02.01.2024

Курс:

Практическая работа к уроку № Lesson_6

--
Способы обхода активации программ

Задание:

Дана программа task-4. Необходимо получить корректный ключ для вашего имени.

Пример запуска программы:
crackme-4.exe

Name: `<your name>`
License Key: `<license key>`

Способы обхода активации программ
Для достижения этой цели может быть использованы различные подходы и способы.
Далее будут рассмотрены следующие способы:

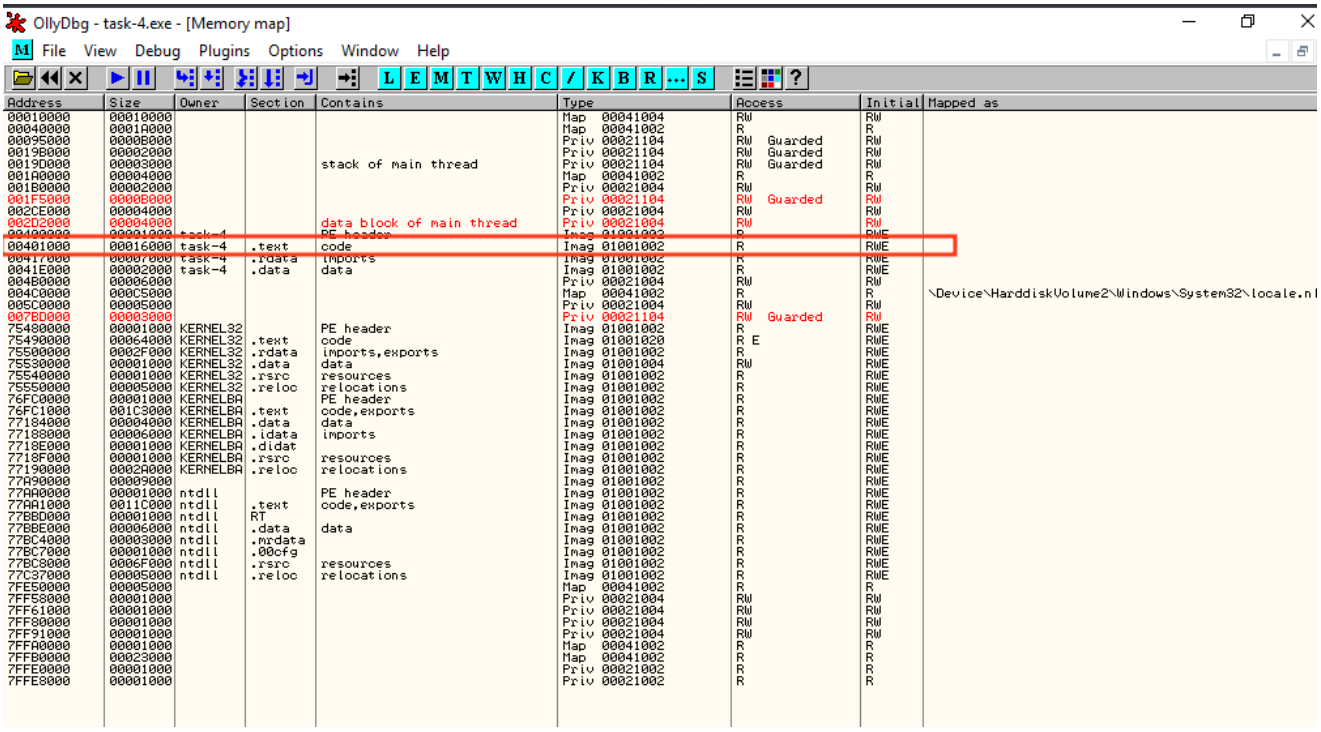
- Продление триального периода
- Создание ключа
- Патчинг
- Перенаправление сетевых запросов

