## Максимальное количество баллов за олимпиаду — 100

**Задание 1.**

**Следы атаки**

Во время анализа журналов веб-приложения и сервера специалист обнаружил фрагменты запросов и события, характерные для разных видов атак. Установите соответствие между фрагментами или событиями и их описаниями.

**Ответ:**

| Фрагмент или событие | Описание атаки |
|---|---|
| `id=1 OR 1=1` | Изменение логики SQL-запроса с помощью добавленного условия (SQL Injection). |
| `../../etc/passwd` | Попытка выйти за пределы разрешённого каталога с помощью переходов к родительским каталогам (Path Traversal). |
| `127.0.0.1; whoami` | Добавление команды операционной системы в значение параметра, которое обрабатывает сервер (Command Injection). |
| `GET /backup/site.zip` → `200 OK` | Попытка получить резервную копию сайта из общедоступного каталога (Exposed Backup). |
| Много неудачных попыток входа с одного адреса за короткое время | Последовательный подбор пароля с помощью многочисленных попыток авторизации (Brute Force). |

**Критерий оценивания:** за каждую верную пару — 2 балла

**Максимальный балл за задание — 10**

**Решение.**

Условие `OR 1=1` меняет логику SQL-запроса. Последовательность `../` позволяет переходить к родительским каталогам и выходить за пределы разрешённой папки. После разделителя `;` к параметру добавлена команда операционной системы `whoami`. Ответ `200 OK` показывает, что сервер успешно выдал архив резервной копии. Многочисленные неудачные попытки входа с одного адреса соответствуют последовательному подбору пароля.

**Задание 2.**

**Почему сервер недоступен?**

Администратор настраивал четыре сервера с ОС Linux. На каждой из них команда `curl https://example.org` не может получить страницу. Администратор выполнил несколько диагностических команд. Все четыре машины подключены к сети `192.168.10.0/24`. Шлюз в Интернет имеет адрес `192.168.10.1`, а DNS-сервер — `192.168.10.53`. Установите соответствие между серверами и возможными причинами недоступности внешних веб-сервисов.

**Сервер 1**

```
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>
    inet 192.168.10.21/24 scope global eth0
$ ip route
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.21
$ ping -c 1 192.168.10.1
64 bytes from 192.168.10.1: icmp_seq=1 ttl=64 time=0.41 ms
$ ping -c 1 1.1.1.1
ping: connect: Network is unreachable
```

**Сервер 2**

```
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST>
    inet 192.168.10.22/24 scope global eth0
$ ip route
default via 192.168.10.1 dev eth0
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.22
$ ping -c 1 192.168.10.1
ping: connect: Network is unreachable
```

**Сервер 3**

```
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>
```

```
    inet 192.168.10.23/24 scope global eth0
$ ip route
default via 192.168.10.1 dev eth0
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.23
$ ping -c 1 192.168.10.1
64 bytes from 192.168.10.1: icmp_seq=1 ttl=64 time=0.37 ms
$ ping -c 1 1.1.1.1
64 bytes from 1.1.1.1: icmp_seq=1 ttl=56 time=15.3 ms
$ getent hosts example.org
Команда getent ничего не вывела.
```

**Сервер 4**

```
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>
    inet 192.168.20.24/24 scope global eth0
$ ip route
default via 192.168.20.1 dev eth0
192.168.20.0/24 dev eth0 proto kernel scope link src 192.168.20.24
$ ping -c 1 192.168.10.1
From 192.168.20.24 icmp_seq=1 Destination Host Unreachable
```

**Ответ:**

| Сервер | Причина недоступности |
|---|---|
| 1 | Отсутствует маршрут по умолчанию. |
| 2 | Сетевой интерфейс неактивен. |
| 3 | Не работает разрешение доменных имён. |
| 4 | Сетевой интерфейс настроен на неправильную подсеть. |

**Критерий оценивания:** за каждую верную пару — 2.5 балла

**Максимальный балл за задание — 10**

**Решение.**

Для доступа к внешним веб-сервисам сетевой интерфейс должен быть активен, хост должен иметь адрес в нужной подсети, в таблице маршрутизации должен существовать маршрут в Интернет, а служба DNS должна разрешать имена. Подсеть `192.168.10.0/24` имеет маску `255.255.255.0`; шлюз — `192.168.10.1`. Команда `ip addr show eth0` показывает состояние интерфейса и его адрес, `ip route` — маршруты. Значения `UP` и `LOWER_UP` означают, что интерфейс работает.

На сервере 1 интерфейс активен, адрес из нужной подсети назначен и шлюз отвечает, но в выводе `ip route` нет маршрута по умолчанию. Ошибка `Network is unreachable` при обращении к `1.1.1.1` подтверждает этот вывод.

На сервере 2 отсутствуют признаки `UP` и `LOWER_UP`: интерфейс неактивен. Сервер 3 успешно обращается к внешнему IP-адресу `1.1.1.1`, но команда `getent hosts example.org` ничего не выводит, то есть имена не разрешаются. На сервере 4 назначен адрес `192.168.20.24/24` вместо адреса в подсети `192.168.10.0/24`.

**Задание 3.**

**Зелёный банк**

Файлы: `2_1011_01-linux`, `2_1011_01-windows.exe`.

Один зелёный банк попросил нас протестировать его веб-клиент на безопасность, но все наши безопасники разом ушли в отпуск. Может, вы попробуете? Войдите в личный кабинет пользователя `useruser` и получите данные его банковских счетов.

Формат ответа: `vsosh{text}`.

