Фаховий коледж ракетно-космічного машинобудування
Дніпровського національного університету імені Олеся Гончара

ЗВІТ
з лабораторних робіт
з дисципліни «Основи кібербезпеки»

Спеціальність 123 Комп’ютерна інженерія
Група КС-21-1

Виконав Антіпов К.І.
Перевірила Панферова Я.В.

2024-2025

ЗМІСТ

ЛАБОРАТОРНА РОБОТА № 3

Тема: Реверс-інженеринг програми “level 1”. Аналіз логіки роботи програми, розробка кількох рішень обходу пароля та створення генератора ключів.

Мета: Дослідити процес реверс-інжинірингу crackme-програми “level1”, зрозуміти логіку її роботи, запропонувати кілька методів вирішення завдання, а також створити keygen для генерації коректного паролю.

Хід роботи

1. ПОСТАНОВКА ЗАВДАННЯ

Провести реверс-інжиніринг crackme-програми рівня “level1”. Завдання передбачає:

– Детальний аналіз логіки програми – вивчення коду для визначення алгоритму перевірки серійного номера.

– Розробку кількох рішень – запропонувати методи, що дозволяють отримати коректний пароль

– Розробку keygen – створити програму, яка генерує серійний номер відповідно до алгоритму crackme.

2. Декомпіляція програми за допомогою Ghidra

Для декомпіляціх програми використовується Ghidra — інструмент для реверс-інжинірингу.

Після відкриття виконуваного файлу в Ghidra ми отримуємо список реалізованих функцій у програмі.

Рисунок 1.1 – Список реалізованих функцій у програмі

Серед них нас найбільше цікавлять дві функції:

– main - головна функція програми, з якої починається виконання. Її аналіз дозволяє зрозуміти загальну структуру та логіку роботи програми, а також виявити, які інші функції викликаються в процесі.

– checkPass - функція, що відповідає за перевірку введеного користувачем пароля. Саме тут зазвичай реалізовано алгоритм перевірки коректності серійного номера чи ключа.

Лістинг 1.1 – Функція main

```
undefined8 main(void)
     {
       int iVar1; // // змінна типу bool, яка зберігає результат
аутентифікації
     undefined local_48 [64]; // char[64] змінна для зберігання
введеного паролю користувача для подальшої перевірки (максимальний
розмір 64).

       printf("Welcome to Easy Crack Me");
       printf("What is the Secret ?");

       // Введення паролю користувача через scanf та запис його у
змінну local_48
       __isoc99_scanf(&DAT_00102032,local_48);

       // Перевірка введеного пароля за допомогою функції checkPass
iVar1 = checkPass(local_48);
       if (iVar1 == 0) {
         printf("Better luck next time. :(");
       } else {
         printf("You are correct :)");
       }

       return 0;
     }
```

Лістинг 1.2 – Функція checkPass

```
char checkPass(char *param_1)

{
  char cVar1;

  if (*param_1 == 's') { // перевірити перший символ у паролі
    cVar1 = param_1[1]; // якщо успішно, то записуємо
наступний символ у змінну для подальших порівнянь
    if ((((cVar1 == 'u') && (cVar1 = param_1[2], cVar1 == 'd'))
&&
        (cVar1 = param_1[3], cVar1 == 'o')) &&
       (((cVar1 = param_1[4], cVar1 == '0' && (cVar1 =
param_1[5], cVar1 == 'x')) &&
        ((cVar1 = param_1[6], cVar1 == '1' && (cVar1 =
param_1[7], cVar1 == '8')))))) {
      cVar1 = '\x01';
    }
  } else {
    cVar1 = '\0';
  }
  return cVar1;
}
```

Наступним кроком є анадіз цих функцій для виявлення механізму перевірки та визначення слабких місць програми, які можна використати для обходу захисту.

