

Реализация системы удаленного выполнения команд с использованием шифрования

Матвеева Алёна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель исследования заключается в разработке и реализации системы удаленного выполнения команд с использованием шифрования на основе шифра Цезаря. В статье представлена реализация клиент-серверной системы на языке Python, обеспечивающего удаленное выполнение команд, загрузку файлов и создание скриншотов с использованием шифра Цезаря для защиты передаваемых данных. Разработанное решение демонстрирует базовые принципы сетевого взаимодействия и криптографии, что делает его актуальным для образовательных целей и использования в качестве основы для более сложных приложений.

Ключевые слова: Python, сокет, клиент, сервер

Implementation of a remote command execution system using encryption

Matveeva Alyona Sergeevna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of the research is to develop and implement a remote command execution system using encryption based on the Caesar cipher. The article presents the implementation of a client-server system in Python, which provides remote command execution, file upload, and screenshot creation using Caesar cipher to protect transmitted data. The developed solution demonstrates the basic principles of networking and cryptography, which makes it relevant for educational purposes and use as a basis for more complex applications.

Keywords: Python, socket, client, server

1 Введение

1.1 Актуальность

В современном мире, где объемы передаваемых данных через сети постоянно растут, обеспечение безопасности информации становится одной из ключевых задач. Удаленное выполнение команд и передача файлов между клиентом и сервером являются распространенными сценариями в сетевых приложениях, однако такие операции требуют надежной защиты от несанкционированного доступа и перехвата данных. В связи с этим

актуальной задачей является разработка решений, которые обеспечивают не только функциональность, но и базовый уровень безопасности передаваемой информации. Кроме того, рост интереса к автоматизации задач и удаленному управлению системами повышает востребованность подобных решений в различных областях, включая IT-инфраструктуру, IoT-устройства и образовательные проекты. Таким образом, исследование и реализация систем удаленного выполнения команд с использованием шифрования остаются актуальными и востребованными в контексте современных технологических трендов.

1.2 Обзор исследований

К. А. Коротун и Е. С. Филонова рассмотрели способы шифрование текстов средствами языка программирования Python [1]. Я. А. Марков в своей статье изучил способы дешифрования классических шифров [2]. В статье А.А. Заикина рассматриваются некоторые аспекты разработки программного практикума по криптоанализу классических криптосистем [3]. В исследовании О. О. Козеевой и И. В. Чухраева с помощью языка Python разработали модуль аутентификации автоматизированной информационной системы [4]. А.А.Гладких используя язык программирования python, разработал расширение системы автоматизированного проектирования [5]. А. С. Никитин и Д.А.Москвин рассмотрели вопросы, связанные с разработкой эффективных средств оценки безопасности компьютерных сетей на языке программирования Python [6].

1.3 Цель исследования

Цель исследования заключается в разработке и реализации системы удаленного выполнения команд с использованием шифрования на основе шифра Цезаря.

2 Материалы и методы

В данном исследовании для реализации клиент-серверной системы использовалась среда разработки Visual Studio Code (VS Code), которая предоставляет удобные инструменты для написания, отладки и тестирования кода на языке Python. Python был выбран благодаря своей простоте, широким возможностям для сетевого программирования и наличию богатой стандартной библиотеки.

3 Результаты

Вначале нужно написать программный код для сервера:

```
import socket
import subprocess
import pyautogui
def caesar_cipher(text, shift):
    """Шифрует или дешифрует текст с использованием шифра Цезаря."""
    result = []
    for char in text:
```

```

        if char.isalpha(): # Проверяем, является ли символ буквой
            shift_base = ord('a') if char.islower() else ord('A')
            # Сдвиг символа с учетом русского алфавита
            result.append(chr((ord(char) - shift_base + shift) % 32 +
shift_base))
        else:
            result.append(char) # Не шифруем не буквенные символы
    return ''.join(result)

def execute_command(command):
    """Выполняет команду и возвращает результат."""
    try:
        output = subprocess.check_output(command, shell=True,
stderr=subprocess.STDOUT)
        decoded_output = output.decode('cp866', errors='replace')
        encrypted_output = caesar_cipher(decoded_output, 5) # Шифруем вывод
        return f"Закодировано: {encrypted_output}\nРаскодировано:
{decoded_output}"
    except subprocess.CalledProcessError as e:
        decoded_output = e.output.decode('cp866', errors='replace')
        encrypted_output = caesar_cipher(decoded_output, 5) # Шифруем вывод
        return f"Закодировано: {encrypted_output}\nРаскодировано:
{decoded_output}"
    except Exception as e:
        return str(e) # Возвращаем текст ошибки

def handle_client(client_socket):
    while True:
        command = client_socket.recv(1024).decode('cp1251') # Декодируем
входящую команду
        if not command:
            break

        if command.startswith("cmd"):
            output = execute_command(command[4:])
        elif command.startswith("download"):
            filename = command[9:]
            try:
                with open(filename, "rb") as f:
                    file_data = f.read()
                    output = f"Файл загружен. Размер: {len(file_data)}
байт."
            except FileNotFoundError:
                output = "Файл не найден."
        elif command == "screenshot":
            screenshot = pyautogui.screenshot()
            screenshot.save("screenshot.png")
            output = "Скриншот сохранен как screenshot.png."
        else:
            output = "Неизвестная команда"

        # Шифруем вывод и отправляем клиенту
        encrypted_output = caesar_cipher(output, 5) # Шифруем вывод
        response = f"Закодировано: {encrypted_output}\nРаскодировано:
{output}"
        client_socket.send(response.encode('cp866')) # Отправляем ответ с
правильной кодировкой

