МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт искусственного интеллекта
Базовая кафедра №252 – информационной безопасности

ЛАБОРАТОРНАЯ РАБОТА №4 ПО ПРЕДМЕТУ «РАЗРУШАЮЩИЕ ПРОГРАММНЫЕ ВОЗДЕЙСТВИЯ»

Студент группы ККСО-01-20 *Семин В.В.*

Преподаватель *Старший преподаватель*
Трошков Вадим Евгеньевич

Москва 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Требуется провести статический анализ файла и отразить следующие результаты:

1) указать, какие строки нужно заменить, чтобы программа выдавала только положительный ответ;

2) разработать программу, которая на любое имя выдаёт правильный серийный номер.

1 ПЕРВИЧНЫЙ АНАЛИЗ ФАЙЛА

Рассмотрим PE заголовок файла с помощью PE View версии 0.9.9.0 (Рисунок 1.1). Он имеет магическое число MZ. Следовательно это исполняемый файл.

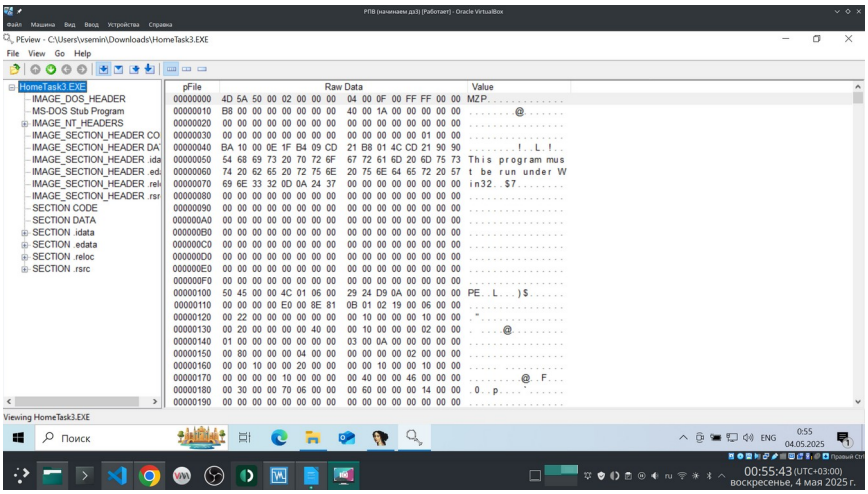

Рисунок 1.1 — PE заголовок файла.

Рассмотрим список сегментов программы (Рисунок 1.2) с помощью ПО IDA Pro 9.1.250226. Права доступа сегментов выставлены корректно. У сегмента кода выставлены права на чтение и исполнение. У сегмента данных выставлены права на чтение и запись.

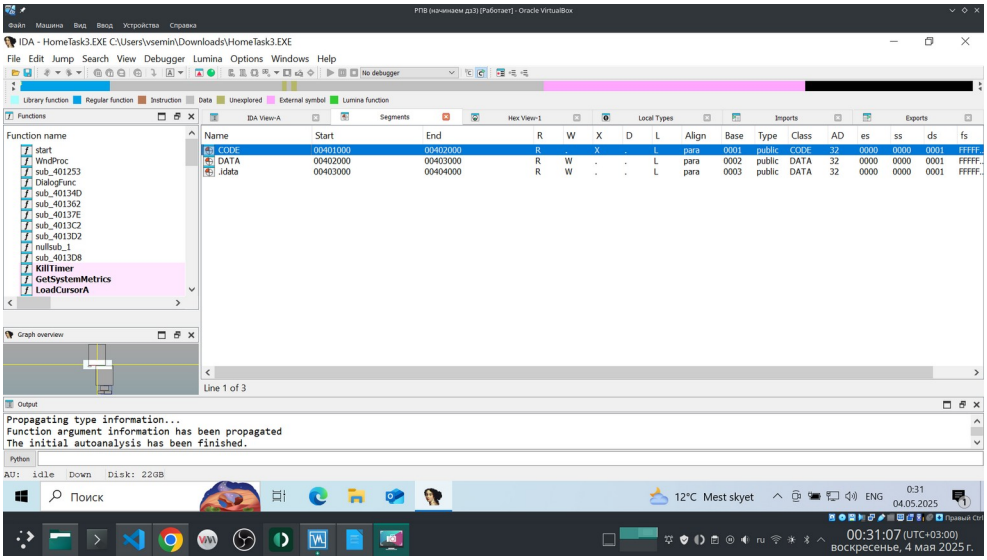

Рисунок 1.2 — Сегменты программы

Получим данные о файле из открытых источников. Для этого возьмём контрольную сумму файла с помощью 7-Zip 24.09 (Рисунок 1.3)