3. Аналіз функцій main та checkPass

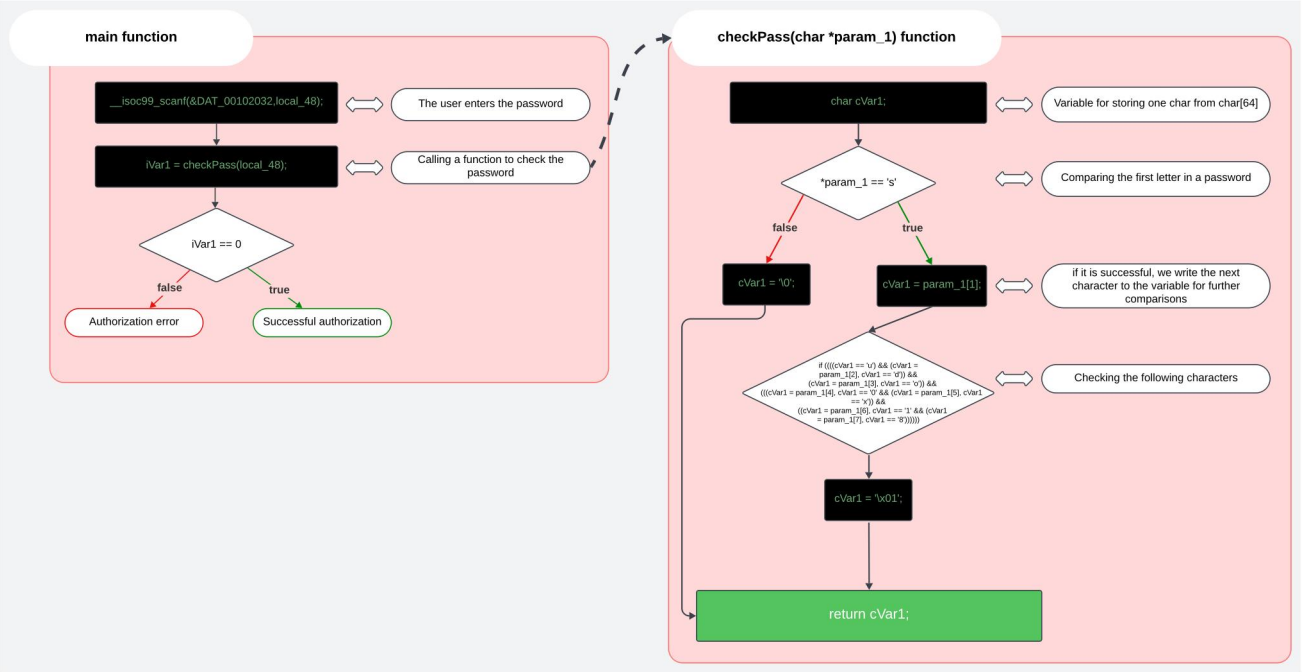

Рисунок 1.2 – Блоксхема для функцій main та checkPass

Загальний алгоритм:

– Користувач вводить пароль.

– Програма передає цей пароль у функцію checkPass.

– checkPass виконує побуквену перевірку, порівнюючи пароль із закладеними критеріями.

– У разі успіху програма повідомляє про успішну авторизацію, в іншому випадку — про помилку.

4. Логічна уразливість у функції checkPass

У коді функції checkPass міститься уразливість, яка дозволяє успішно виконати перевірку паролю з частково коректним паролем.
Дана уразливість виникає через логічну помилку та некоректний контроль значень, що повертаються.

