**МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ В ЦЕНТРАЛЬНОЙ АЗИИ**

**Направление Информационные Технологии**

**Выпускная квалификационная работа**

**На тему:** «Разработка защищенной системы анализа безопасности IoT на примере гостиничного комплекса Novotel»

Выполнил: ст. Арсений Радченко

гр. IT-120

Научный руководитель: ст. преп. Ашымова А.Ж.

Токмок 2024

# **Оглавление**

[**Оглавление 2**](#_ak12dm1edd7c)

[**Введение 4**](#_2gqnqo46tveq)

[1. Актуальность 4](#_t3lcuy58buu4)

[2. Цели и задачи исследования 4](#_gjn7sr7qndrg)

[**Глава 1. Анализ и разработка требований 6**](#_t8chfyyoe9h0)

[1.1 Обзор и анализ существующих систем анализа защищенности IoT устройств 6](#_sf6qkl7f9kh0)

[1.2 Обоснование необходимости разработки 7](#_upqwat6rjkk0)

[1.3 Функциональные требования 8](#_86ruthhotnx)

[*1.3.1 Функции, доступные пользователю 11*](#_hs0n5fejdol)

[1.4 Функциональные требования 12](#_22jo4x3d33ti)

[*1.4.1 Обнаружение устройств 12*](#_jcsaxqk731ur)

[*1.4.2 Анализ защищенности 14*](#_ew41v9q516xk)

[*1.4.3 Эмуляция атак 15*](#_gg66nexc2qz6)

[*1.4.4 Генерация отчетов 15*](#_ocgsmb3u6jek)

[1.5 Нефункциональные требования 18](#_q9grp5ua3ee6)

[*1.5.1 Производительность: 18*](#_m2jg1um3jjvq)

[*1.5.2 Надежность 18*](#_7bab1jsuqoh)

[*1.5.3 Безопасность 18*](#_h4isn2o1hxu)

[*1.5.4 Удобство использования 18*](#_8o45h9ohn1og)

[*1.5.5 Масштабируемость 19*](#_johixztj7yty)

[1.6 Обоснование выбора инструментов разработки 19](#_xr01s3su6rql)

[1.7 Вывод 20](#_apxuc5dlcyo2)

[**Глава 2. Проектирование системы 21**](#_eko2d06opyun)

[2.1 IDEF0 21](#_w1r7hl2g0e13)

[2.2 Детальная архитектура системы 23](#_33lppxa7oxm7)

[*2.1.1 Модуль обнаружения устройств 24*](#_q94cyz89xcoz)

[*2.1.2 Модуль анализа защищенности 24*](#_edosrbpvcrbm)

[*2.1.3 Модуль эмуляции атак 25*](#_7aymx1kqa4h4)

[*2.1.4 Модуль визуализации 26*](#_yh56w629d4yl)

[*2.1.5 Модуль генерации отчетов 26*](#_10lpck9uwmov)

[2.3 Взаимодействие компонентов 27](#_9axx3rkxbvcu)

[2.4 Организация структурных данных 28](#_vyedptx0fe6p)

[2.5 Вывод 29](#_7m5fvkdwlux2)

[**Глава 3. Реализация системы анализа защищенности IoT-устройств 30**](#_h55ocnpyizf0)

[3.1. Выбор технологий 30](#_fprthvhrcp1w)

[3.2 Разработка компонентов 31](#_evky4avj18nh)

[*3.2.1 Модуль обнаружения устройств 31*](#_6hi88i52cl3v)

[*3.2.2 Модуль анализа защищенности 32*](#_teda4w2rl3cr)

[*3.2.3 Модуль эмуляции атак 32*](#_phbz9fio46c4)

[*3.2.4 Модуль эмуляции атак 37*](#_z1isi6wys516)

[3.3 BadUSB 39](#_a2tu69j0qer3)

[*3.3.1 Загрузка и сохранение настроек 39*](#_b6k8pdqpdflw)

[*3.3.2 Обработка событий 39*](#_ddqlzm9ehxfg)

[*3.3.3 Инициализация 40*](#_6ktmtlvn19co)

[*3.3.4 Освобождение ресурсов 41*](#_yb1ku3fwlr0t)

[*3.3.5 Точка входа 41*](#_27h4k5k61ty)

[3.4 Модуль визуализации 48](#_r888eq5ibwim)

[3.5 Модуль генерации отчетов 48](#_1awg2bcyfw60)

[3.6 Тестирование системы 48](#_3bk6gxo799yu)

[3.7 Вывод 49](#_vgrkpv7onbl6)

[**Заключение 50**](#_pxt1cb7bm4wv)

[**Ссылки 52**](#_faaj3c6gt9t3)

# 

# 

# 

# **Введение**

Тема: Разработка системы для анализа защищенности IoT устройств и эмуляции атак с использованием Flipper Zero

## **1. Актуальность**

“В настоящее время количество IoT устройств стремительно растет, что приводит к увеличению уязвимых устройств. Атаки на IoT устройства могут иметь серьезные последствия, такие как кража данных, нарушение работы критической инфраструктуры и шантаж”[[1]](#footnote-0). Существующие системы анализа защищенности IoT устройств не всегда подходят для анализа защищенности устройств и не всегда могут эмулировать атаки. Разработка системы, которая позволит анализировать защищенность IoT устройств и эмулировать атаки с использованием Flipper Zero, является актуальной.

## **2. Цели и задачи исследования**

Цель: Разработать систему для анализа защищенности IoT устройств и эмуляции атак с использованием Flipper Zero.

Задачи:

* Проанализировать существующие системы анализа защищенности IoT устройств и эмуляции атак.
* Определить функциональные требования к системе.
* Разработать архитектуру системы.
* Реализовать систему.
* Провести тестирование системы.
* Подготовить документацию к системе.

**3. Объект и предмет исследования:**

Объект исследования: IoT устройства.

Предмет исследования: методы анализа защищенности IoT устройств и эмуляции атак с использованием Flipper Zero.

**4. Степень разработанности темы:**

Существует ряд исследований, посвященных анализу защищенности IoT устройств и эмуляции атак. Однако не существует системы, которая бы позволяла анализировать защищенность IoT устройств и эмулировать атаки с использованием Flipper Zero.

**5. Методы решения поставленных задач:**

* Анализ литературы: изучение существующих исследований по анализу защищенности IoT устройств и эмуляции атак.
* Моделирование: разработка модели системы.
* Программирование: реализация системы.
* Тестирование: тестирование системы на работоспособность и соответствие требованиям.
* Анализ результатов: анализ результатов тестирования и корректировка системы.

