Анализ бинарников - Lab3

Матвей Русаков m.rusakov@innopolis.university SD-03

Май 2025

1 First Task

1.1 Описание программы

Программа на C++ принимает аргумент командной строки и сравнивает его с «паролем», который генерируется на лету функцией `plouf`. Если введённый пароль совпадает, выводится сообщение об успехе, иначе — ошибка.

1.1.1 Main функция

Ниже приведён дизассемблированный вид функции `main`:

```
undefined4 main(int param_1, undefined4 *param_2) {
    char *pcVar1;
    bool bVar2;
    ostream *poVar3;
    undefined4 uVar4;
    allocator local_1e;
    allocator local_1d;
    string local_1c[4];
    string local_18[4];
    string local_14[4];
    undefined4 *local_10;

    local_10 = &param_1;
    if (param_1 < 2) {
        pcVar1 = (char *)*param_2;
        poVar3 = std::operator<<((ostream *)std::cerr, "usage : ");
        poVar3 = std::operator<<(poVar3, pcVar1);
        poVar3 = std::operator<<(poVar3, " password");
        std::ostream::operator<<(poVar3, std::endl<>);
        uVar4 = 5;
    }
    else {
        std::allocator<char>::allocator();
        std::string::string(local_14, &DAT_08048dc4, &local_1d);
        std::allocator<char>::allocator();
```

```
            std::string::string(local_18, &DAT_08048dcc, &local_1e);
            plouf(local_1c, local_18, local_14);
            std::string::~string(local_18);
            std::allocator<char>::~allocator((allocator<char> *)&local_1e);
            std::string::~string(local_14);
            std::allocator<char>::~allocator((allocator<char> *)&local_1d);
            bVar2 = std::operator==(local_1c, (char *)param_2[1]);
            if (bVar2) {
                poVar3 = std::operator<<((ostream *)std::cout, "Bravo, tu peux valider...");
                std::ostream::operator<<(poVar3, std::endl<>);
            }
            else {
                poVar3 = std::operator<<((ostream *)std::cout, "Password incorrect.");
                std::ostream::operator<<(poVar3, std::endl<>);
            }
            uVar4 = 0;
            std::string::~string(local_1c);
        }
        return uVar4;
    }
```

1.1.2 Функция `plouf`

Функция `plouf` принимает три аргумента (результирующую строку, ключ и зашифрованную строку) и применяет поэлементное XOR-сложение между байтами ключа и данных.

```
string *plouf(string *param_1, uint param_2, uint param_3) {
    byte bVar1;
    byte *pbVar2;
    char *pcVar3;
    allocator local_21;
    int local_20;

    std::allocator<char>::allocator();
    std::string::string(param_1, "", &local_21);
    std::allocator<char>::~allocator((allocator<char> *)&local_21);
    local_20 = 0;
    while (true) {
        pcVar3 = (char *)std::string::operator[](param_2);
        if (*pcVar3 == '\0') break;
        pbVar2 = (byte *)std::string::operator[](param_2);
        bVar1 = *pbVar2;
        std::string::length();
        pbVar2 = (byte *)std::string::operator[](param_3);
        std::string::operator+=(param_1, *pbVar2 ^ bVar1);
        local_20 = local_20 + 1;
    }
```

```
    return param_1;
}
```

1.2 Данные

В памяти были обнаружены два массива байтов:

- Ключ (`DAT_08048dc4`):

  $$18\ D6\ 15\ CA\ FA\ 77\ 00$$

- Зашифрованные данные (`DAT_08048dcc`):

  50 B3 67 AF A5 0E и так далее (будет в питон скрипте)

Важно: последний байт ключа (`00`) — это нулевой символ конца строки, используемый C++ для терминации строки, и **не участвует в шифровании**.

1.3 Процесс дешифрования

Алгоритм дешифрования прост: каждый байт зашифрованной строки XOR-ится с очередным байтом ключа, ключ повторяется циклически каждые 6 байт.

