**Министерство науки и высшего образования Российской Федерации**
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**
**(ФГБОУ ВО «АмГУ»)**

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.03.02 – Информационные системы и технологии
Направленность (профиль) образовательной программы Безопасность информационных систем

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«____»_____2022 г.

# БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка защищенного загрузчика для встраиваемых систем интернета вещей

Исполнитель
студент группы 855-об _____ (подпись, дата) И.А. Олейник

Руководитель
доцент, канд. техн. наук _____ (подпись, дата) А.В. Бушманов

Консультант
по безопасности и экологичности
доцент, канд. техн. наук _____ (подпись, дата) А.Б. Булгаков

Нормоконтроль
инженер кафедры _____ (подпись, дата) В.Н. Адаменко

Благовещенск 2022

**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**
**(ФГБОУ ВО «АмГУ»)**

Факультет математики и информатики
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
«____» _____

# З А Д А Н И Е

К выпускной квалификационной работе студента Олейник Ильи Антоновича

1. Тема дипломной работы: Разработка защищенного загрузчика для встраиваемых систем интернета вещей.

(утверждена приказом от 05.04.2022 №679-уч)

2. Срок сдачи студентом законченной работы: 21.06.2022 г.

3. Исходные данные к выпускной квалификационной работе: отчет о прохождении преддипломной практики, нормативная документация, специальная литература.

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов): обоснование необходимости разработки и определение требований, проектирование программного продукта, оценка надежности и качества функционирования объекта проектирования, описание способов защиты информации для программы, обоснование безопасности и экологичности продукта.

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.): техническое задание.

6. Консультанты по дипломной работе:

по безопасности и экологичности – Булгаков А.Б., доцент, кандидат технических

наук.

7. Дата выдачи задания: _____

Руководитель дипломной работы: Бушманов А.В., доцент, кандидат технических наук.

Задание принял к исполнению: _____

# РЕФЕРАТ

Бакалаврская работа содержит 66 с., 15 рисунков, 2 таблицы, 22 источника, 1 приложение.

ЗАЩИТА, ВСТРАИВАЕМЫЕ СИСТЕМЫ, ФАЙЛОВАЯ СИСТЕМА, КРИПТОГРАФИЯ, ПРОГРАММНЫЙ ПРОДУКТ

В данной работе были исследованы источники угроз для программного обеспечения встраиваемых систем на этапе загрузки и обновления.

Цель бакалаврской работы: разработка программного продукта для защиты встраиваемых систем от внешних угроз на этапе загрузки с применением симметричных и ассиметричных алгоритмов криптографии.

Выполнение работы включает основные этапы:

– Обоснование необходимости разработки и формирование требований к продукту;

– Проектирование и реализация программы;

– Разработка руководства пользователя;

– Описание способов защиты информации для программы;

– Обоснование безопасности и экологичности продукта.

Результатом бакалаврской работы является разработанный программный продукт, который позволяет обеспечить гарантии аутентичности и целостности исполняемого программного кода и данных в вычислительном модуле встраиваемой системы.

СОДЕРЖАНИЕ

Введение 9
1 Анализ и характеристика предметной области 10
1.1 Состояние рынка встраиваемых систем 10
1.2 Методы защиты встраиваемых систем 12
1.3 Обзор существующих решений 14
1.3.1 MCUboot 14
1.3.2 ZEPboot 14
1.3.3 wolfBoot 15
2 Проектирование загрузчика 17
2.1 Цели и задачи разработки 17
2.2 Описание функционала программного продукта 17
2.3 Описание функциональных модулей 18
2.4 Алгоритмическое обеспечение 20
2.4.1 Разработка алгоритма управления загрузкой 20
2.4.2 Разработка алгоритмов работы с файловой системой 22
2.4.3 Разработка алгоритма загрузки исполняемых файлов 23
2.5 Обоснование выбора инструментов разработки 24
3 Разработка программного продукта 29
3.1 Реализация файловой системы 29
3.2 Реализация модуля загрузки исполняемых файлов формата ELF 35
3.3 Реализация вспомогательных компонентов 37
3.4 Расчет надежности ПО 39
4 Информационная безопасность 40
4.1 Аутентификация 41
4.2 Шифрование данных 41
4.3 Модель нарушителя безопасности 41
5 Безопасность и экологичность 45
5.1 Безопасность 45

5.1.1 Требования к помещению для работы с ПЭВМ 45
5.1.2 Требования к освещению на рабочих местах с ПЭВМ 46
5.1.3 Требования к уровням шума и вибрации 48
5.1.4 Требования к микроклимату рабочего места с ПЭВМ 48
5.1.5 Требования к организации рабочих мест с ПЭВМ 49
5.1.6 Требования к ПЭВМ 50
5.2 Экологичность 52
5.3 Чрезвычайные ситуации 54
5.3.1 Действия при пожаре 54
Заключение 56
Библиографические ссылки 57
Библиографический список 58
Приложение А 61

# НОРМАТИВНЫЕ ССЫЛКИ

В настоящей бакалаврской работе использованы ссылки на следующие стандарты и нормативные документы:

ГОСТ 2.104-68 ЕСКД Основные надписи

ГОСТ 2.105-95 ЕСКД Общие требования к текстовым документам

ГОСТ 2.111-68 ЕСКД Нормоконтроль

ГОСТ 12.1.007-76 Система стандартов безопасности труда (ССБТ). Вредные вещества. Классификация и общие требования безопасности

ГОСТ 7.1-2003 Библиографическое описание документа. Общие требования и правила составления

# ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

IOT – интернет вещей;

ЭЦП – электронная цифровая подпись;

AES – (Advanced Encryption Standard) блочный алгоритм шифрования;

SHA – (Secure Hash Algorithm) семейство алгоритмов хеширования данных;

ECDSA – (Elliptic Curve Digital Signature Algorithm) алгоритм для ЭЦП основанный на эллиптических кривых;

ASLR – (Address Space Layout Randomization) алгоритм трансформации адресного пространства исполняемой программы случайным образом;

RTOS – (Real-Time Operating System)

ELF – (Executable and Linkable Format) формат исполняемых файлов использующийся на UNIX системах.