# 

# **Глава 1: Анализ и разработка требований**

## **1.1 Обзор и анализ существующих систем анализа защищенности IoT устройств**

В настоящее время существует множество систем для анализа защищенности IoT устройств. Эти системы можно разделить на несколько категорий:

* **Системы сканирования:** эти системы сканируют сеть и выявляют IoT устройства. Они могут также собирать информацию об устройствах, такую как тип устройства, версия прошивки и открытые порты.
* **Системы тестирования на проникновение:** эти системы используют различные методы для выявления уязвимостей в IoT устройствах. Они могут включать сканирование уязвимостей, fuzzing и атаки на основе словаря.
* **Системы эмуляции атак:** эти системы эмулируют различные типы атак на IoT устройства. Это позволяет администраторам безопасности оценить защищенность своих устройств.

Некоторые из наиболее популярных систем анализа защищенности IoT устройств включают:

* **Nessus:** Nessus - это широко используемый сканер уязвимостей, который может использоваться для сканирования IoT устройств.
* **Nmap:** Nmap - это еще один популярный сканер уязвимостей, который может использоваться для сканирования IoT устройств.
* **Metasploit:** Metasploit - это платформа тестирования на проникновение, которая может использоваться для выявления уязвимостей в IoT устройствах.
* **OWASP ZAP:** OWASP ZAP - это бесплатный веб-браузерный сканер уязвимостей, который может использоваться для сканирования IoT устройств.
* **IoT Inspector:** IoT Inspector - это коммерческая система анализа защищенности IoT устройств, которая включает сканирование, тестирование на проникновение и эмуляцию атак.

**Несмотря на наличие различных систем, существует ряд недостатков:**

* **Неполное покрытие:** не все системы охватывают все типы IoT устройств и все типы уязвимостей.
* **Ложные срабатывания:** некоторые системы генерируют ложные срабатывания, что может привести к ненужным затратам времени и ресурсов.
* **Сложность:** некоторые системы сложны в использовании и требуют специальных знаний.
* **Отсутствие эмуляции атак:** не все системы могут эмулировать атаки на IoT устройства.

## **1.2 Обоснование необходимости разработки**

Разработка системы, которая позволит:

* **Обнаруживать IoT устройства в сети,** поддерживая различные протоколы (RFID, NFC, Infared, BLE, Bad USB, Sub-GHz).
* **Анализировать защищенность IoT устройств,** выполняя сканирование прошивки, тестирование на проникновение и анализ сетевого трафика.
* **Эмулировать атаки на IoT устройства** с использованием Flipper Zero, настраивая параметры атаки и наблюдая за ходом атаки в реальном времени.
* **Визуализировать результаты** в понятном виде, предоставляя подробную информацию о выявленных уязвимостях.
* **Генерировать отчеты** о результатах анализа защищенности и эмуляции атак.

**Такая система позволит администраторам безопасности:**

* Эффективно управлять безопасностью IoT устройств в сети.
* Выявлять и устранять уязвимости IoT устройств.
* Оценивать устойчивость IoT устройств к атакам.
* Соответствовать требованиям безопасности.

## **1.3 Функциональные требования**

**Обнаружение устройств:**

* Сканирование сети для обнаружения IoT устройств с поддержкой протоколов:
  + RFID (Radio Frequency Identification) - для идентификации объектов с помощью радиоволн.
  + NFC (Near Field Communication) - для бесконтактного обмена данными на малом расстоянии.
  + Infrared (Инфракрасный) - для передачи данных с помощью инфракрасного излучения.
  + BLE (Bluetooth Low Energy) - для энергосберегающей беспроводной связи на короткие расстояния.
  + Bad USB - для атак на устройства через USB-порты, маскируясь под легитимные устройства.
  + Sub-GHz (Sub-Gigahertz) - для связи на частотах ниже 1 ГГц, часто используемой в системах автоматизации зданий и промышленном Интернете вещей.
* Определение типа, версии и производителя устройства.
* Фильтрация результатов сканирования по различным критериям (тип устройства, производитель, версия прошивки и т.д.).

**Анализ защищенности:**

* Сканирование прошивки устройства на наличие уязвимостей с использованием баз данных уязвимостей (CVE, NVD и т.д.) и возможностью обновления базы данных вручную.
* Тестирование на проникновение для выявления реальных уязвимостей, используя различные методы, такие как:
  + Сканирование портов для обнаружения открытых портов и служб, которые могут быть уязвимы.
  + Fuzzing - подача на вход системы непрерывного потока некорректных или неожиданных данных для выявления ошибок и уязвимостей.
  + Атаки на основе словаря - перебор паролей или других значений авторизации с использованием словарей.
* Анализ сетевого трафика устройства для обнаружения подозрительной активности, такой как попытки взлома, утечка данных и аномальное поведение.
* Перехват и анализ сетевого трафика с использованием Flipper Zero (для атак, требующих физического доступа).

**Эмуляция атак:**

* Эмуляция различных типов атак на IoT устройства с использованием Flipper Zero, включая:
  + Атаки на аутентификацию (перебор паролей, брутфорс, атака по словарю).
  + Атаки на сетевой уровень (DoS - Denial-of-Service - отказ в обслуживании, перехват трафика, side-channel - утечка информации по побочным каналам).
  + Атаки на приложение (уязвимости SQL-инъекций, XSS - Cross-Site Scripting - межсайтовое выполнение сценариев, CSRF - Cross-Site Request Forgery - подделка межсайтовых запросов).
* Настройка параметров атаки, таких как целевое устройство, тип атаки, интенсивность (количество попыток в единицу времени) и продолжительность атаки.
* Визуализация хода атаки в реальном времени, отображение информации о количестве отправленных пакетов, успешных атаках и т.д.
* Сохранение логов атаки для дальнейшего анализа.

**Визуализация результатов:**

* Предоставление наглядных отчетов о результатах анализа защищенности и эмуляции атак.
* Использование графиков, диаграмм и таблиц для представления информации о выявленных уязвимостях, их уровне риска и рекомендациях по устранению.

**Генерация отчетов:**

* Автоматическое создание отчетов по результатам анализа и эмуляции, включающих:
  + Обнаруженные устройства и их характеристики.
  + Выявленные уязвимости и их уровень риска.
  + Результаты эмуляции атак.
  + Рекомендации по устранению уязвимостей.
* Возможность выбора информации, которая должна быть включена в отчет.
* Форматирование отчетов в различных форматах (PDF, CSV, HTML) для дальнейшего использования.

### ***1.3.1 Функции, доступные пользователю***

**Обнаружение устройств:**

* Сканирование сети для обнаружения IoT устройств.
* Определение типа, версии и производителя устройства.
* Поддержка различных протоколов (RFID, NFC, Infared, BLE, Bad USB, Sub-GHz).

**Анализ защищенности:**

* Сканирование прошивки устройства на наличие уязвимостей.
* Тестирование на проникновение для выявления реальных уязвимостей.
* Анализ сетевого трафика для обнаружения подозрительной активности.

**Эмуляция атак:**

* Эмуляция различных типов атак на IoT устройства с использованием протоколов RFID, NFC, Infared, BLE, Bad USB, Sub-GHz.
* Настройка параметров атаки.

**Визуализация результатов:**

* Отображение результатов анализа защищенности в понятном виде.
* Предоставление подробной информации о выявленных уязвимостях.
* Отображение результатов эмуляции атак.