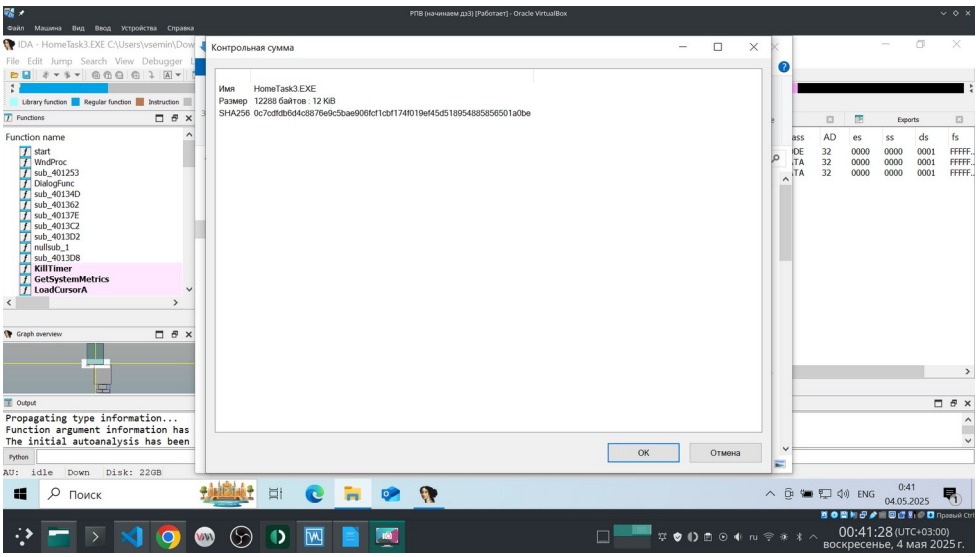

Рисунок 1.3 — Взятие контрольной суммы файла программы

Воспользуемся полученной контрольной суммой, чтобы получить информацию о программе на ресурсе virustotal (Рисунок 1.4). Данный файл детектируется как троянская программа Crackme.exe.

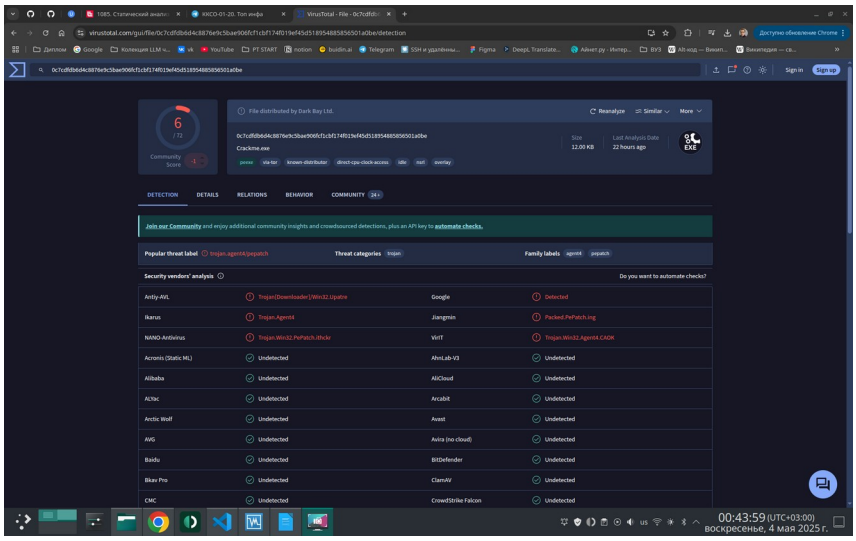

Рисунок 1.4 — Информация о программе из открытых источников

Рассмотрим зависимости программы с помощью PE Explorer версии 1.9 R6 (Рисунок 1.5). В качестве зависимостей файл имеет несколько системных библиотек Windows.

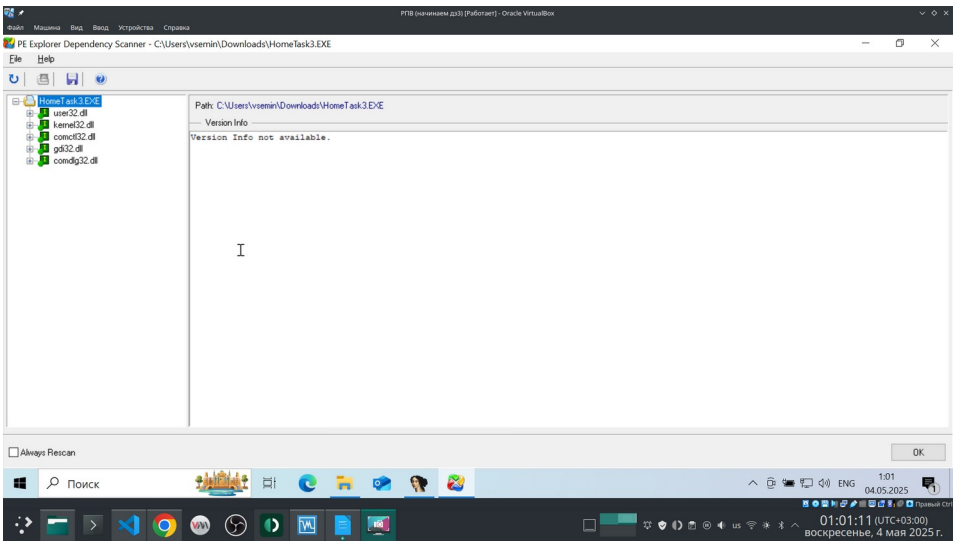

Рисунок 1.5 — Зависимости программы

2 ВОССТАНОВЛЕНИЕ АЛГОРИТМА

Для анализа программы используется ПО IDA Pro 9.1.250226.

