Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

«Технологии и методы программирования»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Выполнила:

Чу Ван Доан – Студент группы N3347

(подпись)

Проверил:

Ищенко Алексей Петрович

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

Содержание

1. Задание

А) Выполняется в локальной операционной системе.

1. Создать текстовый документ (sys.tat), в котором будет содержаться «Системная информация».

2. Написать программу-инсталлятор sys_doc.exe для этого документа, которая под видом установки обновления (с отображением строки прогресса обновления) к какой-нибудь программе (например, Блокнот или Paint):

- Запрашивает у пользователя папку (должен быть вариант использования существующей папки и вариант создания собственной) для копирования «Системной информации».
- Записывает в папку файл с исполняемым кодом программы secur.exe (аналог требований к template.tbl из лабораторной работы No1), защищающей sys.tat.
- Собирает (возможную) информацию о компьютере, на котором устанавливается программа.
- Кодирует эту информацию и записывает в файл sys.tat.
- Подписывает её личным ключом пользователя программы и записывает подпись, например, в реестр Windows в раздел HKEY_CURRENT_USER\Software\Фамилия_студента как значение параметра Signature.
- Запускает secur.exe для защиты sys.tat от несанкционированного доступа.
- Прописывает запуск программы secur.exe при выполнении функции Open для sys.tat, чтобы защита срабатывала и после перезагрузки ОС (есть несколько способов такой «привязки»).

3. В саму программу защиты secur.exe включить следующий функционал:

- Запрос у пользователя информации об имени раздела реестра с электронной цифровой подписью (фамилией студента).
- Считывание подписи из указанного выше раздела реестра, которая проверяется с помощью открытого ключа пользователя.
- Разрешение или запрет просмотра «Системной информации» в файле sys.tat в зависимости от правильности указания ключа.

4. При неудачной проверке работа защищаемой программы должна прекращаться с выдачей соответствующего сообщения.

5. Собираемая о компьютере информация включает в себя как минимум:

- Имя пользователя,
- Имя компьютера,
- Конфигурацию компьютера (память и процессор, как минимум) и версию ОС.

Б) Выполняется в локальной сети (или виртуальной).

1. Создать скрипт, который удалённо и незаметно для пользователя (пользователь открывает какую-нибудь веб-страничку от создателя скрипта) собирает информацию о нём, его компьютере и системе (п.5 предыдущего задания) и записывает её на какой-либо

локальный сетевой диск (доступный создателю скрипта) в папку с именем IP или Mac-адреса пользовательской машины.

2. Продумать доступ к этой информации (можно писать на удалённый диск).

3. Протестировать на 3–5 клиентах и получить статистику о них.

2. Ход работы

2.1. Задание a

Ваша программа состоит из трёх основных файлов: key.py, sys_doc.py и secur.py.

Файл key.py

Этот файл отвечает за создание пары ключей RSA (открытый и закрытый ключи), создание подписи для сообщения и сохранение соответствующих файлов. Конкретно:

1. Создание пары ключей RSA: Функция generate_key_pair() создаёт пару ключей RSA длиной 2048 бит.
   - Открытый ключ: Сохраняется в файл public_key.pem.
   - Закрытый ключ: Используется для подписи сообщения.

2. Создание и сохранение подписи:
   - Сообщение "ChuVanDoanN3347" подписывается закрытым ключом.
   - Подпись сохраняется в файл signature.sig.

3. Запуск файла:
   - При запуске key.py создаётся пара ключей RSA, открытый ключ сохраняется в public_key.pem, а подпись сохраняется в signature.sig.
   - Отображается уведомление о том, что открытый ключ был сохранён и подпись была создана.

Файл sys_doc.py

Этот файл создаёт файл sys.tat, и его содержимое зашифровано. Так как этот файл недоступен для просмотра, я могу только основываться на вашем описании и связывать его с файлом secur.py.

Файл secur.py

Этот файл отвечает за проверку подписи и расшифровку файла sys.tat. Конкретно:

1. Загрузка и проверка подписи:

- Загрузка подписи: Функция load_signature() считывает подпись из файла signature.sig.

- Загрузка открытого ключа: Пользователь должен ввести путь к открытому ключу (public_key.pem). Функция load_public_key_from_file() загружает открытый ключ из этого файла.

