Анализ бинарного файла - Lab9

Matthew Rusakov m.rusakov@innopolis.university SD-03

May 2025

Предисловие

Я изучил бинарник с помощью ghidra и нейросетей. Бинарник реализует проверку серийного номера (serial) на основе имени пользователя. Программа принимает два аргумента командной строки: имя пользователя и серийный номер в формате `XXXX-YYYY-ZZZZ`, где каждая часть интерпретируется как 32-битное шестнадцатеричное число.

1 Первая функция: `FUN_00101100`

Основная функция (`main`) выполняет следующий алгоритм:

- Проверяет количество аргументов командной строки:
  - Если `argc == 1`, выводит строку помощи.
  - Если `argc == 3`, проверяет длину строки серийного номера: должна быть ровно 0x1c (28) символов.
- При корректной длине парсит серийный номер формата `%x-%x-%x` в три числа.
- Вызывает функцию `FUN_00101470` для проверки соответствия.

2 Функция проверки: `FUN_00101470`

Функция проверяет три значения серийного номера на соответствие значениям, полученным из имени пользователя:

1. Вызывает `FUN_00101380`, которая генерирует MD5-хеш из строки (имени пользователя) с дополнительными преобразованиями.
2. Вызывает `FUN_001012b0` дважды, чтобы получить два результата по данным серийного номера и статическим таблицам.
3. Проверяет, совпадают ли соответствующие значения:
   - `param_2[0] == local_18`
   - `param_2[1] == local_14`
   - `param_2[2] == local_10`

3 Генерация MD5: FUN_00101380

Данная функция формирует MD5-хеш следующим образом:

1. Проверяет, что длина имени пользователя не превышает 0x80 символов.
2. Копирует имя в буфер `acStack_1a9`, затем реверсирует строку.
3. Конкатенирует реверсированную строку с оригинальной и сохраняет в `local_128`.
4. Выполняет `MD5(local_128, 256)`.

Итог: результат хеша используется как база для сравнения с серийным номером.

4 Обработка серийного номера: FUN_001012b0

Функция преобразует части серийного номера, используя следующие шаги:

- Обрабатывает входное значение (серийный номер) и данные таблиц (возможно, секретные, находящиеся по адресу `DAT_...`).
- На каждой итерации вызывает `FUN_00101290`, которая считает количество единичных битов в числе.
- Используется схема **битового xor и сдвига**, результатом чего являются два итоговых значения, записываемые в `param_1[0]` и `param_1[1]`.

5 Подсчет битов: FUN_00101290

Обычная функция подсчета установленных битов (popcount) в 32-битном числе. Используется для повышения энтропии и смешивания данных.

6 Вывод

Вся логика программы направлена на проверку того, соответствует ли серийный номер определённому имени пользователя. Алгоритм включает:

- Хеширование имени с помощью модифицированного MD5.
- Преобразование серийного номера и сравнение с хешом.
- Скрытые данные в памяти программы (таблицы `DAT_...`) играют ключевую роль.

Итоговая схема

$$\texttt{username} \xrightarrow{\text{rev+concat}} \texttt{MD5} \rightarrow \texttt{target values}$$
$$\texttt{serial} \xrightarrow{\text{split}} \texttt{transformation} \rightarrow \texttt{candidate values}$$
$$\texttt{if match} \Rightarrow \texttt{OK, else} \Rightarrow \texttt{Try again}$$

Решение?

Я попробовал сделать генератор на питоне, который будет создавать value для ключа по такому же формату. Я вставлю их в dumps.log в ассемблер виде, а также сам скрипт и массивы байтов вы можете найти на github

```
m@hp:~/PycharmProjects/reverse-engineering-course/Lab9$ python3 lab_data/generator.py
Сгенерированный серийный номер для 'admin': 1A50-51E4-4BB4
Сгенерированный серийный номер для 'testuser': 6A9E-EE5C-84C2
Сгенерированный серийный номер для 'averylongusernameexample': B7D2-A894-1F46
```

НО я так и не понял, правильные это ключи или нет

```
m@hp:~/PycharmProjects/reverse-engineering-course/Lab9$ ./9 admin 1A50-51E4-4BB4
m@hp:~/PycharmProjects/reverse-engineering-course/Lab9$ ./9 matt serial
m@hp:~/PycharmProjects/reverse-engineering-course/Lab9$
```

Список литературы

[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