Функция на Рисунке 2.1 выводит сообщение о успешной активации с помощью системного вызова *MessageBoxA*. Назовём её *showGoodWorkWindow*.

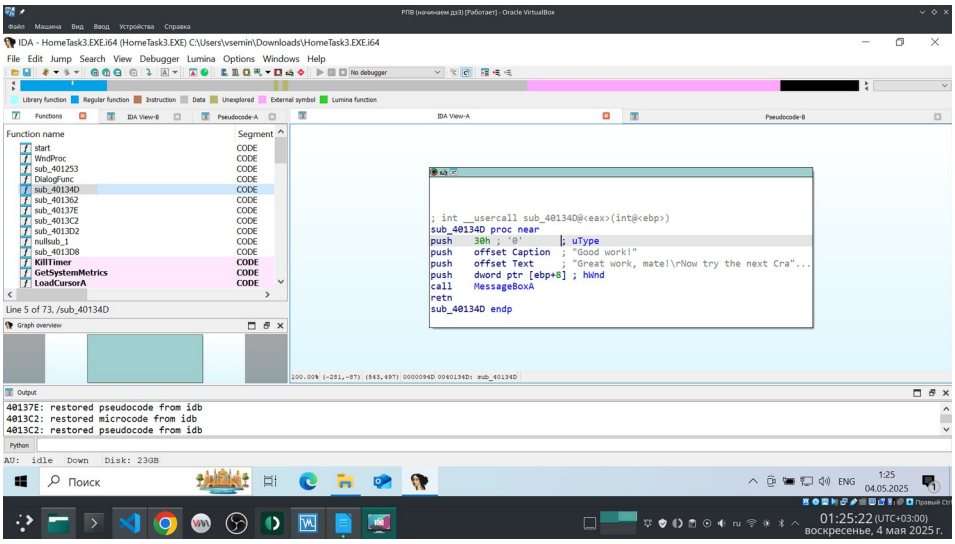

Рисунок 2.1 — Функция, оповещающая об успешной активации

Функция на Рисунке 2.2 делает системный вызов *MessageBeep(0)* - звук системной ошибки. Далее она выводит сообщение о неуспешной активации с помощью системного вызова *MessageBoxA*. Назовём её *showNoLuckWindow*.

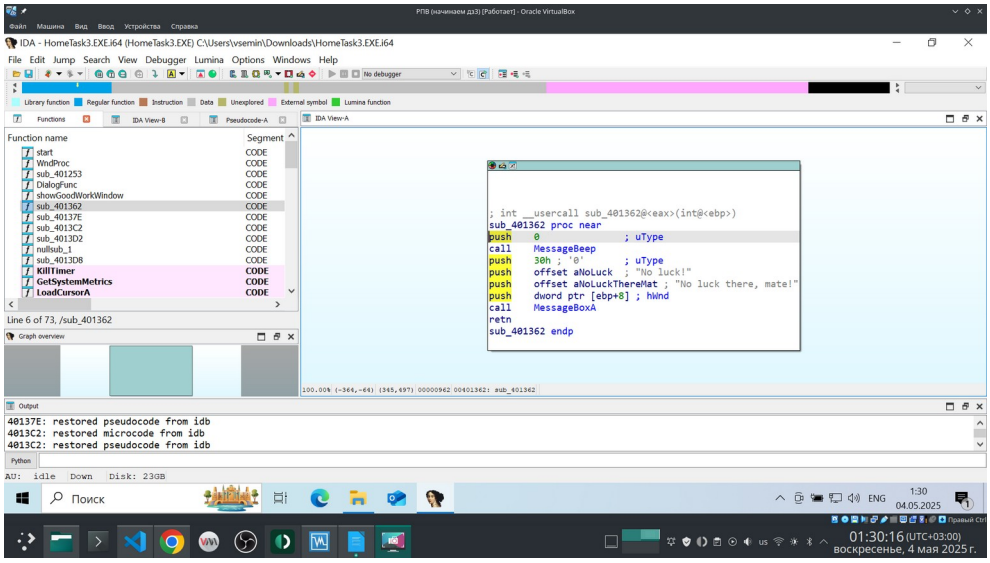

Рисунок 2.2 — Функция, оповещающая об неуспешной активации

Функция на Рисунке 2.3 вычисляет сумму байтов строки. Назовём её *strSum*.