**Ответ:** `vsosh{n3tw0rk_1nt3r4ct10n}`

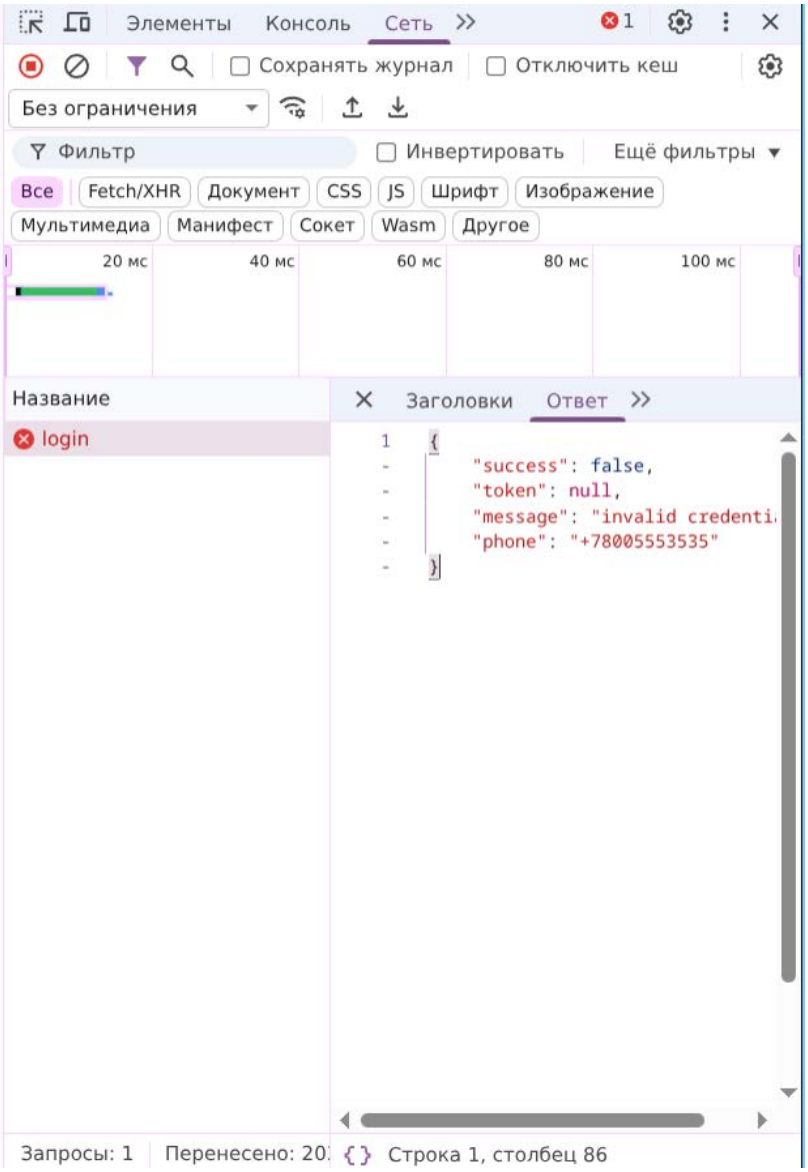
**Критерий оценивания:** точное совпадение ответа — 10 баллов

**Максимальный балл за задание — 10**

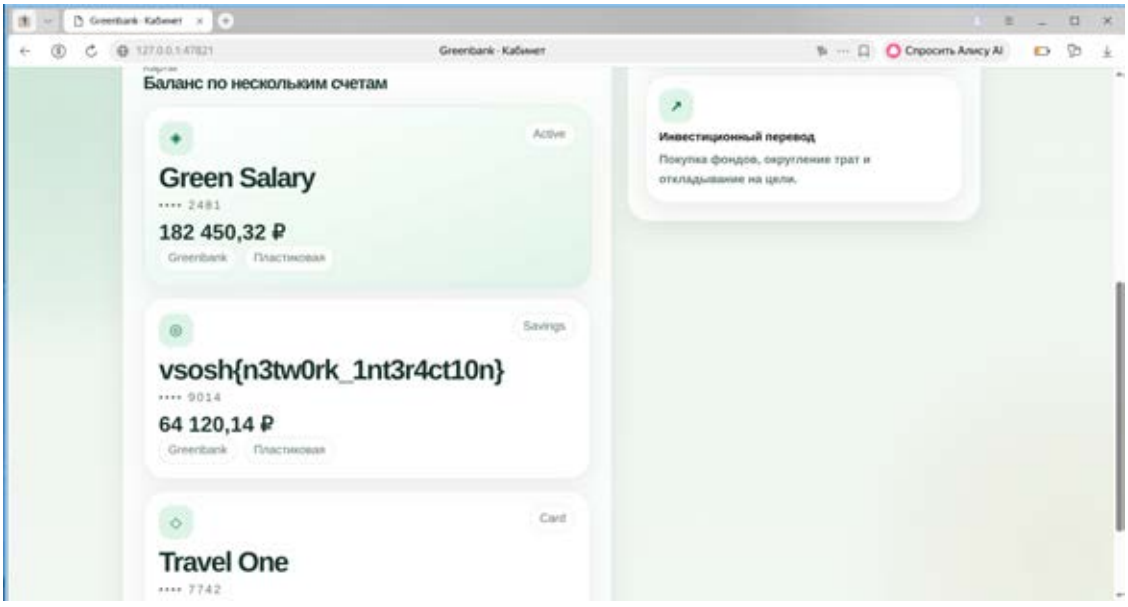
**Решение.**

Запускаем файл для своей операционной системы. Открывается страница Greenbank. Форма входа требует имя пользователя и пароль, но пароль в условии не указан. На странице сброса пароля требуется номер телефона для получения кода по SMS, который тоже не дан. Исследуем запросы при входе через вкладку «Сеть» в инструментах разработчика браузера.