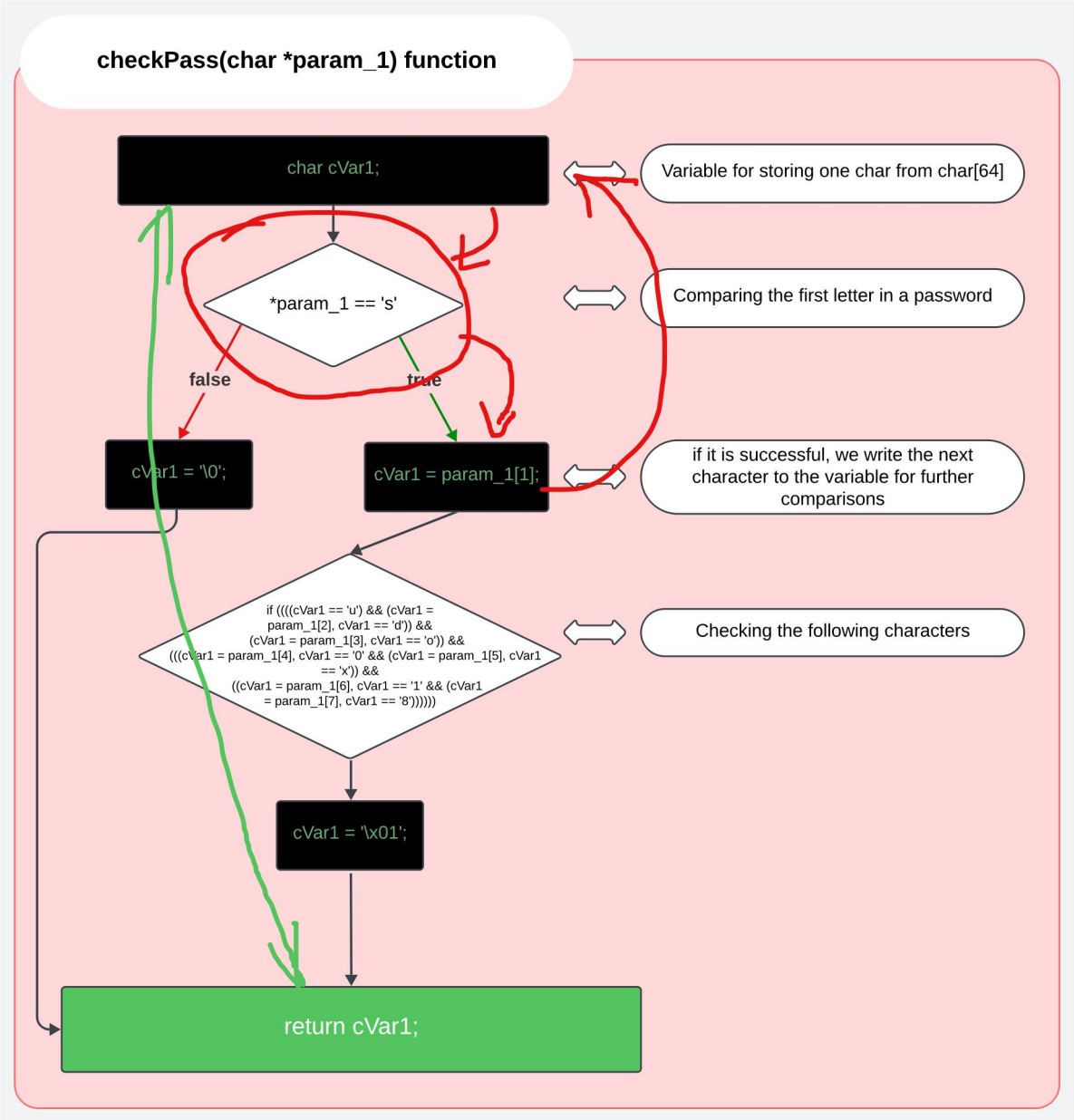

Рисунок 1.3 – Блоксхема для візуального показу вразливості

Повернення '\x01' відбувається не тоді, коли перевіряється весь рядок, а в результаті успішного виконання однієї з вкладених умов. Таким чином, навіть рядок, що починається з допустимого символу, може пройти перевірку, якщо решта функції виконується без суворої перевірки.