```

```
def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(("0.0.0.0", 9999))
    server.listen(5)
    print("Сервер запущен и ожидает подключения...")

    while True:
        client_socket, addr = server.accept()
        print(f"Подключен к {addr}")
        handle_client(client_socket)
        client_socket.close()

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

1. Импорт библиотек:

- socket: используется для создания сетевых соединений.
- subprocess: позволяет выполнять команды операционной

системы.

- pyautogui: используется для создания скриншотов экрана.

2. Функция caesar_cipher:

- Реализует шифр Цезаря для шифрования и дешифрования текста.

Каждый символ текста сдвигается на заданное количество позиций в алфавите. Если символ не является буквой, он остается неизменным.

3. Функция execute_command:

- Выполняет команду, полученную от клиента, с помощью

subprocess.check_output.

- Возвращает зашифрованный и расшифрованный вывод команды.

Если команда завершилась с ошибкой, возвращает сообщение об ошибке.

4. Функция handle_client:

- Обрабатывает входящие соединения от клиентов.
- Получает команды от клиента, определяет тип команды

(выполнение команды, загрузка файла или создание скриншота) и вызывает соответствующие функции.

- Шифрует вывод и отправляет его обратно клиенту.

5. Функция main:

- Создает серверный сокет, который слушает входящие соединения на порту 9999.

• Принимает подключения от клиентов и передает их в функцию handle_client для обработки.

Теперь нужно прописать программный код для клиента:

```
import socket

def caesar_cipher(text, shift):
    """Дешифрует текст с использованием шифра Цезаря."""
    result = []
    for char in text:
        if char.isalpha(): # Проверяем, является ли символ буквой
```

```

        shift_base = ord('a') if char.islower() else ord('A')
        # Сдвиг символа с учетом русского алфавита
        result.append(chr((ord(char) - shift_base - shift) % 32 +
shift_base))
    else:
        result.append(char) # Не шифруем не буквенные символы
    return ''.join(result)

# Список доступных команд
available_commands = """
Доступные команды:
1. cmd <команда> - Выполнить команду на сервере.
   Пример: cmd dir
2. download <имя_файла> - Загрузить файл с сервера.
   Пример: download example.txt
3. screenshot - Сделать скриншот экрана.
   Пример: screenshot
"""

def send_command(command):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(("127.0.0.1", 9999))

    client.send(command.encode('cp1251')) # Отправляем команду с правильной
кодировкой

    response = client.recv(4096)
    try:
        decrypted_response = response.decode('cp866') # Декодируем ответ
        print(decrypted_response) # Выводим расшифрованный текст
    except UnicodeDecodeError:
        print("Не удалось декодировать ответ сервера.")

    client.close()