- Проверка подписи: Функция verify_signature() проверяет подпись для сообщения "ChuVanDoanN3347". Если подпись действительна, программа продолжает расшифровку.

2. Расшифровка файла sys.tat:

- Если подпись действительна, программа считывает ключ из файла key.key и расшифровывает содержимое sys.tat, используя библиотеку cryptography.fernet.

- После расшифровки файл sys.tat открывается в редакторе nano, чтобы пользователь мог его прочитать.

- После просмотра пользователем содержимое файла sys.tat снова шифруется.

3. Запуск файла:

- При запуске secur.py пользователю нужно ввести путь к открытому ключу.

- Если подпись действительна, файл sys.tat расшифровывается и открывается для просмотра в редакторе nano. Затем файл снова шифруется для защиты информации.

Обзор работы программы

1. Запустить key.py для создания пары ключей RSA и подписи:

- Открытый ключ сохраняется в public_key.pem.

- Подпись сохраняется в signature.sig.

2. Запустить sys_doc.py для создания зашифрованного файла sys.tat.

3. Запустить secur.py для проверки подписи и расшифровки файла sys.tat:

- Ввести путь к файлу public_key.pem для проверки.

- Если проверка успешна, файл sys.tat расшифровывается и открывается в текстовом редакторе nano.

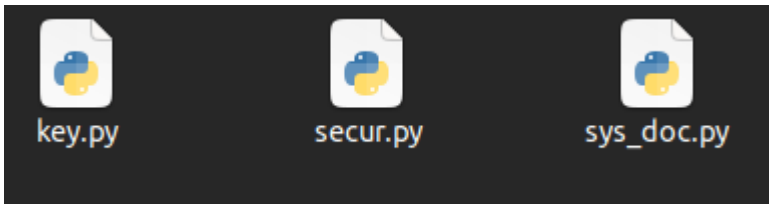

Рисунок 1 - Файлы программы

```
chu@chu-Latitude-5510:~/Documents/Technologies_and_methods_of_programming/Lab_3/A$ python3 key.py
Открытый ключ был сохранен в public_key.pem
Подпись была сохранена в файл.
```

Рисунок 2 - Выполнение программы

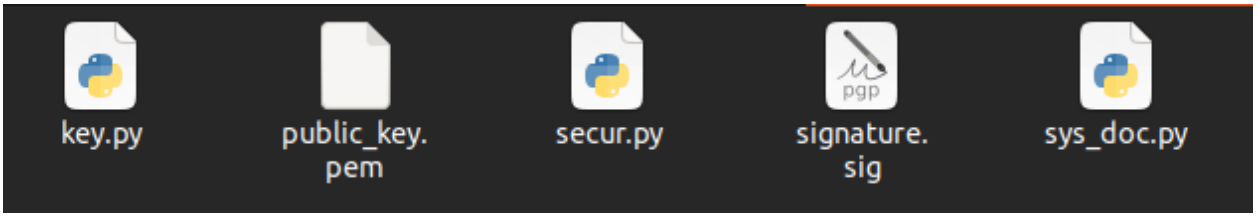

Рисунок 3 - Созданные файлы

```
chu@chu-Latitude-5510:~/Documents/Technologies_and_methods_of_programming/Lab_3/A$ python3 sys_doc.py
Завершено. Файл sys.tat был создан и зашифрован в текущем каталоге.
```

Рисунок 4 - Выполнение программы для сбора информации о системе

Рисунок 5 - Созданные файлы

```
chu@chu-Latitude-5510:~/Documents/Technologies_and_methods_of_programming/Lab_3/A$ cat sys.tat
gAAAAABnN8bEm7J3MATnp4uo3bA0viPBCb9GbbZ1GXzawtdH_Ji0-tC0khbKsuD6qryF7jiEuc0cmddkgERU8varEkmXjWS3m7zyLDlUqTdFFwfUBpOD3pxuBJxuQ8sf
PF1sLVo0FlsAiVbh8Nm1EgM7_SodQ5CkkARyWsTuwWrnshq0usXPvwkwc4MsUBZFcbW9m6G2PSjkDJq4Euu0_ho-AMCv8Kz8w6ZzAMKspQqN6K7xkxO-mRKY7H-MrAeM
ScbGm3O6RPXBxX9ivMamNHAIshYwFVcOdg==chu@chu-Latitude-5510:~/Documents/Technologies_and_methods_of_programming/Lab_3/A$
```