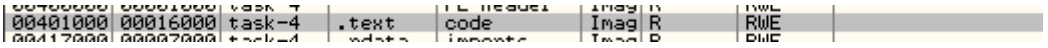
Запускаем в CMD *task-4.exe*

Insert name: Andrew
License key: qwerty

Запускаем в OllyDbg *task-4.exe*

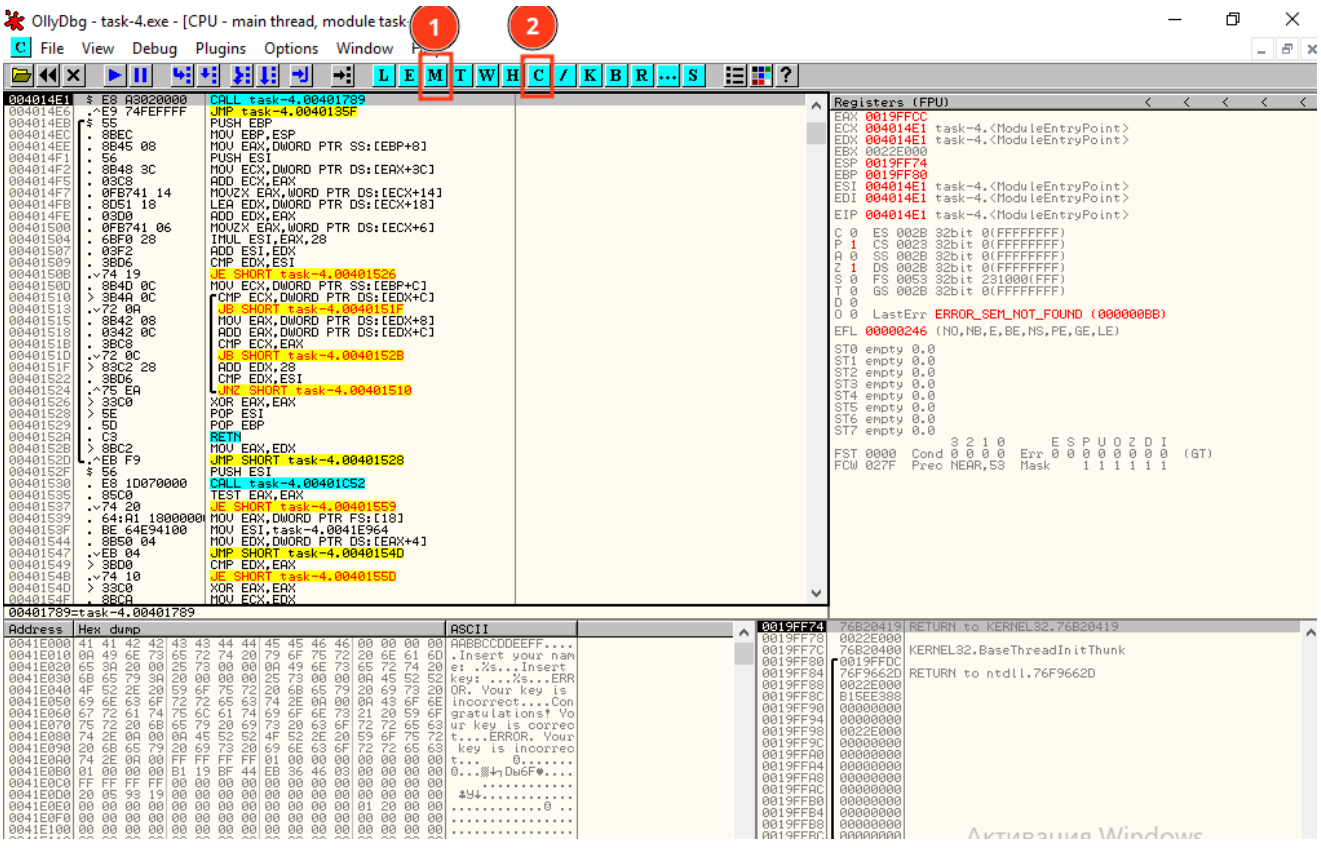
Окно ***Memory***



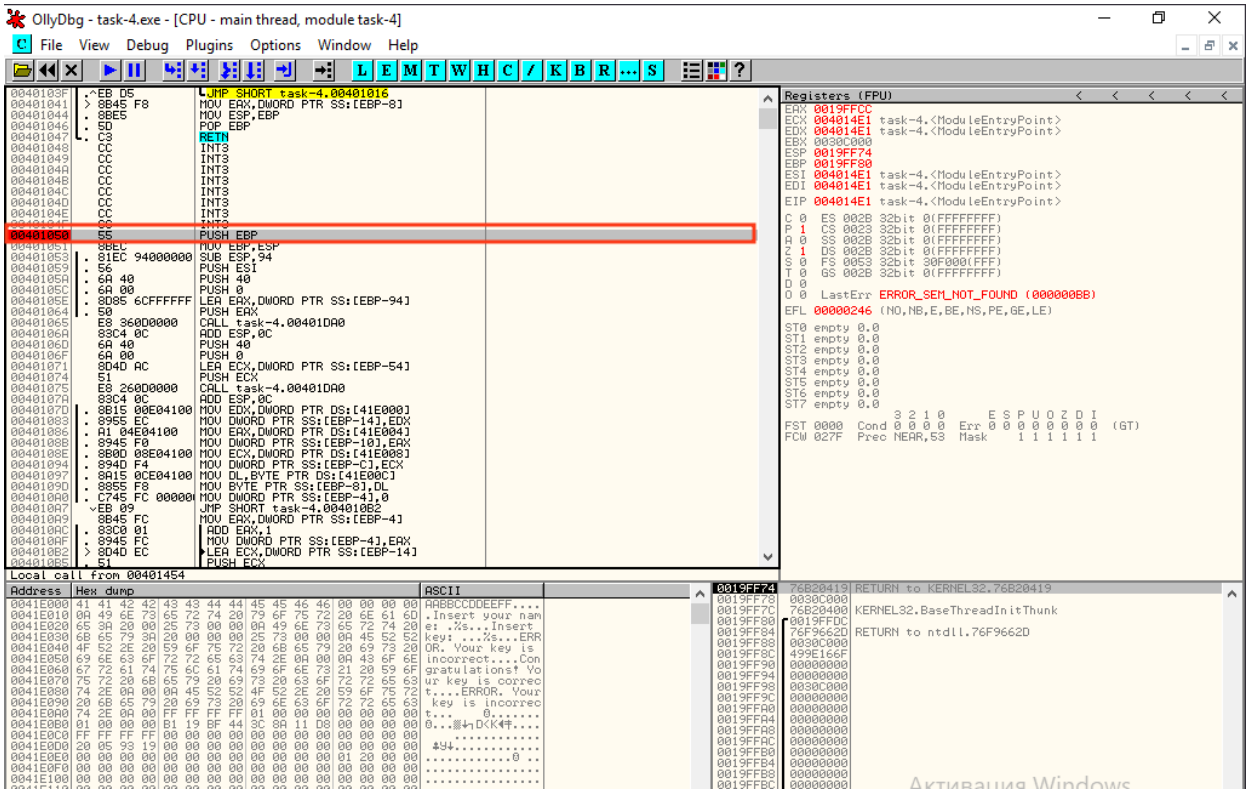


- Переходим по этому адресу (C)