# ВВЕДЕНИЕ

Безопасность встраиваемых систем с каждым днем приобретает все большую значимость в современном мире. Статистика показывает ежегодный рост количества встраиваемых устройств. Согласно данным DataProt в анализе состояния рынка IOT устройств [3] были выявлены следующие тенденции:

– Количество активных устройств 2021 году достигло 10 миллиардов;

– Рост атак на IOT устройства в 2019 составил 300 %, а в 2020 году эти устройства составляли треть всего зараженного оборудования;

– 48 % опрошенных компаний признают, что не обладают возможностью выявления несанкционированного проникновения в IOT инфраструктуру.

Становится очевидно, что фактор количества устройств и их плохая защищенность ведут к росту производимых на них атак. Эти атаки могут иметь разные источники происхождения, но так или иначе сводятся к двум целям: кража информации (ключи шифрования, данные пользователя и т.д.) и получение контроля над устройством. Загрузчик системы является первым компонентом, который получает контроль над системой. От его защищенности зависит вся безопасность устройства.

Основными целями загрузчика являются: обновление кода, исполняемого на устройстве; аутентификация кода следующего этапа загрузки и предоставление для него специальной среды исполнения.

Для достижения этих целей необходимо:

– провести анализ возможных механизмов защиты;

– определить механизмы защиты подлежащие реализации;

– выбрать инструменты для разработки;

– разработать программный продукт.

# 1 АНАЛИЗ И ХАРАКТЕРИСТИКА ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Состояние рынка встраиваемых систем

Устройства интернета вещей уже прочно вошли в повседневную жизнь. Об этом свидетельствует статистика подключенных к интернету устройств, разделенная по категориям. В 2020 году количество IOT соединений впервые превзошло число обычных. Из 21,7 миллиарда – 11,7 миллиардов (или 54 %) являлись запросами от устройств интернета вещей. К 2025 году ожидается, что эта цифра вырастет до 30 миллиардов, а количество устройств на одного человека будет приблизительно равно 4.


Рисунок 1 – темпы роста IOT рынка

Такой быстрый рост связан с удобствами, которые приносят эти устройства. Так, например, в сфере здравоохранения IOT позволяет пациентам самостоятельно отслеживать основные показатели своего здоровья (давление, сахар в крови и т.д.), это в свою очередь позволяет более оперативно находить и лечить возникающие болезни. В сфере продаж они позволяют повысить эффективность рекламных кампаний за счет использования т.н. «маячков», а также упростить управление цепочкой поставок и управления складскими операциями.

В это же время, уровень защищенности этих устройств остается под вопросом. Организации, использующие IOT устройства, довольно медленно реагируют на угрозы безопасности. Это происходит в основном потому, что компании не обладают должными ресурсами, а безопасность встраиваемых систем имеет довольно высокий порог вхождения. Другая проблема заключается в увеличении разнообразия устройств, что создает дополнительные сложности с их управлением.