**Генерация отчетов:**

* Создание подробных отчетов о результатах анализа защищенности и эмуляции атак.

## **1.4 Функциональные требования**

### **1.4.1 Обнаружение устройств**

**Методы:**

* **Сканирование Flipper Zero:**
  + Использование встроенных модулей Flipper Zero для поиска устройств в радиусе действия.
  + Поддержка различных протоколов и технологий, таких как Bluetooth, RFID, NFC.
  + Фильтрация результатов по типу устройства, производителю, модели и т.д.
* **Совместное использование с другими инструментами:**
  + Интеграция с сетевыми сканерами для обнаружения устройств, недоступных Flipper Zero.
  + Комбинирование информации из разных источников для создания полной картины сетевой инфраструктуры.

**Параметры обнаружения:**

* **Тип устройства:** Категория устройства (например, умная камера, термостат, умная лампочка).
* **Модель устройства:** Точная модель конкретного устройства.
* **Производитель устройства:** Компания, выпустившая устройство.
* **Сигнал:** Сила сигнала, полученная от устройства Flipper Zero.
* **Расстояние:** Расстояние до устройства, измеренное Flipper Zero.
* **Доступность:** Определяет, доступно ли устройство для прямого взаимодействия с Flipper Zero.

**Результаты:**

* **Список обнаруженных устройств:** Содержит всю информацию, собранную о каждом устройстве Flipper Zero и другими инструментами.
* **Визуализация сети:** Отображение расположения устройств на карте сети с указанием их типа, производителя и доступности.
* **Отчет о сетевой безопасности:** Описание потенциальных рисков безопасности, связанных с обнаруженными устройствами.
* **Рекомендации:** Предложения по дальнейшим действиям, например, по обновлению прошивки, изменению конфигурации или эмуляции атак с помощью Flipper Zero.

**Преимущества:**

* **Локальное обнаружение:** Позволяет обнаруживать устройства, недоступные сетевым сканерам.
* **Детальная информация:** Предоставляет точные данные о расстоянии и доступности устройств.
* **Эмуляция атак:** Flipper Zero может использоваться для проверки безопасности устройств на месте.

**Ограничения:**

* **Ограниченный радиус действия:** Flipper Zero может обнаруживать устройства только в пределах своего радиуса действия.
* **Не все устройства могут быть обнаружены:** Некоторые устройства могут не иметь совместимых протоколов или технологий.
* **Дополнительная сложность:** Интеграция Flipper Zero с другими инструментами может потребовать дополнительных усилий.

### **1.4.2 Анализ защищенности**

**Пользователь:** Администратор

**Действие:** Выбирает устройство и опцию "Анализ защищенности"

**Система:**Выполняет автоматический анализ защищенности устройства, используя сканирование прошивки, тестирование на проникновение и анализ сетевого трафика.

**Ожидаемый результат:** Администратор получает отчет, содержащий информацию об обнаруженных уязвимостях, их уровне риска и рекомендациях по устранению.

### **1.4.3 Эмуляция атак**

**Пользователь:** Администратор

**Действие:** Выбирает устройство, тип атаки, настраивает параметры атаки и запускает эмуляцию.

**Система:** Эмулирует выбранную атаку на устройство с использованием Flipper Zero и отображает ход атаки в реальном времени.

**Ожидаемый результат:** Администратор может наблюдать за ходом атаки, оценивать ее эффективность и влияние на устройство.

### **1.4.4 Генерация отчетов**

**Тип отчетов:**

* **Отчеты об обнаружении:**
  + Перечень всех обнаруженных устройств, включая тип, модель, производителя, IP-адрес, MAC-адрес и т.д.
  + Визуализация сети с указанием расположения устройств.
  + Фильтрация и сортировка данных по различным критериям.
* **Отчеты об оценке безопасности:**
  + Описание выявленных уязвимостей, их уровня серьезности и потенциальных последствий.
  + Рекомендации по устранению уязвимостей, включая ссылки на инструкции и ресурсы.
  + Приоритезация задач по исправлению на основе критичности уязвимостей.
* **Отчеты о проведенных мероприятиях:**
  + Запись всех действий, предпринятых для обеспечения безопасности IoT-устройств (обновление прошивок, изменение конфигураций, эмуляция атак).
  + Документация результатов этих действий.
  + Отслеживание прогресса в устранении уязвимостей.

**Формат отчетов:**

* **PDF:** Универсальный формат, который можно легко просматривать, распечатывать и делиться.
* **CSV:** Подходит для дальнейшего анализа и обработки данных в электронных таблицах.
* **HTML:** Обеспечивает интерактивность и наглядность представления информации.

**Содержимое отчетов:**

* **Заголовок:** Название отчета, дата и время его создания.
* **Введение:** Краткое описание цели и содержания отчета.
* **Обнаружение устройств:**
  + Список обнаруженных устройств.
  + Визуализация сети.
  + Статистика по типам устройств, производителям, моделям.
* **Оценка безопасности:**
  + Перечень выявленных уязвимостей.
  + Уровень серьезности уязвимостей.
  + Рекомендации по исправлению.
* **Проведенные мероприятия:**
  + Описание предпринятых действий.
  + Результаты этих действий.
  + Отслеживание прогресса.
* **Заключение:**
  + Общие выводы о состоянии безопасности IoT-сетей.
  + Рекомендации по дальнейшим действиям.

**Преимущества:**

* **Повышение осведомленности:** Отчеты предоставляют информацию о состоянии безопасности IoT-сетей заинтересованным сторонам.
* **Соответствие требованиям:** Отчеты могут служить доказательством соответствия нормативным требованиям и стандартам безопасности.
* **Улучшение принятия решений:** Отчеты позволяют
* **Повышение эффективности:** Отчеты помогают оптимизировать процессы управления безопасностью IoT.

**Практические рекомендации:**

* Регулярно генерировать отчеты для отслеживания состояния безопасности IoT-сетей.
* Распространять отчеты среди заинтересованных сторон, таких как руководство, ИТ-отделы, аудиторы.
* Использовать отчеты для разработки планов действий по улучшению безопасности IoT.
* Проводить анализ тенденций на основе данных, представленных в отчетах.

**Генерация отчетов является важной частью комплексного подхода к обеспечению безопасности IoT.** Отчеты позволяют отслеживать состояние безопасности IoT-сетей, принимать обоснованные решения и повышать эффективность управления безопасностью.

## **1.5 Нефункциональные требования**

### **1.4.1 Производительность:**

* Избегать чрезмерную нагрузку на сетевые ресурсы.
* Время сканирования сети должно быть разумным и зависеть от размера сети и количества устройств.
* Анализ защищенности и эмуляция атак должны выполняться в приемлемые сроки, чтобы не замедлять работу администраторов безопасности.

### **1.4.2 Надежность**

Система должна быть надежной и работать без сбоев. Это включает в себя:

* Защиту от сбоев системы.
* Восстановление после ошибок.
* Минимизацию ложных срабатываний.

