1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. **Институт кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА № 6**

1. «Защита программного обеспечения от нелегального использования»
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. 4831001/20003 Черникова В.М.

1. Руководитель
2. асс. преподавателя Орел Е.М.
4. Санкт-Петербург

2023

# 1. Цель работы

Приобретение навыков по защите приложений от нелегального использования, по анализу исполняемых ходов в отсутствие исходных текстов и по применению способов защиты программ от дизассемблирования и отладки.

# 2. Формулировка задания

В данной работе необходимо:

1.Создать простую программу на языке С запрашивающую пароль.

2.Пользуясь исходным текстом программы и утилитой HIEW отыскать в исполняемом файле команду перехода при проверке правильности пароля и изменить ее таким образом, чтобы при вводе любого пароля программа принимала пользователя как авторизованного.

3.Получить и модифицировать код программы Nag.exe таким образом, чтобы сразу после появления всплывающего окна появлялась кнопка «Ок»

4.Распаковать исполняемый файл Guard.exe, модифицировать код таким образом, чтобы наличие отладчика не проверялось и при вводе любого пароля программа принимала пользователя как авторизованного.

5.Для разработанной в начале работы программы создать обфускатор на уровне исходных кодов.

# 3. Ход работы

1. В процессе выполнения лабораторной работы был написан простой код, воспринимающий пароль, который хранится в отдельном файле, и разрешающий доступ для пользователя. Для выполнения поставленной задачи просмотр и редактирование дисассемблер-кода использованы программы HIEW и Ida Pro. Для безусловного перехода применяется команда jmp (переход в любом случае), а условия обозначаются JE (перейти, если равно) и JZ (перейти, если нуль).

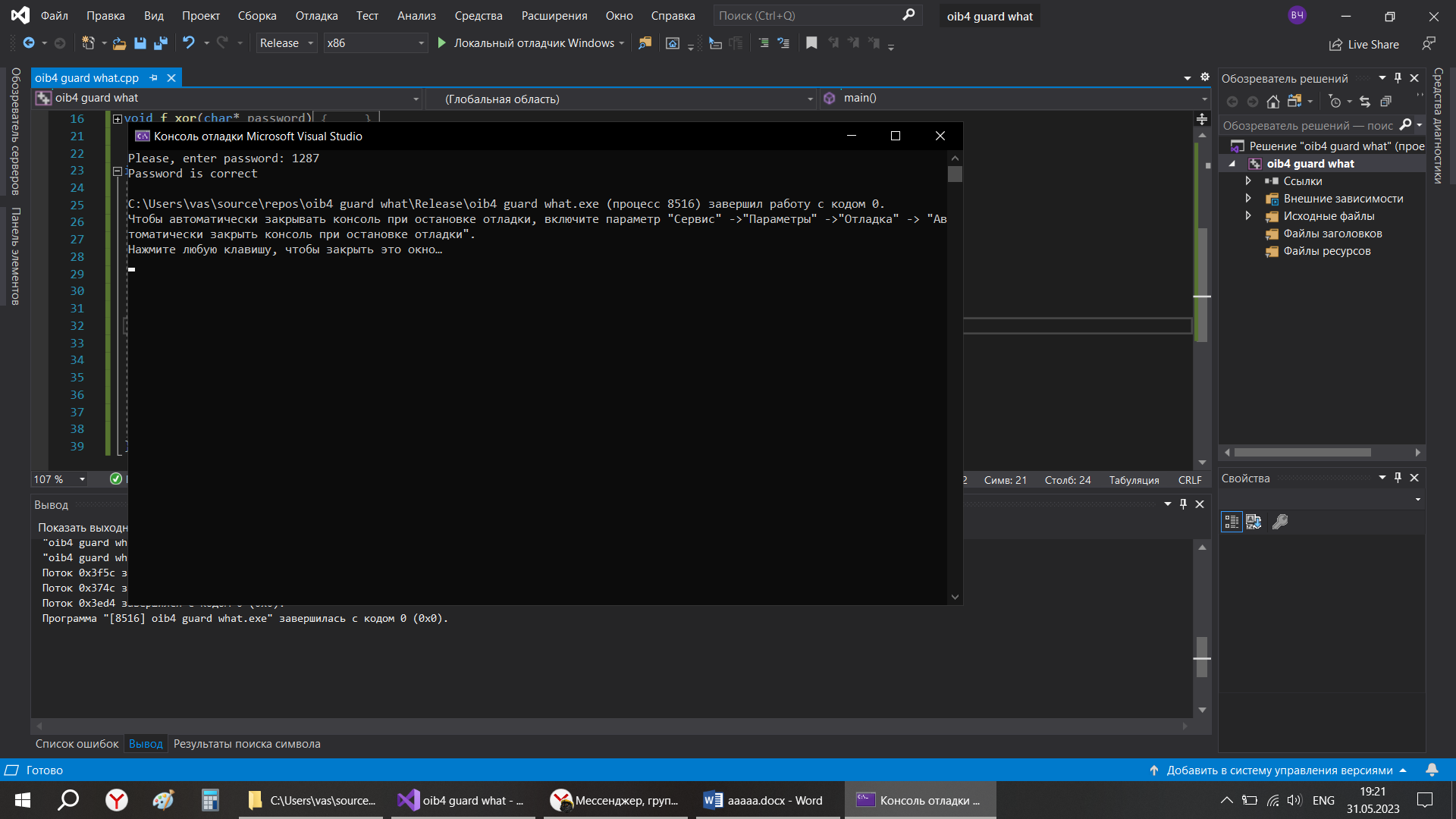
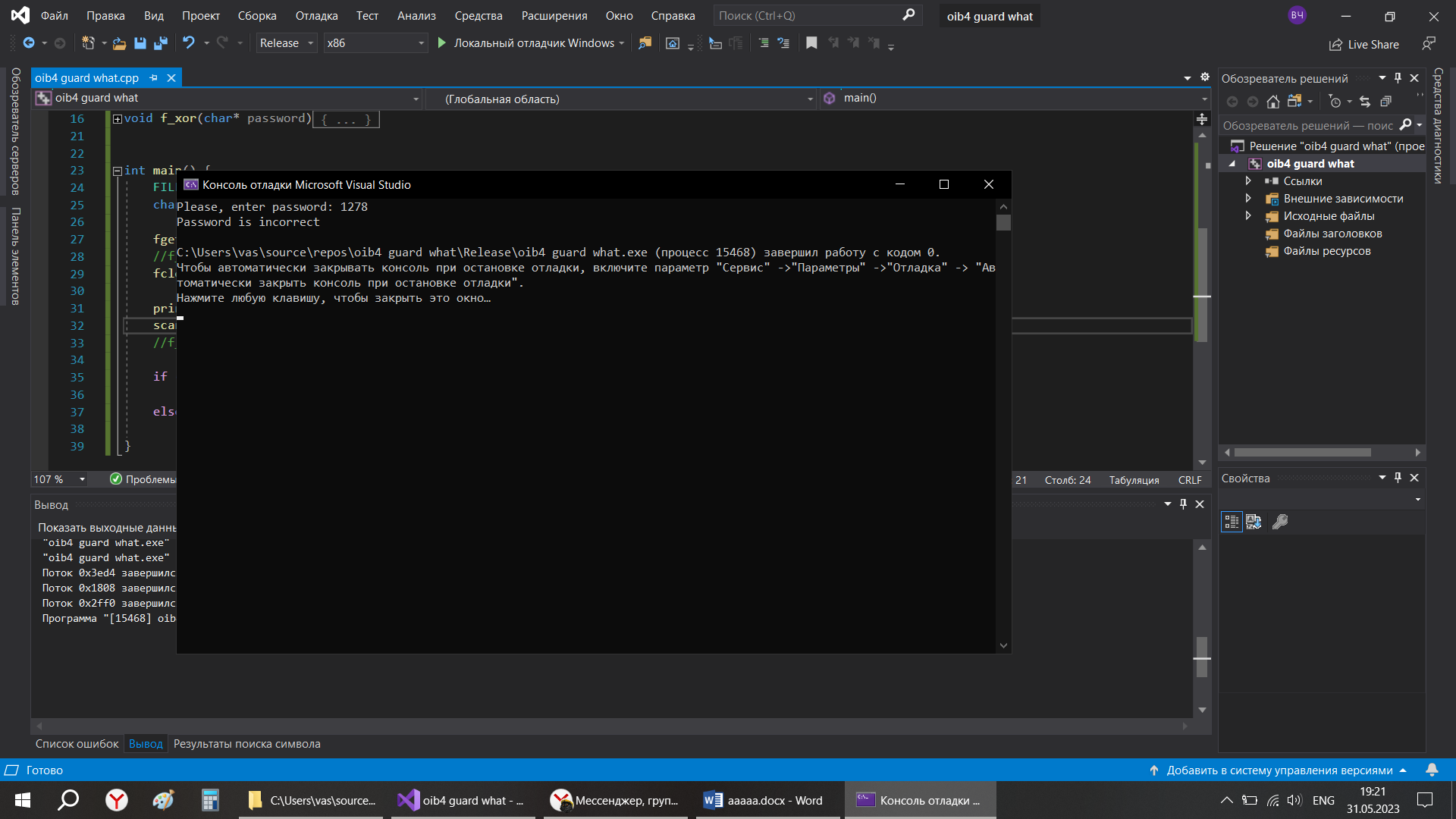