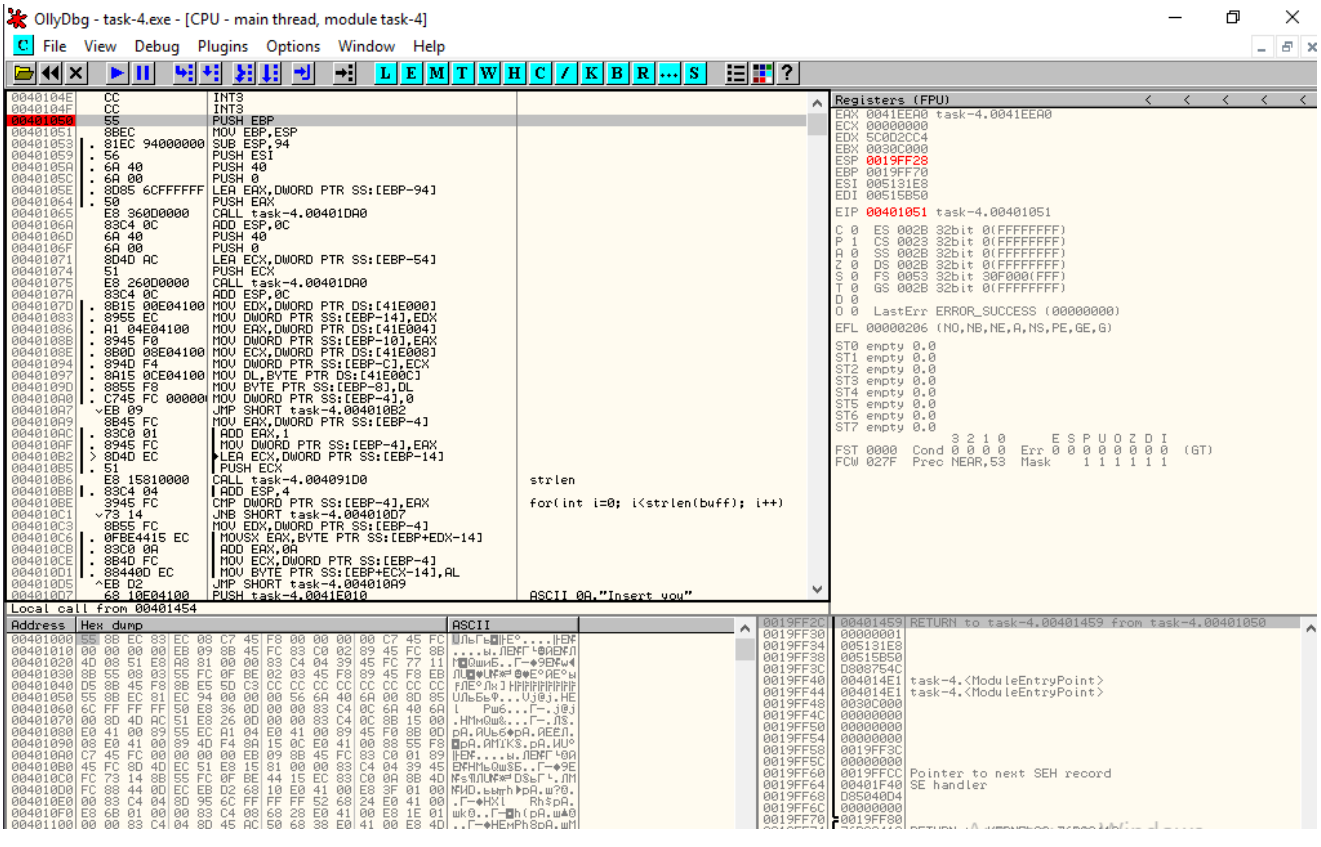
Ctrl+G	Follow expression
	00401000

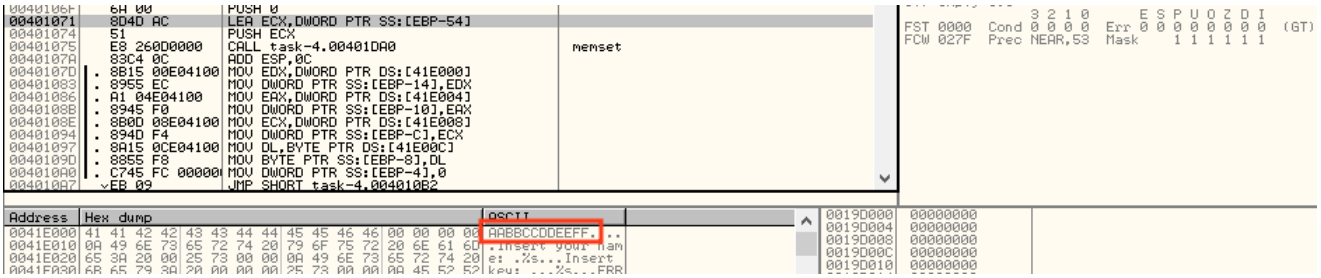


- Точка останова



Run application	Main	Debug|Run	F9



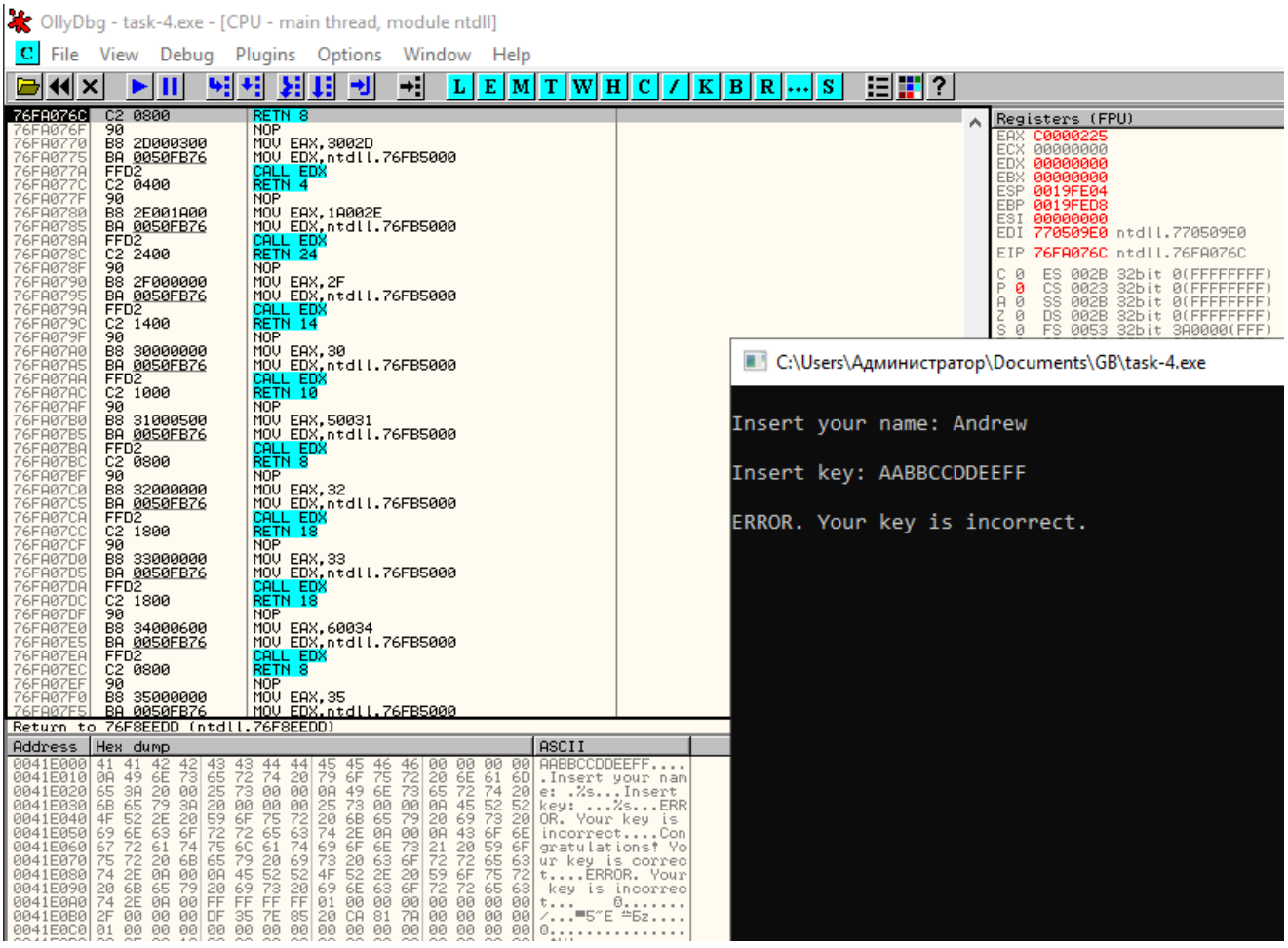


Try:

Incorrect:

AABBCCDDEEFF

abcedefghiklmnopqrstuvwxyz



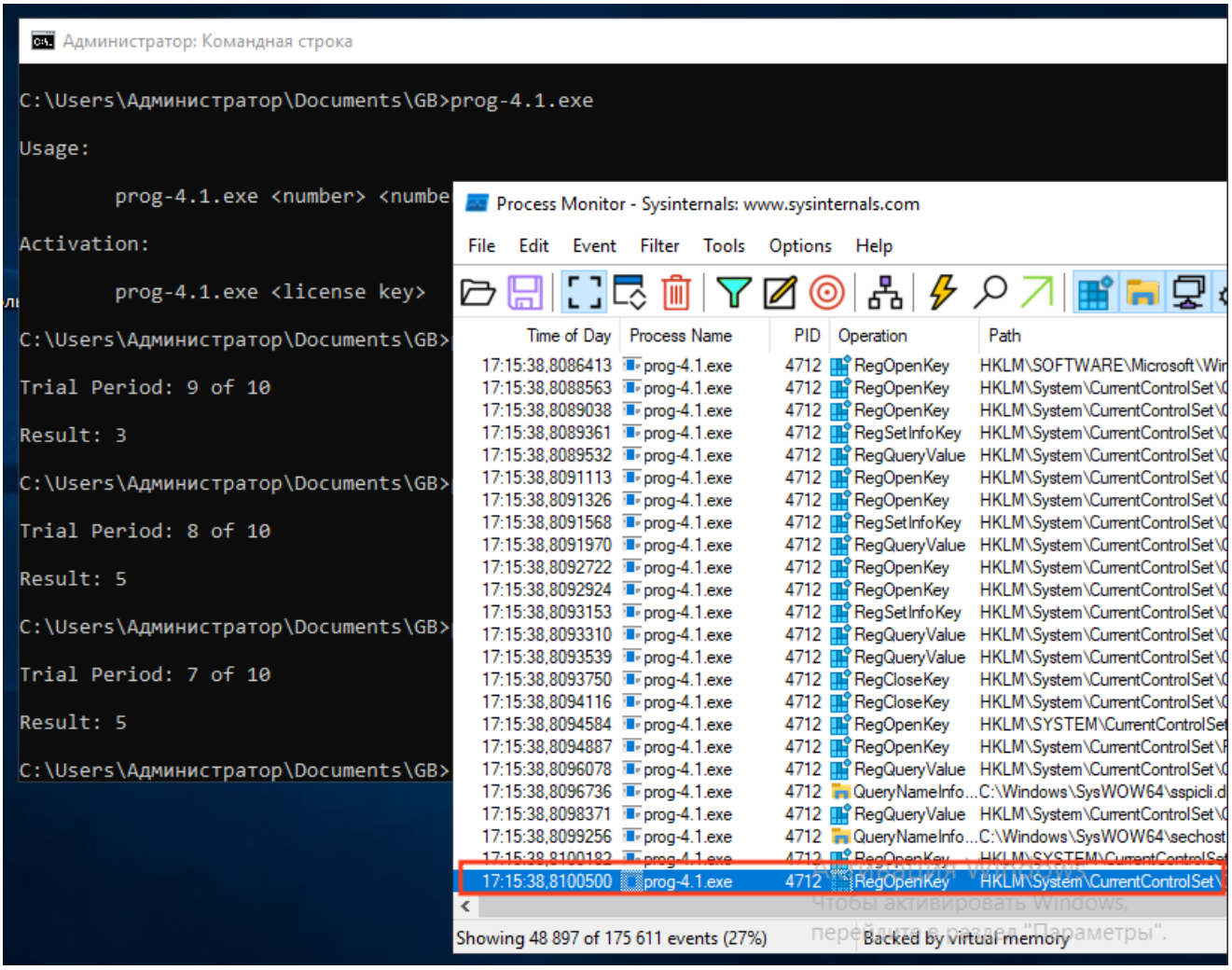

```
for(int i = 0; i < strlen(buff); i++)

if (func(key) + strlen(name) + func("AABBCCDD") == 0x514)
        printf("correct");
else
        printf("incorrect");

        for (int i = 0, j = 0; i < strlen(buff); i+=2)

        j = j + data(i)
```

Пример ***prog_4.1.exe***:

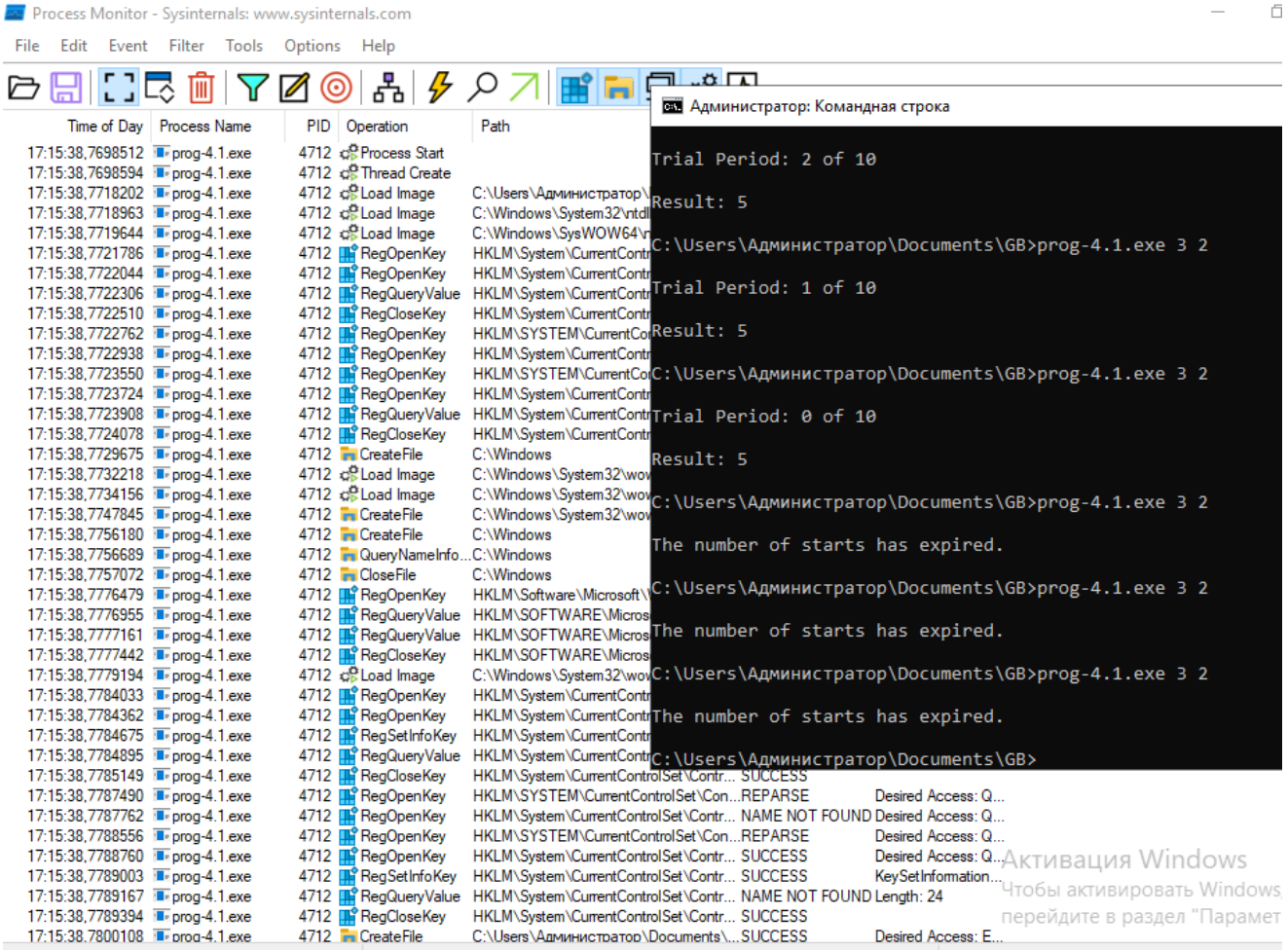
1. Продление триального периода:

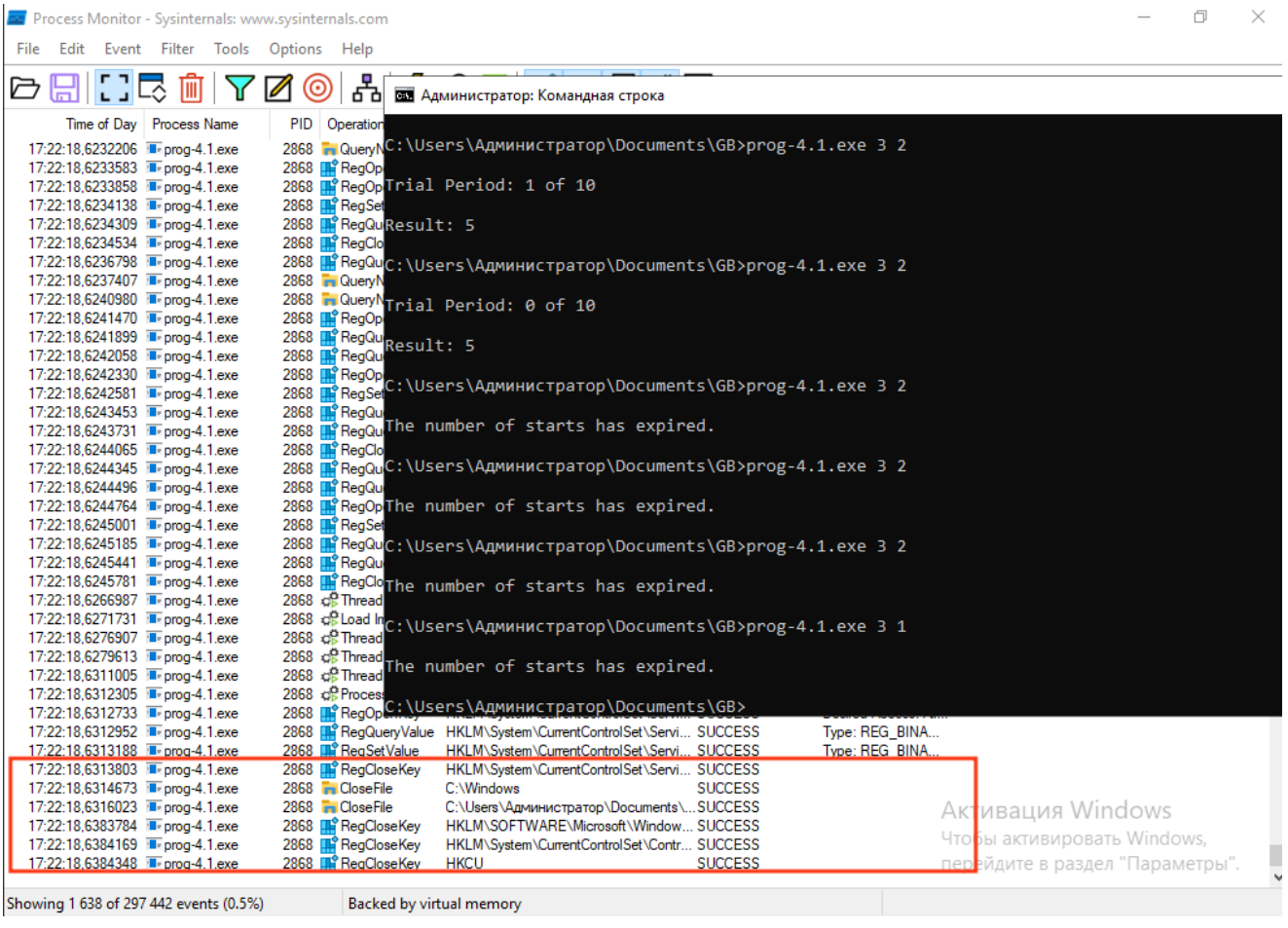
Устанавливаем ***procmon***
https://learn.microsoft.com/en-us/sysinternals/downloads/procmon

