

Документация проекта: Wi-Fi Warden

Система пассивного анализа безопасности беспроводных и проводных сетей

1. Общее описание

Wi-Fi Warden — это консольное приложение на Python для комплексного пассивного анализа сетевой безопасности. Система обнаруживает потенциальные угрозы и уязвимости в локальных сетях без генерации дополнительного трафика.

Основные характеристики:

- **Пассивный анализ:** не вмешивается в работу сети
- **Модульная архитектура:** легко расширяемый функционал
- **Консольный интерфейс:** простота использования
- **Кроссплатформенность:** работает на Windows, Linux, macOS
- **Open Source:** полный доступ к исходному коду

2. Технические требования

2.1. Системные требования

- **ОС:** Windows 10/11
- **Python:** версия 3.11 или выше
- **Права доступа:** для некоторых функций требуются права администратора/root

2.2. Зависимости

Основные библиотеки

scapy $\geq 2.5.0$

requests $\geq 2.28.0$

dnspython $\geq 2.3.0$

colorama $\geq 0.4.6$

3. Установка и настройка

3.1. Клонирование репозитория

```
git clone https://github.com/tagir-valitov/wifi-warden.git  
cd wifi-warden
```

3.2. Установка зависимостей

```
pip install -r requirements.txt
```

4. Структура проекта

```
text  
wifi-warden/  
    └── modules/          # Основные модули анализа  
        ├── arp_monitor.py  # Мониторинг ARP-трафика  
        ├── portscan_monitor.py  # Обнаружение сканирования портов  
        ├── gateway_monitor.py  # Проверка сетевого шлюза  
        ├── dos_monitor.py  # Мониторинг DDoS-атак  
        ├── tls_check.py  # Проверка TLS-шифрования  
        ├── dns_check.py  # Анализ DNS-безопасности  
        └── wifi_scanner.py  # Сканирование Wi-Fi сетей  
    └── engine/            # Движок оценки рисков  
        └── risk_engine.py  # Движок оценки рисков  
    └── utils/              # Вспомогательные утилиты  
        ├── logger.py       # Система логирования  
        └── config_loader.py  # Загрузчик конфигураций  
    └── main.py            # Главный файл приложения  
    └── requirements.txt  # Зависимости проекта  
    └── README.md         # Основная документация  
    └── config.yaml       # Файл конфигурации
```

5. Функциональные модули

5.1. Модуль arp_monitor.py

Назначение: Обнаружение ARP-spoofing атак

Алгоритм работы:

1. Пассивное прослушивание ARP-пакетов
2. Построение таблицы соответствий IP-MAC

3. Выявление конфликтов (несколько MAC для одного IP)
4. Расчет уровня риска на основе аномалий

Выходные данные: arp_risk (0-100)

5.2. Модуль portscan_monitor.py

Назначение: Обнаружение сканирования портов

Алгоритм работы:

1. Анализ частоты подключений к портам
2. Выявление паттернов сканирования
3. Классификация типов сканирования
4. Расчет уровня риска

Выходные данные: ports_risk (0-100)

5.3. Модуль gateway_monitor.py

Назначение: Проверка корректности сетевого шлюза

Алгоритм работы:

1. Определение текущего шлюза по умолчанию
2. Проверка стабильности соединения
3. Сравнение с ожидаемыми параметрами
4. Обнаружение подмены шлюза

Выходные данные: gateway_risk (0-100)

5.4. Модуль dos_monitor.py

Назначение: Обнаружение DDoS-атак

Алгоритм работы:

1. Мониторинг интенсивности трафика
2. Анализ пакетов на предмет флуда
3. Выявление аномальных паттернов
4. Расчет уровня угрозы

Выходные данные: dos_risk (0-100)

5.5. Модуль `tls_check.py`

Назначение: Проверка безопасности TLS-соединений

Алгоритм работы:

1. Проверка доступности HTTPS-сервисов
2. Анализ поддерживаемых протоколов и шифров
3. Проверка сертификатов (срок действия, цепочка доверия)
4. Выявление устаревших настроек

Выходные данные: `tls_risk` (0-100)

5.6. Модуль `dns_check.py`

Назначение: Анализ DNS-безопасности

Алгоритм работы:

1. Мониторинг DNS-запросов и ответов
2. Обнаружение признаков DNS-spoofing
3. Проверка поддержки DNSSEC
4. Анализ DNS-серверов