Рис. 1-2 — Правильная работа программы

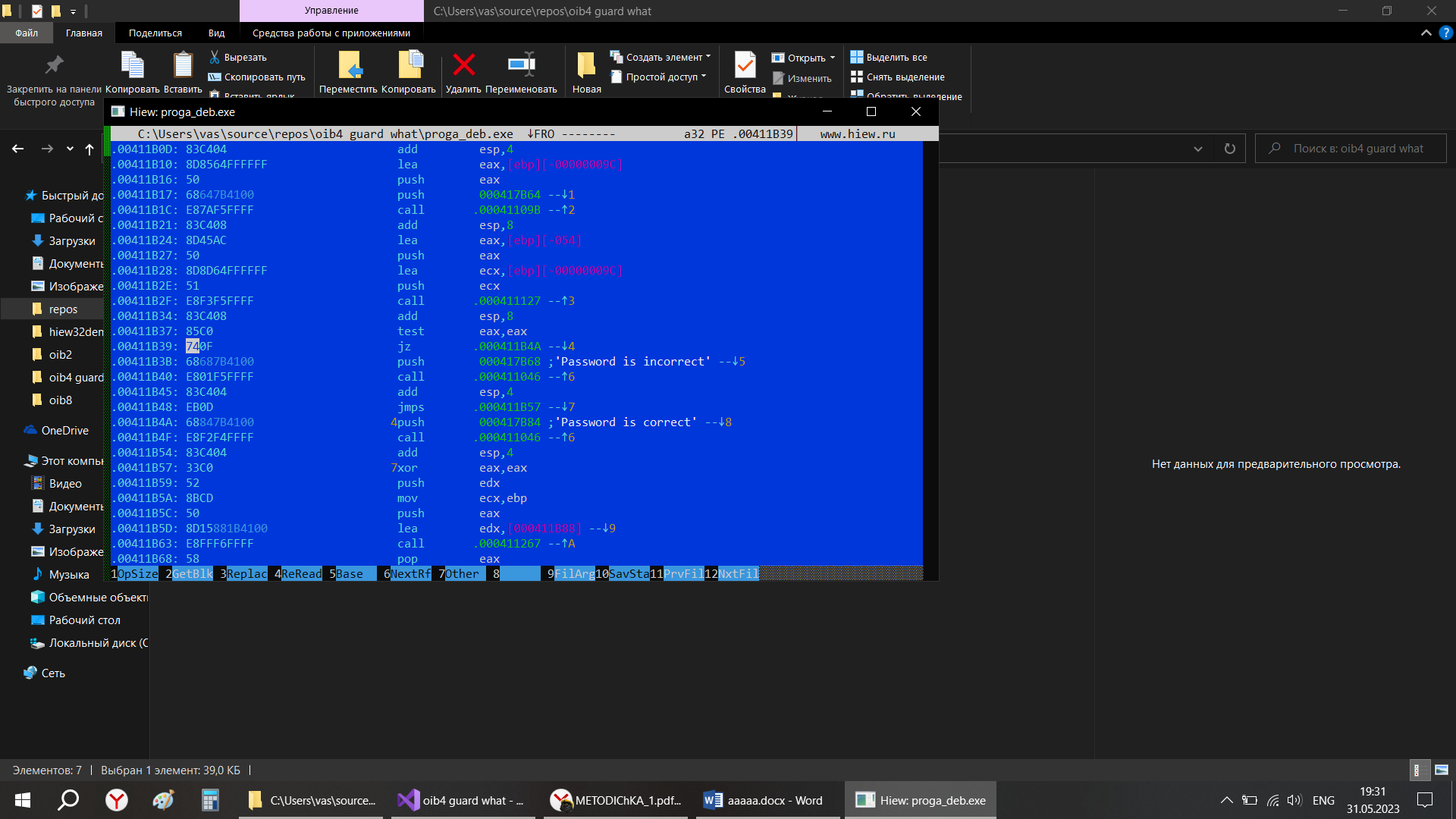
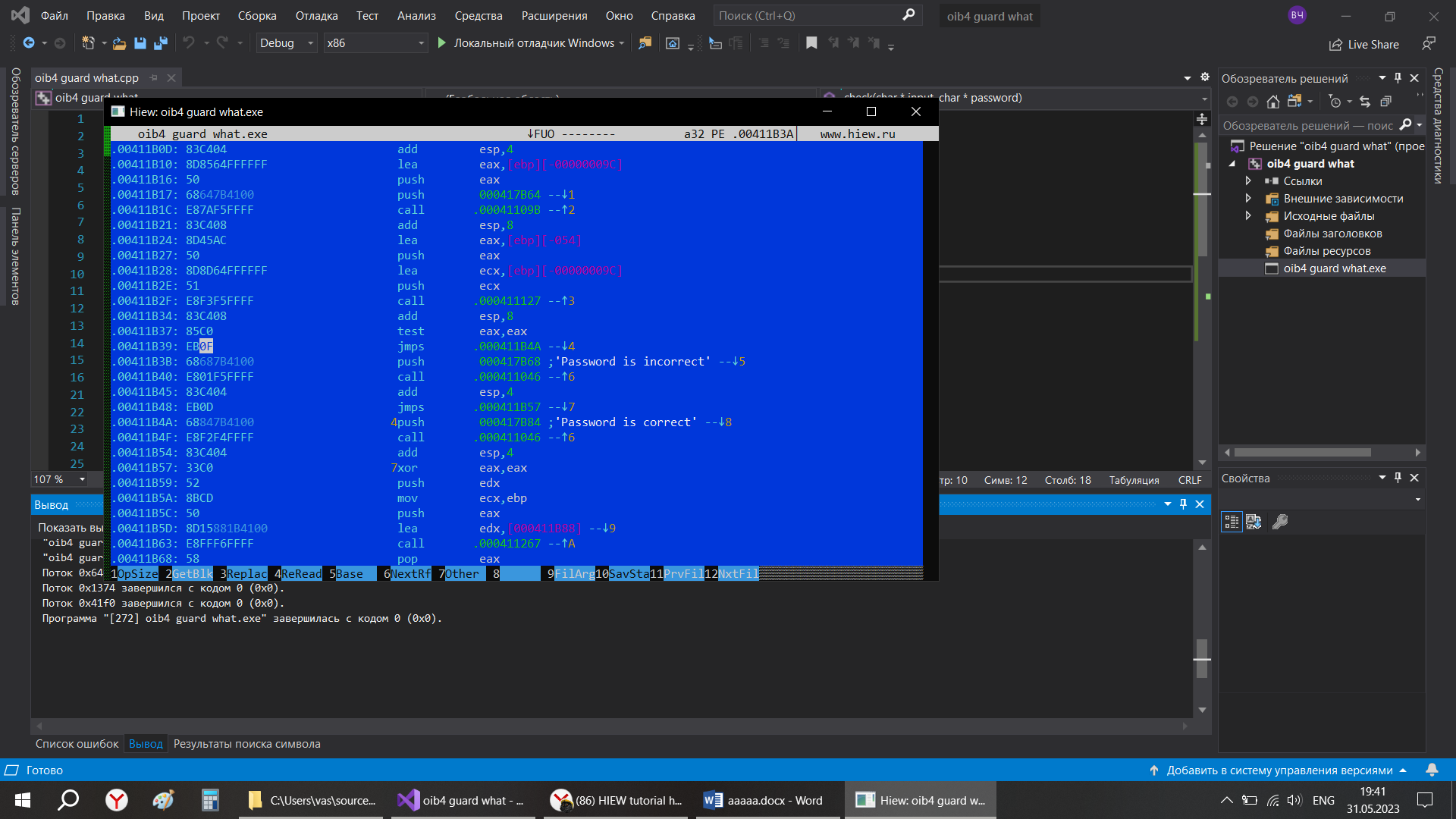


Рис. 3-4 —Изменение на безусловный прыжок

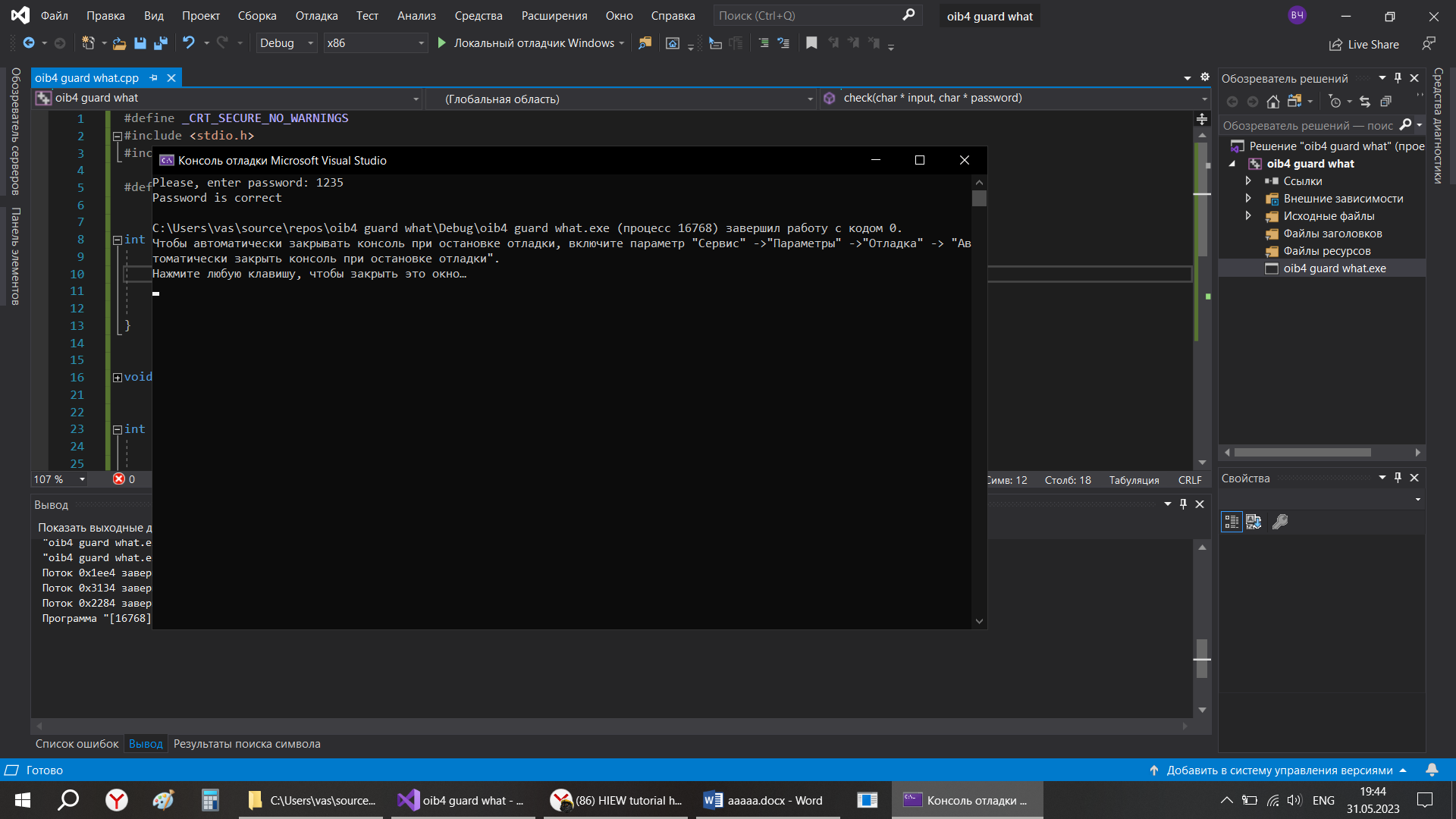


Рис. 5 — Работа измененной программы

2. В выданном файле Nag.exe спустя 10 секунд появляется окно с таймером на 100 милисекунд, которое блокирует любое взаимодействие с основной программой. С помощью Hex-редактора IDA была найдена процедура SetTimer, которая отвечвет за счет часов на кнопке