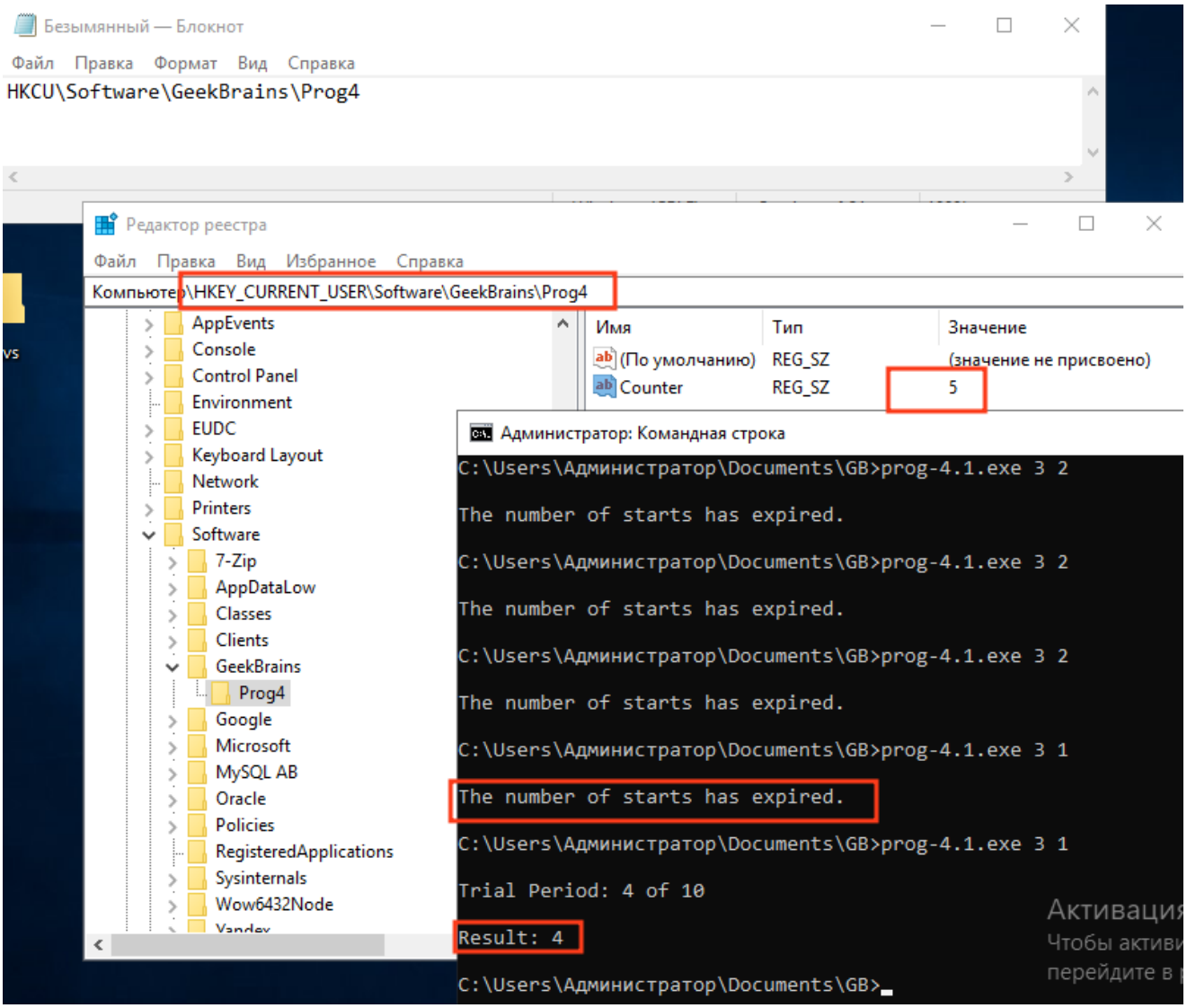
```
C:\Users\Администратор\Documents\GB>prog-4.1.exe 2 1

Trial Period: 9 of 10

Result: 3
```



Add Filter -> Process Name: prog-4.1.exe



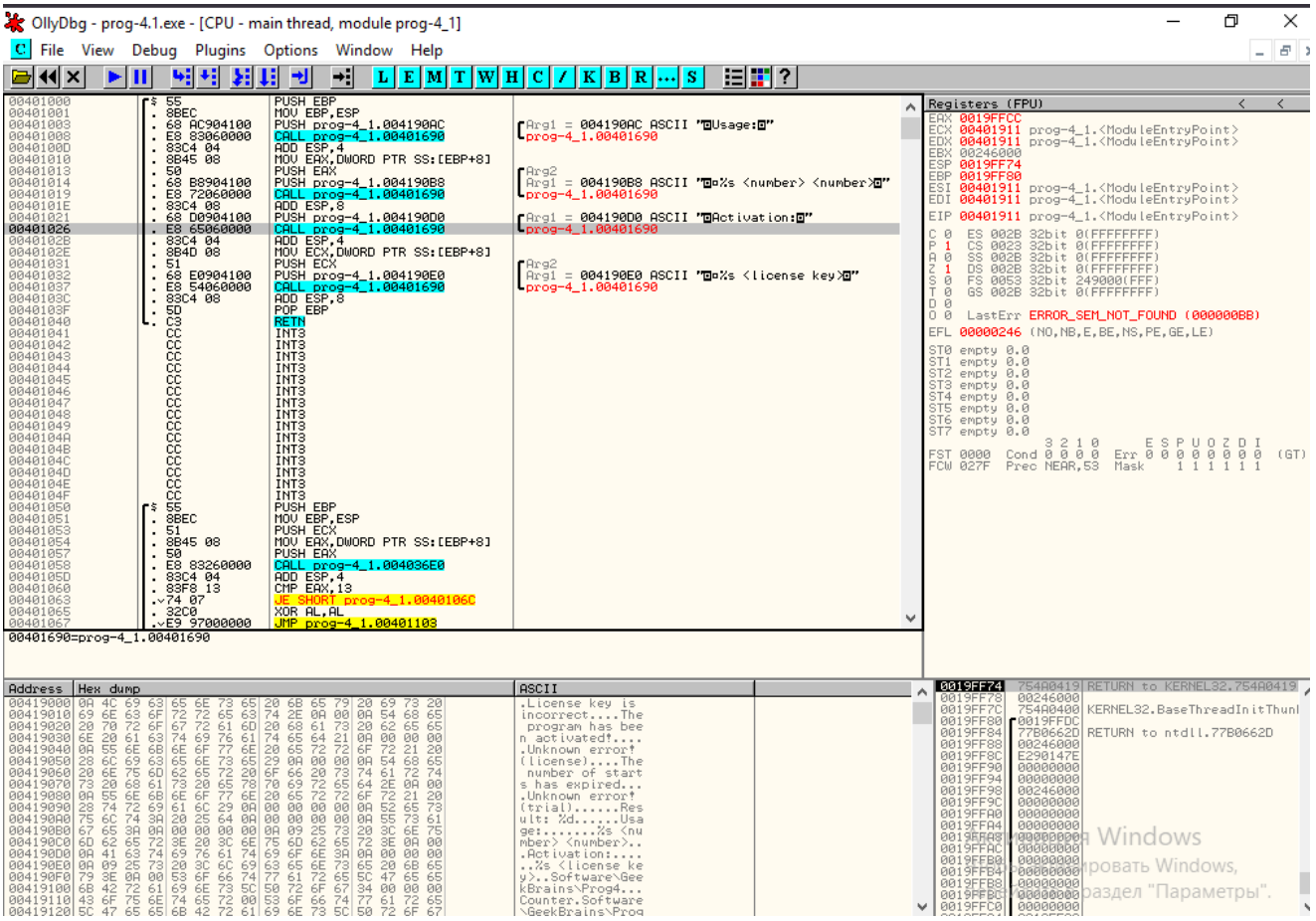




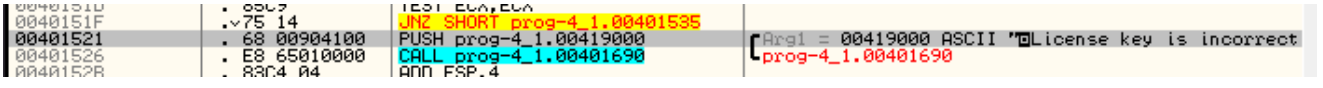
2. Создание ключа:

Запускаем в CMD ***prog-4.1.exe***

```
C:\Users\Администратор\Documents\GB>prog-4.1.exe

Usage:

        prog-4.1.exe <number> <number>

Activation:

        prog-4.1.exe <license key>

C:\Users\Администратор\Documents\GB>prog-4.1.exe 123123

License key is incorrect.
```



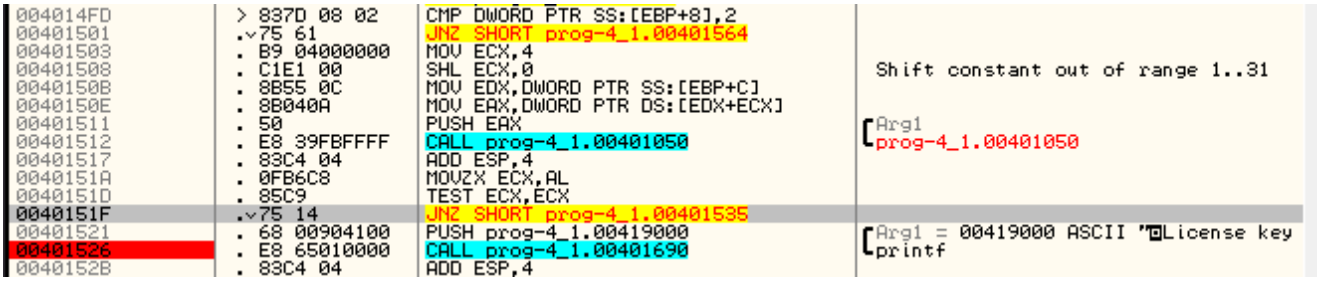
Находим строку в одной из функций. Это строка передается в качестве единственного аргумента в функцию `0x00401690`