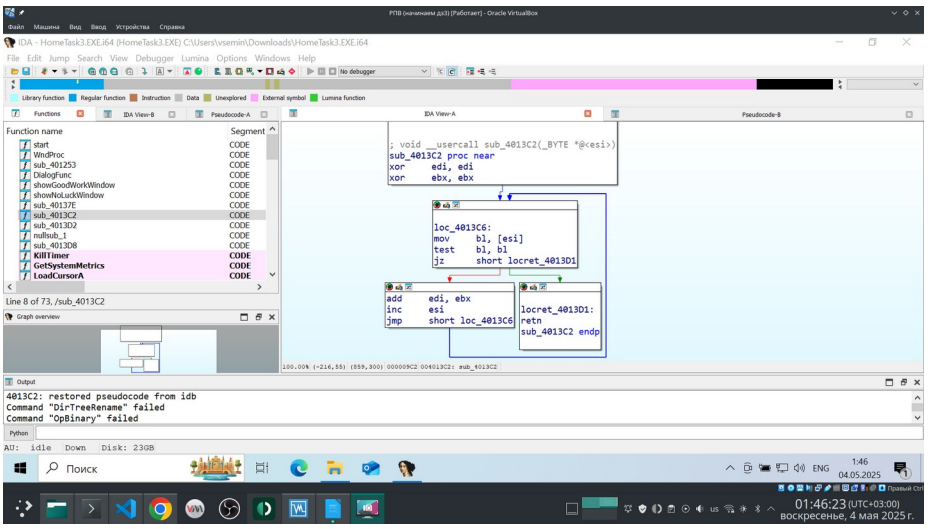

Рисунок 2.3 — Функция, вычисляющая сумму байтов строки

Функция на Рисунке 2.4 переводит символ в строке в верхний регистр. Назовём её *upperCharInString*.

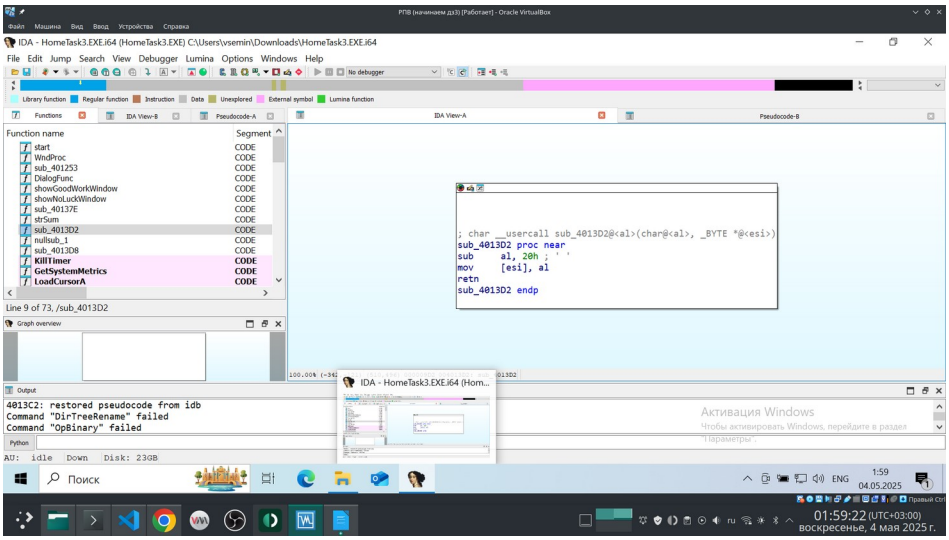

Рисунок 2.4 — Функция перевода символа в строке в верхний регистр.

Функция на Рисунке 2.5 и является той, которая высчитывает серийный номер по названию. Назовём её *getSerialNumber*.

Алгоритм её следующий. Она итерируется по строке. Если достигнут конец строки, то она вызывает функцию *strSum* и возвращает её результат, побитово сложенный по модулю два с значением 5678h.

Если в строке встречается символ, ASCII-код которого меньше чем у символа ’A’, то она выводит окно аналогично функции *showNoLuckWindow*.

При этом функция возвращает код возврата *MessageBoxA*. *MessageBoxA* возвращает код кнопки, нажатой пользователем в выведенном ей окне. В случае данной программы окно будет иметь только кнопку «OK», имеющую код 1. То есть возвращаемым значением в данной ветке исполнения функции *getSerialNumber* всегда будет 1.

Если в строке встречается символ, ASCII-код которого не меньше чем у символа ’Z’, то вызывается функция *upperCharInString*. Для самого символа ’Z’ вызов данной функции довольно странен. Возможно это ошибка программиста.