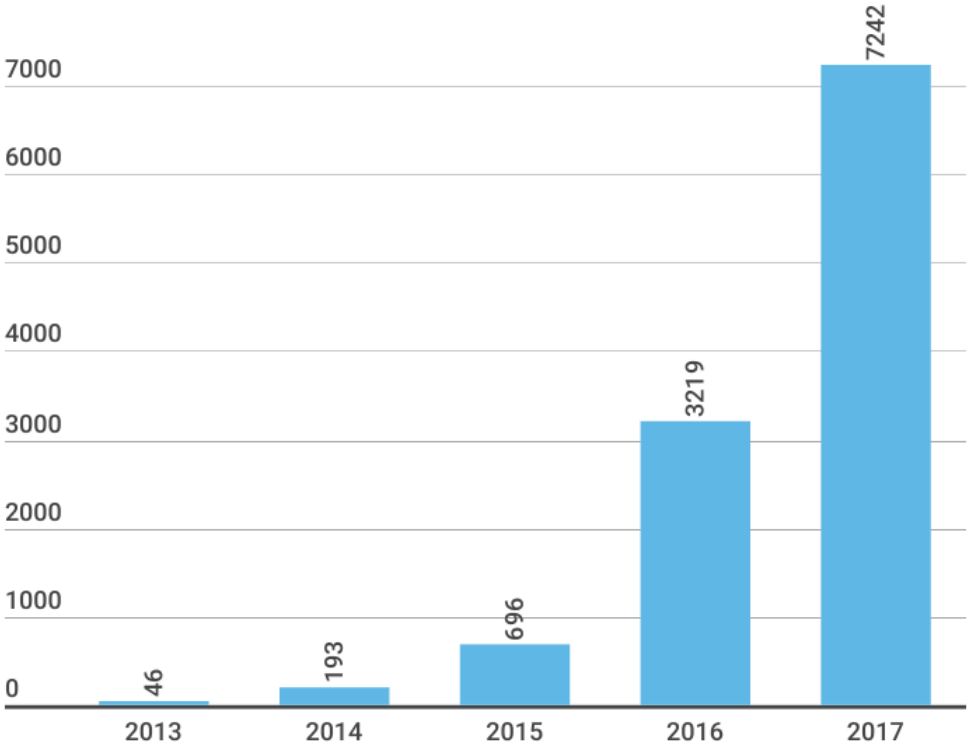


Рисунок 2 – количество образцов вредоносного ПО для IOT устройств

Исследование IOT устройств за 2020 год [4] показало, что из одного миллиона пользовательских устройств, 50 процентов имели уязвимое программное обеспечение. Большая часть из них была на 5-7 лет позади текущих обновлений, что делает эти устройства идеальной целью для злоумышленников. Другие 50 процентов устройств имели заводские учетные данные, которые могут быть легко угаданы. Для повышения безопасности требуется регулярно выпускать новые исправления, обновления прошивки и изменять данные авторизации.

### 1.2 Методы защиты встраиваемых систем

Защита встраиваемых систем делится на 2 типа:

– аппаратная – для защиты системы используются аппаратные средства, уже установленные внутри нее. Это могут быть непрозрачные криптографические подсистемы, отдельные ядра, выполняющие роль системного монитора и контроля доступа, а также механизмы защиты от модификации прошивки. Главным достоинством такого подхода является повышенная надежность по сравнению с чисто программным подходом. Но есть и недостатки. Главный из них это стоимость и более высокая сложность реализации. Так же в большинстве случаев отсутствует возможность внесения исправлений для найденных уязвимостей. Еще одним существенным недостатком является не универсальность данных систем защиты, что в общем случае означает написание ПО для нового устройства с нуля;

– программная – для защиты системы используются криптографии и специальные технологии для усложнения реализации атак на устройство. Эти методы относительно просты в реализации и достаточно эффективны. В отличии от аппаратных методов реализация программной защиты обладает свойством переносимости с платформы на платформу и позволяет легко устранять обнаруженные уязвимости. Главный недостаток – более низкая надежность защиты относительно аппаратной защиты.

Базой защиты любой информационной системы является криптография. С ее помощью можно защищать секреты системы и аутентифицировать информацию извне и внутри системы. Для этого используются алгоритмы хеширования, симметричного и ассиметричного шифрования.

Для защиты информации при ее транспортировке в и из системы используются симметричные криптографические шифры. Они помогают сокрыть конфиденциальную системы от внешнего мира. На сегодняшний день основным алгоритмом является AES в различных блочных режимах. Его надежность была доказана временем и в ближайшее время изменений не предвидится. Существуют две вариации: с длиной ключа 128 бит и 256 бит.

Для аутентификации данных в системе обычно используют хеширование и ЭЦП на базе ассиметричных алгоритмов шифрования. В качестве данных может выступать все что угодно, от программного обеспечения до сетевых пакетов. Для защищаемых данных сначала получают хэш и затем его зашифровывают с помощью приватного ключа. Существует множество популярных алгоритмов хеширования: MD-5, SHA-1, SHA-256, SHA-3, BLAKE2, BLAKE3. Однако наибольшее распространение в сфере защиты информации получили алгоритмы серии SHA. В качестве ассиметричного алгоритма шифрования (для ЭЦП) широкое распространение получили двое: RSA и ECDSA. По большей части, их основное отличие заключается длине ключа. Для RSA обычно применяются ключи с длиной в 2048-4096 бит, а для ECDSA 192-256 бит.

Могут так же применяться специальные методы программной защиты. Они являются низкоуровневыми и практически невидимы для ПО устройства. Один из них это т.н. «маркер стэка». Он позволяет обнаруживать выход за границы допустимого пространства стэка для текущей процедуры. Это позволяет обнаруживать и предотвращать некоторые классы атак, которые используют логические ошибки в коде программы. Этот механизм поддерживается всеми ведущими компиляторами. Другой технологией защиты является ASLR. Ее идея заключается в использовании всего адресного пространства аппаратной платформы как потенциальной области для исполнения кода программы. При каждой загрузке программы генератор случайных чисел определяет место ее загрузки. Это изменяет расположение регионов памяти, что усложняет проведение атак, основывающихся на знании адресов переменных и функции в памяти для последующей их модификации.

## 1.3 Обзор существующих решений

Для лучшего представления о направлении разработки были рассмотрены несколько реализаций загрузчика с упором на безопасность.

### 1.3.1 MCUboot

Данный ПО позиционируется как безопасный загрузчик для 32 битных

микроконтроллеров. В нем определяются общие интерфейсы для загрузчика и организация флэш-памяти системы. Проект содержит в себе как код самого загрузчика, так и утилиты, необходимые для создания загрузочных образов и работы с ним.

Достоинства:

– Хорошая документация;

– Поддержка различных RTOS;

– Поддержка нескольких слотов для образа прошивки;

– Поддержка шифрования и ЭЦП для образа прошивки;

– Поддержка исполнения кода из оперативной памяти.

Недостатки:

– Нет поддержки позиционно независимого кода;

– Нет возможности простого добавления файлов, используемых прошивкой;

– Нестандартный формат исполняемого файла прошивки.