Рисунок 6 - Файл sys.tat был зашифрован и не может быть прочитан

```
chu@chu-Latitude-5510:~/Documents/Technologies_and_methods_of_programming/Lab_3/A$ ls
key.key  key.py  public_key.pem  secur.py  signature.sig  sys_doc.py  sys.tat
chu@chu-Latitude-5510:~/Documents/Technologies_and_methods_of_programming/Lab_3/A$ pwd
/home/chu/Documents/Technologies_and_methods_of_programming/Lab_3/A
```

Рисунок 7 - Точно определите расположение текущего каталога, содержащего файлы.

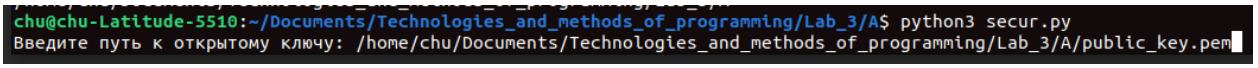

Рисунок 8 - Выполнение программы для дешифрования файла sys.tat

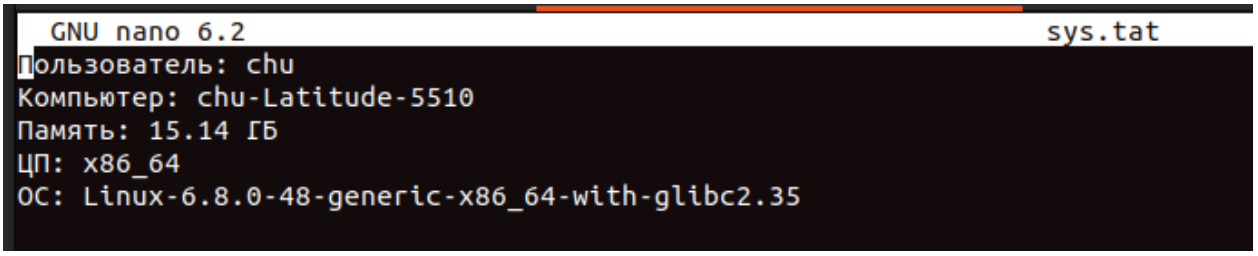

Рисунок 9 - Результат

- File key.py

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes, serialization

PUBLIC_KEY_FILE = "public_key.pem"
SIGNATURE_FILE = "signature.sig"

def save_signature_to_file(signature):
    try:
        with open(SIGNATURE_FILE, 'wb') as f:
            f.write(signature)
        print("Подпись была сохранена в файл.")
    except Exception as e:
        print(f"Ошибка при сохранении подписи в файл: {e}")

def generate_key_pair():
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )

    public_key = private_key.public_key()
    with open(PUBLIC_KEY_FILE, "wb") as f:
        f.write(
            public_key.public_bytes(
                encoding=serialization.Encoding.PEM,
                format=serialization.PublicFormat.SubjectPublicKeyInfo
            )
        )
    print(f"Открытый ключ был сохранен в {PUBLIC_KEY_FILE}")
```

```
    message = b"ChuVanDoanN3347"
    signature = private_key.sign(
        message,
        padding.PKCS1v15(),
        hashes.SHA256()
    )

    save_signature_to_file(signature)

def main():
    generate_key_pair()