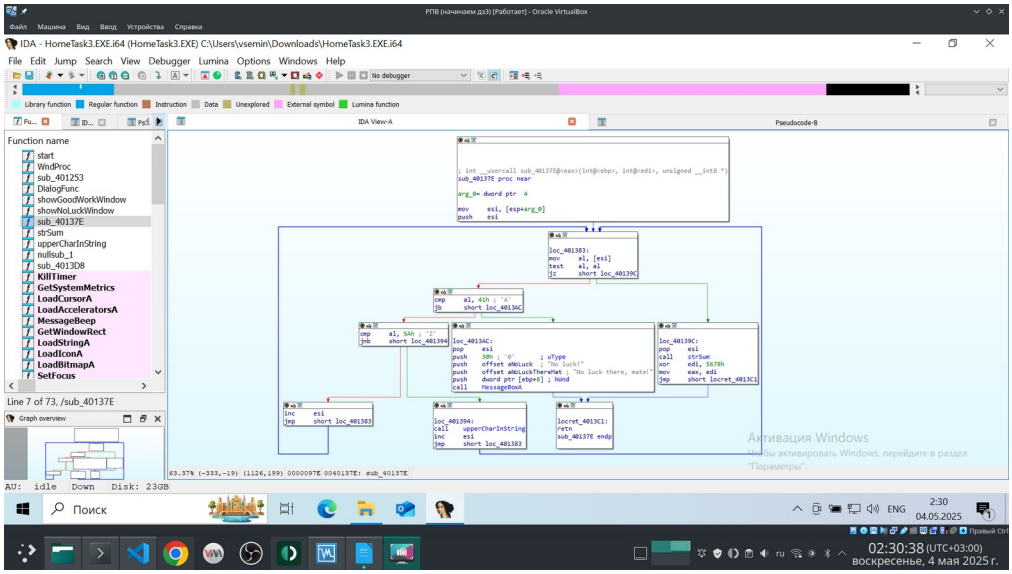

Рисунок 2.5 — Функция перевода символа в строке в верхний регистр

Функция на Рисунке 2.6 преобразует строку в число и осуществляет побитовое сложение по модулю 2 результата и константы 1234h. Назовём её *stoiGamma*.

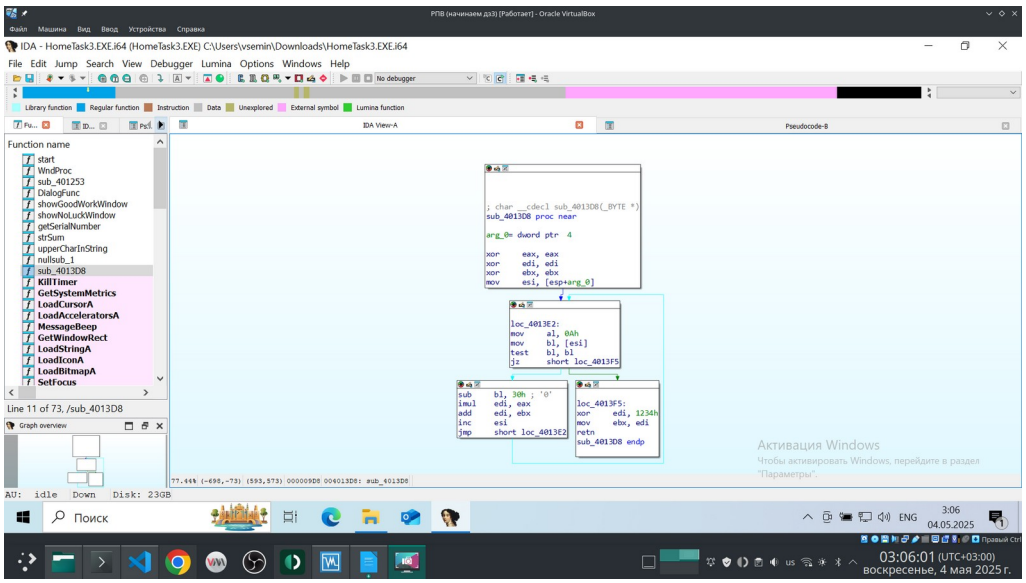

Рисунок 2.6 — Функция преобразования строки в число

В функции *WndProc* (Рисунок 2.7) видим, что программа выводит окно с положительным результатом в случае совпадения результатов функций *stoiGamma* и *getSerialNumber*. Иначе выведется окно с отрицательным результатом.