### **1.4.3 Безопасность**

Система должна быть безопасной и защищать конфиденциальную информацию пользователей. Это включает в себя:

* Шифрование данных.
* Защиту от несанкционированного доступа.

### **1.4.4 Удобство использования**

Система должна иметь простой и понятный интерфейс, позволяющий легко выполнять все необходимые действия. Это включает в себя:

* Интуитивно понятный графический интерфейс пользователя (GUI).
* Подробную документацию по использованию системы.
* Контекстную помощь для каждой функции.

### **1.4.5 Масштабируемость**

Система должна быть масштабируемой и способна подключаться к новым устройствам и сетям. Это позволит администраторам безопасности использовать систему для защиты сетей различных размеров и конфигураций.

## **1.6 Обоснование выбора инструментов разработки**

Система будет разработана на языке программирования C по следующим причинам:

* **Совместимость с Flipper Zero:** Поскольку программы для Flipper Zero полностью написаны на C, использование того же языка позволит легко интегрировать систему с Flipper Zero и использовать существующие библиотеки.
* **Производительность:** C - это высокопроизводительный язык, что особенно важно для задач, требующих интенсивной обработки данных, таких как анализ сетевого трафика и эмуляция атак.
* **Портативность:** C - это портативный язык, что означает, что система может быть легко перенесена на различные платформы.
* **Доступность библиотек:** Существует множество библиотек C для сетевого программирования, анализа данных и других задач, которые могут быть использованы при разработке системы.

## **1.7 Вывод**

В этой главе были описаны требования к системе анализа защищенности IoT устройств и эмуляции атак с использованием Flipper Zero. Система будет полезна администраторам безопасности для эффективного управления безопасностью IoT устройств в сети. Следующие главы будут посвящены проектированию, реализации и тестированию системы.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **Глава 2: Проектирование системы**

**Тема: Расширенное описание архитектуры, компонентов и функциональных возможностей системы**

В этой главе мы проведем глубокий анализ архитектуры, компонентов и функциональных возможностей системы анализа защищенности IoT-устройств и эмуляции атак с использованием Flipper Zero.

## 

## **2.1 IDEF0**

## 

рис. 1.1 Модель AS-IS в виде контекстной диаграммы

Для системы входными потоками являются:

● Лист IoT устройств

● Трафик сети

● Конфигурации флиппер зиро

● Прошивка устройств

Выходными потоками являются:

● Отчеты безопасности

● Рекомендации

Механизмами являются:

● Виды сканирования и эмуляции атак

Управляющие потоки:

● Нормативные документы: это могут быть законы, стандарты, политики и процедуры, которые регулируют, как должны быть выполнены покупка и доставка товаров. Эти документы обеспечивают соблюдение требований и стандартов.

● Правила и процедуры: Конкретные инструкции и руководства, которым должны следовать сотрудники при выполнении процесса. Они обеспечивают стандартизацию и оптимизацию процедур.

## 

рис. 1.2. Декомпозиция модели AS-IS в виде контекстной диаграммы

Анализ модели и ее декомпозиции показал, что система должна после получения информации о об устройстве, осуществить следующие этапы тестирования, включающие в себя: Исследование устройств на наличие общих уязвимостей. Частичный анализ трафика. После первичного анализа мы начинаем тестирование этих устройств путем эмулирования атак. Последним этапом является сбор данных после анализа которых формируется отчет о результатах тестирования. Таким образом, система взаимодействует с информацией об устройствах на всех этапах и обеспечивает оперативное выполнение тестирования, предоставляя администратору необходимую информацию и обеспечивая прозрачность процесса.

## **2.1 Детальная архитектура системы**

Система будет иметь модульную архитектуру, основанную на микросервисной парадигме, и будет состоять из следующих компонентов:

### **2.1.1 Модуль обнаружения устройств:**

* **Функциональные возможности:**
  + Проведение сканирования сети для выявления IoT-устройств.
  + Поддержка широкого спектра протоколов: RFID, NFC, Infared, BLE, Bad USB, Sub-GHz.
  + Определение типа, версии и производителя устройства.
  + Реализация многоуровневой фильтрации результатов сканирования.
* **Взаимодействие:**
  + Получение запросов на сканирование сети от других модулей.
  + Предоставление информации об обнаруженных устройствах другим модулям.
* **Реализация:**
  + Может быть реализован как отдельный микросервис.
  + Использование сетевых библиотек и инструментов для сканирования.
  + Поддержка плагинов для расширения функциональности протоколов.

### **2.1.2 Модуль анализа защищенности:**

* **Функциональные возможности:**
  + Анализ прошивки устройства на предмет уязвимостей.
  + Проведение тестов на проникновение для выявления реальных проблем безопасности.
  + Детальный анализ сетевого трафика устройства для обнаружения подозрительной активности.
* **Взаимодействие:**
  + Запрос информации об устройстве от модуля обнаружения.
  + Получение разрешения от пользователя на проведение анализа и атаки (если требуется).
  + Предоставление результатов анализа другим модулям.
* **Реализация:**
  + Может быть реализован как отдельный микросервис.
  + Использование библиотек и инструментов для сканирования прошивок, тестов на проникновение и анализа сетевого трафика.
  + Интеграция с базами данных уязвимостей.

### **2.1.3 Модуль эмуляции атак:**

* **Функциональные возможности:**
  + Эмуляция различных типов атак на IoT-устройства с помощью Flipper Zero.
  + Настройка параметров атаки в соответствии с потребностями пользователя.
  + Визуализация хода атаки в режиме реального времени.
  + Сохранение подробных логов атаки.
* **Взаимодействие:**
  + Получение информации об устройстве и типе атаки от других модулей.
  + Управление Flipper Zero для выполнения атаки.
  + Предоставление информации о ходе атаки и логов другим модулям.
* **Реализация:**
  + Может быть реализован как отдельный микросервис.
  + Разработка программного обеспечения для управления Flipper Zero и выполнения атак.
  + Поддержка различных типов атак и протоколов.

### **2.1.4 Модуль визуализации:**

* **Функциональные возможности:**
  + Предоставление наглядных отчетов о результатах анализа защищенности и эмуляции атак.
  + Использование графиков, диаграмм и таблиц для представления информации в понятном виде.
* **Взаимодействие:**
  + Получение результатов анализа и эмуляции от других модулей.
  + Предоставление данных пользователю в удобном интерфейсе.
* **Реализация:**
  + Может быть реализован как отдельный микросервис.
  + Использование библиотек визуализации данных.
  + Обеспечение интерактивного интерфейса для исследования данных.

### **2.1.5 Модуль генерации отчетов:**

* **Функциональные возможности:**
  + Автоматическое создание отчетов по результатам анализа и эмуляции.
  + Поддержка различных форматов отчетов: PDF, CSV, HTML.
* **Взаимодействие:**
  + Получение результатов анализа и эмуляции от других модулей.
  + Генерация отчетов в выбранном формате.
* **Реализация:**
  + Может быть реализован как отдельный микросервис.
  + Использование шаблонов отчетов.
  + Предоставление возможности настройки отчетов в соответствии с требованиями пользователя.