Вводим `useruser` и произвольный пароль. При выборе запроса `/api/auth/login` во вкладке «Ответ» видим: даже при ошибке аутентификации сервер возвращает номер телефона, привязанный к аккаунту.




Переходим к сбросу пароля и вводим полученный номер. В ответе на запрос кнопки «Получить код» из-за ошибки сервера оказывается сам код. Вводим его в форму, устанавливаем новый пароль и заходим в кабинет `useruser`. Название одного из счетов содержит флаг.




**Задание 4.**

**Почти одинаковые ключи**

Рекомендуемые инструменты: VS Code, Python, библиотека pycryptodome, библиотека sympy. Файлы: `2_1011_04-код.py`, `2_1011_04-текст.txt`.

Разработчик реализовал шифрование секретных сообщений с использованием RSA. Фрагмент генерации ключей и шифрования на языке Python находится в файле `2_1011_04-код.py`. Разработчик утверждает, что восстановить сообщение невозможно без знания $p$ и $q$, но решил помочь нам убедиться в этом, оставив в качестве подсказки файл `2_1011_04-текст.txt`. Восстановите исходное сообщение.

Формат ответа: `vsosh{text}`.

**Ответ:** `vsosh{cl0s3_pr1m3s}`

**Критерий оценивания:** точное совпадение ответа — 10 баллов

**Максимальный балл за задание — 10**

**Решение.**

В файлах даны открытый ключ $n$, $e$, шифротекст $c$ и фрагмент генератора простых чисел:

```
p = getPrime(96)
q = nextprime(p + 10**17)
```

В RSA $n = pq$. Число $q$ выбирается близко к $p$, поэтому для разложения $n$ подходит метод Ферма:

$$n = a^2 - b^2 = (a - b)(a + b), \qquad a = \frac{p+q}{2}, \quad b = \frac{q-p}{2}.$$

Начинаем с $a = \lceil \sqrt{n} \rceil$, увеличиваем $a$ до тех пор, пока $a^2 - n$ не станет полным квадратом $b^2$. Тогда $p = a - b$ и $q = a + b$. Это можно сделать так:

```
import math

n = 15693732479202231915380417392395821536691283045220865822 09
a = math.isqrt(n)
if a * a < n:
    a += 1
while True:
    b2 = a * a - n
    b = math.isqrt(b2)
    if b * b == b2:
        p = a - b
        q = a + b
        break
    a += 1
print(p, q)
```

Получаем $p = 396153158250222922535609875 27$ и $q = 396153158251222922535609877 67$. Находим $\varphi(n) = (p-1)(q-1)$ и закрытую экспоненту $d \equiv e^{-1} \pmod{\varphi(n)}$. Затем вычисляем $m = c^d \bmod n$ и переводим число обратно в байты:

```
e = 65537
c = 14970984586229452543687743050140672697852815206198748863
phi = (p - 1) * (q - 1)
d = pow(e, -1, phi)
m = pow(c, d, n)
message = m.to_bytes((m.bit_length() + 7) // 8, "big")
print(message.decode())
```

Вывод программы: `vsosh{cl0s3_pr1m3s}`.

**Задание 5.**

**Удалённый документ**

Рекомендуемые инструменты: HxD HEX editor, Python. Файл: `2_1011_05_архив.zip`.

Администратор обнаружил архив резервной копии. По журналу системы известно, что ранее архив содержал файл `secret.txt`, однако сейчас стандартные программы просмотра архивов его не отображают. Сам архив при этом частично открывается, а некоторые находящиеся в нём файлы доступны. Прочитайте содержимое этого секретного файла, которое представляет собой осмысленный текст.

Формат ответа: `vsosh{text}`.