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

- File sys_doc.py

```
import os
import socket
import platform
import psutil
from cryptography.fernet import Fernet
import shutil

# Создание и сохранение ключа шифрования
key = Fernet.generate_key()
cipher_suite = Fernet(key)

with open('key.key', 'wb') as key_file:
    key_file.write(key)

# Функция для шифрования файла
def encrypt_file(filepath):
    with open(filepath, 'rb') as file:
        file_data = file.read()
    encrypted_data = cipher_suite.encrypt(file_data)

    with open(filepath, 'wb') as file:
        file.write(encrypted_data)

# Сбор информации о системе
def gather_system_info():
    user_name = os.getlogin()
```

```
    computer_name = socket.gethostname()
    memory_info = psutil.virtual_memory().total / (1024 3)
    cpu_info = platform.processor()
    os_version = platform.platform()

    system_info = (
        f"Пользователь: {user_name}\n"
        f"Компьютер: {computer_name}\n"
        f"Память: {memory_info:.2f} ГБ\n"
        f"ЦП: {cpu_info}\n"
        f"ОС: {os_version}"
    )
    return system_info

# Сохранение информации о системе в файл sys.tat в текущем каталоге
def save_sys_info_to_file(system_info):
    sys_tat_path = 'sys.tat'
    with open(sys_tat_path, 'w') as file:
        file.write(system_info)
    return sys_tat_path

def main():
    # Сбор информации о системе
    system_info = gather_system_info()
    # Сохранение информации в sys.tat в текущем каталоге
    filepath = save_sys_info_to_file(system_info)

    # Шифрование файла sys.tat
    encrypt_file(filepath)

    # Копирование файла secur.py и key.key в текущий каталог, если они не
существуют
    if not os.path.exists('./secur.py'):
        shutil.copy('secur.py', '.')

    if not os.path.exists('./key.key'):
        shutil.copy('key.key', '.')

    print("Завершено. Файл sys.tat был создан и зашифрован в текущем
каталоге.")

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

- File secur.py

```
import os
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes, serialization
from cryptography.exceptions import InvalidSignature
from cryptography.fernet import Fernet
import subprocess

PUBLIC_KEY_FILE = "public_key.pem"
SIGNATURE_FILE = "signature.sig"

def load_signature():
    try:
        with open(SIGNATURE_FILE, 'rb') as f:
            signature = f.read()
        return signature
    except FileNotFoundError:
        print(f"Файл подписи не существует: {SIGNATURE_FILE}")
        return None

def load_public_key_from_file(public_key_file):
    try:
        with open(public_key_file, "rb") as key_file:
            public_key = serialization.load_pem_public_key(
                key_file.read(),
                backend=default_backend()
            )
        return public_key
    except FileNotFoundError:
        print(f"Не найден файл открытого ключа: {public_key_file}")
        return None

def verify_signature(public_key, message, signature):
    try:
        public_key.verify(
            signature,
            message,
            padding.PKCS1v15(),
            hashes.SHA256()
        )
        return True
    except InvalidSignature:
        return False
```

```
def decrypt_file(filepath, key):
    cipher_suite = Fernet(key)
    with open(filepath, 'rb') as file:
        encrypted_data = file.read()
    decrypted_data = cipher_suite.decrypt(encrypted_data)

    with open(filepath, 'wb') as file:
        file.write(decrypted_data)

def encrypt_file(filepath, key):
    cipher_suite = Fernet(key)
    with open(filepath, 'rb') as file:
        file_data = file.read()
    encrypted_data = cipher_suite.encrypt(file_data)

    with open(filepath, 'wb') as file:
        file.write(encrypted_data)

def main():
    signature = load_signature()
    if not signature:
        return

    public_key_file = input("Введите путь к открытому ключу: ").strip()
    public_key = load_public_key_from_file(public_key_file)
    if not public_key:
        return

    message = b"ChuVanDoanN3347"

    if verify_signature(public_key, message, signature):
        with open('key.key', 'rb') as key_file:
            key = key_file.read()
        decrypt_file('sys.tat', key)
        subprocess.run(['nano', 'sys.tat'])
        encrypt_file('sys.tat', key)
    else:
        print("Подтверждение не удалось! Невозможно открыть файл sys.tat.")

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

2.2. Задание б

-Описание: Программа позволяет собирать системную информацию удалённо и записывать её в файл (system_info.txt) на доступном сетевом диске. Пользователю достаточно зайти на веб-страницу приложения, чтобы активировать процесс сбора информации.

-Установка и требования:

- ТребуютсяPython версии 3.7+,FastAPI,Uvicorn ирsutil.
- Пользователю необходимо клонировать репозиторий с GitHub и установить зависимости с помощью команды pip.
- Запустить приложение можно с помощью команды uvicorn app:app --host 0.0.0.0 --port 8000 --reload.