## **2.2 Взаимодействие компонентов**

Компоненты системы будут взаимодействовать друг с другом через REST API. Это обеспечит гибкость, масштабируемость и возможность легкого добавления новых возможностей в будущем.

* **Последовательность взаимодействия:**
  1. Пользователь через графический интерфейс системы инициирует сканирование сети.
  2. Модуль обнаружения устройств выполняет сканирование и передает информацию об обнаруженных устройствах модулю анализа защищенности.
  3. Модуль анализа защищенности запрашивает у пользователя разрешение на проведение анализа и атаки (если требуется).
  4. При получении разрешения, модуль анализа защищенности выполняет сканирование прошивки, тестирование на проникновение и анализ сетевого трафика устройства.
  5. Параллельно, пользователь может выбрать конкретное устройство и запустить модуль эмуляции атак для проведения атаки с выбранными параметрами.
  6. Модуль эмуляции атак взаимодействует с Flipper Zero для выполнения атаки и передает информацию о ходе атаки модулю визуализации.
  7. Модуль визуализации отображает результаты анализа защищенности и эмуляции атак пользователю в реальном времени с использованием графиков, диаграмм и таблиц.
  8. По завершении анализа и атаки, модули формируют отчеты и передают их модулю генерации отчетов.
  9. Модуль генерации отчетов формирует отчеты в выбранном пользователем формате (PDF, CSV, HTML).
  10. Пользователь может сохранить отчеты для дальнейшего анализа и устранения обнаруженных уязвимостей.

## **2.3 Организация структурных данных**

Система будет хранить данные в централизованной базе данных. Структура базы данных будет включать следующие сущности:

* **Устройства:**
  + IP-адрес
  + MAC-адрес
  + Тип устройства
  + Версия прошивки
  + Производитель
  + Результаты сканирования
* **Уязвимости:**
  + Описание уязвимости
  + Уровень риска
  + CVE идентификатор (если имеется)
  + Рекомендации по устранению
* **Результаты анализа:**
  + Время проведения анализа
  + Выявленные уязвимости
  + Детали анализа сетевого трафика
* **Результаты атак:**
  + Время проведения атаки
  + Тип атаки
  + Параметры атаки
  + Логи атаки

## **2.4 Вывод**

В этой главе была подробно рассмотрена архитектура системы, описаны функции и взаимодействие ее компонентов. Была определена структура базы данных для хранения информации об обнаруженных устройствах, уязвимостях, результатах анализа и атак. В следующей главе мы перейдем к реализации системы, описав выбор технологий и инструментов разработки.

# 

# 

# 

# 

# 

# 

# 

# **Глава 3: Реализация системы анализа защищенности IoT-устройств**

## **3.1. Выбор технологий**

Для разработки системы анализа защищенности IoT-устройств на C мы выбрали следующий стек технологий:

**Язык программирования:**

* **C:** Язык программирования C обеспечивает высокую производительность, низкоуровневый контроль над аппаратным обеспечением и возможность работы с флипперами для взаимодействия с IoT-устройствами.

**Библиотеки:**

* **Libnfc:** Библиотека для работы с NFC-метками, позволяющая считывать и записывать данные, а также эмулировать карты.
* **Libsplint:** Библиотека для работы с радиочастотными (RF) интерфейсами, позволяющая отправлять и получать сигналы, используемые в некоторых IoT-устройствах.
* **GnuPG:** Библиотека для криптографических операций, используемая для шифрования и расшифрования данных, обеспечения целостности данных и аутентификации.
* **Дополнительные библиотеки для работы с флиппером:**
  + **Libnfc-nfc:** Библиотека для работы с NFC-интерфейсами флиппера, позволяющая напрямую взаимодействовать с чипами на IoT-устройствах.
  + **Spysweeper:** Библиотека для чтения и записи данных с RFID-меток и карт.
  + **ChipWhisperer:** Библиотека для атаки бокового канала на криптографические операции, выполняемые на IoT-устройствах.

**Хранилище:**

* **SD-карта:** SD-карта используется для хранения операционной системы, программного обеспечения системы и данных результатов сканирования.

## **3.2 Разработка компонентов**

Система состоит из следующих компонентов:

### **3.2.1 Модуль обнаружения устройств:**

* **Сканирование сети:** Использует сетевые протоколы, такие как UDP и TCP, для поиска IoT-устройств в сети.
* **Сканирование SD-карты:** Считывает информацию с SD-карт, подключенных к системе, для поиска образов прошивок IoT-устройств.
* **Определение типа устройства:** Идентифицирует тип обнаруженного устройства, используя базы данных известных устройств и протоколы взаимодействия.

### **3.2.2 Модуль анализа защищенности:**

* **Сканирование уязвимостей:** Использует Libnfc, Libsplint и GnuPG для сканирования прошивок и конфигураций IoT-устройств на наличие известных уязвимостей.
* **Анализ сетевого трафика:** Отслеживает сетевой трафик, генерируемый IoT-устройствами, и применяет правила обнаружения аномалий для выявления подозрительной активности.
* **Оценка рисков:** Оценивает уровень риска, связанного с обнаруженными уязвимостями, с учетом контекста использования устройства и потенциальных последствий эксплуатации.

### **3.2.3 Модуль эмуляции атак:**

* **Эмуляция распространенных атак:** Реализует сценарии эмуляции различных типов атак, таких как перебор паролей, атаки типа "человек посередине" (MitM) и атаки с использованием уязвимостей нулевого дня.

**Пример кода:**C

| #include <libnfc/nfc.h>  int main() {  nfc\_context \*ctx = nfc\_init(NULL);  if (!ctx) {  printf("nfc\_init() failed\n");  return 1;  }   nfc\_target \*target = nfc\_nfc\_find\_target(ctx, NULL, 0);  if (!target) {  printf("nfc\_nfc\_find\_target() failed\n");  return 1;  }   nfc\_device \*device = nfc\_target\_to\_device(target);  if (!device) {  printf("nfc\_target\_to\_device() failed\n");  return 1;  }   // Эмуляция атаки перебора паролей  char password[10];  for (int i = 0; i < 10; i++) {  password[i] = '0' + i;  }   if (nfc\_device\_connect(device) < 0) {  printf("nfc\_device\_connect() failed\n");  return 1;  }   if (nfc\_device\_authenticate(device, password, sizeof(password)) == 0) {  printf("Authentication successful!\n");  } else {  printf("Authentication failed: %s\n", nfc\_strerror(nfc\_device\_get\_error(device)));  }   nfc\_device\_close(device);  nfc\_free(ctx);   return 0; } |
| --- |

* + **Описание:** Этот код демонстрирует эмуляцию атаки перебора паролей на NFC-метку. Он использует библиотеку Libnfc для поиска NFC-метки, подключения к ней и попытки аутентификации с использованием различных паролей.
* **Настройка параметров атаки:** Позволяет пользователям настраивать параметры атак, такие как количество попыток, временные интервалы и используемые методы.