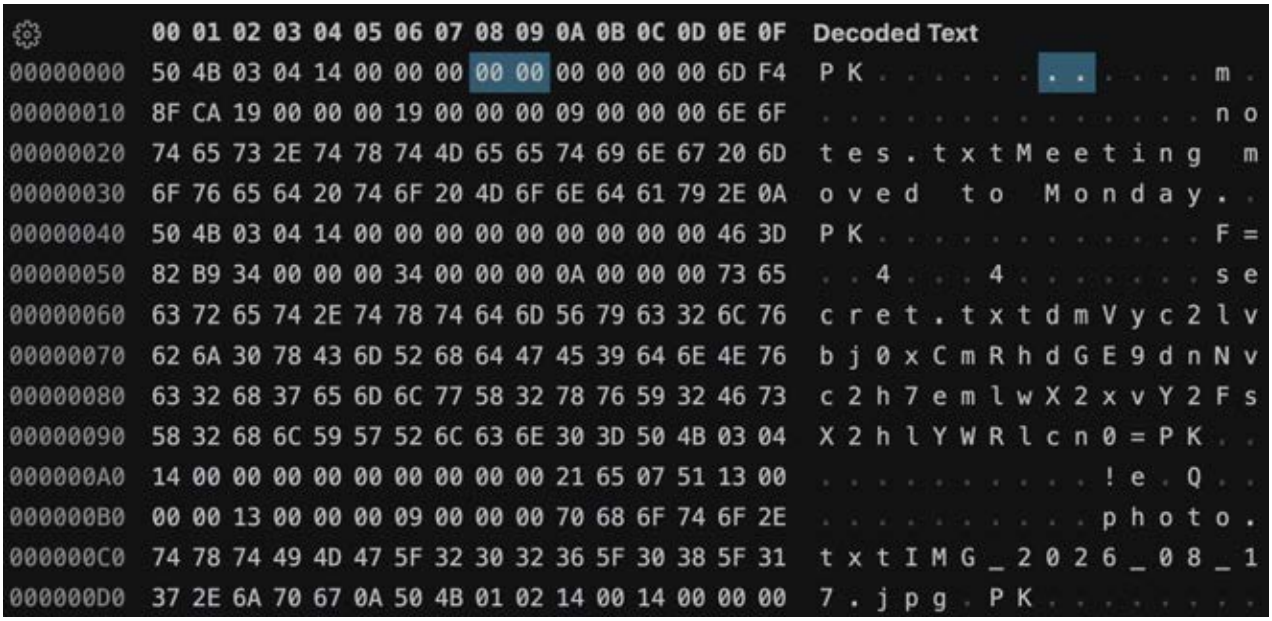
**Ответ:** `vsosh{zip_local_header}`

**Критерий оценивания:** точное совпадение ответа — 10 баллов

**Максимальный балл за задание — 10**

**Решение.**

Обычный архиватор показывает только файлы `notes.txt` и `photo.txt`. Открываем архив в hex-редакторе и ищем сигнатуры локальных заголовков ZIP: `50 4B 03 04` (`PK`). В одном из заголовков обнаруживаем имя `secret.txt`: данные файла физически остались в архиве, хотя он не показывается обычным просмотрщиком.




Поле метода сжатия содержит `00 00`, то есть содержимое хранится без сжатия. После имени файла читаем строку `dmVyc2lvbj0xCmRhdGE9dnNvc2h7emlwX2xvY2FsX2hlYWRlcn0=`. Она имеет вид Base64. Декодируем её, например с помощью Python:

```
import base64
print(base64.b64decode(
    "dmVyc2lvbj0xCmRhdGE9dnNvc2h7emlwX2xvY2FsX2hlYWRlcn0="
).decode())
```

В результате видим содержимое `secret.txt`, включающее флаг `vsosh{zip_local_header}`.

```
>>> base64.b64decode("dmVyc2lvbj0xCmRhdGE9dnNvc2h7emlwX2xvY2FsX2hlYWRlcn0=").decode()
'version=1\ndata=vsosh{zip_local_header}'
```

**Задание 6.**

**Файл, которого нет**

Рекомендуемые инструменты: Exiftool, Wireshark, Python. Файл: `2_1011_06_трафик.pcap`.

В ходе расследования подозрительной активности был получен дамп сетевого трафика. Аналитики установили, что перед инцидентом пользователь передавал изображение через внутренний веб-сервис. Ваша задача — найти информацию, скрытую в изображении.

Формат ответа: `vsosh{text}`.

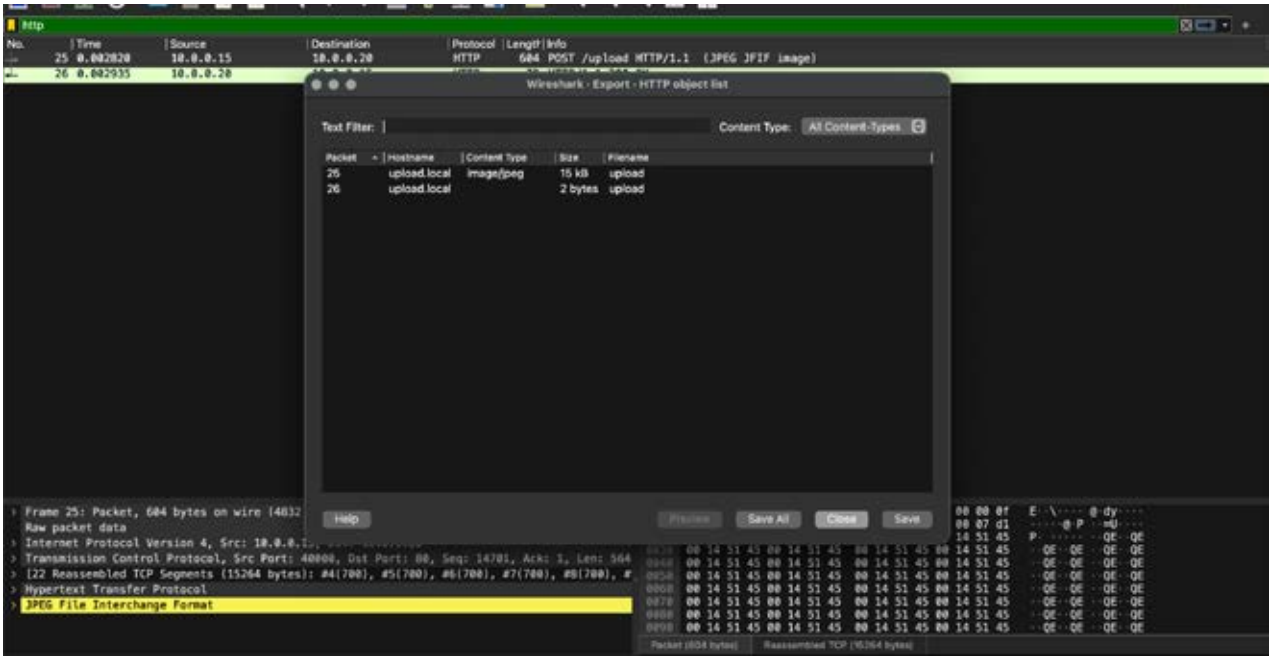
**Ответ:** `vsosh{metadata_is_also_evidence}`

**Критерий оценивания:** точное совпадение ответа — 10 баллов

**Максимальный балл за задание — 10**

**Решение.**

Открываем дамп в Wireshark и отфильтровываем HTTP-трафик. Через HTTP передавалось изображение JPEG. В меню «Файл» выбираем «Экспорт объектов» → «HTTP» и экспортируем объект из пакета № 25.