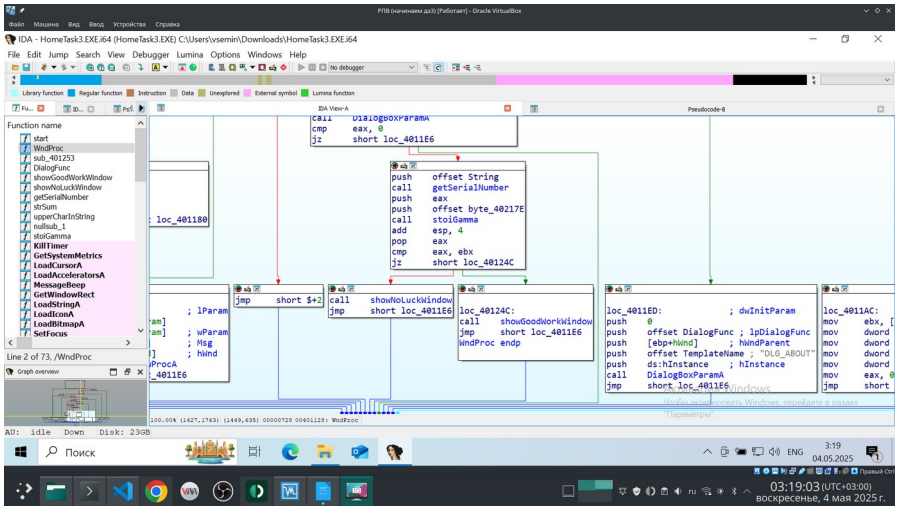

Рисунок 2.7 — Условие положительного результата

Интересно то, что функция *WndProc* не проверяет успешность генерации серийного номера в функции *getSerialNumber*. Соответственно, если ввести некорректное с точки зрения логики программы название (содержащее символ, ASCII-код которого меньше чем у символа ’A’), то выведется два окна с результатом. Одно из *getSerialNumber*, другое из *WndProc*.

Как было описано выше, результат *getSerialNumber* при невалидном названии всегда будет 1. Следовательно, если в качестве серийного номера для невалидного названия ввести $4661 = 0x1234 \oplus 1$, то выведется два окна: одно с отрицательным результатом, другое с положительным.

Функция на Рисунке 2.8 обрабатывает введённые в окне регистрации данные. Назовём её *parseRegisterData*.

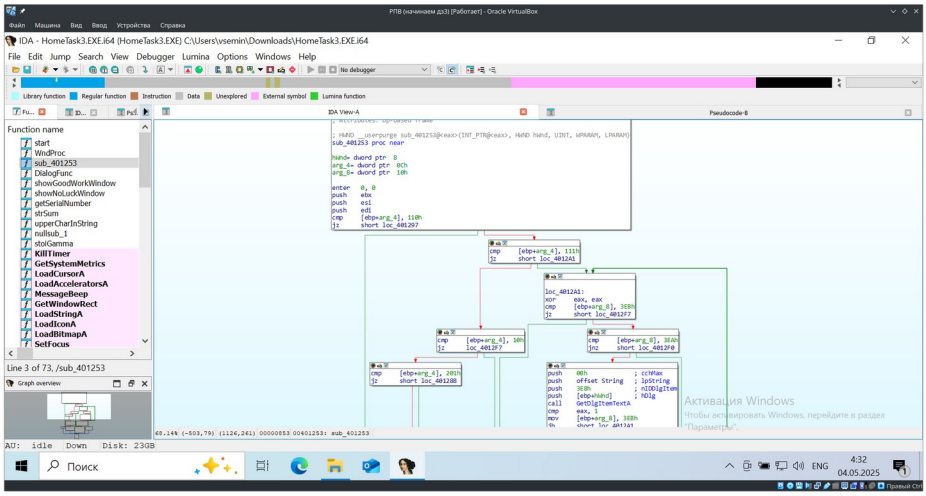

Рисунок 2.8 — Функция обработки данных, введённых в окне регистрации

Граф вызовов программы представлен на Рисунке 2.9.

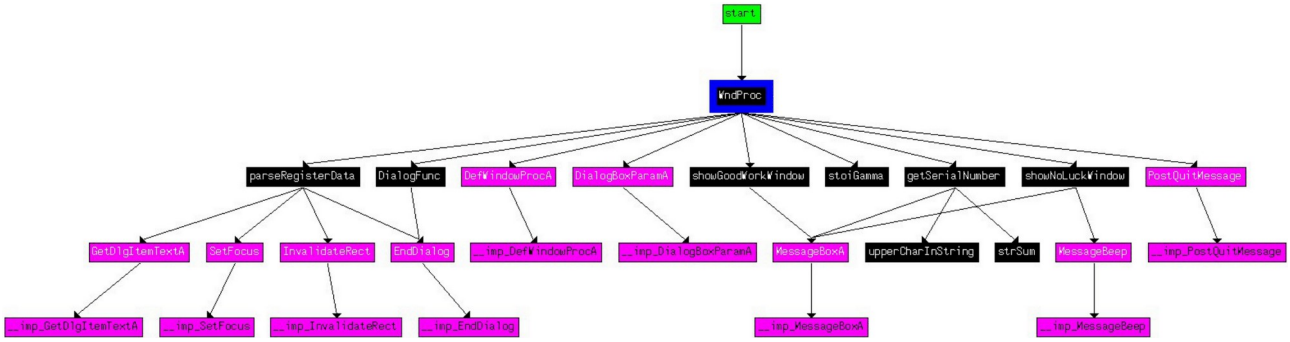

Рисунок 2.9 — Граф вызовов

3 ПРОГРАММА ДЛЯ ПОЛУЧЕНИЯ СЕРИЙНОГО НОМЕРА

Код программы для генерации серийного номера, разработанной на основе результатов анализа в главе 2, представлен в Листинге 3.1. Языком программирования, на котором написана программа, является C.