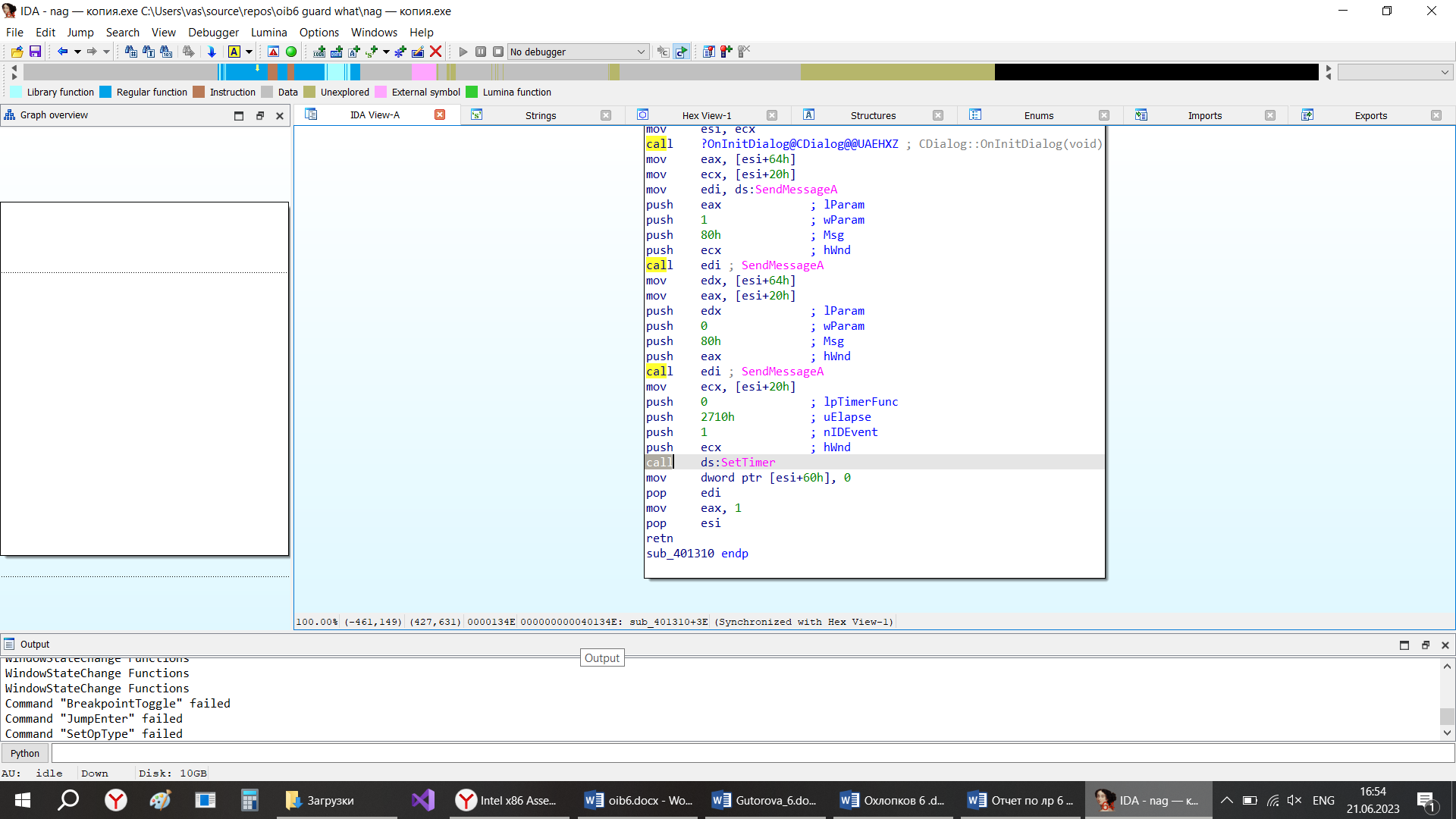


Рис.6 — Вызов команды SetTimer и изначальные данные

Она вызывается в 2 случаях: таймер появления окна и таймер внутри окна. На вход из стека ей подаются переменные с идентификатором окна, номер события, переменная до скольки считать таймеру и переременная функции таймера.

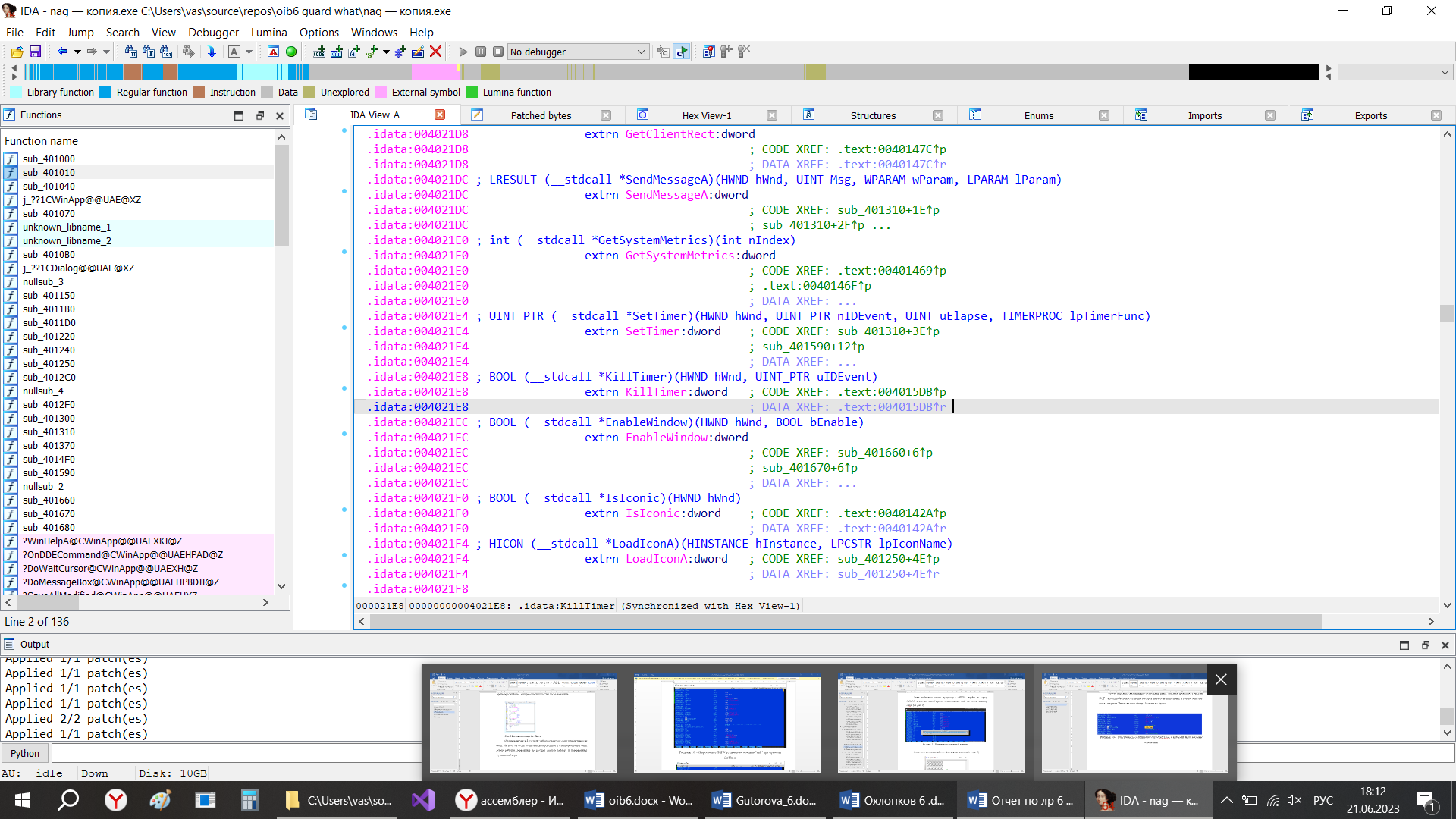


Рис.6 — Описание SetTimer и KillTimer. Прыжки, где вызываются эти функции

Чтобы в окне nagscreen сразу показывалась кнопка OK, была изменена в ассемблере строчка сравнения в функции KillTimer, чтобы таймер заканчивался сразу.