Если экспортированный файл не имеет расширения, добавляем `.jpeg`. На самом изображении нужного текста нет. Просматриваем метаданные с помощью `exiftool`: несколько текстовых полей содержат фрагменты строки Base64. Последний фрагмент определяется знаком `=`, которым дополняют Base64-строки. Располагаем фрагменты в порядке, дающем осмысленный текст, и декодируем их функцией `base64.b64decode(...).decode()`. Искомая информация находится в метаданных изображения.

**Задание 7.**

**Чужой профиль**

Исходный код мини-приложения: `2_1011_07_code.py`.

В школьной лаборатории разработали сервис для хранения профилей пользователей. Профиль можно экспортировать в виде строки, а затем импортировать обратно. Разработчик решил использовать встроенный механизм Python `pickle`, поскольку он позволяет сохранять объекты Python. Вам предоставлен листинг программы. Разберитесь, как приложение работает с передаваемыми данными, и восстановите пропуски в утверждениях.

**Утверждение 1.** При обращении к `/import` переданные пользователем данные сначала извлекаются из HTTP-запроса, а затем передаются функции

**Ответ:**

- `encode_profile`
- ✓ `decode_profile`
- `get_user`
- `create_user`

**Критерий оценивания:** точное совпадение ответа — 2 балла

**Утверждение 2.** В функции `decode_profile` сначала выполняется

**Ответ:**

- Хеширование с помощью SHA-256
- ✓ Декодирование Base64
- Преобразование данных в JSON
- Проверка имени пользователя

А затем результат передаётся функции `pickle.loads()`.

**Критерий оценивания:** точное совпадение ответа — 2 балла

**Утверждение 3.** Почему использование `pickle.loads()` для данных, полученных от пользователя, опасно?

**Ответ:**

- Pickle автоматически шифрует все загружаемые данные
- ✓ При десериализации специально сформированного объекта могут выполняться заложенные в нём действия
- Base64 позволяет выполнить JavaScript в браузере пользователя
- Pickle разрешает доступ к базе данных только администраторам

**Критерий оценивания:** точное совпадение ответа — 2 балла

**Утверждение 4.** Кодирование профиля с помощью Base64

**Ответ:**

- Полностью защищает данные от изменения

- Шифрует данные секретным ключом
- ✓ Лишь представляет исходные байты в другом виде и само по себе не делает их безопасными
- Запрещает выполнение `pickle.loads()`

**Критерий оценивания:** точное совпадение ответа — 2 балла

**Утверждение 5.** Для хранения передаваемого профиля, состоящего из обычных полей вроде имени, языка и темы оформления, вместо `pickle` целесообразно использовать

**Ответ:**

- ✓ JSON
- `pickle.loads()`
- `eval()`
- `os.system()`

**Критерий оценивания:** точное совпадение ответа — 2 балла

**Максимальный балл за задание — 10**

**Решение.**

Обработчик `/import` извлекает данные профиля из запроса. Затем он вызывает функцию `decode_profile(data)`:

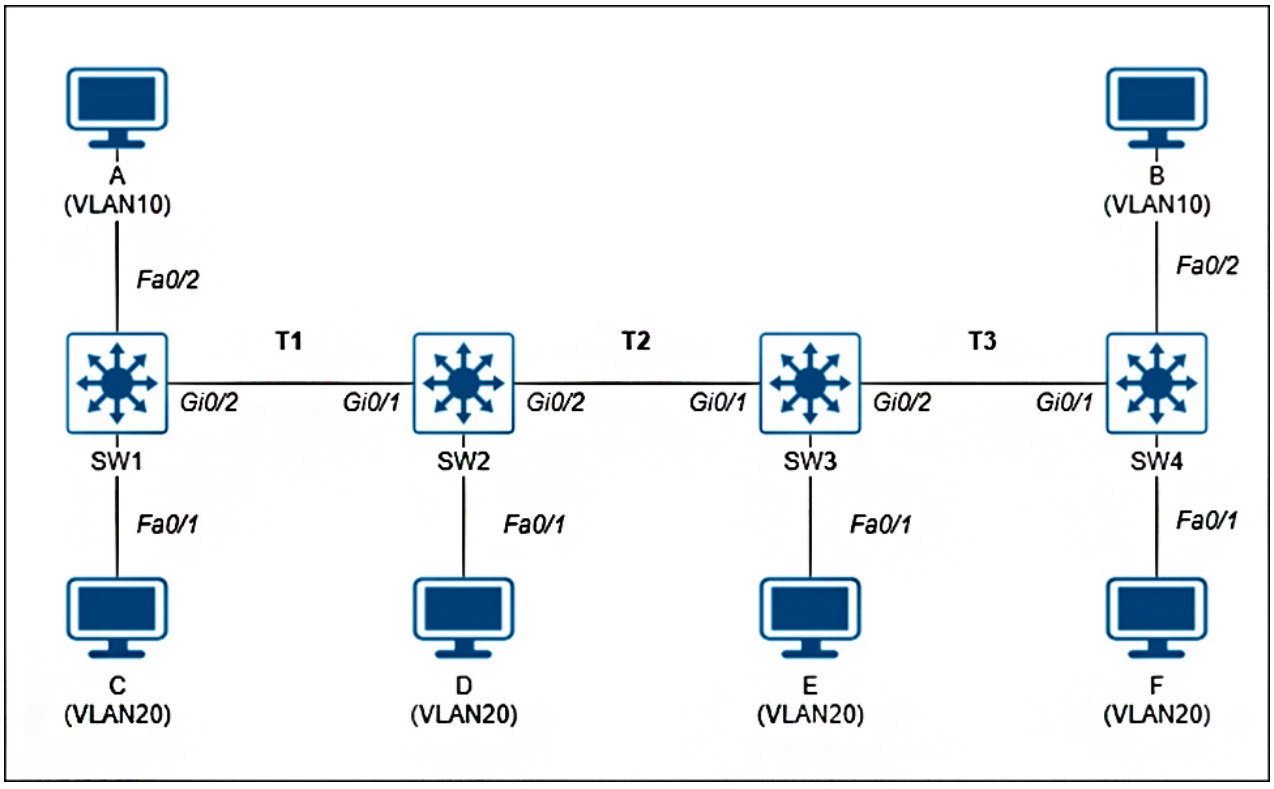
```
def decode_profile(data):
    raw = base64.b64decode(data)
    return pickle.loads(raw)
```