Листинг 3.1 — Программа для генерации серийного номера

```
#include <stdio.h>
int strSum(char *name) {
	int result;
	int i = 0;
	while (name[i]) {
		result += name[i];
		++i;
	}
	return result;
}
int main(int argc, char **argv) {
	if (argc < 2) {
		printf("Запуск: %s <имя>\n", argv[0]);
		return -1;
	}
	char *name = argv[1];
	int i = 0;
	while (name[i]) {
		if (name[i] < 'A') {
			printf("4661\n");
			return 0;
		}
		if (name[i] >= 'Z')
			name[i] -= 32;
		++i;
	}
	printf("%d\n", strSum(name) ^ 0x1234 ^ 0x5678);
	return 0;
}
```

4 МОДИФИКАЦИЯ ПРОГРАММЫ ДЛЯ ОБХОДА ПРОВЕРКИ

На Рисунке 4.1 можно увидеть, что открытие окна регистрации происходит при успешном сравнении поля *wParam* с значением 66h.

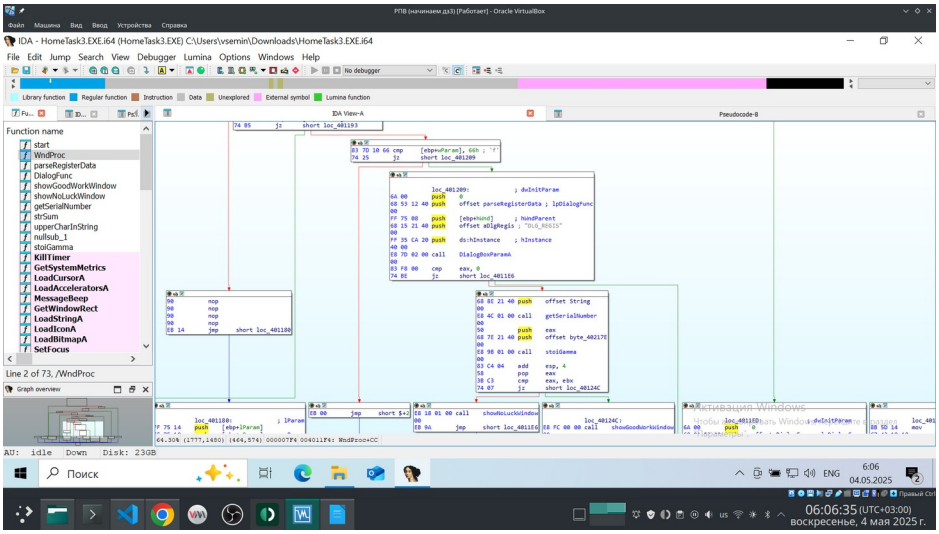

Рисунок 4.1 — Обработка нажатия на кнопку регистрации

Можно изменить параметр в данном условном переходе, чтобы перемещение происходило сразу на вызов *showGoodWorkWindow*. Для этого надо рассчитать разность между адресом, на котором находится вызов *showGoodWorkWindow*, и адресом следующей за условным переходом инструкции.

На Рисунке 4.2 видно, что адрес следующей за условным переходом инструкции равен 0x004011E4.

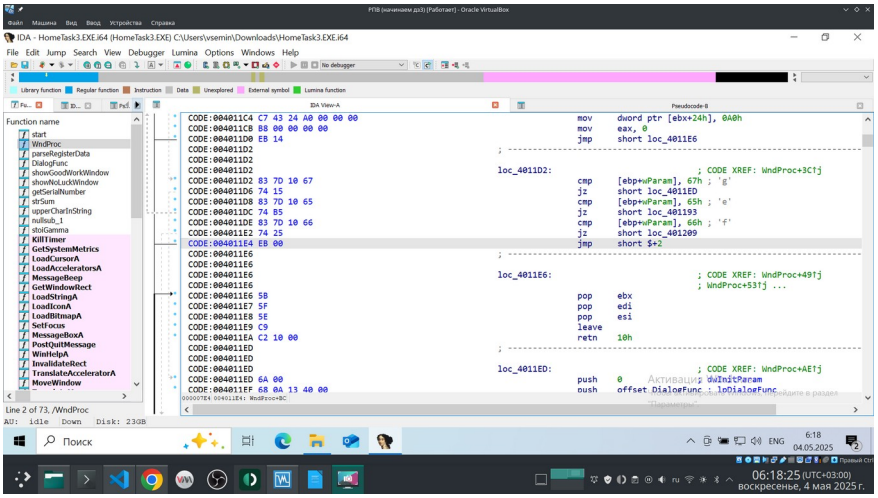

Рисунок 4.2 — Адрес следующей за условным переходом инструкции

На Рисунке 4.3 видно, что вызов *showGoodWorkWindow* расположен по адресу 0x0040124C.