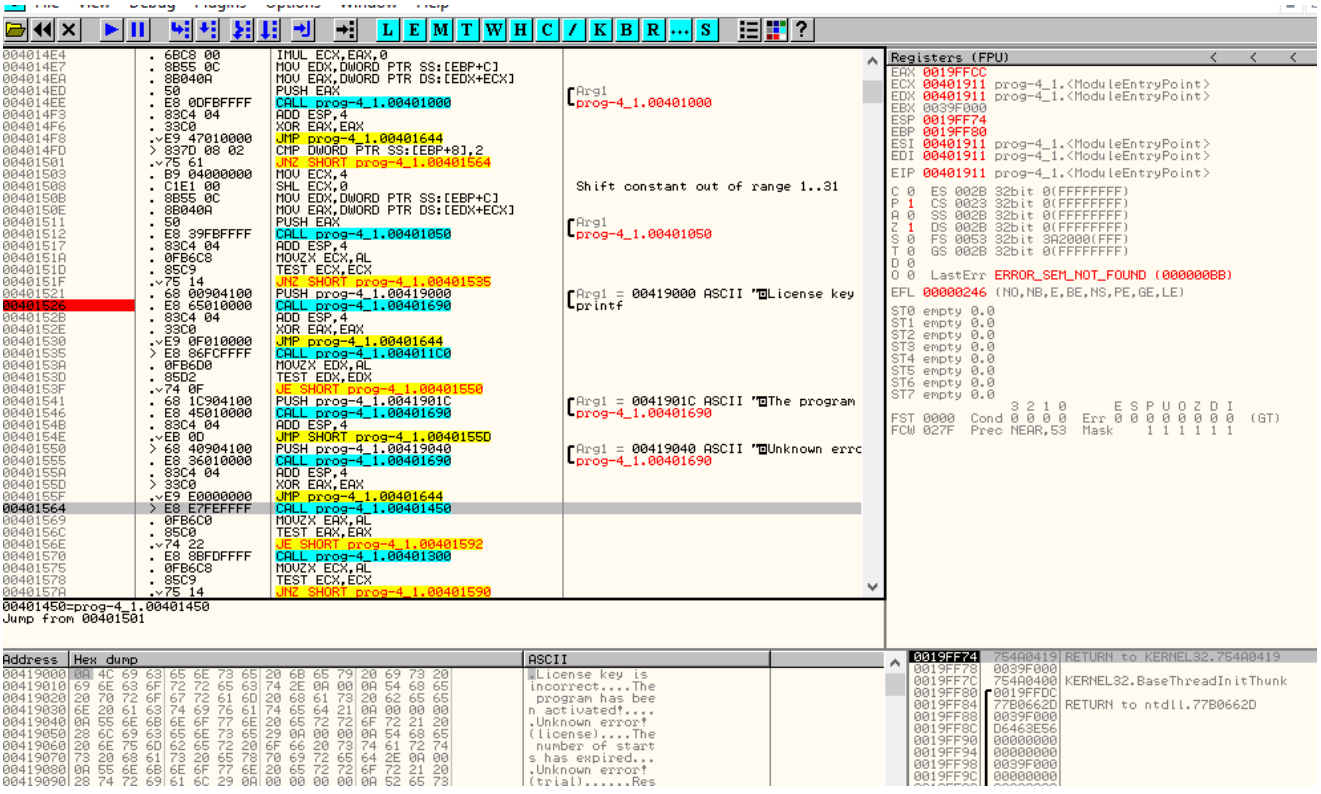
Очень похоже на то, что функция 0x00401690 - это функция printf. Можем поставим точку
останова перед вызовом этой функции и запустить программу с одним параметром.

Мы остановились перед вызовом функции. Видим, что пока в командной строке не было
записей.

Обратимся к коду, который идет перед тем, как будет напечатана строка “License key is incorrect.”, видим вызов функции 0x00401050 и условный переход. Видим, что переход
JNZ указывает на код, где должна быть напечатана нужная нам строка.



Давайте разберемся, от чего зависит результат условия. После вызова функции происходит запись значения из регистра AL в регистр ECX. Далее мы видим оператор TEST,
который выполняет проверка на нулевое значение в регистре ECX. И прыжок будет, если
результат будет не 0. Мы выяснили, что решение о корректности ключа принимает функция 0x00401050. Давайте посмотрим на содержимое этой функции. Кликаем правой кнопкой на инструкции CALL и выбираем Follow.



Поставим точку останова перед инструкцией MOV и перезапустим программу.
Выполняем инструкцию MOV и видим, что адрес в регистре EAX указывает на ключ, который мы передали при запуске программы.
Вызываем эту функцию и видим, что в регистре EAX появилось значение 6. Это значение
похоже на длину нашей строки. Давайте изменим длину ключа (blabla1) и перезапустим
программу.
Видим, что теперь в регистре EAX значение 7, значит функция с адресом 0x004036E0 это
функция strlen. Добавим комментарий.
Далее мы видим, что длина сравнивается со значением 0x13 (19). Мы эту проверку не проходим и попадаем в конец функции и функция завершается со значение 0 в регистре EAX. А мы помним, что нам необходимо добиться того, чтобы функция вернула ненулевое значение.

Ставим ключ с длиной в 19 символов и перезапускаем программу. Мы прошли первую проверку, теперь мы знаем, что ключ должен состоять из 19 символов.
Проверяем ключ в командной строке.

```
prog-4.1.exe 1234567890123456789
License key is incorrect.
```

Ключ всё ещё неправильный. Продолжаем анализ функции.

Далее в регистр ECX помещается значение 1, затем за счет сдвига битов влево на 2 позиции мы получаем значение 4 в регистре ECX. Затем в регистр EDX попадает указатель на ключ. Далее в регистр EAX попадает значение, который находится в строке на 4-й позиции. Это байт 35, который по ASCII таблице равен 5.

И происходит сравнение с байтом 0x2D. И мы попадаем в конец функции с нулевым значением в регистре EAX. Делаем выводы, что 4-й байт должен быть 0x2D. По таблице ASCII это знак “-”.
Изменяем ключ (1234-67890123456789) и перезапускаем программу.

Мы прошли проверку и далее видим еще несколько таких же проверок. На 9-й и 14-й байты. Проверяем ключ в командной строке.

```
prog-4.1.exe 1234-6789-1234-6789
License key is incorrect.
```

Ключ всё ещё неправильный. Продолжаем анализ функции.

Изменяем ключ (1234-6789-1234-6789) и перезапускаем программу.
Далее мы видим набор инструкций, который очень похож на цикл с предусловием.

Выставляет локальная переменная в 0. Затем сравнивается с 0x14 (20). Это длина нашего ключа.

Далее мы видим условный переход JGE, который прыгает на инструкцию MOV, которая кладет значение 1 в регистр AL и функция завершает свою работу. Это то, что нам нужно.
Получается, нам нужно оставаться в цикле пока счетчик не достигнет 20.

Итак, мы попали на первую итерацию цикла. В регистр ECX мы кладем указатель на ключ.
Затем прибавляем счетчик. Видим, это смещение по строке. Первый байт ключа кладем в
регистр EAX. Прибавляем счетчик. Затем в регистр ECX кладем второй байт ключа. В регистре EDX храним сумму первых двух байтов. В регистр ECX кладем третий байт и снова прибавляем его к значение в регистре EDX. Затем тоже самое с третьим байтом. Далее мы кладем четвертый байт в регистр ECX. И теперь с помощью инструкции LEA мы складываем значение в регистре EDX и ECX и вычитаем число 0xC0. В результате получаем число 0x0A (10) в регистре EDX и выполняем сравнение с 0x0A. Они равны и происходит переход к началу цикла. Получается, мы посчитали сумму первых 4 байт и вычли из суммы 192.

Переходим ко второй фазе цикла. Мы увеличиваем счетчик на 5 и на этот раз складываем
следующие 4 числа в ключе. В сумме вы получили 1E и вышли из цикла со значением 0 в
регистре AL.

Получается, что цикл с каждым проходом считает сумму цифр в каждом блоке, вычитает
192 и сравнивает с 10. Первый блок прошел проверку, а второй уже нет. Нам ничего не мешает сделать второй блок и последующие таким же, как первый.