Сначала строка декодируется из Base64, затем вызывается `pickle.loads()`. При десериализации специально сформированного объекта могут выполняться заложенные в нём действия. Base64 лишь меняет представление байтов, поэтому не устраняет эту опасность. Для передачи обычных полей профиля вместо `pickle` целесообразно использовать JSON.

**Задание 8.**

**Где пропал VLAN**

В этой сети устройства разделены на VLAN: логические группы внутри одной физической сети. Устройства одного VLAN могут обмениваться данными через коммутаторы, а устройства разных VLAN изолированы: маршрутизации между ними нет. Компьютеры подключены к access-портам, каждый из которых относится к одному VLAN. Подпись VLAN рядом с компьютером на схеме показывает, к какой группе он должен принадлежать. Коммутаторы соединены транковыми линиями, по которым одновременно передаётся трафик нескольких VLAN. На каждом конце транка задаётся список VLAN, которым разрешено проходить. Если VLAN не разрешён хотя бы на одном конце, его трафик по этой линии не пройдёт. VLAN 10 и 20 созданы на всех коммутаторах, SW1 и SW4 настроены правильно, линии связи и устройства исправны, а остальные параметры не содержат ошибок. После изменения конфигурации часть связей нарушилась. Сопоставьте схему с конфигурациями SW2 и SW3 и определите, какие настройки нужно исправить.




**Фрагменты конфигурации коммутаторов**

| SW2 | SW3 |
|---|---|
| `interface GigabitEthernet0/1`<br>`switchport mode trunk`<br>`switchport trunk allowed vlan 10,20`<br>`interface GigabitEthernet0/2`<br>`switchport mode trunk`<br>`switchport trunk allowed vlan 10,20`<br>`interface FastEthernet0/1`<br>`switchport mode access`<br>`switchport access vlan 10` | `interface GigabitEthernet0/1`<br>`switchport mode trunk`<br>`switchport trunk allowed vlan 10`<br>`interface GigabitEthernet0/2`<br>`switchport mode trunk`<br>`switchport trunk allowed vlan 10,20`<br>`interface FastEthernet0/1`<br>`switchport mode access`<br>`switchport access vlan 20` |

**Результаты проверок связи**

| Проверка | Результат |
|---|---|
| C ↔ E | Нет связи |
| A ↔ B | Связь есть |
| E ↔ F | Связь есть |
| D ↔ F | Нет связи |
| C ↔ D | Нет связи |

**а)** Какие интерфейсы настроены неправильно?

**Ответ:**

- SW2, порт Gi0/1
- ✓ SW3, порт Gi0/1
- ✓ SW2, порт Fa0/1
- SW3, порт Fa0/1
- SW2, порт Gi0/2
- SW3, порт Gi0/2

**Критерий оценивания:** точное совпадение ответа — 5 баллов

**б)** Какой набор изменений полностью восстановит связь внутри требуемых VLAN и сохранит их разделение? Все неуказанные настройки остаются прежними.

**Ответ:**

- Исправить только настройки SW2: на Fa0/1 назначить VLAN 20, остальное не менять
- Изменить только SW3: на Gi0/1 разрешить VLAN 10 и 20, остальное не менять
- Поправить оба транка на SW2: на Gi0/2 разрешить только VLAN 20, а на Fa0/1 назначить VLAN 20
- ✓ Сделать два изменения: на SW2 Fa0/1 назначить VLAN 20 и на SW3 Gi0/1 разрешить VLAN 10 и 20
- Переназначить access-порты: всем устройствам A–F выдать VLAN 10

**Критерий оценивания:** точное совпадение ответа — 5 баллов

**Максимальный балл за задание — 10**

**Решение.**

SW2 Fa0/1 подключён к устройству D. На схеме оно относится к VLAN 20, однако порт назначен в VLAN 10. Поэтому D не обменивается данными с C и F из VLAN 20. На SW3 Gi0/1, конце транка T2, разрешён только VLAN 10. Противоположный порт SW2 Gi0/2 разрешает оба VLAN, но на стороне SW3 трафик VLAN 20 блокируется. Это объясняет отсутствие связи C ↔ E. Для полного исправления на SW2 Fa0/1 задаём `switchport access vlan 20`, а на SW3 Gi0/1 — `switchport trunk allowed vlan 10,20`. VLAN 10 на транке сохраняется, и разделение устройств по VLAN не нарушается.

**Задание 9.**

**Последний переключатель**

Файл: `2_1011_09_code.cpp`.

В лаборатории обнаружили программу управления электронным замком. Программа запрашивает код, и если код правильный, появляется сообщение об успехе. Однако код был утерян. Необходимо исследовать исходный код программы и восстановить правильный код.

Формат ответа: `vsosh{text}`.