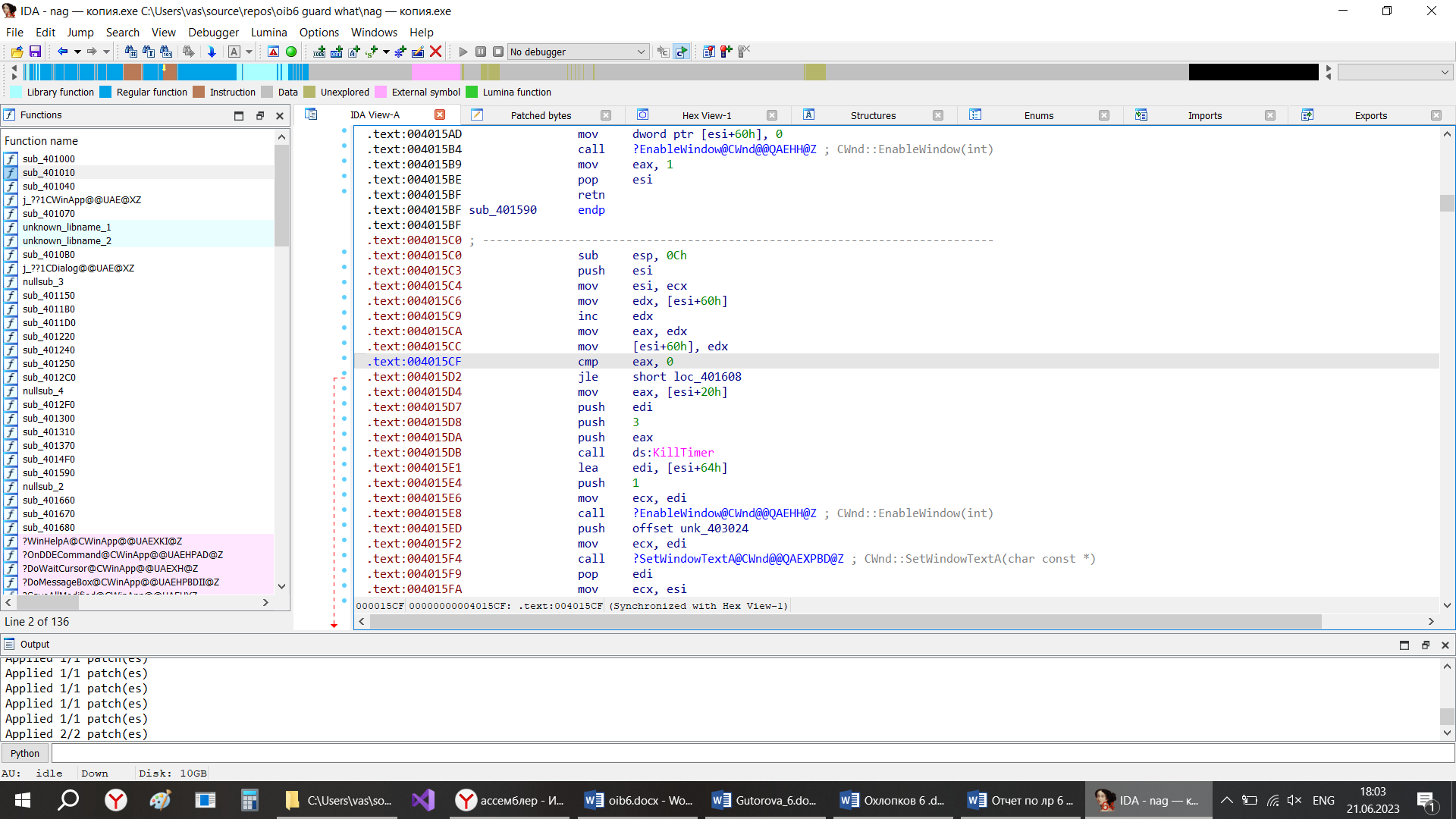


Рис 7. — Изменение байтов в функции, которая вызывает KillTimer

В переменной uElapse хранится число, ответственное за время, через которое окно nagscreen будет появляться. Поэтому, чтобы окно не появлялось, можно просто заполнить данную переменную максимально возможным значением.

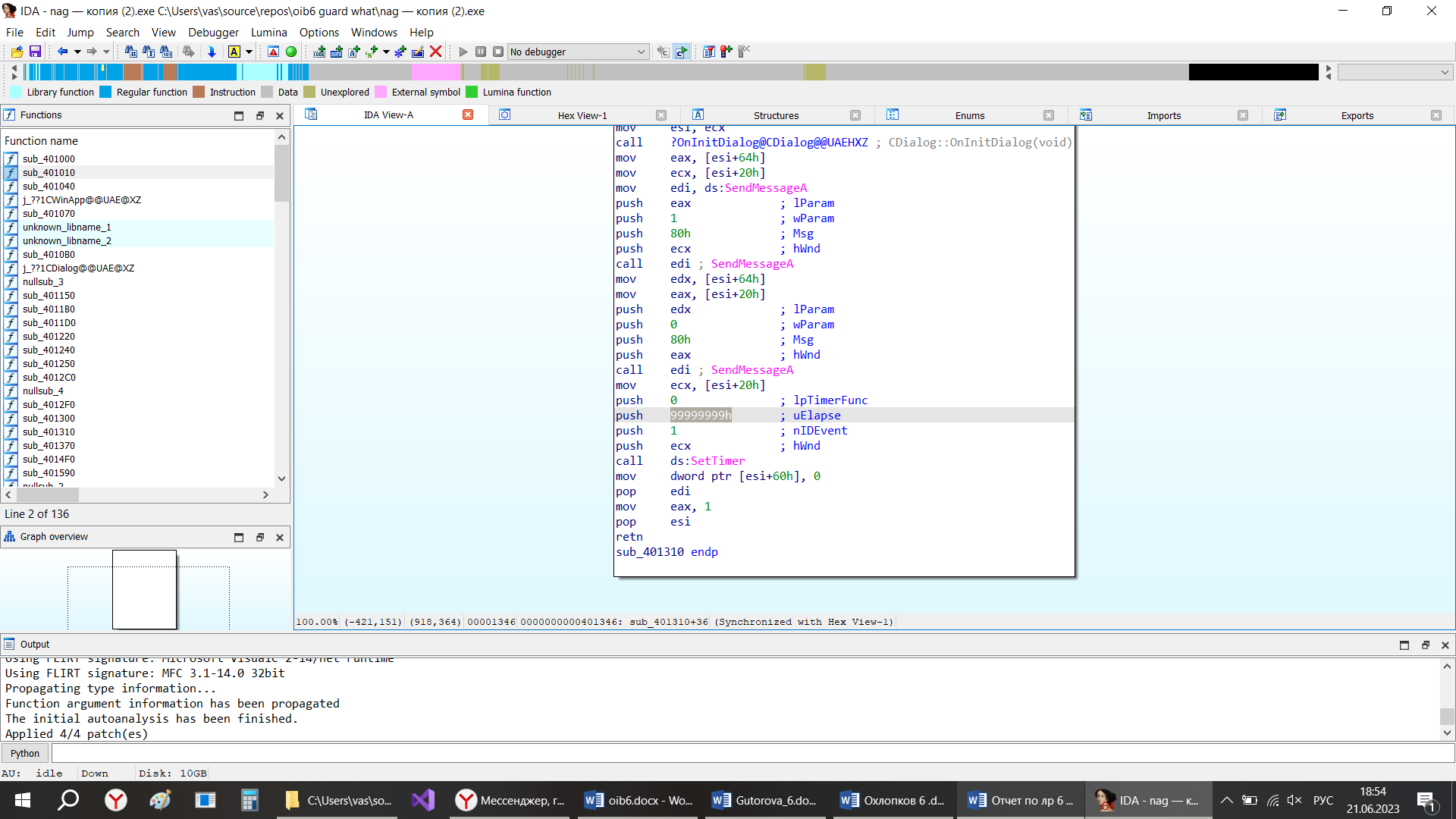


Рис. 8 — Изменённое значение uElapse

3. Программа GUARD.exe просит ввести пароль. При вводе неверного пароля программа выдает ошибку.

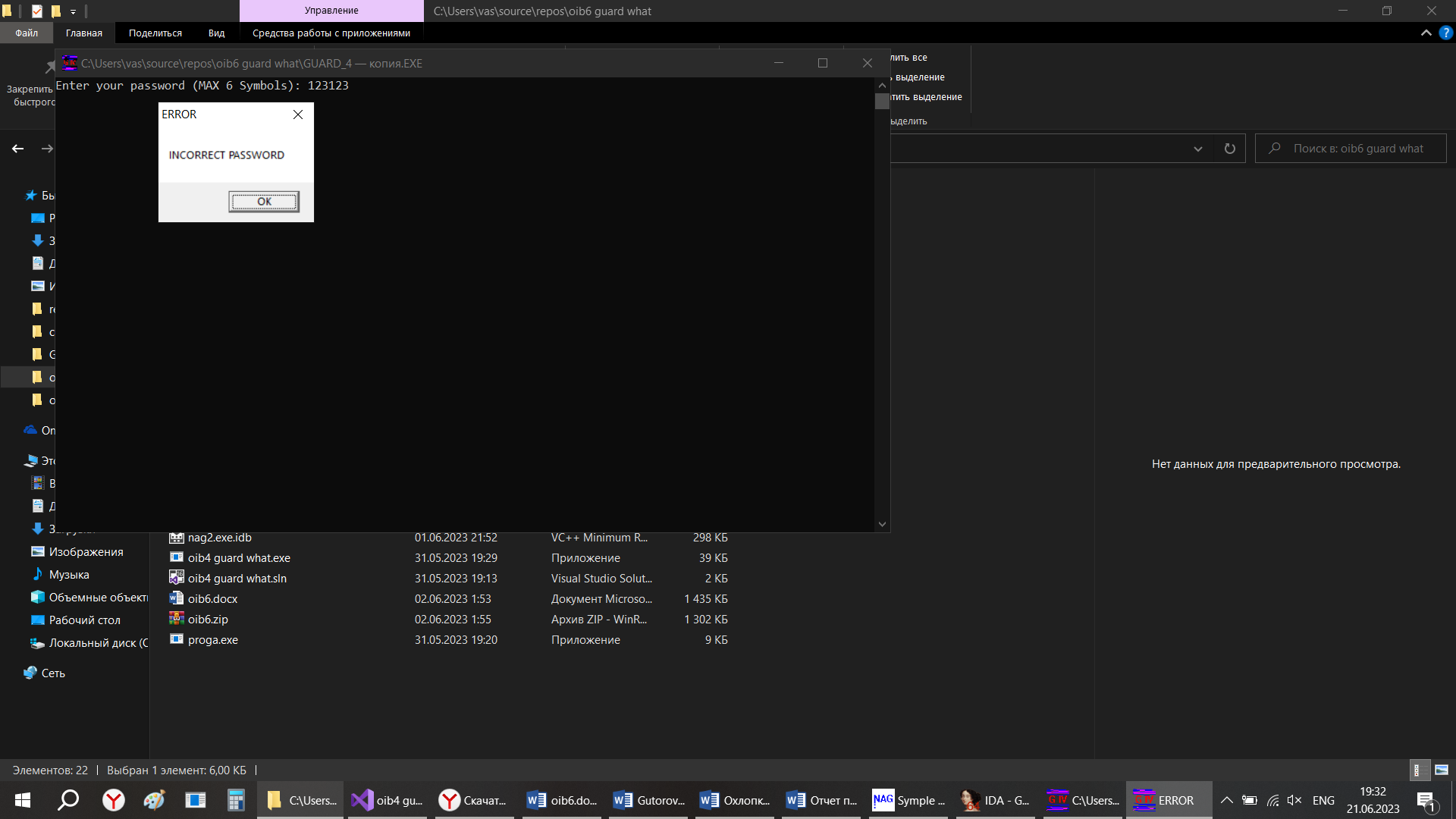
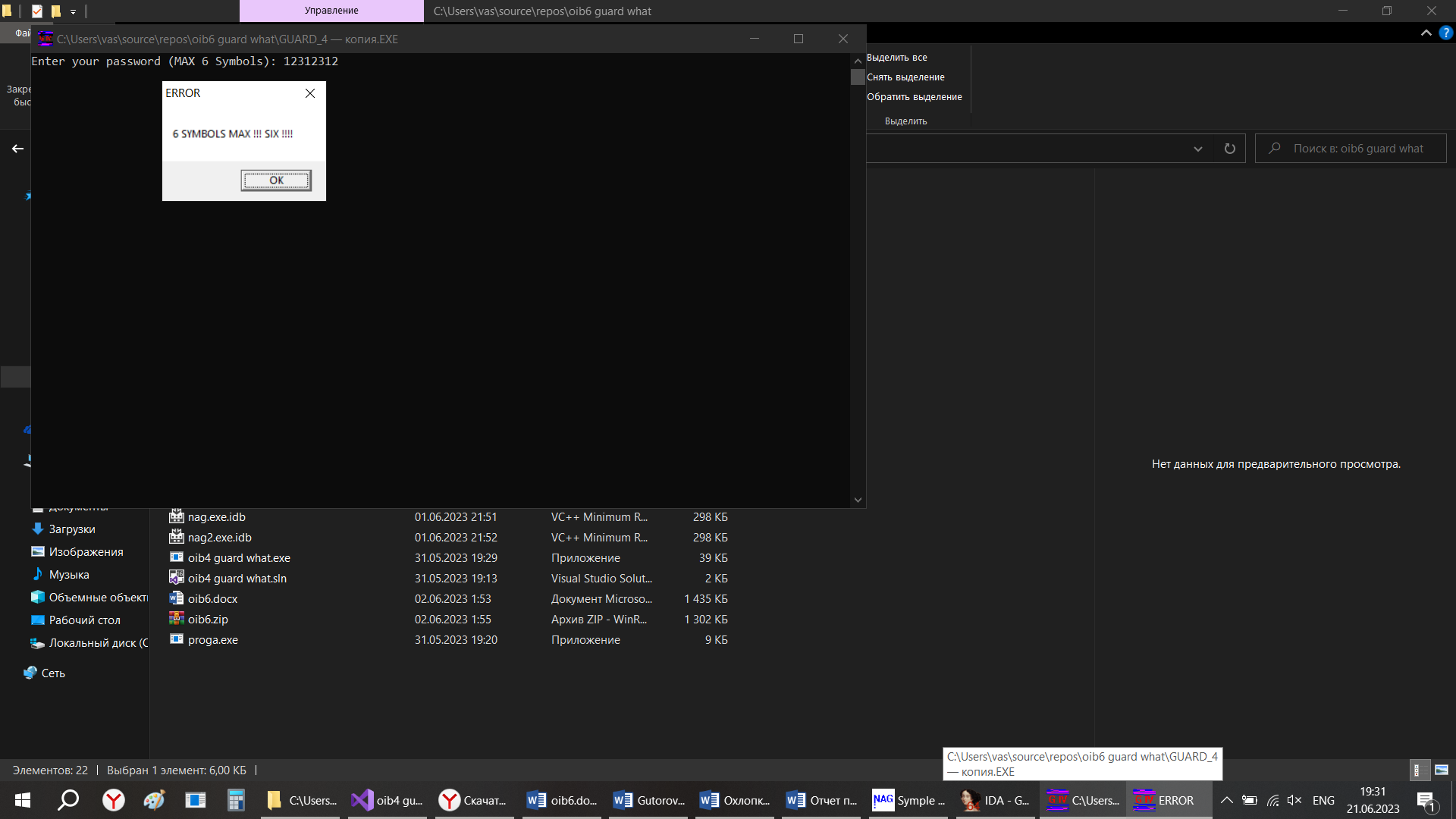