-Использование:

- Пользователь переходит по адресу http://<IP-адрес_сервера>:8000, чтобы увидеть приветственное сообщение.
- Для сбора информации о системе пользователь переходит по адресу /system-info.
- Собранная информация сохраняется в файл system_info.txt в указанной директории, с учётом IP-адреса клиента.

main.py - Основной файл приложения

Файл main.py является ядром веб-приложения, разработанного с использованием FastAPI, включающим конечные точки / и /system-info для обработки HTTP-запросов пользователей.

-Маршрут /: Возвращает приветственное сообщение, направляющее пользователя перейти на /system-info для сбора системной информации.

-Маршрут /system-info:

- При переходе пользователя по данному маршруту собирается информация о системе, включая:

-Имя пользователя: Получается из переменной окружения (USER).

-Имя компьютера: Получается с помощью socket.gethostname().

-Использование CPU: Используется psutil.cpu_percent(interval=1) для получения информации об уровне загрузки CPU.

-Объём оперативной памяти: Общий объём RAM (в ГБ).

-Версия операционной системы: Получается с помощью platform.platform().

-IP-адрес пользователя, отправившего запрос.

- Затем информация записывается в файл system_info.txt в указанную директорию, в зависимости от IP-адреса пользователя, для удобства управления и отслеживания.

Принцип работы программы

1.Запуск сервера: При запуске команды uvicorn сервер FastAPI запускается и готов принимать запросы от пользователей.

2.Переход на приветственную страницу: Пользователь переходит на корневой адрес (/), чтобы увидеть приветственное сообщение и получить инструкцию перейти на /system-info.

3.Сбор системной информации: При переходе на /system-info система автоматически собирает информацию, записывая её в файл system_info.txt на сетевом диске. Это позволяет администраторам сети легко собирать и управлять информацией с нескольких удалённых устройств.

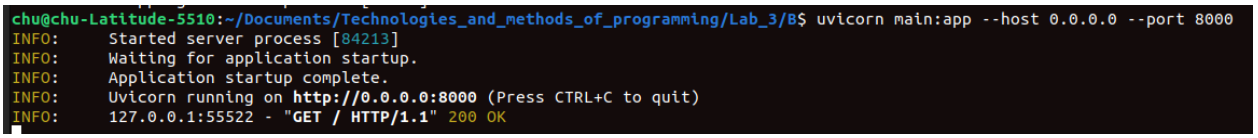

Рисунок 10 - Выполнение программы

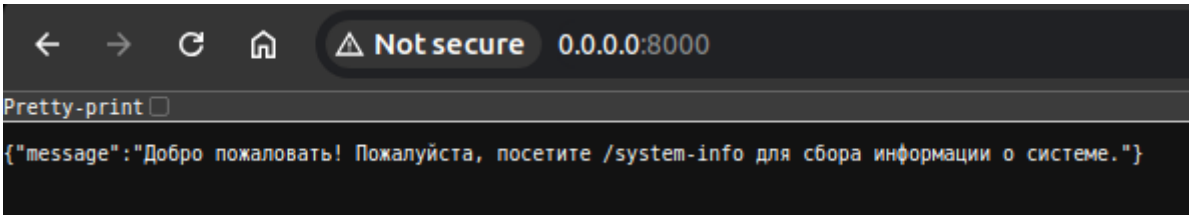

Рисунок 11 - Программа запускается в вебе.

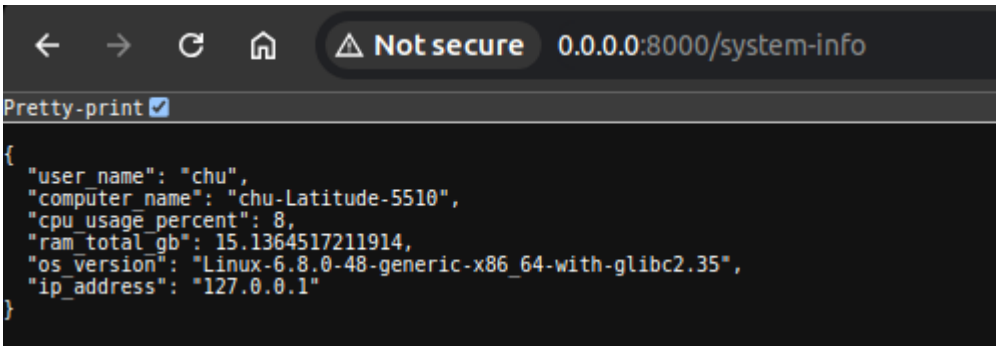

Рисунок 12 - Доступ для сбора информации о системе.