**Ответ:** `vsosh{st4t3_m4ch1n3}`

**Критерий оценивания:** точное совпадение ответа — 10 баллов

**Максимальный балл за задание — 10**

**Решение.**

В исходном коде проверка находится в функции `verify`. Массив `table` содержит 20 значений, значит строка должна содержать 20 символов. Начальное состояние — `0x5C`. Для каждого символа программа применяет XOR с текущим

состоянием, затем функцию `step`: XOR с `keys[i]`, сложение с `0x19` и циклический сдвиг вправо на два бита. Полученный байт сравнивается с `table[i]`. После этого состояние обновляется с использованием преобразованного байта, то есть `table[i]`, а не исходного символа.

Обращаем действия функции `step` в обратном порядке: циклический сдвиг влево на два бита, вычитание `0x19`, XOR с ключом. Последним XOR с текущим состоянием восстанавливаем символ. Скрипт с массивами из исходного файла:

```
table = [
    0x21, 0x8D, 0x0B, 0x98, 0x27,
    0x6E, 0x0B, 0x62, 0x01, 0x28,
    0x9F, 0x44, 0xC4, 0xBE, 0x36,
    0xF7, 0x44, 0xA1, 0xA7, 0xB9
]
keys = [
    0x41, 0x13, 0x77, 0x22, 0x58,
    0x05, 0x31, 0x66, 0x14, 0x27,
    0x53, 0x09, 0x6A, 0x18, 0x44,
    0x72, 0x2B, 0x35, 0x5E, 0x11
]
state = 0x5C
result = ""

for i in range(len(table)):
    x = table[i]
    x = ((x << 2) | (x >> 6)) & 0xff
    x = (x - 0x19) & 0xff
    x ^= keys[i]
    x ^= state
    result += chr(x)
    state = (state + table[i] + i) & 0xff

print(result)
print(f"Final state: 0x{state:02X}")
```

Результат: `vsosh{st4t3_m4ch1n3}`, итоговое состояние `0x6D`, как и требует завершающая проверка программы.

**Задание 10.**

**TokenBazaar**

Файлы: `2_1011_02-linux`, `2_1011_02-windows.exe`.

TokenBazaar — подпольная площадка, где торгуют «токенами доступа» (JWT). Авторизация на сайте устроена так же, как аутентификация в реальных JWT-based сервисах: после входа сервер выдаёт JWT, и именно он потом подтверждает каждый запрос. У сайта есть встроенная «JWT-мастерская» (отладчик в стиле jwt.io) — доступна сразу, без входа: можно раскодировать, отредактировать и заново собрать любой токен, посмотреть его «отпечаток» (blob) и оценку редкости от 0.0 до 10.0. На витрине (доступна только вошедшим) виден продавец `vault` — у его лота самая высокая редкость на рынке. Но ваша цель не «выкупить лот», а зайти на сайт под чужим аккаунтом: собрать JWT, который сервер примет как настоящую сессию продавца `vault`, и на своей странице «Мои токены» забрать то, что там лежит.

Формат ответа: `vsosh{text}`.

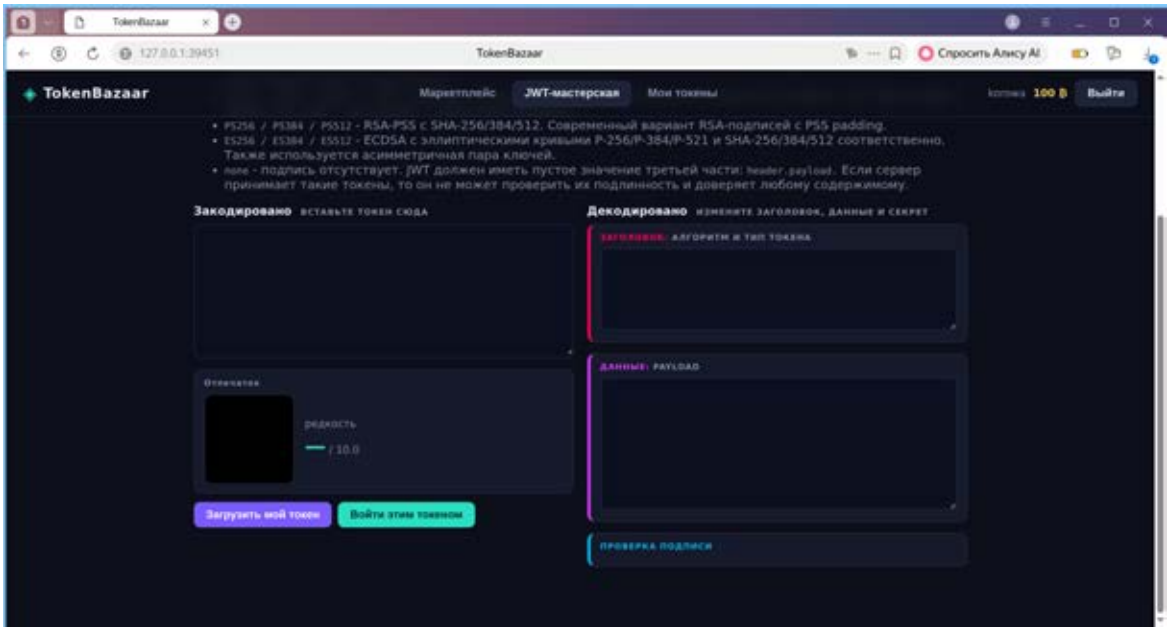
**Ответ:** `vsosh{n0ne_4lg0_0pens_th3_vault}`