5. Розробка скрипта keygen для генерації паролів

[Посилання на GitHub репозиторій]

Лістинг 1.3 – Bash скрипт для генерації паролів

```
#!/bin/bash
      available_chars=({a..z} {A..Z} {0..9})
      min_length=2
      password_count=5

      echo "Generated 5 passwords:"
      for (( i = 0; i <= password_count; i++ )); do
        max_length=$((RANDOM % 8 + 1));
        password="s"

        for (( j = 0; j < max_length; j++ )); do
          random_char="${available_chars[RANDOM                    %
${#available_chars[@]}]}"
          password+="$random_char"
        done

        echo $password
      done
```

6. Перевірка роботи keygen

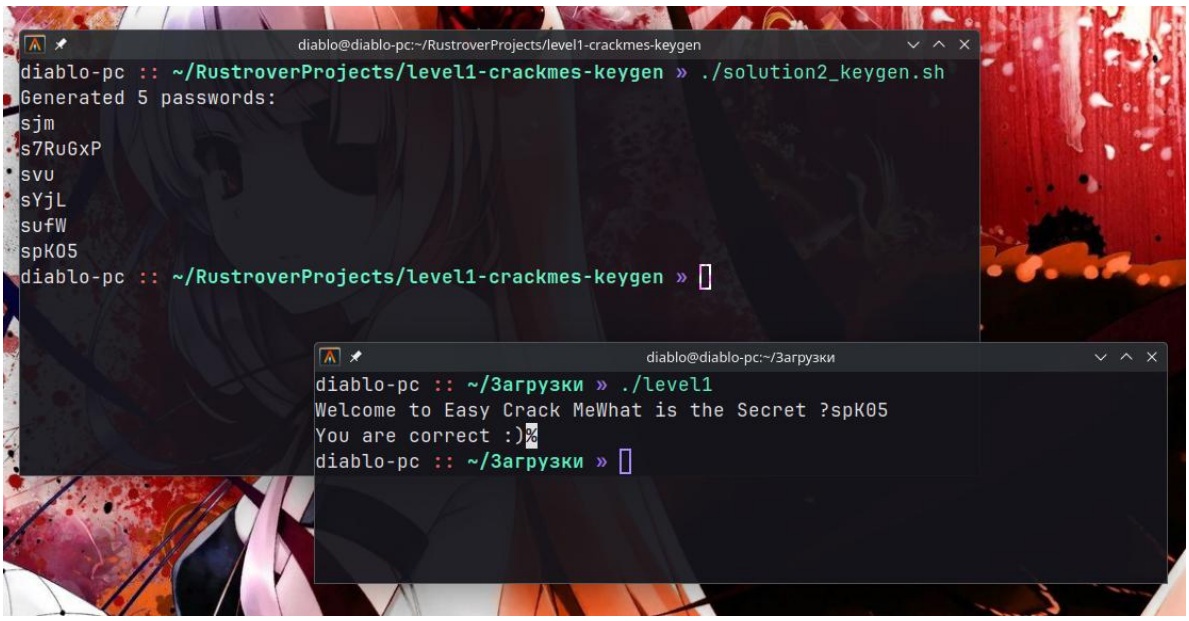

Рисунок 1.4 – Генерація валідних паролей за допомогою скрипта keygen

7. Як можна усунути цю вразливість

– Треба переконатися, що весь пароль перевіряється до кінця, і тільки тоді функція повинна повертати успішний результат.

– Також можна спростити код і зробити його більш читабельним

Лістинг 1.4 – Моє рішення щодо усунення цієї вразливості

```
char checkPass(char *param_1) {
  const char *correct_pass = "sudo0x18";

  for (int i = 0; i < 8; i++) {
    if (param_1[i] != correct_pass[i]) {
      return '\0';
    }
  }

  if (param_1[8] != '\0') {
    return '\0';
  }

  return '\x01';
}
```

Висновок: У процесі роботи було виконано детальний аналіз програми, виявлено основні алгоритми перевірки паролю, реалізован підхід для отримання валідного ключа та створено keygen.

Це дало змогу зрозуміти принципи роботи програм із захистом від несанкціонованого доступу, покращити навички реверс-інжинірингу та вміння роботи з мовою програмування C++.