam_response *response;
        retval = pam_get_item(pamh, PAM_CONV, (const void **) &conv);
        pmsg[0] = &msg[0];
```

```
        msg[0].msg_style = PAM_PROMPT_ECHO_OFF;
        msg[0].msg = malloc(100);
        snprintf(msg[0].msg, 180, "Your vector is %d %d %d %d\n",
random_vector[0], random_vector[1],
                 random_vector[2]);
        retval = conv->conv(1, (const struct pam_msg **) pmsg, &response,
conv->appdata_ptr);
        int user_arr[4][4] = {{1, 1,  1,  1},
                              {1, 2,  4,  8},
                              {1, 3,  9, 27},
                              {1, 4, 16, 64}};
        int user_answer[4] = {0};
        for (int i = 0; i < 4; ++i) {
            for (int j = 0; j < 4; ++j) {
                user_answer[i] += user_arr[i][j] * random_vector[j];
            }
        }
        right = malloc(100);
        snprintf(right, 40, "%d %d %d %d", user_answer[0] % MAX_V,
user_answer[1] % MAX_V, user_answer[2] % MAX_V,
                 user_answer[3] % MAX_V);
        if (!(strcmp(right, response->resp))) {
            return PAM_SUCCESS;
        } else {
            return PAM_AUTH_ERR;
        }
    }
    return PAM_SUCCESS;
}

PAM_EXTERN int pam_sm_setcred(pam_handle_t *pamh, int flags, int argc, const
char **argv) {
    unsigned int ctrl;
    int retval;
    retval = PAM_SUCCESS;
    return retval;
}
#ifdef PAM_STATIC
struct pam_module _pam_unix_auth_modstruct = {
"pam_test",
pam_sm_authenticate,
pam_sm_setcred,
NULL,
```

```
NULL,
NULL,
NULL,
};
#endif
```

Скомпилируем наш PAM-модуль с помощью:

sudo apt-get install libpam0g-dev gcc

библиотека Подключаемых Модулей Аутентификации (PAM)

Содержит динамическую C библиотеку для Linux-PAM, которая позволяет администратору локальной системы выбрать способ, с помощью которого приложения будут производить аутентификацию пользователей. Другими словами, без переписывания и пересборки совместимых с PAM приложений, можно переводить эти приложения на разные механизмы авторизации. Вообще можно производить обновление локальной системы аутентификации не меняя сами приложения

gcc -fPIC -c pam_test.c - собираем с независимостью от сдвига адресов и без привязки линкера.

ld -x --shared -o pam_test.so pam_test.o - делаем динамическую библиотеку с удалением локальных символов

Поместим его в каталог с другими модулями и назначим рута владельцем:

sudo chown root:root pam_test.so - назначаем владельцем библиотеки рута

sudo cp pam_test.so /lib/x86_64-linux-gnu/security/ - здесь хранятся pam-модули

Дальше необходимо поменять конфиги PAM.

Для этого в файле */etc/pam.d/login* пропишем нашу аутентификацию:

sudo vim /etc/pam.d/login

auth requisite pam_test.so

После всех этих действий наш PAM-модуль активен, хоть он и не отменяет стандартной аутентификации по простому паролю.

Проверим его работу:

Парольная матрица пользователя any:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{bmatrix}$$

При вводе пароля 48 139 326 645 и после ввода обычного
пароля для any получаем доступ в систему.
Теперь отключим стандартную аутентификацию с помощью редактирования файла
/etc/pam.d/login и попробуем заново*:*

При вводе 79 139 239 397 мы получаем доступ в систему, при этом обычный пароль запрошен не был.

Вывод: мы написали рабочий PAM-модуль, который использует необычный способ аутентификации, устойчивый к брутфорсу. Вероятность угадать пароль крайне мала, даже с учетом того, что мы используем небольшие значения. При увеличении размерности матрицы и простого модуля случайных чисел вероятность угадать пароль будет стремиться к нулю.