1.3.2 ZEPboot

Данный загрузчик основан на RTOS Zephyr. Он может запускать и обновлять образы прошивки. Имеется поддержка двух типов образов:

– приложение, которое предоставляет функции обновления и основные функции (устанавливается в слот 0);

– приложение, которое предоставляет только функции обновления (устанавливается в слот 0) и основное приложение, которое будет исполняться из слота 1.

Это повышает отказоустойчивость при обновлении.

Достоинства:

– Поддержка нескольких слотов для образа прошивки;

– Поддержка шифрования и ЭЦП для образа прошивки;

– Поддержка исполнения кода из оперативной памяти.

Недостатки:

– Нет поддержки позиционно независимого кода;

– Нет возможности простого добавления файлов, используемых прошивкой;

– Нестандартный формат исполняемого файла прошивки;

– Нет поддержки различных RTOS;

– Повышенный износ флэш-памяти из-за алгоритма переключения слотов.

1.3.3 wolfBoot

Загрузчик предназначен для 32 битных микроконтроллеров. Он имеет минималистичный аппаратный программный уровень, что ведет к отсутствию необходимости написания специального кода для поддержки RTOS. Написан командой разработчиков библиотеки wolfSSL.

Достоинства:

– Хорошая документация;

– Отсутствие необходимости интеграционного кода для RTOS;

– Поддержка нескольких слотов для образа прошивки;

– Поддержка шифрования и ЭЦП для образа прошивки;

– Поддержка доверенного модуля платформы.

– Поддержка эмуляции флэш-памяти через внешний интерфейс коммуникации.

Недостатки:

– Нет поддержки позиционно независимого кода;

– Нет возможности простого добавления файлов, используемых прошивкой;

– Нестандартный формат исполняемого файла прошивки.

Основываясь на данных анализа, были сделаны выводы, что функции шифрования образа прошивки присутствуют во всех рассмотренных загрузчиках и являются необходимыми. Также были выявлены потенциальные возможности улучшения защиты за счет добавления поддержки файловой системы и загрузки исполняемых модулей с возможностью перемещения (для ASLR).

# 2 ПРОЕКТИРОВАНИЕ ЗАГРУЗЧИКА

## 2.1 Цели и задачи разработки

Цель – использовать знания в области низкоуровневого программирования и информационной безопасности для создания усовершенствованного загрузчика для встраиваемых систем (в частности микроконтроллеров). Для реализации цели необходимо выполнить следующие задачи:

– Спроектировать функциональные модули приложения;

– Выбрать оптимальные средства для разработки;

– Разработать утилиты для работы с компонентами программного продукта;

– Разработать программный продукт.

## 2.2 Описание функционала программного продукта

Разрабатываемый загрузчик должен обеспечивать высокую защиту от попыток проникновения внутрь системы и попыток кражи информации из фалов прошивки. Для этого требуется реализовать следующие функции:

– Универсальный программный интерфейс инициализации аппаратной платформы. Он должен включать в себя: настройку платформозависимых аппаратных компонентов, запуск системы тактирования, функции для работы с флэш-памятью, функции вывода отладочных сообщений, определения карты памяти устройства и интерфейс обновления образа прошивки.

– Реализовать среду исполнения высокоуровневого кода загрузчика.

– Реализовать алгоритм запуска приложения и обновления образа прошивки.

– Возможность конфигурации загрузчика на этапе компиляции.

– Систему вывода отладочных сообщений.

– Поддержку шифрования и цифровой подписи загрузочного образа системы. ЭЦП должна проверятся при каждом запуске системы для предотвращения исполнения модифицированного кода.

– Поддержку хранения файлов (с поддержкой атрибутов) в загрузочном

образе системы.

– Программный интерфейс, используемый в приложении, для взаимодействия с загрузчиком. Он должен позволять управлять работой загрузчика.

– Поддержка загрузки одного из стандартных форматов исполняемых файлов.

– Поддержка запуска приложений с перемещаемым кодом. Это необходимо для реализации ASLR.

### 2.3 Описание функциональных модулей и их взаимодействие

Для упрощения дальнейшей разработки загрузчика была разработана схема взаимодействия функциональных модулей (рис. 3).