Рис. 9-10 — Работа GUARD.exe

При попытке загрузить программу в «IDA Pro» не получается проанализировать дизассемблированный код. Значит он упакован. Чтобы узнать тип упаковки используется программа «PEiD». В нее загружается программа и в строке «EP Section» будет указан тип упаковки, в данном случае UPX 1 (см. рис. 11).

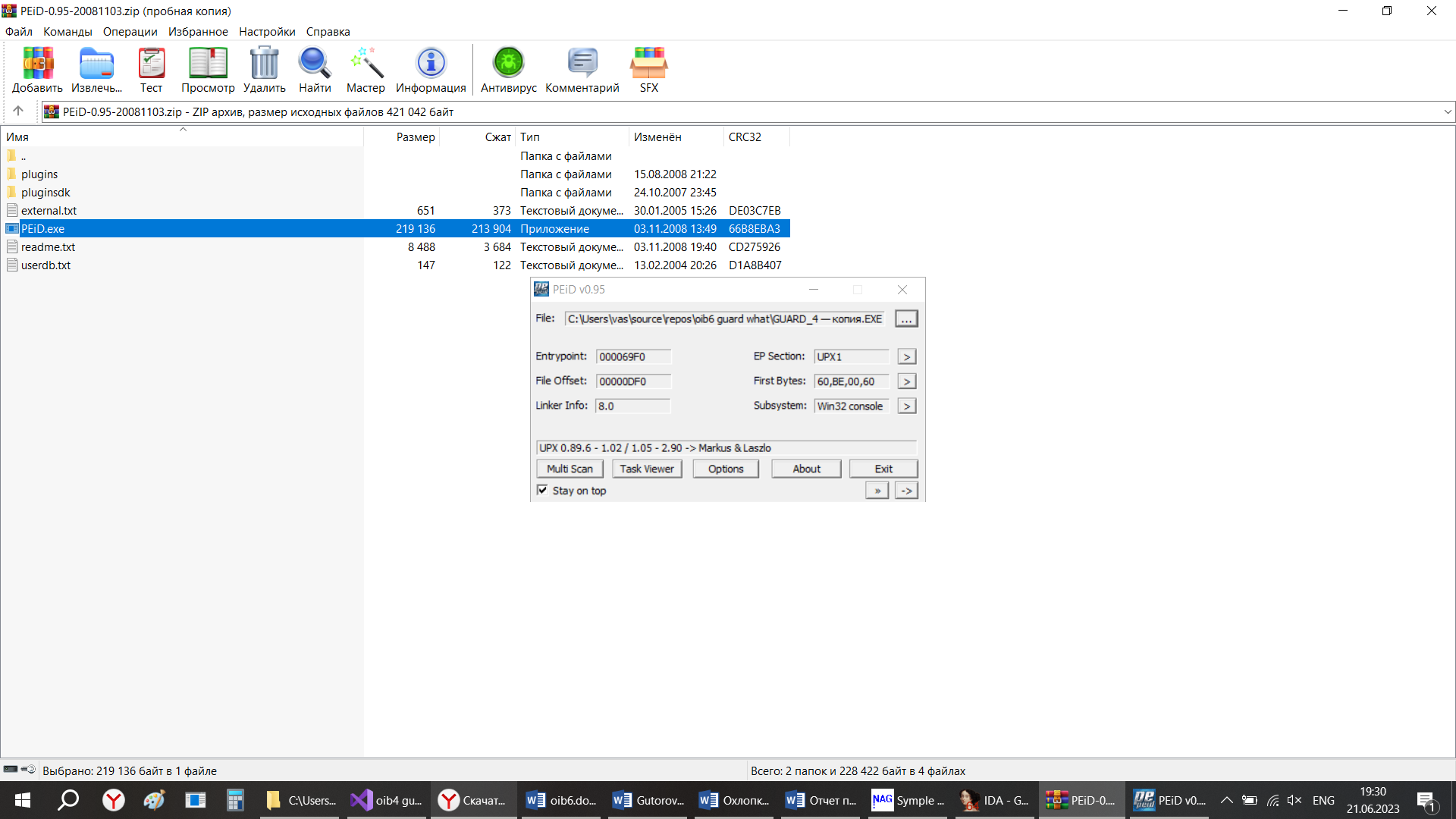


Рис. 11 — Тип упаковки GUARD.exe

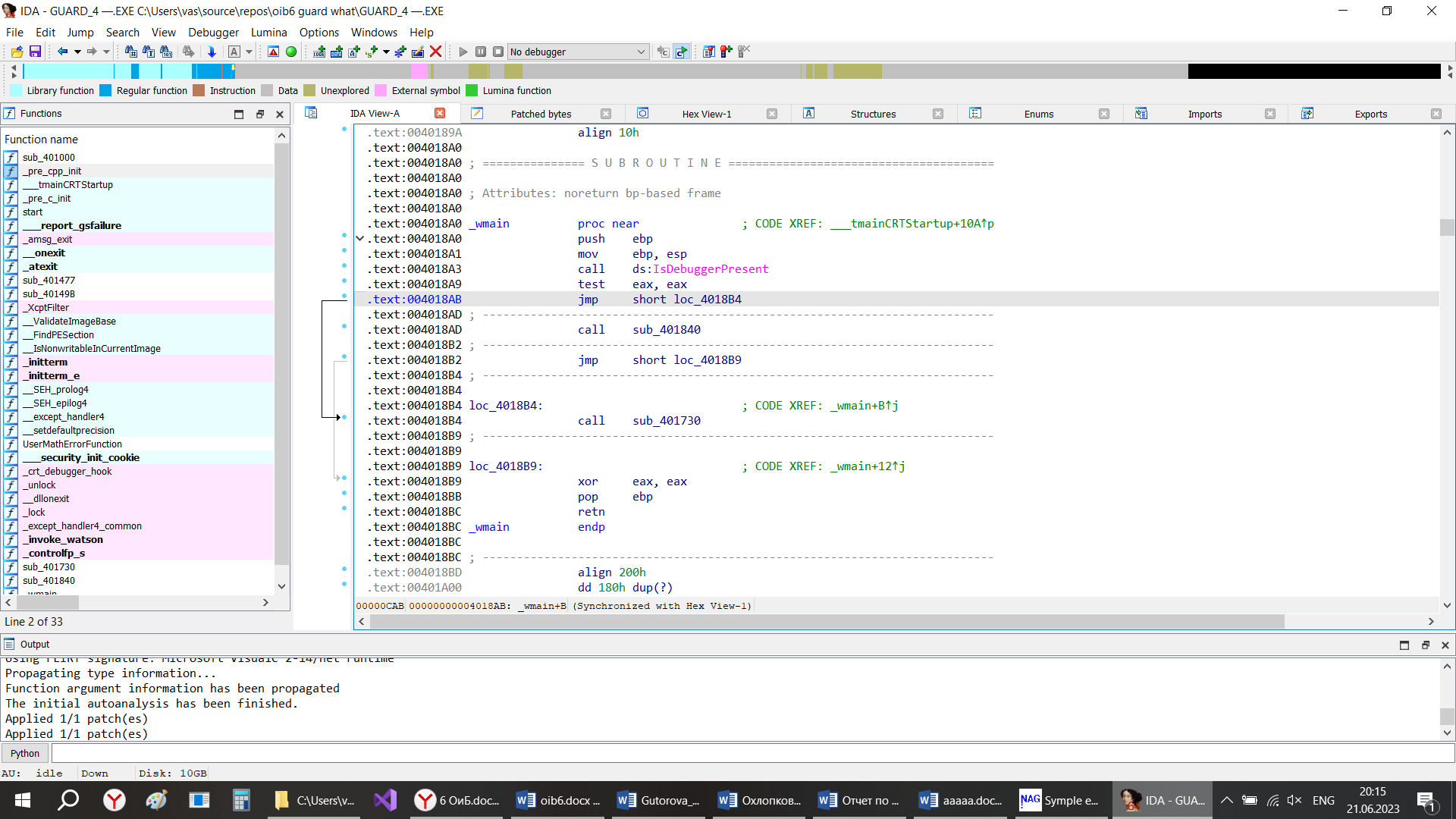
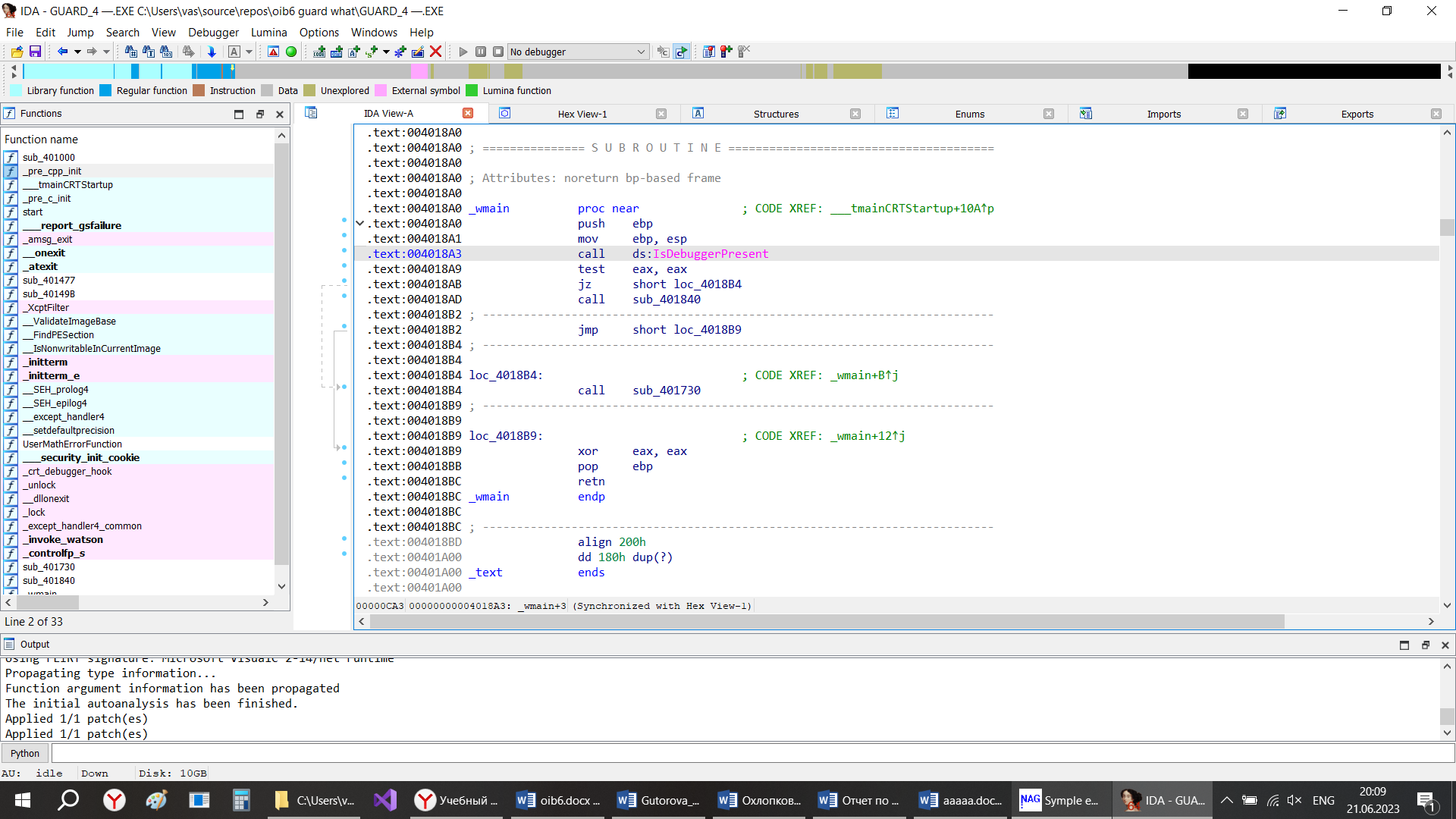


Рис. 12-13 — Проверка на дебаггер и изменение условного прыжка jz на безусловный jmp

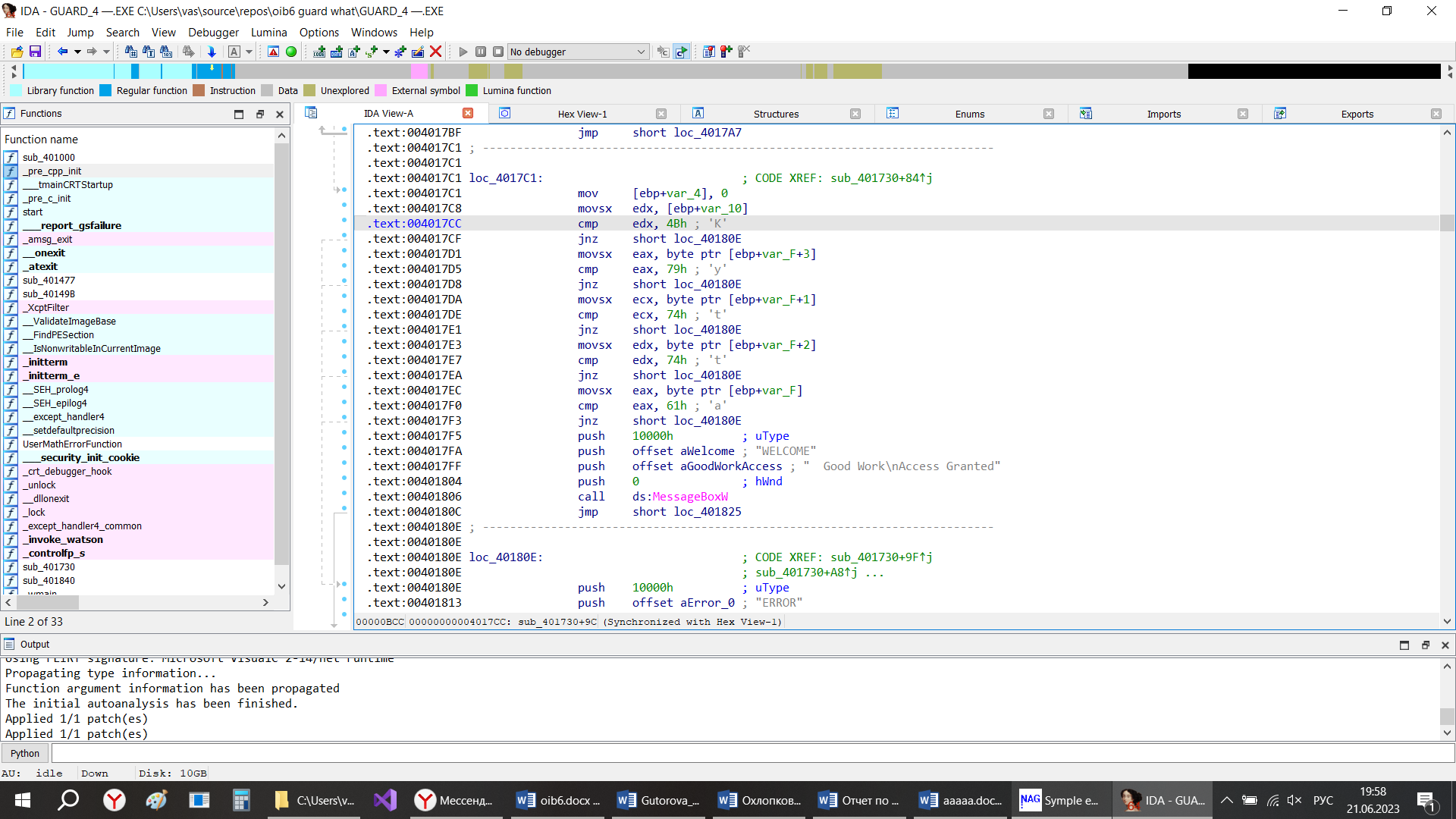
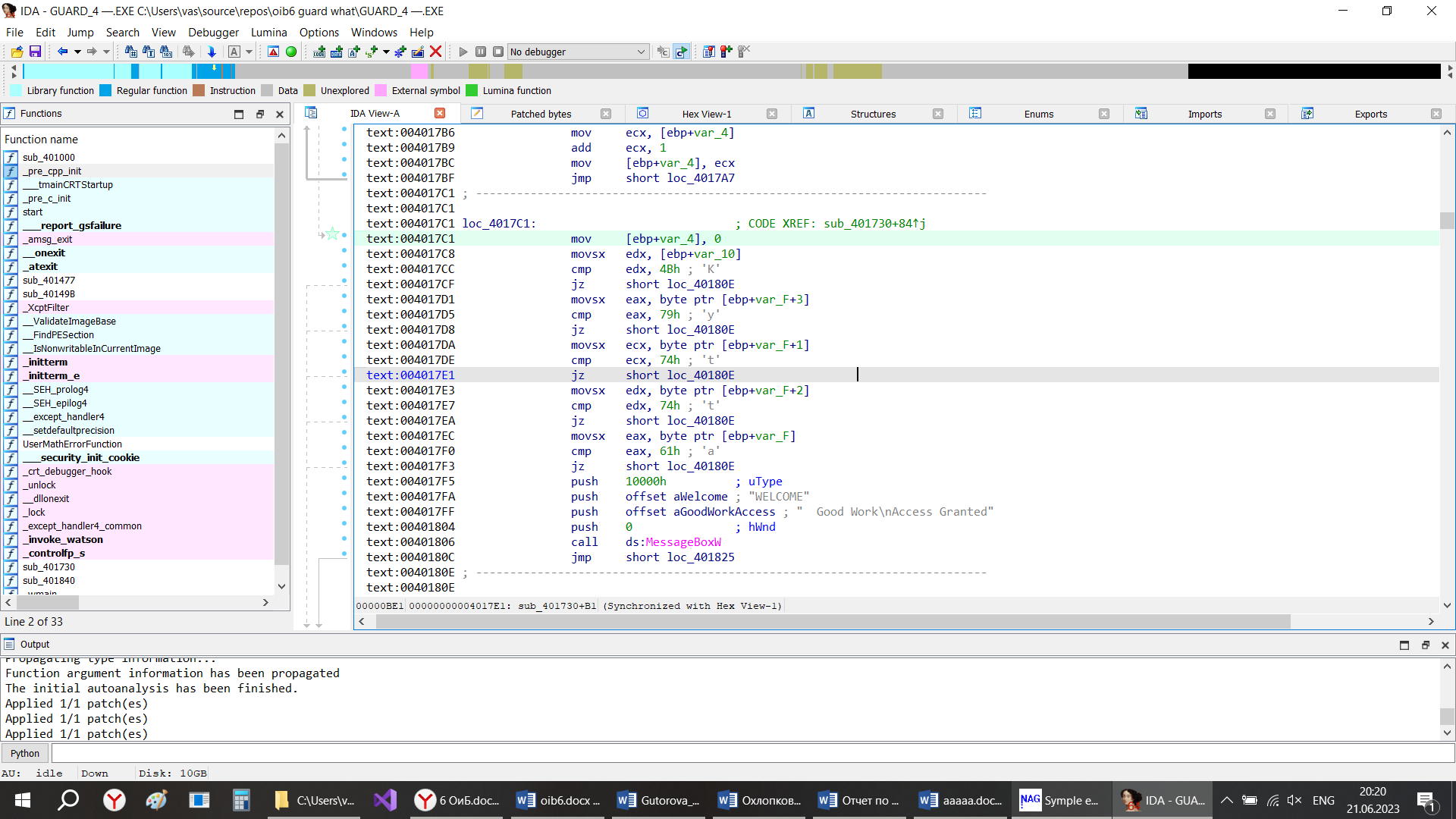
 

