Задание: (Усложненный вариант) Придумать и написать свой PAM-модуль (сложная авторизация действий).

Task: (Complicated version) Invent and write your own PAM module (complex authorization of actions).

Ход работы

На основе кода https://github.com/beatgammit/simple-pam на языке C был написан PAM-модуль и программа, тестирующая его.

PAM-модуль:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <security/pam_appl.h>
#include <security/pam_modules.h>
#include <unistd.h>
#include <pwd.h>

PAM_EXTERN int pam_sm_setcred( pam_handle_t *pamh, int flags, int argc, const char **argv ) {
	return PAM_SUCCESS;
}

PAM_EXTERN int pam_sm_acct_mgmt(pam_handle_t *pamh, int flags, int argc, const char **argv) {
	return PAM_SUCCESS;
}

PAM_EXTERN int pam_sm_authenticate( pam_handle_t *pamh, int flags,int argc, const char **argv ) {
	int ret;
	const char* user;
	ret = pam_get_user(pamh, &user, "Username: ");

	printf("\nHi %s!\n", user);

	if (ret != PAM_SUCCESS) {
		printf("Something went wrong...\n");
		return ret;
	}

	struct passwd *user_passwd = getpwnam(user);
	if (user_passwd == NULL) {
		printf("User %s doesn't exist\n", user);
		exit(1);
	}

	char answ[20];
	printf("Or not %s\n", user);
```

```
        sleep(1);
        printf("How can you prove it's you?\n");
        sleep(1);
        printf("Haha, just kidding... ");
        sleep(1);
        printf("Or not.\n");
        sleep(1);
        printf("Who are you?\n");
        scanf("%s", answ);

        if (strcmp(answ, user) != 0) {
                printf("You're not %s, bye...\n", user);
                exit(1);
        }

        return PAM_AUTH_ERR;
}
```

Тест:

```
#include <security/pam_appl.h>
#include <security/pam_misc.h>
#include <stdio.h>

const struct pam_conv conv = {misc_conv, NULL};

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

        if(argc != 2) {
                printf("Usage: app [username]\n");
                exit(1);
        }

        pam_handle_t *pamh = NULL;
        int ret;
        const char *username = argv[1];

        ret = pam_start("pam_test", username, &conv, &pamh);

        if (ret == PAM_SUCCESS) {
                printf("Credentials accepted.\n");
                ret = pam_authenticate(pamh, 0);
        }

        printf("\n");

        if (ret == PAM_SUCCESS) {
                printf("Account is valid.\n");
                ret = pam_acct_mgmt(pamh, 0);
        }
```

```
        if (ret == PAM_SUCCESS) {
                printf("Authenticated\n");
        } else {
                printf("Not Authenticated\n");
        }

        if (pam_end(pamh, ret) != PAM_SUCCESS) {
                pamh = NULL;
                printf("check_user: failed to release authenticator\n");
                exit(1);
        }

        return ret == PAM_SUCCESS ? 0 : 1;

}
```

Для успешного запуска теста добавим две строчки в файл /etc/pam.d/common-auth, как показано на вышеупомянутом сайте.

```
auth sufficient mypam.so
account sufficient mypam.so
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
```

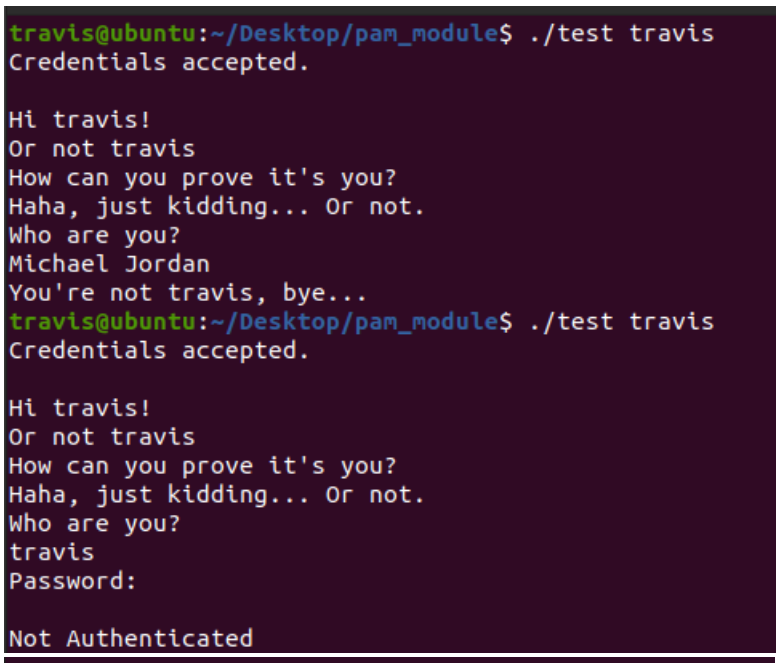
Скомпилируем модуль и залинкуем объектный файл с PAM:

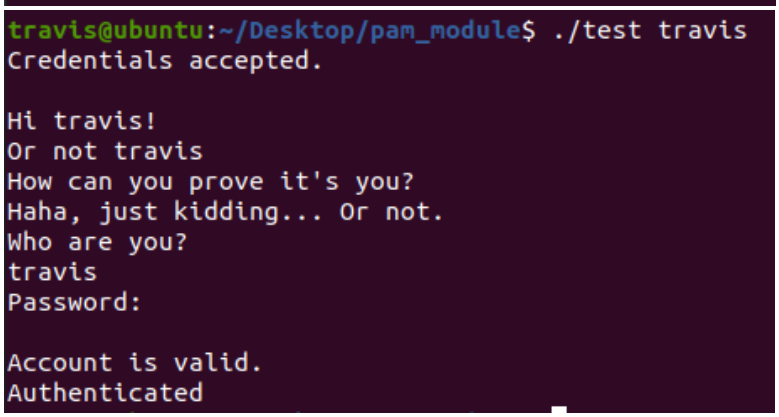
```
travis@ubuntu:~/Desktop/pam_module$ gcc -fPIC -fno-stack-protector -c mypam.c
travis@ubuntu:~/Desktop/pam_module$ sudo ld -x --shared -o /lib/x86_64-linux-gnu/security/mypam.so mypam.o
```

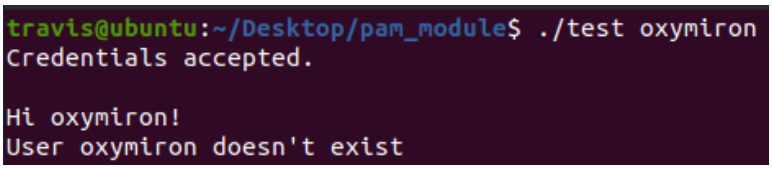
Далее скомпилируем тест:

```
travis@ubuntu:~/Desktop/pam_module$ gcc -o test test.c -lpam -lpam_misc
```

Наконец, протестируем программу:







В первом запуске попытка аутентификации была провалена, во втором - был некорректно введен пароль, в третьей попытке данные были введены правильно, и в последнем запуске был получен отказ вследствие ввода имени несуществующего пользователя.