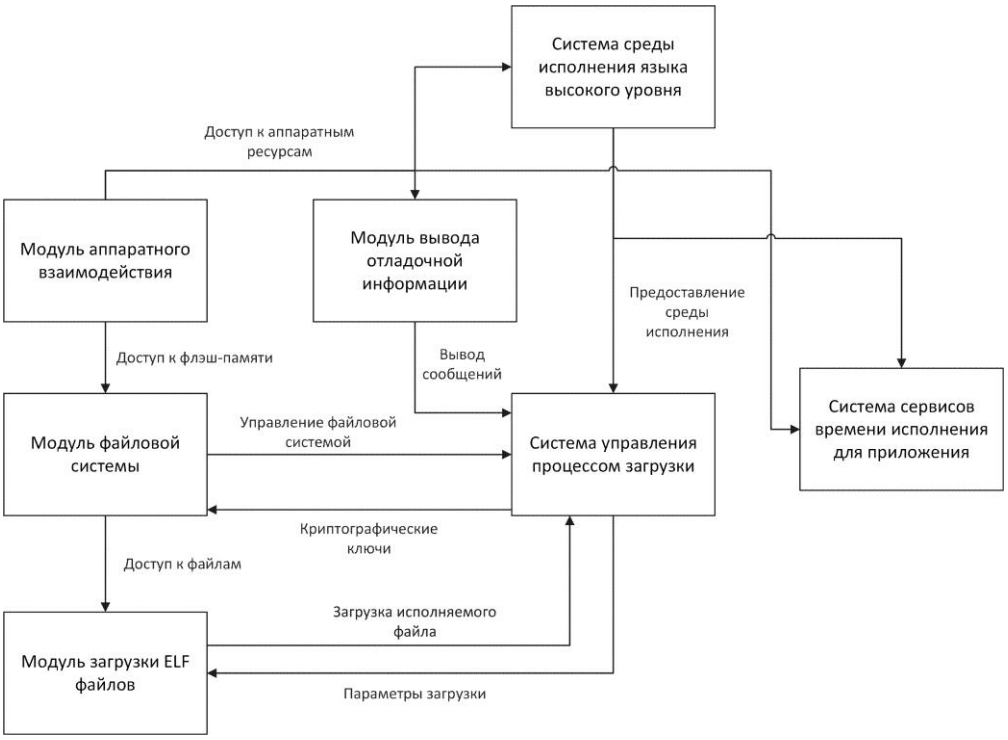


Рисунок 3 – Схема взаимодействия функциональных модулей

Функциональные модули включают:

– систему инициализации и работы среды исполнения языка высокого уровня;

– систему управления процессом загрузки, которая принимает основные решения;

– модуль аппаратного взаимодействия включает в себя функции управления аппаратными блоками, необходимыми для работы остальных модулей;

– модуль вывода отладочной информации для отображения статуса загрузки;

– модуль файловой системы, который необходим для доступа к файлам, хранящимся в прошивке;

– модуль загрузки ELF файлов, который необходим для запуска исполняемых файлов, а также для поддержки ASLR;

– система сервисов времени исполнения для приложения служит программным интерфейсом для доступа к информации, сохранённой в загрузчике, а также для изменения его параметров конфигурации.

Была построена диаграмма обеспечивающих подсистем загрузчика (рис. 4).

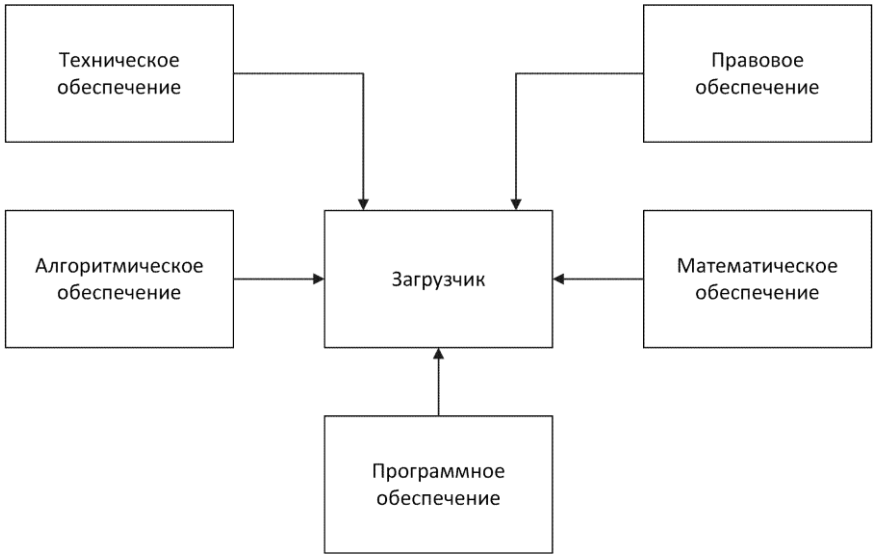


Рисунок 4 – Обеспечивающие подсистемы загрузчика

В обеспечивающие подсистемы входят:

– техническое обеспечение – комплекс технических средств, предназначенных для работы информационной системы, а также соответствующая документация на эти средства и технологические процессы. Его составляют: отладочное оборудование, необходимое для создания загрузчика; устройства, исполняющие код программного продукта.

– математическое обеспечение включает в себя методы математической обработки данных. Например, стандартные библиотеки C/C++ и криптографические библиотеки.

– программное обеспечение включает в себя общесистемные и специальные компоненты. Например, Visual Studio Code, техническая документация библиотек и открытых форматов.

– правовое обеспечение - совокупность правовых норм, определяющих создание, юридический статус и функционирование информационных систем, регламентирующих порядок получения, преобразования и использования информации. Правовые нормы на создание загрузчика определяются лицензиями использованных библиотек и законами Российской Федерации.

– алгоритмическое обеспечение – совокупность использованных алгоритмов.

## 2.4 Алгоритмическое обеспечение

### 2.4.1 Разработка алгоритма управления загрузкой

Основой загрузчика является алгоритм управления загрузкой. Основываясь на данных от других систем и общей конфигурации, он принимает решения по загрузке и обновлению системы. Данный алгоритм взаимодействуют с модулями загрузки исполняемых файлов, файловой системы и вспомогательными модулями. На рисунке 5 представлена блок-схема управляющего алгоритма.


Рисунок 5 – Блок-схема алгоритма управления загрузкой

2.4.2 Разработка алгоритмов работы с файловой системой

Одной из основных систем безопасности загрузчика является файловая система. Хранение и защита файлов в системе. На рисунке 6 представлена блок-схема подключения файловой системы.