Рис. 14-15 — Проверка пароля. Изменение условного прыжка jnz на обратный jz

Из-за того, что условия для окна Welcome были изменены на обратные, то программа теперь предостваляет доступ по любому паролю

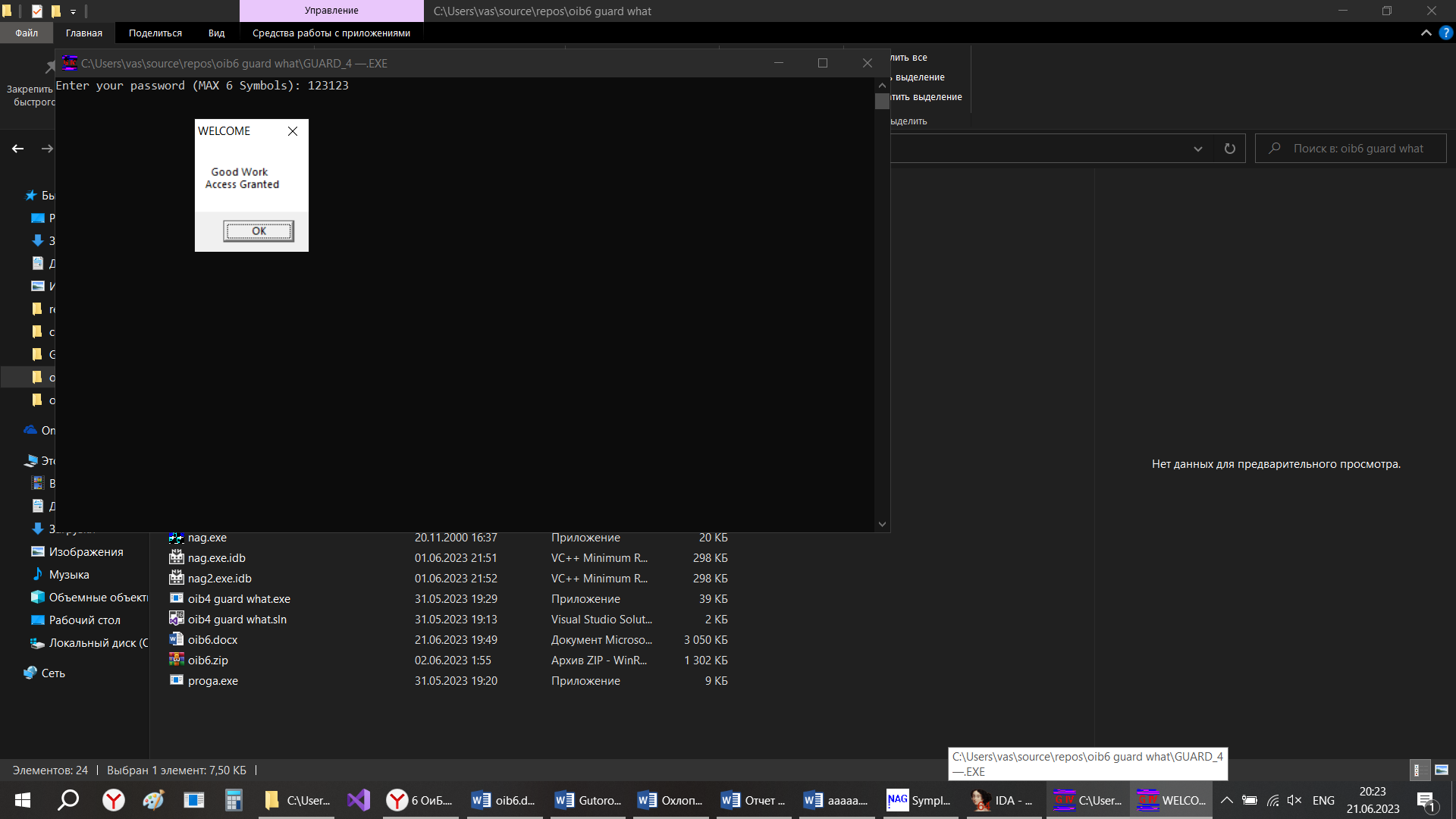
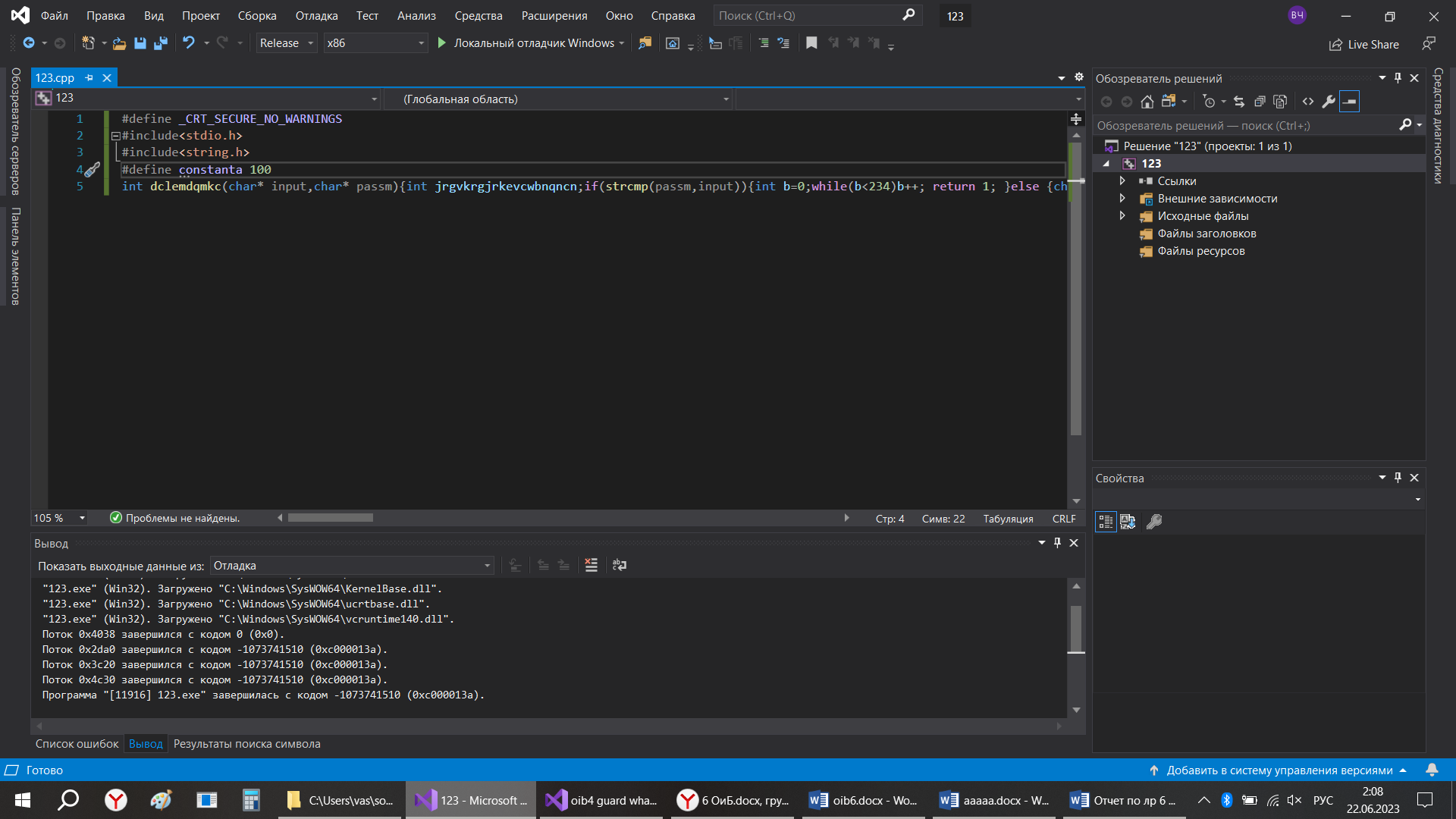


Рис. 16 — Успешное получение доступа

4. Программа обфускатор добавляет в код «мусорные» переменные и функции, а также меняет названия переменных в коде. Обфускация помогает сделать программу более защищенной от отладки и анализа, но ни один из существующих обфускаторов не гарантирует абсолютной невозможности восстановления логики работы программы.