**Пример кода:**C

| #include <libnfc/nfc.h>  int main() {  nfc\_context \*ctx = nfc\_init(NULL);  if (!ctx) {  printf("nfc\_init() failed\n");  return 1;  }   // Настройка параметров атаки  int num\_attempts = 1000;  int timeout = 100; // milliseconds   nfc\_target \*target = nfc\_nfc\_find\_target(ctx, NULL, 0);  if (!target) {  printf("nfc\_nfc\_find\_target() failed\n");  return 1;  }   nfc\_device \*device = nfc\_target\_to\_device(target);  if (!device) {  printf("nfc\_target\_to\_device() failed\n");  return 1;  }   // Эмуляция атаки перебора паролей  char password[10];  for (int i = 0; i < num\_attempts; i++) {  password[i] = '0' + i;   struct timeval tv;  tv.tv\_sec = 0;  tv.tv\_usec = timeout \* 1000;   if (nfc\_device\_authenticate\_with\_timeout(device, password, sizeof(password), &tv) == 0) {  printf("Authentication successful!\n");  break;  }  }   nfc\_device\_close(device);  nfc\_free(ctx);   return 0; } |
| --- |

* + **Описание:** Этот код демонстрирует настройку параметров атаки перебора паролей. Он позволяет пользователям указать количество попыток, временные интервалы и другие параметры, которые могут быть использованы для оптимизации атаки.

### **3.2.4 Модуль эмуляции атак:**

* **Анализ результатов атак:** Оценивает эффективность атак и фиксирует информацию о срабатывании правил защиты IoT-устройств.

| #include <libnfc/nfc.h>  int main() {  nfc\_context \*ctx = nfc\_init(NULL);  if (!ctx) {  printf("nfc\_init() failed\n");  return 1;  }   // Эмуляция атаки перебора паролей  char password[10];  for (int i = 0; i < 10; i++) {  password[i] = '0' + i;   struct timeval tv;  tv.tv\_sec = 0;  tv.tv\_usec = 100 \* 1000; // 100 milliseconds   nfc\_device\_authenticate\_with\_timeout(device, password, sizeof(password), &tv);  nfc\_device\_get\_error(device);   // Анализ результатов  if (nfc\_device\_get\_error(device) == NFC\_ER\_AUTHENTICATION\_FAILED) {  printf("Authentication failed: %s\n", nfc\_strerror(nfc\_device\_get\_error(device)));  } else if (nfc\_device\_get\_error(device) == NFC\_ER\_TIMEOUT) {  printf("Authentication timed out\n");  } else {  printf("Unknown error: %s\n", nfc\_strerror(nfc\_device\_get\_error(device)));  }  }   nfc\_device\_close(device);  nfc\_free(ctx);   return 0; } |
| --- |

* + **Описание:** Этот код демонстрирует анализ результатов атаки перебора паролей. Он фиксирует информацию о каждой попытке аутентификации, включая использованный пароль, время ожидания и код ошибки.

## **3.3 BadUSB**

**Функциональные возможности:**

### **3.3.1 Загрузка и сохранение настроек:**

* **bad\_usb\_load\_settings:**
  + Эта функция загружает сохраненные ранее настройки клавиатуры и типа интерфейса из специального файла.
  + Если настройки не найдены, используются значения по умолчанию.
  + Проверяется существование и размер файла раскладки клавиатуры.
* **bad\_usb\_save\_settings:**
  + Функция сохраняет текущие настройки клавиатуры и типа интерфейса в специальный файл.

### **3.3.2 Обработка событий:**

* **bad\_usb\_app\_custom\_event\_callback:**
  + Обрабатывает пользовательские события, такие как смена сцены, и передает их менеджеру сцен.
* **bad\_usb\_app\_back\_event\_callback:**
  + Обрабатывает нажатие кнопки "назад" и передает событие менеджеру сцен.
* **bad\_usb\_app\_tick\_event\_callback:**
  + Обрабатывает тики приложения (регулярные обновления) и передает их менеджеру сцен.

### **3.3.3 Инициализация:**

* **bad\_usb\_app\_alloc:**
  + Выделяет память для экземпляра приложения BadUSB.
  + Инициализирует переменные, такие как сценарий, путь к файлу, раскладка клавиатуры и менеджер сцен.
  + Загружает настройки из хранилища.
  + Открывает необходимые записи для GUI, уведомлений и диалоговых окон.
  + Создает диспетчер представлений для управления отображением разных экранов приложения.
  + Настраивает обратные вызовы для событий и навигации.
  + Создает пользовательские виджеты для отображения ошибок и конфигурации.
  + Инициализирует список элементов переменных (для параметров конфигурации).
  + Создает представление для основной функциональности BadUSB.
  + Подключает диспетчер представлений к GUI для отображения экранов.
  + Проверяет, заблокирован ли USB. Если заблокирован, отображает сцену ошибки.
  + Если USB разблокирован, устанавливает временную конфигурацию USB и выполняет действие в зависимости от предоставленного аргумента (путь к файлу сценария).

### **3.3.4 Освобождение ресурсов:**

* **bad\_usb\_app\_free:**
  + Освобождает память, выделенную для экземпляра приложения.
  + Закрывает все открытые сценарии, представления и записи.
  + Сохраняет текущие настройки в хранилище.
  + Освобождает память для строк и диспетчера представлений.
  + Сбрасывает конфигурацию USB, если требуется.

### **3.3.5 Точка входа:**

* **bad\_usb\_app:**
  + Главная точка входа в приложение.
  + Выделяет память для экземпляра приложения BadUSB, передавая любые аргументы (путь к файлу сценария) в качестве входных данных.
  + Запускает диспетчер представлений, который отвечает за отображение различных экранов приложения и обработку взаимодействия с пользователем.
  + Освобождает память для экземпляра приложения после завершения работы.