if __name__ == "__main__":
    print(available_commands)
    while True:
        command = input("Введите команду: ")
        send_command(command)

```

1. Импорт библиотеки:

- socket: используется только одна библиотека для создания сетевых соединений.

2. Функция caesar_cipher:

- Реализует дешифрование текста с использованием шифра Цезаря.

Каждый символ текста сдвигается на заданное количество позиций в алфавите. Если символ не является буквой, он остается неизменным.

3. Список доступных команд:

- Определяет команды, которые клиент может отправить серверу, включая выполнение команд, загрузку файлов и создание скриншотов.

4. Функция send_command:

- Создает клиентский сокет и подключается к серверу.
- Отправляет команду, закодированную в cp1251, и ожидает ответа от сервера.

- Декодирует полученный ответ в sr8b6 и выводит его на экран.
- Если декодирование не удалось, выводит сообщение об ошибке.
- 5. Основной блок:**
- При запуске программы выводит список доступных команд.
 - В бесконечном цикле запрашивает у пользователя ввод команды и отправляет ее на сервер.

Выполнение работы кода:

В первую очередь запускаем сервер, а затем запускаем клиента. В консоли клиента выводятся доступные команды, которые он может отправлять на сервер. Когда клиент отправляет на сервер команду «screenshot», сервер делает скриншот экрана и сохраняет его в формате .png, затем кодирует сообщение и отправляет клиенту. Клиент получает зашифрованное сообщение и дешифрует его.

Клиент также может загрузить этот файл, используя команду «download screenshot.png» (рис. 1).

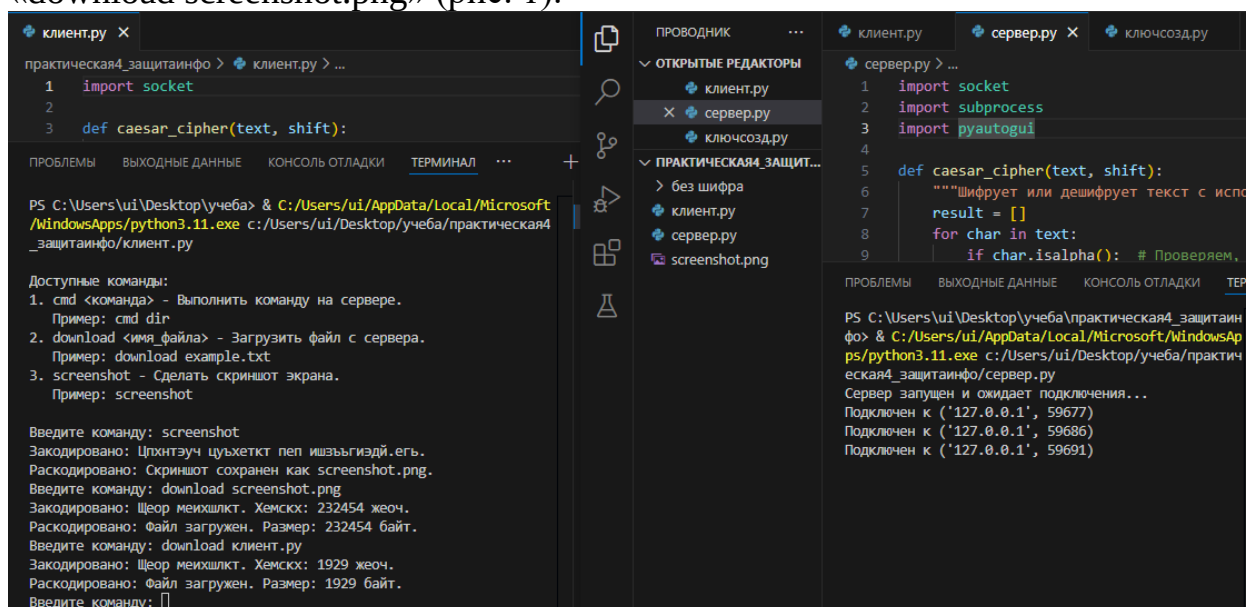


Рисунок 1 – Работа системы клиент-сервер

Также сервер может выполнять такие же встроенные команды, как в командной строке на ПК. В данном примере выполняется команда «cmd dir», используется для отображения списка файлов и папок в текущем каталоге или указанной директории. Сервер также отправляет зашифрованные данные, а клиент их дешифрует (рис. 2).

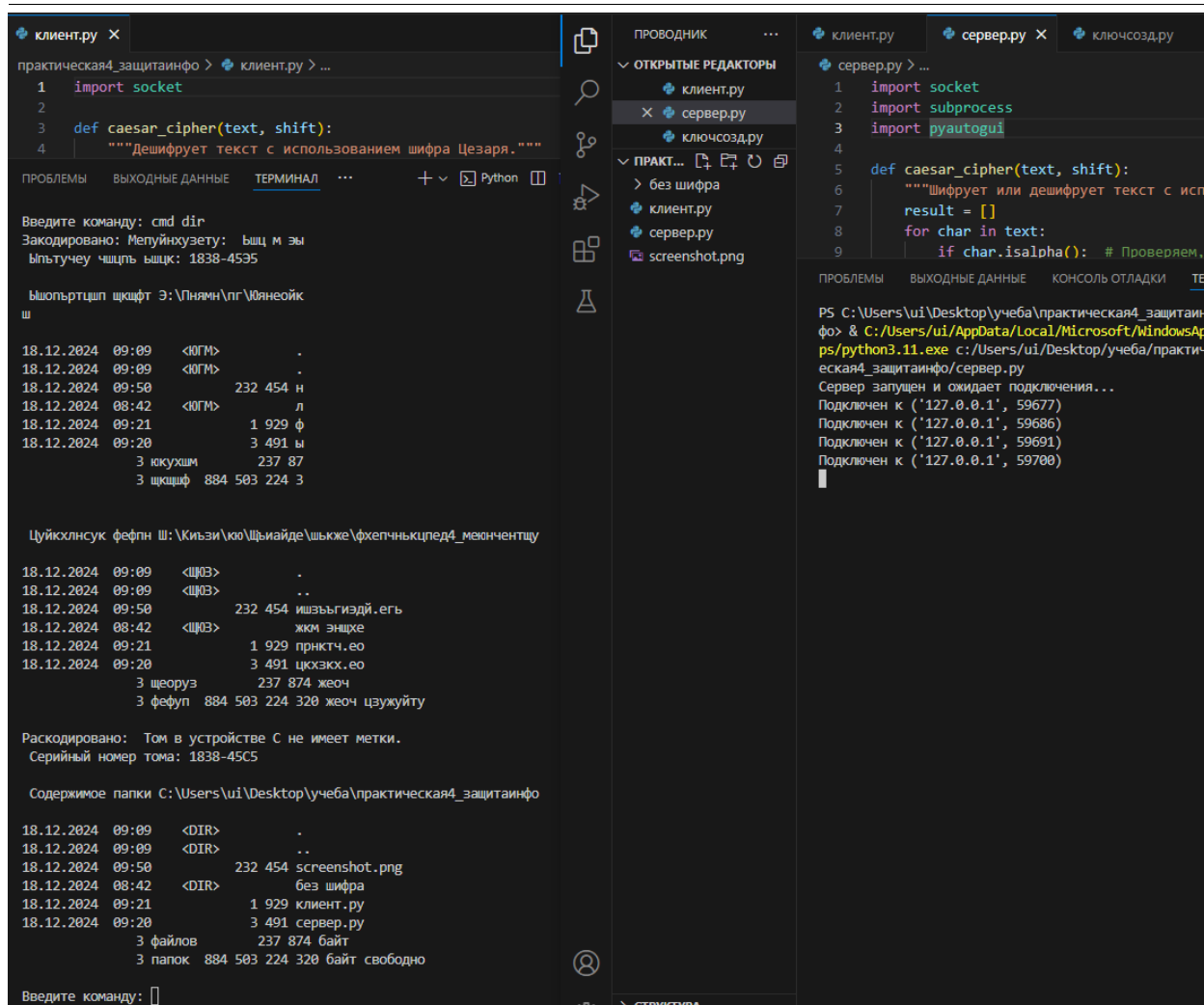


Рисунок 2 – Выполнение команды «cmd dir»

4 Выводы