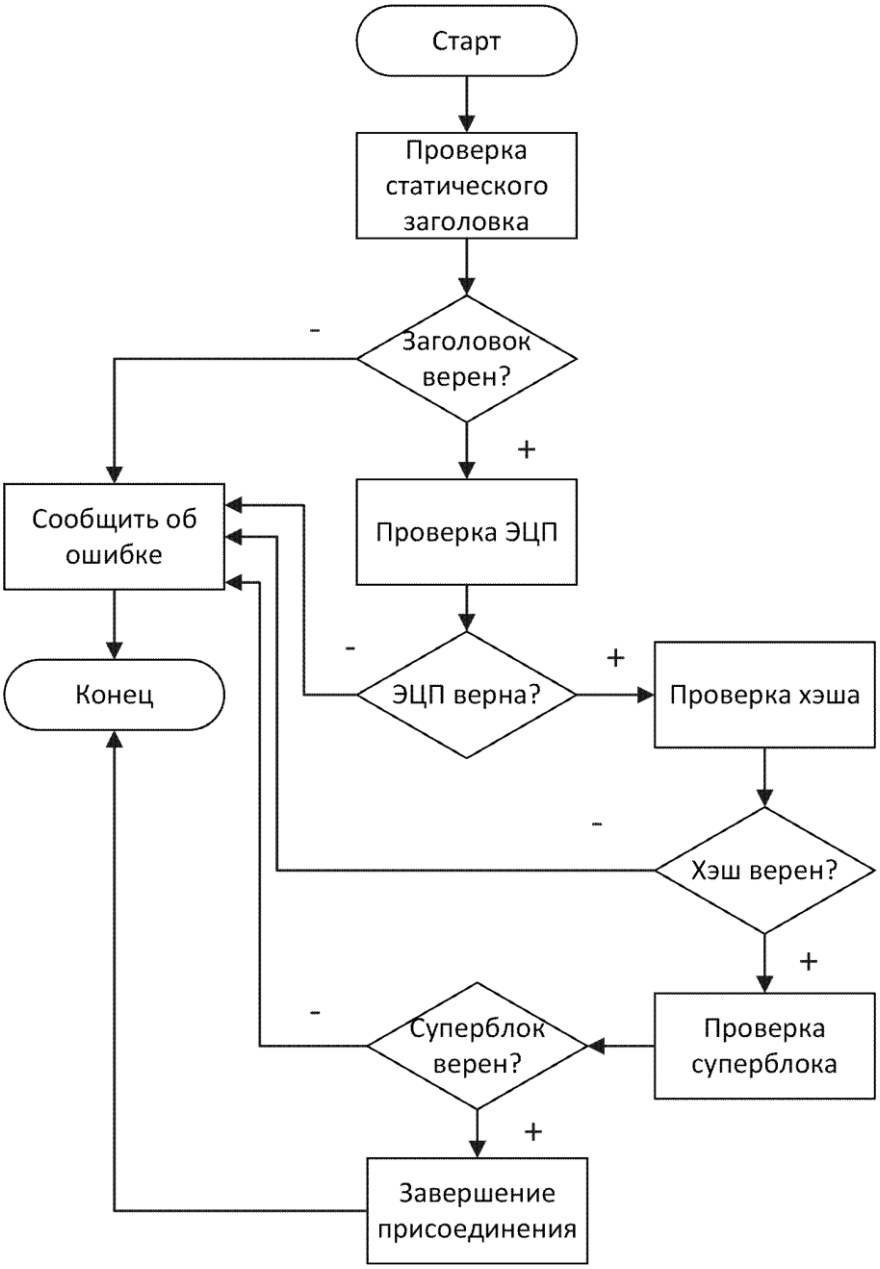


Рисунок 6 – Блок-схема алгоритма подключения файловой системы

На этапе подключения файловой системы происходят проверки структуры, криптографических данных. Это требуется для того, чтобы удостовериться в аутентичности и целостности данных прошивки.

Также был разработан алгоритм поиска файла по заданному пути на рисунке 7.