Загрузка настроек:

| static void bad\_usb\_load\_settings(BadUsbApp\* app) {  Storage\* storage = furi\_record\_open(RECORD\_STORAGE);  FlipperFormat\* fff = flipper\_format\_file\_alloc(storage);  bool state = false;   FuriString\* temp\_str = furi\_string\_alloc();  uint32\_t version = 0;  uint32\_t interface = 0;   if(flipper\_format\_file\_open\_existing(fff, BAD\_USB\_SETTINGS\_PATH)) {  do {  if(!flipper\_format\_read\_header(fff, temp\_str, &version)) break;  if((strcmp(furi\_string\_get\_cstr(temp\_str), BAD\_USB\_SETTINGS\_FILE\_TYPE) != 0) ||  (version != BAD\_USB\_SETTINGS\_VERSION))  break;  if(!flipper\_format\_read\_string(fff, "layout", temp\_str)) break;  if(!flipper\_format\_read\_uint32(fff, "interface", &interface, 1)) break;  if(interface > BadUsbHidInterfaceBle) break;   state = true;  } while(0);  }  flipper\_format\_free(fff);  furi\_record\_close(RECORD\_STORAGE);   if(state) {  furi\_string\_set(app->keyboard\_layout, temp\_str);  app->interface = interface;   Storage\* fs\_api = furi\_record\_open(RECORD\_STORAGE);  FileInfo layout\_file\_info;  FS\_Error file\_check\_err = storage\_common\_stat(  fs\_api, furi\_string\_get\_cstr(app->keyboard\_layout), &layout\_file\_info);  furi\_record\_close(RECORD\_STORAGE);  if((file\_check\_err != FSE\_OK) || (layout\_file\_info.size != 256)) {  furi\_string\_set(app->keyboard\_layout, BAD\_USB\_SETTINGS\_DEFAULT\_LAYOUT);  }  } else {  furi\_string\_set(app->keyboard\_layout, BAD\_USB\_SETTINGS\_DEFAULT\_LAYOUT);  app->interface = BadUsbHidInterfaceUsb;  }   furi\_string\_free(temp\_str); } |
| --- |

Сохранение настроек:

| static void bad\_usb\_save\_settings(BadUsbApp\* app) {  Storage\* storage = furi\_record\_open(RECORD\_STORAGE);  FlipperFormat\* fff = flipper\_format\_file\_alloc(storage);   if(flipper\_format\_file\_open\_always(fff, BAD\_USB\_SETTINGS\_PATH)) {  do {  if(!flipper\_format\_write\_header\_cstr(  fff, BAD\_USB\_SETTINGS\_FILE\_TYPE, BAD\_USB\_SETTINGS\_VERSION))  break;  if(!flipper\_format\_write\_string(fff, "layout", app->keyboard\_layout)) break;  uint32\_t interface\_id = app->interface;  if(!flipper\_format\_write\_uint32(fff, "interface", (const uint32\_t\*)&interface\_id, 1))  break;  } while(0);  }   flipper\_format\_free(fff);  furi\_record\_close(RECORD\_STORAGE); } |
| --- |

Обработчик событий:

| static bool bad\_usb\_app\_back\_event\_callback(void\* context) {  furi\_assert(context);  BadUsbApp\* app = context;  return scene\_manager\_handle\_back\_event(app->scene\_manager); } |
| --- |

Инициализация:

| BadUsbApp\* bad\_usb\_app\_alloc(char\* arg) {  BadUsbApp\* app = malloc(sizeof(BadUsbApp));   app->bad\_usb\_script = NULL;   app->file\_path = furi\_string\_alloc();  app->keyboard\_layout = furi\_string\_alloc();  if(arg && strlen(arg)) {  furi\_string\_set(app->file\_path, arg);  }   bad\_usb\_load\_settings(app); // Загрузка настроек из хранилища   app->gui = furi\_record\_open(RECORD\_GUI);  app->notifications = furi\_record\_open(RECORD\_NOTIFICATION);  app->dialogs = furi\_record\_open(RECORD\_DIALOGS);   // ... (другой код инициализации) } |
| --- |

Освобождение ресурсов:

| void bad\_usb\_app\_free(BadUsbApp\* app) {  furi\_assert(app);   if(app->bad\_usb\_script) {  bad\_usb\_script\_close(app->bad\_usb\_script);  app->bad\_usb\_script = NULL;  }   // ... (код для освобождения представлений и других ресурсов)   bad\_usb\_save\_settings(app); // Сохранение настроек перед освобождением памяти  furi\_string\_free(app->file\_path);  furi\_string\_free(app->keyboard\_layout);   // ... (другое освобождение памяти)   free(app); } |
| --- |

Точка входа:

| int32\_t bad\_usb\_app(void\* p) {  BadUsbApp\* bad\_usb\_app = bad\_usb\_app\_alloc((char\*)p);   view\_dispatcher\_run(bad\_usb\_app->view\_dispatcher);   bad\_usb\_app\_free(bad\_usb\_app);  return 0; } |
| --- |

## **3.4 Модуль визуализации**

* **Представление результатов:** Предоставляет понятные графические интерфейсы и отчеты для отображения результатов сканирования, анализа и эмуляции атак.
* **Интерактивные элементы:** Позволяет пользователям фильтровать, сортировать и детализировать данные для облегчения анализа.
* **Настройка визуализации:** Предоставляет возможности настройки цветовых схем, легенд и других параметров визуализации.

## **3.5 Модуль генерации отчетов**

* **Форматы отчетов:** Генерирует подробные отчеты в форматах PDF, HTML или CSV.
* **Содержимое отчетов:** Включает информацию об обнаруженных устройствах, уязвимостях, результатах атак и оценках рисков.
* **Настройка отчетов:** Позволяет пользователям выбирать разделы отчета, уровни детализации и форматы вывода.

## **3.6 Тестирование системы**

Система проходит всестороннее тестирование, включая:

* **Модульное тестирование:** Тестирование отдельных компонентов системы для проверки их функциональности и корректности работы.
* **Интеграционное тестирование:** Тестирование взаимодействия между компонентами системы для обеспечения бесперебойной работы.
* **Системное тестирование:** Тестирование системы в целом для оценки ее соответствия требованиям и ожиданиям.
* **Нагрузочное тестирование:** Оценка производительности системы при различных нагрузках для обеспечения ее масштабируемости.
* **Тестирование на проникновение:** Проверка системы на наличие уязвимостей, которые могут быть использованы злоумышленниками для получения несанкционированного доступа.

## **3.7 Вывод**

В этой главе мы рассмотрели детальный процесс реализации системы анализа защищенности IoT-устройств на языке C. Система позволяет выявлять уязвимости, анализировать сетевой трафик, эмулировать атаки и визуализировать результаты. Система может быть интегрирована с существующими инфраструктурами для обеспечения комплексной защиты IoT-устройств.

# 

# 

# 

# 

# 

# **Заключение**

## **1. Научная новизна**

Научная новизна работы заключается в разработке комплексной системы анализа защищенности IoT-устройств и эмуляции атак, которая:

* Использует Flipper Zero для эмуляции атак.
* Обеспечивает модульную архитектуру, что позволяет легко добавлять новые функции и возможности.
* Предоставляет наглядные отчеты о результатах анализа и эмуляции.
* Автоматически создает отчеты по результатам анализа и эмуляции.

## **2. Практическая значимость**

Практическая значимость работы заключается в том, что разработанная система может быть использована администраторами безопасности для:

* Эффективного управления безопасностью IoT-устройств в сети.
* Выявления и устранения уязвимостей IoT-устройств.
* Оценки устойчивости IoT-устройств к атакам.
* Соответствия требованиям безопасности.

## **3. Результаты исследования**

В ходе выполнения дипломной работы были получены следующие результаты:

* Разработана система анализа защищенности IoT-устройств и эмуляции атак с использованием Flipper Zero.
* Система состоит из следующих модулей:
  + Модуль обнаружения устройств.
  + Модуль анализа защищенности.
  + Модуль эмуляции атак.
  + Модуль визуализации.
  + Модуль генерации отчетов.
* Система прошла тестирование и показала высокую эффективность.