В статье представлена реализация клиент-серверного приложения с использованием шифра Цезаря для шифрования передаваемых данных. Разработанное решение позволяет выполнять команды операционной системы, загружать файлы и создавать скриншоты на стороне сервера, обеспечивая при этом защиту данных за счет шифрования. Предложенные улучшения, такие как многопоточная обработка клиентов и использование более современных методов шифрования, повышают производительность и безопасность системы. Реализация демонстрирует, что даже простые криптографические методы могут быть эффективно использованы для защиты данных в сетевых приложениях. Данная работа может служить учебным материалом для изучения основ сетевого программирования и криптографии, а также стать отправной точкой для разработки более сложных и безопасных решений.

Библиографический список

1. Коротун К. А., Филонова Е. С. Шифрование текстов средствами языка

- программирования Python // Наука и современность: материалы всероссийской научно-практической. 2022. С. 71.
2. Марков Я. А. К вопросу о способах дешифрования классических шифров // Безопасные информационные технологии. Сборник трудов XII международной научно-технической конференции "Безопасные информационные технологии". М.: МГТУ им. Н.Э.Баумана, 2023. С.166-168.
 3. Заикин А. А. О разработке лабораторного практикума криптоанализа классических шифров // Информационные технологии в науке, бизнесе и образовании. Проблемы обеспечения цифрового суверенитета государства. 2021. С. 31-35.
 4. Козеева О. О., Чухраев И. В. Разработка средствами языка Python модуля аутентификации автоматизированной информационной системы // Электронный журнал: наука, техника и образование. 2017. №. 2. С. 173-180.
 5. Гладких А. А. Использование языка программирования python для разработки расширений систем автоматизированного проектирования // Вестник Екатерининского института. 2014. №. 3. С. 7-16.
 6. Никитин А. С., Москвин Д. А. Применение языка PYTHON для разработки средств оценки безопасности компьютерных сетей // Региональная информатика и информационная безопасность. 2017. С. 235-235.