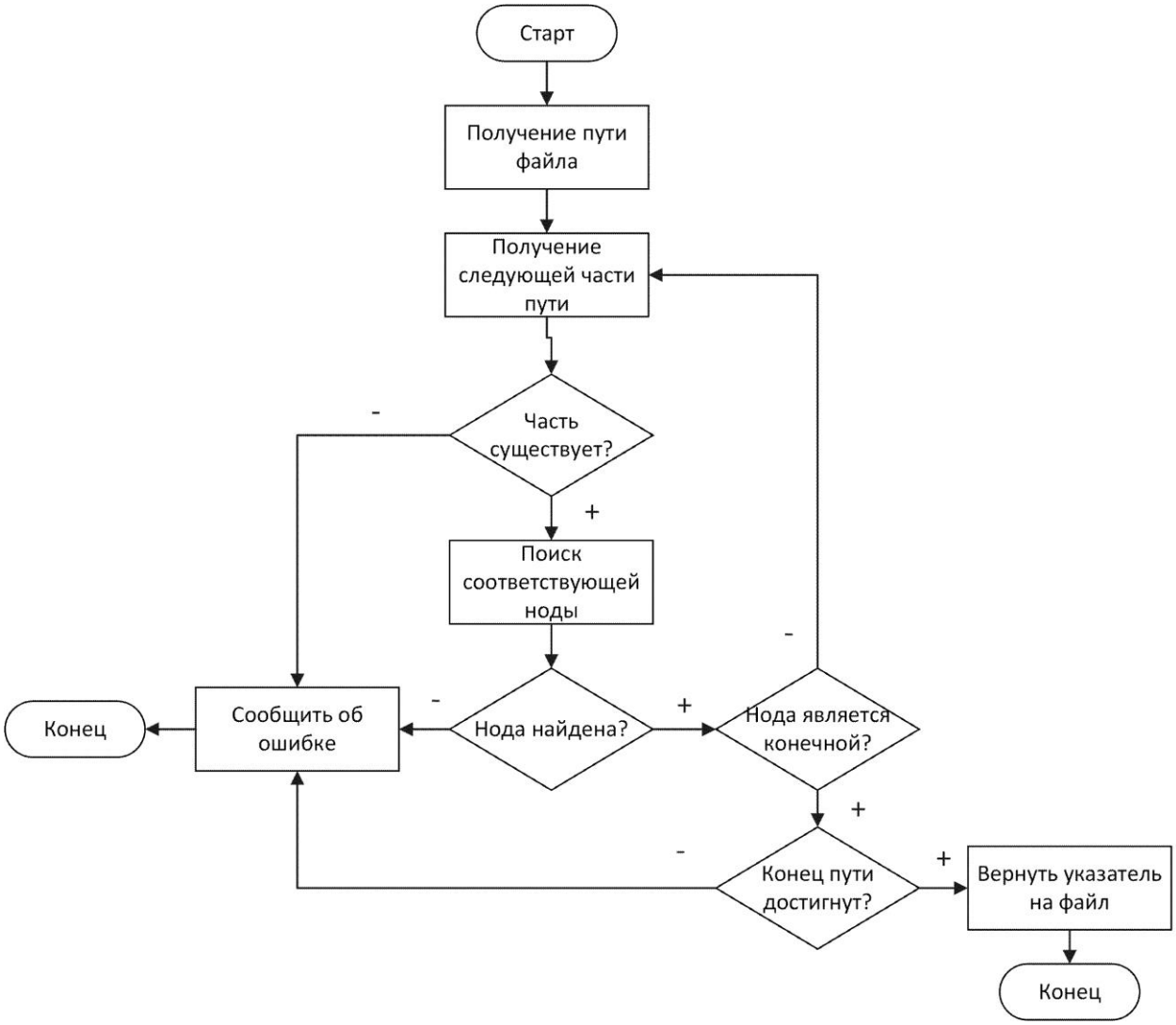


Рисунок 7 – Блок-схема алгоритма поиска файла

Поиск файла происходит с помощью деления пути на части, разделенные символом «/». После этого последовательно для каждой части происходит поиск соответствующей ноды. Если цепочка нод совпадает с заданным путем, то файл найден.

2.4.3 Разработка алгоритма загрузки исполняемых файлов

Для реализации технологии ASLR требуется реализовать алгоритм загрузки исполняемых файлов по выбранному адресу. Часть процесса загрузки связна с верификацией частей ELF файла. Блок-схема загрузки файла представлена на рисунке 8.