Выходные данные: `dns_risk` (0-100)

5.7. Модуль `wifi_scanner.py`

Назначение: Сканирование Wi-Fi сетей

Алгоритм работы:

1. Обнаружение доступных беспроводных сетей
2. Анализ параметров безопасности (WPA2/WPA3)
3. Проверка открытых сетей
4. Сбор информации о точках доступа

Выходные данные: Список сетей с оценкой безопасности

5.8. Модуль `risk_engine.py`

Назначение: Агрегация и оценка рисков

Алгоритм работы:

1. Сбор данных от всех модулей
2. Применение весовых коэффициентов
3. Расчет интегрального показателя безопасности
4. Формирование отчета с рекомендациями

Выходные данные: Общий риск (0-100) и детальный отчет

6. Использование приложения

6.1. Запуск приложения

```
bash  
python main.py
```

6.2. Основные команды

text

Главное меню:

1. Полный анализ сети
2. Проверка Wi-Fi сетей
3. Проверка локальной сети
4. Проверка интернет-безопасности
5. Настройки
6. Выход

6.3. Режимы проверки

1. **Быстрая проверка** (15 секунд) — экспресс-анализ
2. **Обычная проверка** (30 секунд) — стандартный анализ
3. **Непрерывный мониторинг** — постоянное наблюдение

6.4. Интерпретация результатов

- **0-30 баллов:** Низкий риск, сеть защищена
- **31-70 баллов:** Средний риск, требуется внимание
- **71-100 баллов:** Высокий риск, немедленные действия

7. Примеры использования

7.1. Полный анализ домашней сети

Выбираем "1. Полный анализ сети"
Выбираем быструю проверку
Получаем отчет с оценкой безопасности

9. Безопасность и ограничения

9.1. Меры безопасности

- Пассивный анализ (без отправки пакетов)
- Локальная обработка данных
- Без сохранения персональной информации
- Открытый исходный код для аудита

9.2. Ограничения

- Не обнаруживает все типы атак
- Требует права администратора для некоторых функций
- Эффективность зависит от сетевой конфигурации
- Не заменяет полноценные системы безопасности

9.3. Юридические аспекты

- Использование только в законных целях
- Требуется разрешение для анализа чужих сетей
- Соблюдение ФЗ-152 "О персональных данных"
- Соответствие политике ответственного раскрытия

10. Развитие проекта

10.1. Планируемые улучшения

1. Графический интерфейс — версия с GUI
2. Веб-панель управления — удаленный мониторинг
3. Дополнительные модули — анализ новых типов угроз
4. API — интеграция с другими системами
5. Мобильное приложение — проверка сетей со смартфона

10.2. Как внести вклад

1. Форк репозитория
2. Создание feature branch
3. Внесение изменений
4. Создание pull request
5. Обсуждение и код-ревью

10.3. Сообщество

- **Issues:** сообщения об ошибках
- **Discussions:** обсуждение улучшений
- **Wiki:** документация и примеры
- **Releases:** стабильные версии

11. Лицензия

Проект распространяется под лицензией **MIT**. Подробности в файле LICENSE.

12. Контакты

- **Автор:** Tagir Valitov
 - **GitHub:** [tagir-valitov](#)
 - **Репозиторий:** [wifi-warden](#)
 - **Почта:** [укажите при необходимости]
-

Приложение А: Быстрый старт

bash

1. Установка

```
git clone https://github.com/tagir-valitov/wifi-warden.git  
cd wifi-warden  
pip install -r requirements.txt
```

2. Запуск

```
python main.py
```

3. Выбор полного анализа

4. Получение отчета о безопасности

Приложение В: Пример отчета

text

ОТЧЕТ О БЕЗОПАСНОСТИ СЕТИ

Время: 2024-01-15 14:30:00

Сеть: Home-Network

ОБЩАЯ ОЦЕНКА: 15/100 (НИЗКИЙ РИСК)

ДЕТАЛИЗАЦИЯ:

- ARP безопасность: 0/100 ✓
- Защита портов: 20/100 ⚠
- Сетевой шлюз: 0/100 ✓
- DDoS защита: 0/100 ✓
- TLS безопасность: 40/100 ⚠
- DNS безопасность: 10/100 ✓
- Wi-Fi безопасность: 0/100 ✓

РЕКОМЕНДАЦИИ:

1. Проверить открытые порты: 22, 80
2. Обновить настройки TLS на роутере