Проверяем ключ в командной строке.

```
prog-4.1.exe 1234-1234-1234-1234
The program has been activated!
```

ASCII код цифры 1 - это 0x31 (49), цифры 2 - это 0x32 (50) и т.д. Для того, чтобы получить
число, нужно вычесть 48 из каждого байта. Например, 49 - 48 - это 1, 50 - 48 - это 2 и .т.д. Если мы складываем 4 числа в каждом блоке, то в конце нужно вычесть 4 раза по

48. А это и есть 192, которые мы видели в коде Давайте составим другой ключ, но с той же суммой в блоке.

```
prog-4.1.exe 0334-1234-1234-1234
The program has been activated!
```

- Сумма цифр в каждом блоке должна быть равно 10.

2. Перенаправление сетевых запросов:
   Запустить программу prog-4.4.1, убедиться в том, что она работает правильно.

https://1.eu.dl.wireshark.org/win64/all-versions/

Запускаем сниффер. Выбираем сетевой интерфейс, через который мы выходим в сеть Интернет. Видим, что через сетевой интерфейс проходит очень много сетевых пакетов. Сейчас хорошо бы добавить фильтр, но мы пока ничего не знает о протоколе взаимодействия. TCP это или UDP. Используется ли доменное имя или обращение происходит сразу по IP-адресу. Если программа взаимодействует с удаленным сервером в сети Интернет, то у нее точно в коде или каком-нибудь конфиге должен быть указан IP-адрес или доменное имя удаленного сервера, иначе она не будет знать, как с ним связаться. В нашем случае программа prog-4.4.1.exe состоит из одного файла. Давайте с помощью утилиты Strings получим все строки, которые есть в бинарном файле и попробуем найти что-нибудь похожее на IP-адрес или доменное имя.
Рекомендации по установке утилиты Strings.

Ссылка: https://docs.microsoft.com/en-us/sysinternals/downloads/strings
Запускаем утилиту Strings.
strings64.exe prog-4.4.1.exe > out.txt

Открываем файл out.txt и видим более 2 тыс. строк. Это очень много для того, чтобы искать в ручном режиме. Давайте сократим область поиска.
Если доменное имя есть в составе программы, то оно находится в секции данных.
Запускаем утилиту Puppy. Открываем файл prog-4.4.1.exe. Выбираем Section Headers. Видим, что секция данных в файле начинается со смещения 0x17a00.
Утилита Strings принимает смещение только в десятичной системе счисления, поэтому открываем калькулятор и переводим число из шестнадцатиричной в десятичную. Получаем число 96768. Запускаем снова утилиту Strings.
strings64.exe -f 96768 prog-4.4.1.exe > out.txt

Открываем файл out.txt и видим всего 42 строки. И почти сразу находим доменное имя check-key.example.com. Это говорит о том, что программа, скорее всего, обращается к серверу по доменному имени. Значит перед тем, как отправить запрос на проверку ключа, происходит разрешение доменного имени для получения IP-адреса. А для

разрешения доменных имен используется протокол DNS.
Возвращаемся к снифферу и добавляем фильтр по протоколу DNS.

dns:
Давайте еще раз попробуем активировать нашу программу и убедимся в том, что действительно доменное имя check-key.example.com используется программой при активации.
prog-4.4.1.exe blabla

В сетевом трафике мы быстро находим DNS-запрос. В ответе мы видим, что этому доменному имени соответствует IP-адрес 192.168.1.8. Давайте посмотрим, какие сетевые пакеты были отправлены по этому IP-адресу.
dns || ip.dst == 192.168.1.5

Мы видим, что было установлено TCP-соединение с сервером и в одном из пакетов отправлен ключ в открытом виде - “blabla”. Обратите внимание, что порт назначения 1337, это нам понадобится позже.
В ответ была отправлена строка “false” и соединение завершилось. Мы видим, что никаких механизмов контроля целостности сообщений не используется. Мы знаем, что в случае неправильного ключа сервер отвечает строкой “false”, но при этом не знаем, что он отвечает, когда ключ правильный.

Давайте откроем программу prog-4.4.1.exe под отладчиком и найдем код, который обрабатывает ответы от сервера.

Переходим в секцию кода и пробуем найти код, который ссылается на строку “License key
is incorrect”. Поставим точку останова перед условным переходом и перезапустим программу в режиме активации.

Функция 0x00401050 вернула нулевое значение через регистр EAX. Давайте перейдем внутрь этой функции, поставим точку останова и перезапустим программу.
Видим, что внутри функции происходит отправка сетевых пакетов, далее мы получаем ответ от сервера и выполняем сравнение двух строк с помощью функции strcmp. Сравнение ответа происходит со строкой “true”. А если мы указываем при активации программы ключ “blabla”, то мы получаем от сервера строку “false”.
Наша задача сделать так, чтобы сервер прислал строку “true”.
Давайте выполним перенаправление запроса на наш сервер.

Специального для этой цели была разработана программа prog-4-4.2, которая умеет отвечать на все запросы строкой “true”.
Запускаем наш сервер:
prog-4.4.2.exe

Итак, сейчас сервер работает локально на нашем компьютере, теперь нам надо перенаправить запрос на наш сервер. Это можно очень легко сделать, если добавить следующую запись в файл `C:\Windows\System32\hosts`
127.0.0.1 check-key.example.com

Проверим, что теперь доменное имя check-key.example.com разрешается IP-адресом 127.0.0.1.

Открываем командную строку
ping check-key.example.com

Видим, что теперь доменному имени соответствует IP-адрес 127.0.0.1.
Давайте попробуем активировать программу.
prog-4.4.1.exe blabla

The program has been activated
Откроем консоль, где был запущен сервер. Видим, что он принял запрос и ответил строкой “true”.

Открываем сниффер и видим, что новых запросов на настоящие сервер не выполнялось.

4. Патчинг

Откроем программу prog-4.1 под отладчиком и перейдем к функции, где программа печатает строку “License key is incorrect.”. Зададим в качестве аргумента произвольный ключ (blabla).
Поставим точку останова сразу после вызова функции проверки ключа.

Мы видим, что далее проверяется значение, которое вернула функция. Значение кладется в регистр ECX, происходит проверка на ноль и если не ноль, то проверка успешно
пройдена.

Какие есть варианты патчинга?

1. Сделать так, чтобы функция возвращала всегда 1.
2. Сделать так, чтобы в регистр ECX всегда попадала 1 перед проверкой.
3. Изменить условный переход JNZ на безусловный JMP.

Все эти варианты должны работать. Но самый простой вариант, как можно активировать
программу, перед инструкцией JNZ изменить состояние флага ZF. Давайте сделаем это.
Доходим до инструкции JNZ. Кликаем правой кнопкой на флаге ZF - Reset. И видим в консоли надпись - “The program has been activated!”.

Проверяем работу программы в командной строке.
prog-4.1.exe 1 2
Result: 3

Соответствующие изменения были сделаны в реестре и программа считается активированной.

То, что мы сейчас сделали нельзя сохранить в файле. Для того, чтобы изменения можно
было сохранить, нужно изменять инструкции в коде программы.
Давайте удалим записи с реестра, которые сделала программа и восстановим первоначальное состояние.

Открывает regedit и удаляем ветку (`HKEY_CURRENT_USER\Software\GeekBrains\Prog4`).
Мы видим инструкцию MOVZX ECX, AL которая загружает значение в регистре AL в регистр ECX. Она занимает 3 байта. Во время патчинг нельзя менять количество байт, иначе поедут смещения и мы превратим программу в кирпич.

Итак, у нас есть три байта и нам нужно записать значение 1 в регистр ECX. Сразу напрашивается инструкция MOV ECX, 1.
Давайте найдем похожую инструкцию в коде нашей программы и посмотри ее опкод.
Ctrl + F (MOV ECX, 1) - ничего не найдено. Такой инструкции в коде программы нет.
Давайте попробуем найти не с единицей, а с двойкой.
Ctrl + F (MOV ECX, 2). Такая инструкция есть. Ее опкод b9 02000000 и такая инструкция
нам не подойдет, так как она занимает 5 байт. Первый байт это инструкция и 4 байта это единица.
Возвращаемся к нашему коду. Кликаем правой кнопкой - Go to - Origin.
В такие моменты нужно проявить фантазию и придумать, как за 3 байта получить значение 1 в регистре ECX.
Я придумал следующее. Первая инструкция XOR ECX, ECX, ее опкод 33C9 и составляет 2
байта, а вторая это инструкция INC ECX, ее опкод 41 и составляет 1 байт. В сумме это 3 байта и нам это подходит.