**Критерий оценивания:** точное совпадение ответа — 10 баллов

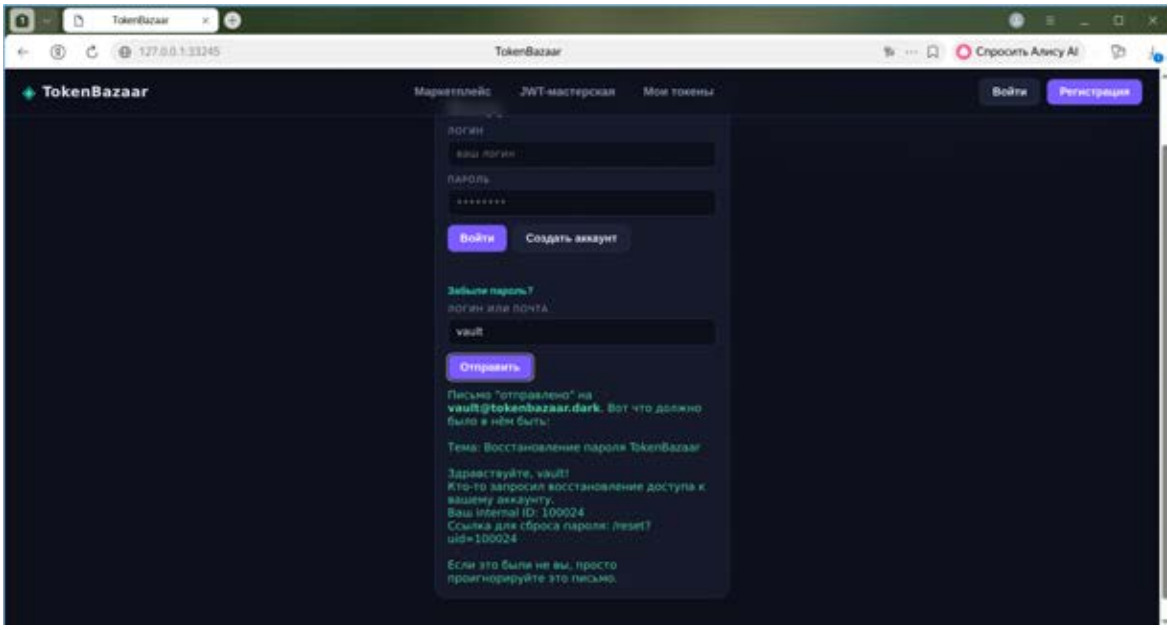
**Максимальный балл за задание — 10**

**Решение.**

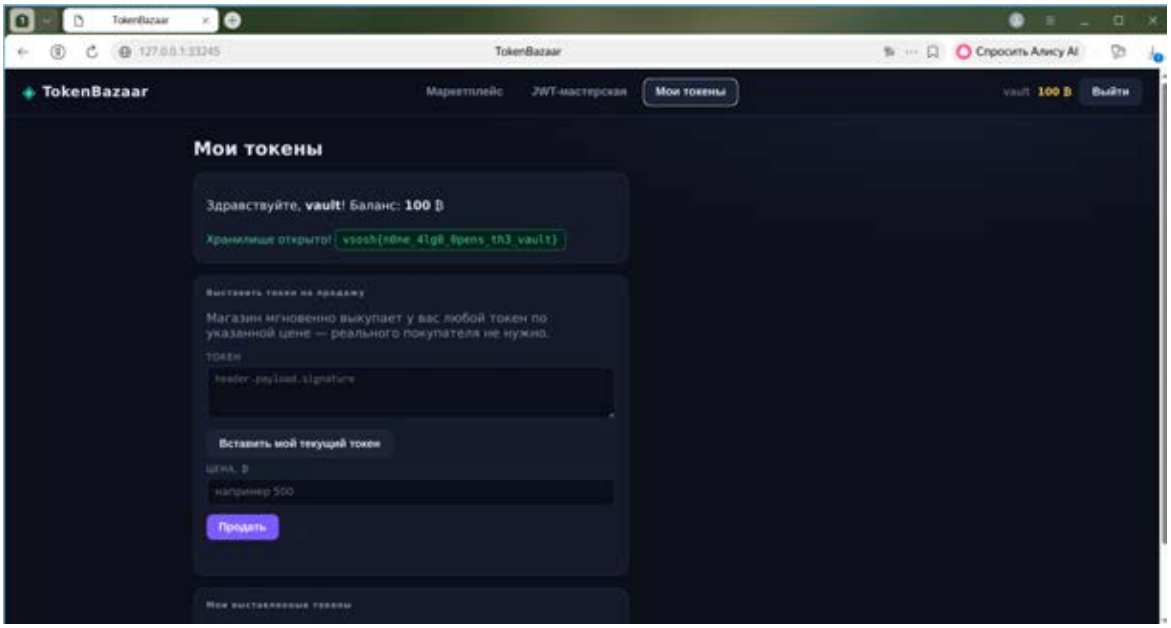
Запускаем файл для своей операционной системы и регистрируем произвольный аккаунт. На витрине выделяется лот «Ключ от хранилища» продавца `vault`; для обычной покупки денег не хватает. На сайте есть разделы «Мои токены» и «JWT-мастерская». Загружаем свой токен в мастерскую и смотрим его поля: токен использует алгоритм `HS256`, содержит имя пользователя `user` и его идентификатор `user_id`.




Если сервер принимает JWT с алгоритмом `none`, подписи у такого токена нет, а значения его полей можно подменить. Для входа под именем `vault` узнаём его идентификатор: выходим из аккаунта, открываем форму «Забыли пароль?» и вводим имя `vault`. Ответ страницы сброса сообщает `user_id=100024`.




Вновь заходим в свой аккаунт, открываем «JWT-мастерскую», меняем в заголовке токена алгоритм на `none`, а в его данных — `user` на `vault` и `user_id` на `100024`. Нажимаем «Войти с этим токеном». На странице «Мои токены» аккаунта `vault` видим флаг.




Сириус.Курсы — для тех,
кто хочет знать больше!