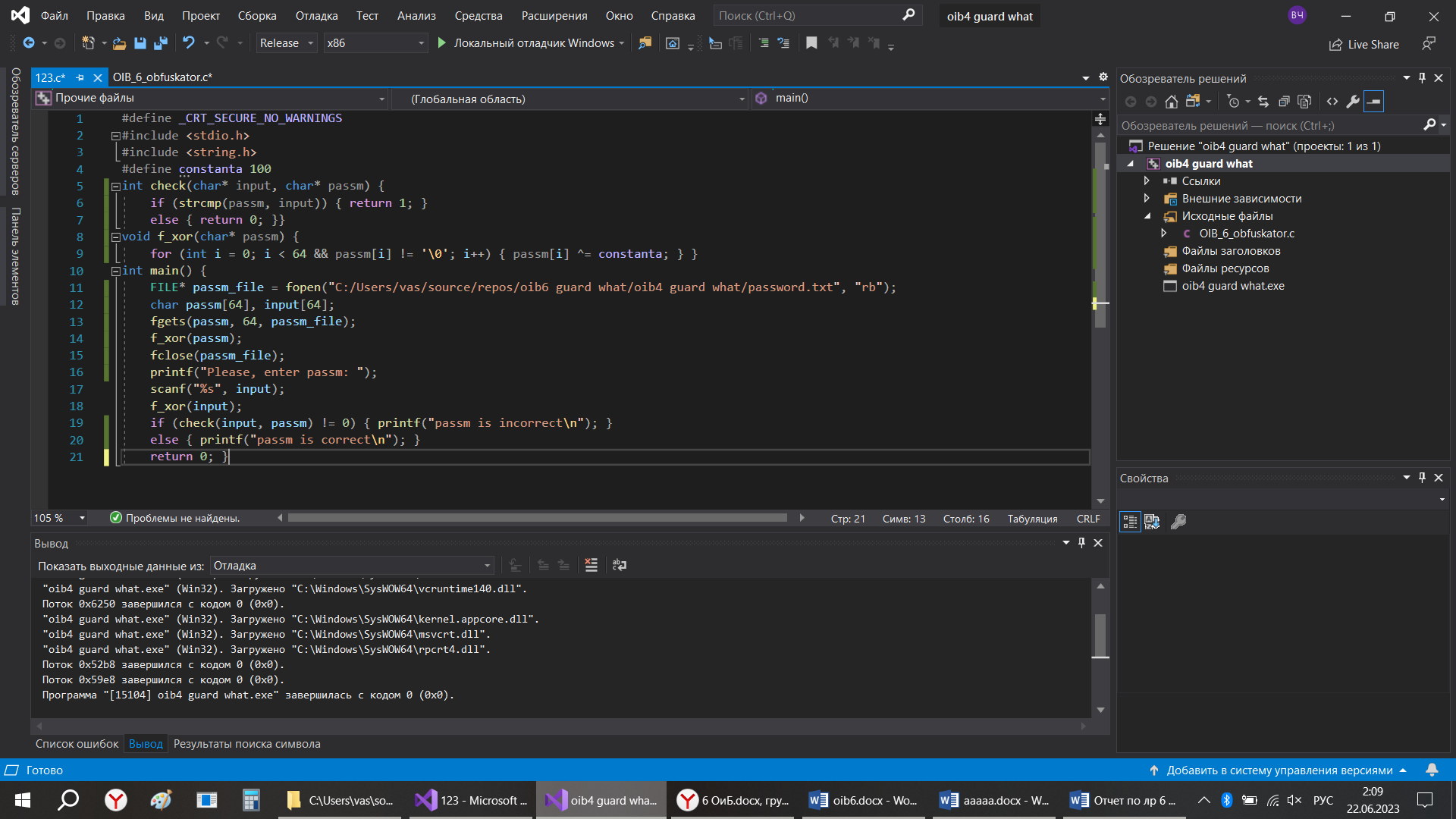


Рис. 17-18 — Код из п.1 до и после работы обфускатора

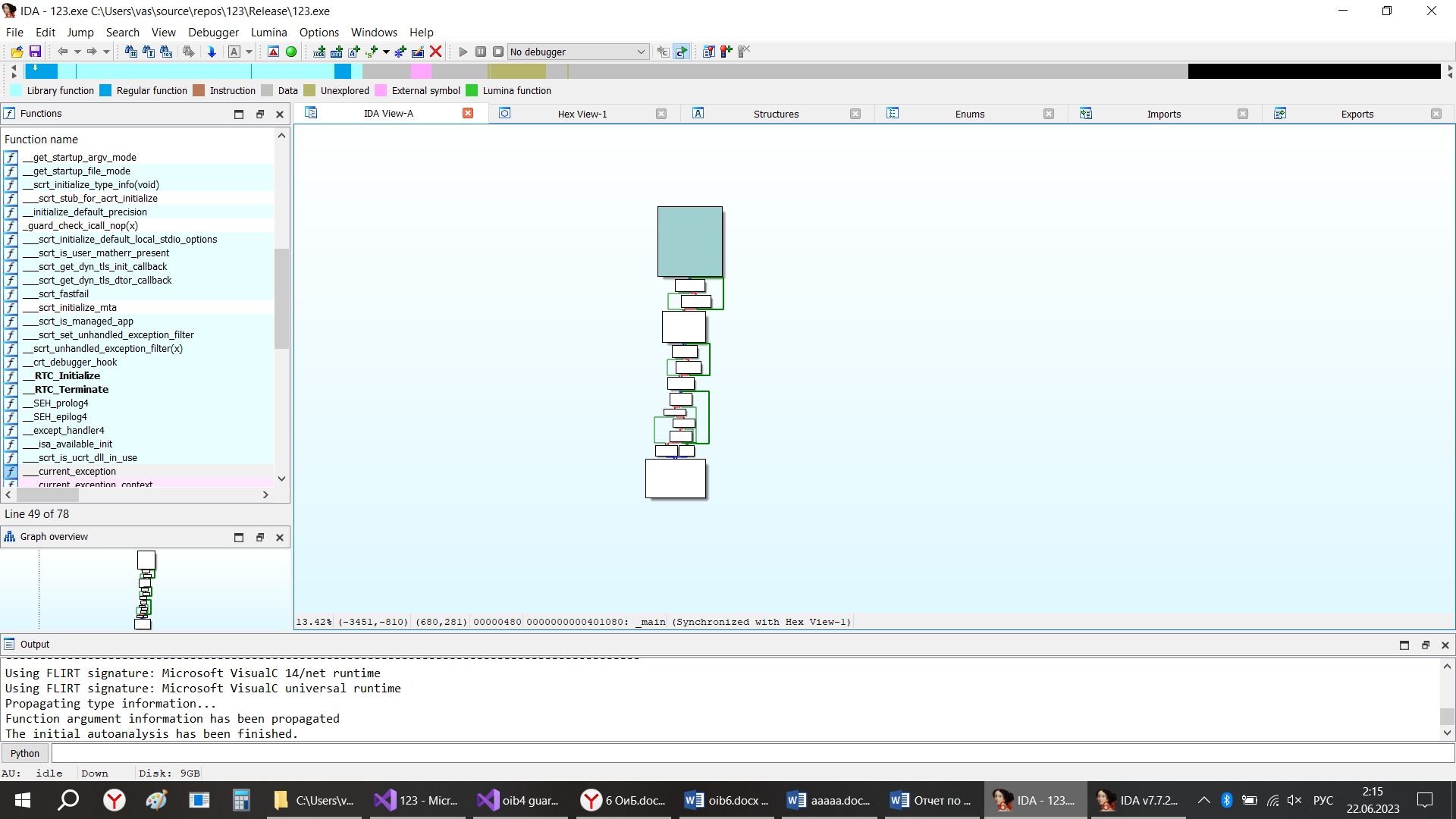
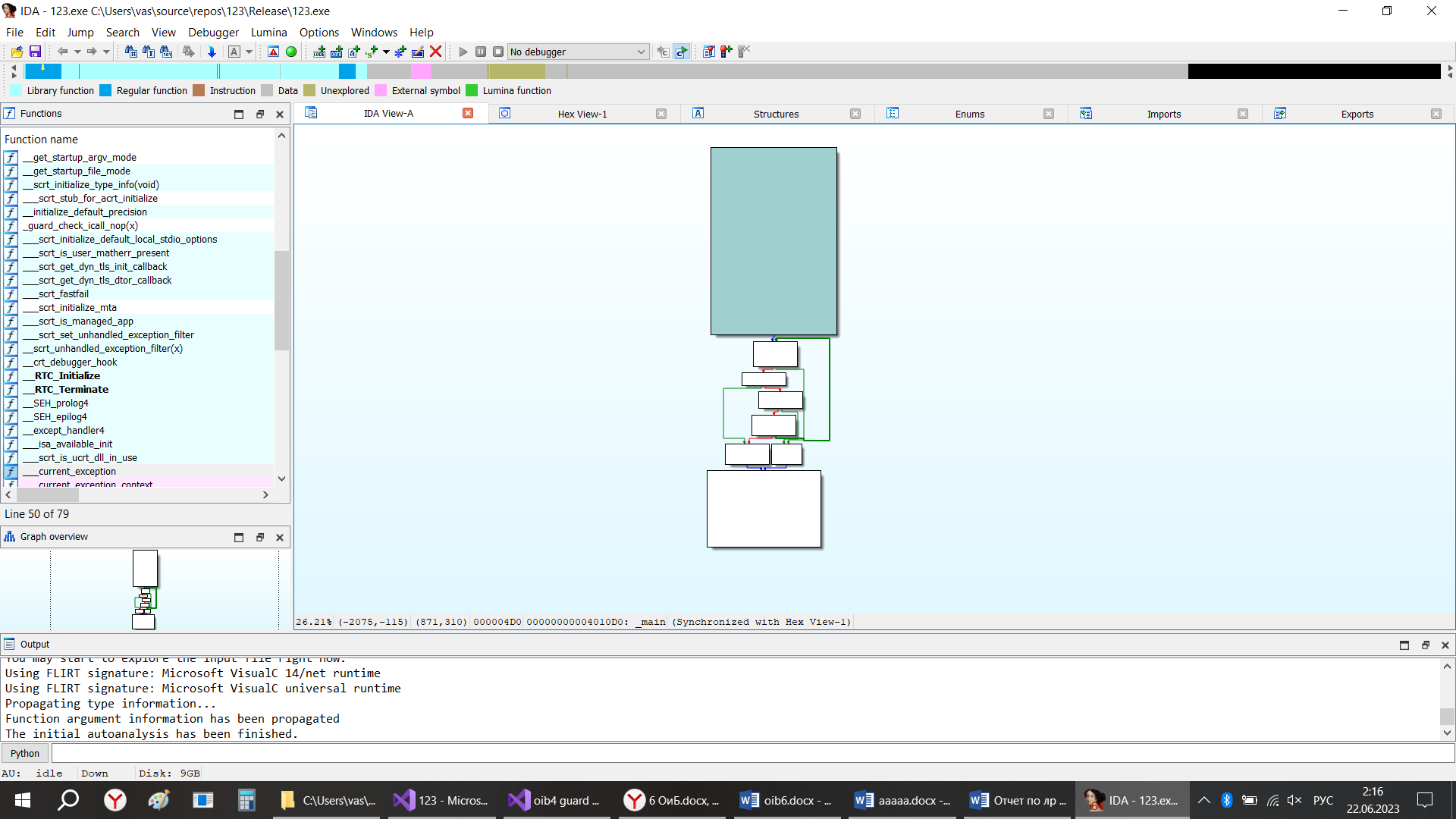


Рис. 19-20 — Граф программы до и после работы обфускатора

# 4. Результаты работы

В результате работы были разработаны программа, запрашивающая пароль, хранящийся в закрытом виде и программа-обфускатор, а также было реализовано обхождение защиты через дисассемблер.

# 5. Вывод

В ходе выполнения работы были изучены различные методы защиты программ от нелегального использования, приобретены навыки по обхождению защиты и взлома приложений путем дисассемблирования.