Python-скрипт для расшифровки:

```
key = [0x18, 0xD6, 0x15, 0xCA, 0xFA, 0x77]  # 6 байт, без \0
encrypted = [
    0x50,0xB3,0x67,0xAF,0xA5,0x0E,0x77,0xA3,0x4A,0xA2,
    0x9B,0x01,0x7D,0x89,0x61,0xA5,0xA5,0x02,0x76,0xB2,
    0x70,0xB8,0x89,0x03,0x79,0xB8,0x71,0x95,0x9B,0x28,
    0x74,0xBF,0x61,0xBE,0x96,0x12,0x47,0x95,0x3E,0xE1,
    0xA5,0x04,0x6C,0xA3,0x73,0xAC,0x89,0x00
]

decrypted = ""
for i in range(len(encrypted)):
    decrypted += chr(encrypted[i] ^ key[i % len(key)])

print(decrypted)
```

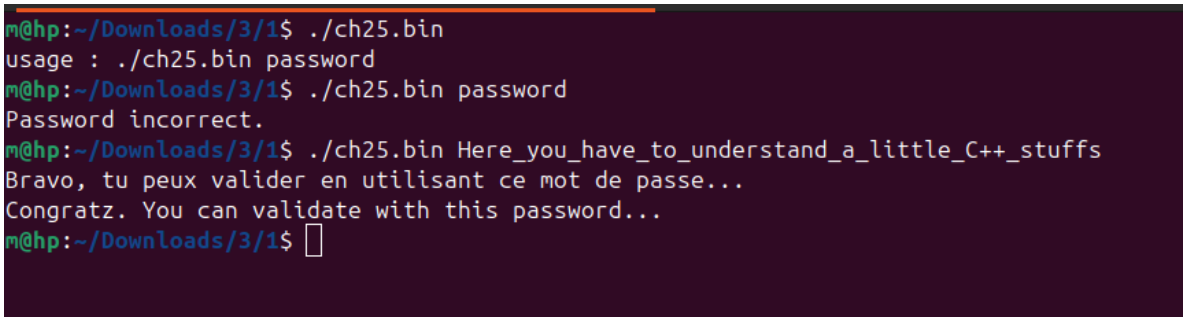
1.4 Результат

Выход скрипта:

Here_you_have_to_understand_a_little_C++_stuffs

Что и является паролем, на мой взгляд

1.5 Тестовые запуски



2 Second Task

2.1 Main функция

```
undefined4 main(void)

{
  char *pcVar1;
  int iVar2;
  undefined4 local_10;

  puts("########################################## ################");
  puts("##        Bienvennue dans ce challenge de cracking        ##" );
  puts("########################################## ################\n");
  printf("username: ");
  pcVar1 = (char *)getString(local_10);
  iVar2 = strcmp(pcVar1,"john");
  if (iVar2 == 0) {
     printf("password: ");
     pcVar1 = (char *)getString(pcVar1);
     iVar2 = strcmp(pcVar1,"the ripper");
     if (iVar2 == 0) {
        printf("Bien joue, vous pouvez valider l'epreuve avec le mot d e passe : %s !\n","987
     }
     else {
        puts("Bad password");
     }
  }
  else {
     puts("Bad username");
  }
  return 0;
}
```

2.2 Описание

Программа реализует простую аутентификационную логику. Пользователю предлагается ввести имя пользователя и пароль. Проверка осуществляется путём прямого сравнения введённых строк с жёстко заданными значениями.

- Если имя пользователя совпадает с `"john"`, то далее проверяется пароль.
- Если пароль совпадает с `"the ripper"`, то выводится сообщение об успехе и секретный ключ.
- В противном случае программа сообщает об ошибке (`"Bad username"` или `"Bad password"`).

Ключ (флаг)

Если логин и пароль указаны верно, программа выводит:

```
Bien joue, vous pouvez valider l'epreuve avec le mot de passe : 987654321 !
```

2.3 Особенности

- Логика крайне простая — всё полностью зашито в бинарник.
- Пароли не шифруются, не хэшируются и не скрыты — можно легко найти через `strings` или дизассемблирование.
- Юзернейм - "john", пароль - "the ripper"
- Ключ - 987654321

2.4 Тестовые запуски

```
m@hp:~/Downloads/3/2$ ./ch2.bin
############################################################
##        Bienvennue dans ce challenge de cracking        ##
############################################################

username: m
Bad username
m@hp:~/Downloads/3/2$ ./ch2.bin
############################################################
##        Bienvennue dans ce challenge de cracking        ##
############################################################

username: john
password: the ripper
Bien joue, vous pouvez valider l'epreuve avec le mot de passe : 987654321 !
m@hp:~/Downloads/3/2$
```

References

[1] GitHub Link: https://github.com/MattWay224/reverse-engineering-course В этом репозитории можно найти все лабы и информацию про каждое задание в каждой лабе