Давайте проверим.
Кликаем правой кнопкой на инструкции Binary - Edit (33C941).
Видим, что код изменился. Давайте выполним эти инструкции. Мы прошли проверку ключа.
Это вариант уже может быть сохранен на диск.
Кликаем правой кнопкой на любое место в коде Copy to executable - All modifications - Copy
All. Появляется новое окно, кликаем на коде правой кнопкой и Save file. Указываем имя новой программы prog-4.1-any-key.exe.

Проверяем работу программы в командной строке.
prog-4.1-any-key.exe qwerty
The program has been activated!

Отлично! Программа теперь активируется с любым ключом.

А теперь давайте усложним задачу и сделаем так, чтобы программа вообще не требовала
активации.

Возвращаемся в отладчик. Откатываем изменения.

Окно Patches - Restore Original Code.

Далее по коду мы видим строку - “The number of starts has expired.”, которая оповещает нас о том, что количество запусков программы исчерпано. Это происходит тогда когда, значение параметра Counter равно 0.

Будет ли напечатана эта строки или нет завист от функции 0x00401300, которая, видимо,
проверяет счетчик.

Мы точно не знаем, что делает эта функция, поэтому просто убрать вызов этой функции
будет неправильно.

Мы пойдем по-другому пути. Сразу после вызова функции мы видим условный переход
JNZ. Именно от него зависит, попадем мы на код, которая завершает работу программы или нет.

Нам важно, чтобы этот переход осуществлялся всегда. Заменим условный переход JNZ
на безусловный JMP.

Кликаем правой кнопкой на инструкции Binary - Edit (EB14).

Видим, что код изменился.

Давайте изменим состояние счетчика в реестре на ноль, чтобы спровоцировать состояние, когда количество попыток запуска закончится.
Counter - 0.

Давайте выполним эти инструкции. Видим в консоли, что программа успешно отработала,
несмотря на то, что счетчик равен 0. Мы прошли проверку ключа.
Это вариант уже может быть сохранен на диск.

Кликаем правой кнопкой на любое место в коде Copy to executable - All modifications - Copy
All. Появляется новое окно, кликаем на коде правой кнопкой и Save file. Указываем имя новой программы prog-4.1-cracked.exe.

Проверяем работу программы в командной строке.
prog-4.1-cracked.exe 1 2
Result: 3

Как мы видим, программа работает несмотря на то, что счетчик равен нулю.

```
// Исходный код программы prog-4.1.c

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#pragma comment(lib, "Advapi32.lib")

void usage(char *);
bool check_key(char *);
int sum(int, int);
bool check_numbers(char *, char *);
bool license(void);
bool trial(void);
bool check_trial(void);
bool isRegistered(void);

int main(int argc, char* argv[]) {

        if (argc == 1 || argc > 3) {
                usage(argv[0]);
                return 0;
        }

        if (argc == 2) {
                if (!check_key(argv[1])) {
                        printf("\nLicense key is incorrect.\n");
                        return 0;
                }

                if (license()) {
                        printf("\nThe program has been activated!\n");
                } else {
                        printf("\nUnknown error! (license)\n");
                }
                return 0;
        }
```

```
        if (isRegistered()) {
                if (!check_trial()) {
                        printf("\nThe number of starts has expired.\n");
                        return 0;
                }
        } else {
                if (!trial()) {
                        printf("\nUnknown error! (trial)\n");
                        return 0;
                }
        }

        if (!check_numbers(argv[1], argv[2])) {
                usage(argv[0]);
                return 0;
        }

        printf("\nResult: %d\n", sum(atoi(argv[1]), atoi(argv[2])));
}

void usage(char *program_name) {
        printf("\nUsage:\n");
        printf("\n\t%s <number> <number>\n", program_name);
        printf("\nActivation:\n");
        printf("\n\t%s <license key>\n", program_name);
}

bool isRegistered(void) {
        HKEY hkey;
        LPCTSTR PATH = TEXT("Software\\GeekBrains\\Prog4");
        LPCTSTR NAME = TEXT("Counter");

        LONG nError = RegOpenKeyEx(HKEY_CURRENT_USER, PATH, 0, KEY_READ,
&hkey);
        if (nError != ERROR_SUCCESS) {
                return false;
    }

        nError = RegGetValue(hkey, NULL, NAME, RRF_RT_REG_SZ, 0, NULL,
NULL);
        if (nError != ERROR_SUCCESS) {
                RegCloseKey(hkey);
                return false;
    }

        RegCloseKey(hkey);

        return true;
}
```

```
bool check_key(char* key) {

        if (strlen(key) != 19) {
                return false;
        }

        if (key[4] != '-' || key[9] != '-' || key[14] != '-') {
                return false;
        }

        for (int block_offset = 0; block_offset < 20; block_offset += 5) {
                if ((key[block_offset] + key[block_offset+1] +
key[block_offset+2] + key[block_offset+3] - 192) != 10) {
                        return false;
                }
        }

        return true;
}

int sum(int a, int b) {
        return a + b;
}

bool check_numbers(char *num1, char *num2) {
        for (int i = 0; i < strlen(num1); i++) {
                if (num1[i] < 48 || num1[i] > 57) {
                        return false;
                }
        }

        for (int i = 0; i < strlen(num2); i++) {
                if (num2[i] < 48 || num2[i] > 57) {
                        return false;
                }
        }

        return true;
}
bool license(void) {

        HKEY hkey;
        LPCTSTR PATH = TEXT("Software\\GeekBrains\\Prog4");
        LPCTSTR NAME = TEXT("Counter");

        char value[] = "10000";

        LONG nError = RegCreateKeyEx(HKEY_CURRENT_USER, PATH, 0, NULL,
REG_OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hkey, NULL);
```

```
        if (nError != ERROR_SUCCESS) {
                return false;
    }

        nError = RegSetValueExA(hkey, NAME, 0, REG_SZ, value,
sizeof(value));
        if (nError != ERROR_SUCCESS) {
                RegCloseKey(hkey);
                return false;
    }

        RegCloseKey(hkey);

        return true;
}

bool trial(void) {

        HKEY hkey;
        LPCTSTR PATH = TEXT("Software\\GeekBrains\\Prog4");
        LPCTSTR NAME = TEXT("Counter");

        char value[] = "9";

        LONG nError = RegCreateKeyEx(HKEY_CURRENT_USER, PATH, 0, NULL,
REG_OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hkey, NULL);
        if (nError != ERROR_SUCCESS) {
                return false;
    }

        nError = RegSetValueExA(hkey, NAME, 0, REG_SZ, value,
sizeof(value));
        if (nError != ERROR_SUCCESS) {
                return false;
    }

        printf("\nTrial Period: %d of 10\n", atoi(value));

        RegCloseKey(hkey);

        return true;
}

bool check_trial(void) {

        HKEY hkey;
        LPCTSTR PATH = TEXT("Software\\GeekBrains\\Prog4");
        LPCTSTR NAME = TEXT("Counter");

        char value[64];
```

```
    DWORD value_size = 64;

        LONG nError = RegOpenKeyEx(HKEY_CURRENT_USER, PATH, 0, KEY_READ,
&hkey);
        if (nError != ERROR_SUCCESS) {
                return false;
    }

        nError = RegGetValue(hkey, NULL, NAME, RRF_RT_REG_SZ, 0, value,
&value_size);
        if (nError != ERROR_SUCCESS) {
                RegCloseKey(hkey);
                return false;
    }

        int tmp = atoi(value);

        if (tmp == 0) {
                RegCloseKey(hkey);
                return false;
        }

        itoa(--tmp, value, 10);

        nError = RegOpenKeyEx(HKEY_CURRENT_USER, PATH, 0, KEY_WRITE,
&hkey);
        if (nError != ERROR_SUCCESS) {
                RegCloseKey(hkey);
                return false;
    }

        nError = RegSetValueExA(hkey, NAME, 0, REG_SZ, value,
sizeof(value));
        if (nError != ERROR_SUCCESS) {
                RegCloseKey(hkey);
                return false;
    }

        RegCloseKey(hkey);

        if (tmp < 10) {
                printf("\nTrial Period: %d of 10\n", atoi(value));
        }

        return true;
}
```

```
Компиляция:
> cl prog-4.1.c -Od /Gs- /GS- /link /DYNAMICBASE:NO /NXCOMPAT:NO
Запуск:
> prog-4.1.exe
```

Выполнил: AndreiM