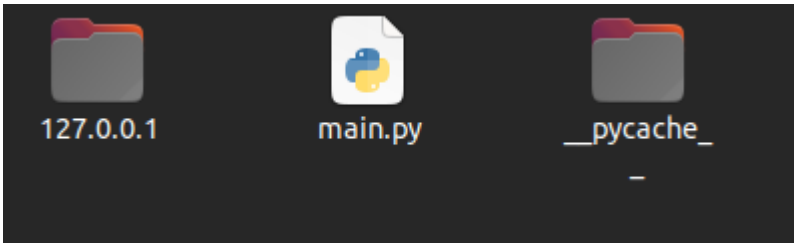

Рисунок 13 - Информация о системе сохранена в каталоге.

```
chu@chu-Latitude-5510:~/Documents/Technologies_and_methods_of_programming/Lab_3/B/127.0.0.1$ cat system_info.txt
user_name: chu
computer_name: chu-Latitude-5510
cpu_usage_percent: 9.4
ram_total_gb: 15.136451721191406
os_version: Linux-6.8.0-48-generic-x86_64-with-glibc2.35
ip_address: 127.0.0.1

user_name: chu
computer_name: chu-Latitude-5510
cpu_usage_percent: 8.0
ram_total_gb: 15.136451721191406
os_version: Linux-6.8.0-48-generic-x86_64-with-glibc2.35
ip_address: 127.0.0.1
```

Рисунок 14 - Просмотр информации о системе, сохраненной в файле.

- File main.py

```
from fastapi import FastAPI, Request

import psutil

import socket

import platform

import os


app = FastAPI()


def write_info_to_network_disk(data, identifier):

   # Путь к директории в Linux

   network_path =
f"/home/chu/Documents/Technologies_and_methods_of_programming/Lab_3/B/{ide
ntifier}"
```

```
    os.makedirs(network_path, exist_ok=True)


    # Запись информации в файл
    with open(f"{network_path}/system_info.txt", "a") as f:  # Используем "a" для добавления информации
        for key, value in data.items():
            f.write(f"{key}: {value}\n")
        f.write("\n")  # Добавляем пустую строку между записями


@app.get("/")
def read_root():
    return {"message": "Добро пожаловать! Пожалуйста, посетите /system-info для сбора информации о системе."}


@app.get("/system-info")
async def get_system_info(request: Request):
    # Получение IP-адреса клиента
    ip_address = request.client.host


    # Получение имени пользователя и имени компьютера
    user_name = os.environ.get('USER')  # Получаем имя пользователя
    computer_name = socket.gethostname()


    # Получение информации о ЦП и оперативной памяти
    cpu_info = psutil.cpu_percent(interval=1)
    ram_info = psutil.virtual_memory().total / (1024 ** 3)  # Преобразуем в ГБ
```

```
# Получение версии операционной системы
os_version = platform.platform()

# Сбор информации
system_info = {
    "user_name": user_name,
    "computer_name": computer_name,
    "cpu_usage_percent": cpu_info,
    "ram_total_gb": ram_info,
    "os_version": os_version,
    "ip_address": ip_address
}

# Запись информации на сетевой диск
write_info_to_network_disk(system_info, ip_address)

return system_info  # Возвращаем информацию о системе клиенту
```

Заключение

Разработаны программы, удовлетворяющие требованиям задач.

Первая программа обеспечивает, что только при успешной проверке с использованием действительного открытого ключа пользователь сможет расшифровать и прочитать содержимое файла sys.tat. Это добавляет уровень безопасности для содержимого файла.

Приложение "Сборщик системной информации" позволяет легко и эффективно собирать системную информацию. Сочетание FastAPI и поддерживающих библиотек, таких как psutil, даёт возможность быстро собирать и сохранять информацию о системах с удалённых устройств без необходимости непосредственного взаимодействия с каждым из них.