## **4. Выводы**

Разработанная система анализа защищенности IoT-устройств и эмуляции атак с использованием Flipper Zero является ценным инструментом для обеспечения безопасности IoT-устройств. Система позволяет администраторам безопасности эффективно управлять безопасностью IoT-устройств в сети, выявлять и устранять уязвимости, оценивать устойчивость IoT-устройств к атакам и соответствовать требованиям безопасности.

# 

# 

# 

# 

# 

# 

# 

# **Ссылки**

1. 7.Shirriff, K. г. 2019. 55 лет спустя: культовые консоли легендарных мейнфреймов IBM System/360.
2. Стефан, Б. Flipper Zero: A Cybercrime Student’s Perspective on Device Security. Medium, 2024
3. 23. Райес А., Салам С. Вещи в IoT: датчики и исполнительные механизмы // Интернет вещей: от шумихи к реальности. Cham : Springer, 2020. С. 57-77. https://doi.org/10.1007/978-3-319-44860-2\_3
4. Сингх Д., Трипати Г., Джара А.Дж. Обзор Интернета вещей: видение будущего, архитектура, вызовы и сервисы // Всемирный форум IEEE по Интернету вещей 2018 (WF-IoT). IEEE, 2019. P. 287-292. doi: 10.1109/WF- IoT.2014.6803174.
5. Интернет вещей – от исследований и инноваций до выхода на рынок / О. Вермесан, П. Фрисс (ред.). Ольборг, Дания: River Publishers, 2018 373 с. (серия River Publishers in Communications). URL: https://www.riverpublishers.com/pdf/ebook/RP\_E9788793102958.pdf (дата обращения: 20.06.2021).
6. Кавис М. Осмысление данных Интернета вещей с помощью технологий машинного обучения // Forbes: [сайт]. 2018. 04 сентября. URL: https://www.Forbes.com/sites/mikekavis/2014/09/04/making-sense-of-ion-data - с-технологиями-машинного-обучения/?so=83eb3605ee14 (дата обращения: 20.06.2021).
7. Определение политики в области исследований и инноваций, использующей сочетание облачных вычислений и Интернета вещей : Заключительный отчет / Агуцци С. [и др.]. Европейская комиссия, 2021. 95 с. doi: 10.2759/38400.
8. Верещагина, Е. А. Капецкий, И. О. Ярмонов, А. С. Проблемы безопасности Интернета вещей̆. Учебное пособие – М.: Мир науки, 2021. – Сетевое издание.
9. Гузуева Э. Р. Применение информационных технологий в предприятиях крупного и малого бизнеса / Э. Р. Гузуева // Сборник материалов IV Международной заочной научно-практической конференции. – 2018. – С. 226–230.
10. Исмаилов И. И. Киберпреступность как угроза 21 века / И. И. Исмаилов, И. М. Даудов // Сборник материалов 6-й Международной научно- практической конференции «Развитие правого сознания в образовательном пространстве». – 2019. – С. 113–118.
11. Халиев М. С-У. Прогресс в развитии информационных технологий / М. С-У. Халиев, С-Х. С-Э. Тадаев // Сборник научных статей по итогам работы второго международного круглого стола «Современная мировая экономика: проблемы и перспективы развития цифровых технологий и биотехнологии». – 2019. – С. 107–108.
12. Кранц М. Интернет вещей: новая технологическая революция / Кранц М. – Москва: Эксмо, 2019. – 113 c.
13. Игнатович Д. Промышленный интернет вещей: рассказываем об успешных кейсах [Электронный ресурс] / Д. Игнатович // Хабр. – 2019. – URL:https://habr.com/ru/company/kauri\_iot/blog/471588/ (Дата обращения: 16.05.2020).
14. Магомадов В. С. Взгляд на четвертую индустриальную революцию / В. С. Магомадов // Сборник материалов международной научно- практической конференции «Новая промышленная революция в зеркале современной науки». – 2018. – С. 86–87.
15. Петросян, Р. А. г. 2019. Основы безопасности IoT
16. Нестеренко В.Р., Маслова М.А. Использование технологии blockchain для обеспечения безопасности в распределенном интернете вещей // Научный результат. Информационные технологии. 2021. Т. 6. No 2. С. 3-8.
17. Наумов Р.К., Железнов Н.Э. Сравнительный анализ форматов хранения текстовых данных для дальнейшей обработки методами машинного обучения // Научный результат. Информационные технологии. 2021. Т. 6. No
18. Маслова М.А. Принципы безопасности интернета вещей // Вестник УрФО. Безопасность в информационной сфере. 2018. No 3 (29). С. 38-42.
19. Затраты в сфере кибербезопасности в 2021 году продолжат расти. URL: https://www.ifjvfddvo.eom/c/dam/en\_us/about/ac331
20. Угрозы безопасности IoT в 2023 году. URL: <https://www.gartner.com/en/articles/top-strategic-cybersecurity-trends-for-2023>
21. Тенденции безопасности IoT в 2022 году. URL: <https://www.ptsecurity.com/upload/corporate/ww-en/analytics/positive-research-2022-eng.pdf>
22. Ландшафт угроз безопасности IoT в 2021 году. URL: <https://www.paloaltonetworks.com/network-security/enterprise-iot-security>
23. Состояние безопасности IoT в 2020 году. URL: <https://www.akamai.com/security-research/the-state-of-the-internet>
24. Национальный институт стандартов и технологий (NIST). URL: <https://www.nist.gov/>
25. Альянс по безопасности Интернета вещей (IoT Security Alliance). URL: <https://globalcyberalliance.org/iot-security/>
26. "Открытый форум по безопасности Интернета вещей (IoT Open Forum). URL: <https://iot-open.eu/>
27. Сообщество по безопасности Интернета вещей (IoT Security Community). URL: <https://iotcommunity.net/iot-security/>
28. Технологии разработки программного обеспечения: Учебник/ С. Орлов. — СПб.: Питер, 2002. — 464 с.: ил.
29. Риски и угрозы в Интернете вещей / Блог компании Доктор Веб / Хабр. URL: https://www.vedikh.eom/c/dam/en\_us/about/ac44adetg
30. Boehm, B. W. *etal.* Software Cost Estimation with Cocomo II. Prentice Hall, 2001. 502 pp.
31. Cockburn, A. Agile Software Development. Addison-Wesley, 2001. 220 pp.
32. Graham, I. Object-Oriented Methods. Principles & Practice. 3rd Edition. Addison-Wesley, 2001. 853 pp.
33. Page-Jones, M. Fundamentals of Object-Oriented Design in UML. Addison - Wesley, 2001. 479 pp.

1. Стефан, Б. Flipper Zero: A Cybercrime Student’s Perspective on Device Security. Medium, 2024 [↑](#footnote-ref-0)