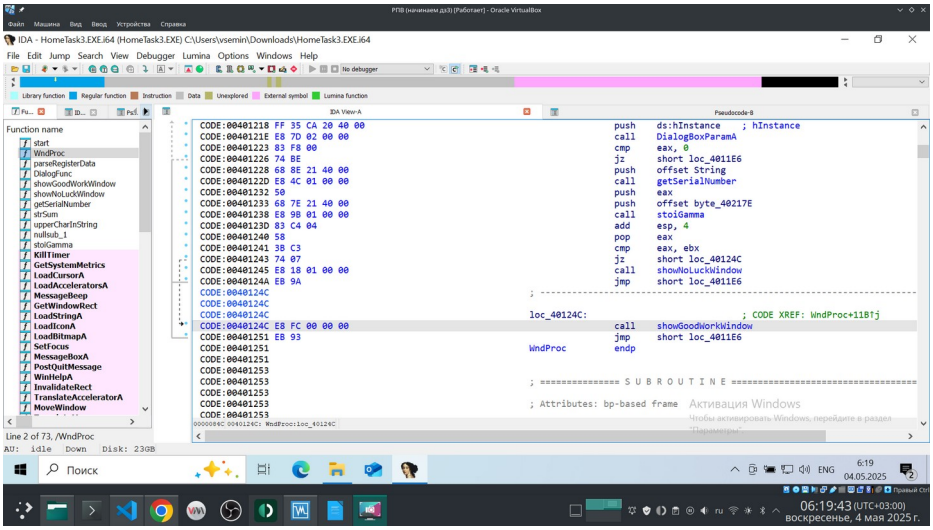

Рисунок 4.3 — Адрес вызова showGoodWorkWindow

Тогда параметр условного перехода нужно заменить с 0x25 на 0x68=0x0040124C−0x004011E4. Сделаем это с помошью SweetScape 010 Editor v15.0 (Рисунок 4.4).

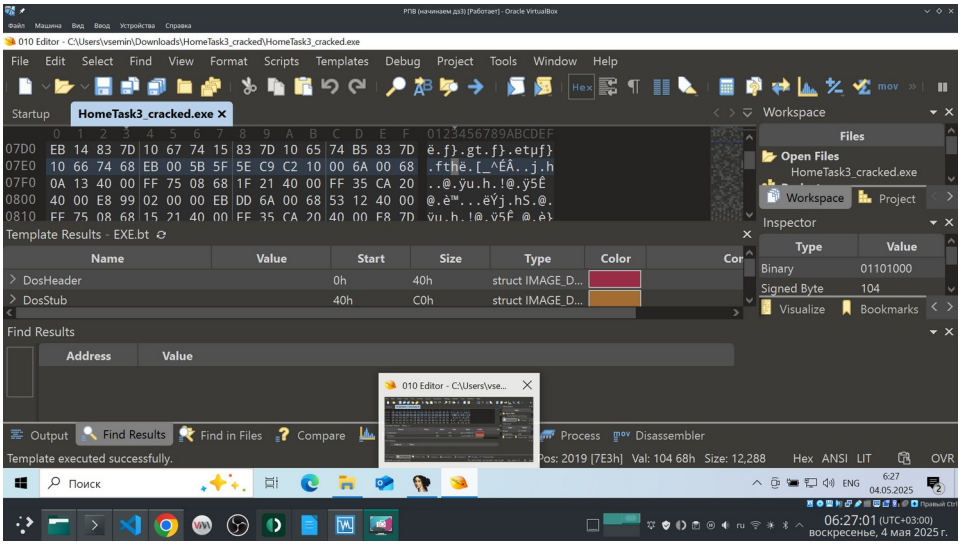

Рисунок 4.4 — Замена параметра условного перехода

Теперь по нажатии кнопки «Register» в приложении сразу выводится окно с сообщением об успешном результате.

ЗАКЛЮЧЕНИЕ

В рамках данной лабораторной работы проведён анализ исполняемого файла, потенциально несущего вредоносное воздействие.

Первым этапом являлся первичный анализ файла. Была вычислена его контрольная сумма, которая была использована для получения информации о файле из открытых источников. По полученной информации сделан предварительный вывод о том, что файл является троянской программой Crackme.exe.

Кроме этого были проверены PE заголовок файла и список его зависимостей. Полученные данные помогли определить тип файла — исполняемый 32-битный файл Windows.

Также были проверены сегменты данного исполняемого файла, но в них подозрительных привилегий обнаружено не было.

Вторым этапом являлась попытка восстановить алгоритм работы программы. Для этого программа была подвергнута анализу с использованием интерактивного дизассемблера IDA Pro. В результате данного анализа восстановлен алгоритм генерации серийного номера.

На основе результатов анализа разработана программа, способная генерировать серийный номер для любого имени пользователя. Кроме этого получена модифицированная версия исследуемой программы, которая на попытку регистрации сразу даёт положительный ответ.