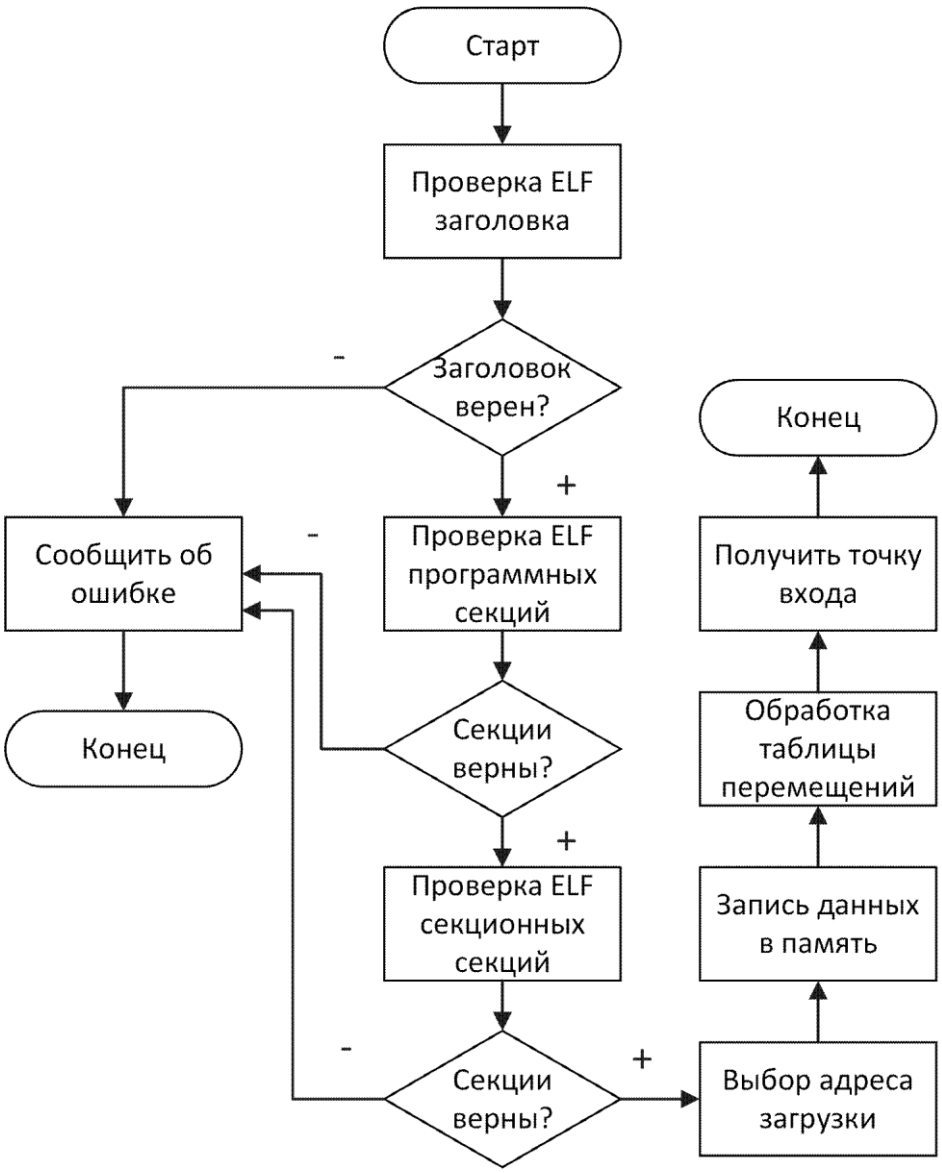


Рисунок 8 – Блок-схема алгоритма загрузки файлов

### 2.5 Обоснование выбора инструментов разработки

Выбор инструментов для разработки загрузчика является определяющим этапов в проектировании. Для данного проекта требуется выбрать язык разработки, библиотеки для криптографии и иных функций, а также дополнительных инструментов.

Для разработки программ, взаимодействующих с оборудованием, используются языки системного программирования. Одним исключением является ассемблер. Это язык наиболее низкого уровня, и не может быть заменен на что-либо другое, так как некоторые операции (инициализация регистров, переключение режимов работы процессора и т.д.) не могут выполнены из языка высокого уровня.

Из языков системного программирования высокого уровня используются: C, C++ и Rust. Конечно, вместо языка высокого уровня может использоваться ассемблер, но это всегда ведет к снижению эффективности разработки и возрастанию количества ошибок.

Язык программирования «C» является наиболее старым языком из 3-х. Это означает, что язык является проверенным инструментом разработки. Но при этом, это также означает, что с технологической точки зрения он не является наиболее безопасным. С точки зрения поддержки другими инструментами, он является наиболее поддерживаемым. Каждая аппаратная платформа имеет компилятор «C» и он может быть единственным поддерживаемым языком на ней. Для данного языка существуют огромное количество библиотек для любых задач. Так как среда исполнения довольно мала, производительность остается высокой.

Язык программирования «C++» является вторым по популярности среди разработчиков встраиваемых систем. Он более современный чем «C». Новые версии стандарта выпускаются каждые 3 года организацией C++ standards committee. Это позволяет держать язык в хорошем состоянии и добавлять новые функции. Стандартная библиотека шаблонов (STL) имеет большое количество проверенных функций для решения общих задач. За счет использования шаблоном можно достичь полуавтоматического управления ресурсами системы, что уменьшает количество ошибок логики, допущенных программистом. Еще одним важным достоинством является возможность взаимодействия с кодом, написанным на «C», что позволяет использовать не только библиотеки, написанные на «C++», но на «C». Производительность языка находится примерно на уровне «C».

И третьим языком является «Rust». Это сравнительно новый язык, созданный компанией Mozilla. Главной целью языка был упор на безопасность. Это достигается при помощи синтаксиса языка и специального механизма – проверки заимствований. Это позволяет избегать логических ошибок, которые могут быть выявлены еще на этапе компиляции. При этом производительность

языка и требования к системе лишь незначительно выше чем у предыдущих двух. На этом достоинства языка заканчиваются. Количество написанных библиотек сравнительно невелико. Есть возможность взаимодействия с кодом, написанным на «C», но для этого требуется написать специальный интерфейс конвертации, что не всегда целесообразно. Язык поддерживается только одним компилятором, что затрудняет его применение на разных аппаратных платформах.

Проанализировав все варианты был выбран «C++», так как это современный язык, имеющий широкую поддержку среди других инструментов и большое количество библиотек.

В качестве системы сборки проекта был выбран Make. Она универсальная и довольно простая по сравнению с другими, но обладает большей гибкостью. Это позволит легко реализовать дополнительные шаги сборки, необходимые для загрузчика. Языком утилит для упрощения сборки и отладки загрузчика был выбран Python. Решение было принято из-за простоты работы с ним и скорости написания кода, так как не требуется высокая производительность.

Для выбранного языка существуют два основных межплатформенных компилятора: gcc и clang.

Gcc поддерживает последние версии стандарта языка «C++» и имеет множество параметров конфигурации компиляции для разных аппаратных платформ. Это позволяет реализовать практически любой функционал, требующий поддержки со стороны компилятора.

Clang также поддерживает последние версии стандарта языка «C++», но при этом количество опций конфигурации для разных аппаратных платформ существенно меньше. Например, на нем нельзя реализовать приложение с функцией перемещаемого кода.

Оба компилятора являются занимают лидирующие позиции и генерируют быстрый код. Однако в связи с невозможностью реализации полного функционала на clang, было принято решение использовать только gcc. Для

отладки кода во время разработки будет использоваться соответствующая программа-отладчик gdb.

Для минимизации количества ошибок и ускорения процесса разработки, было решено использовать отдельные библиотеки для криптографических алгоритмов (ECDSA, SHA256, AES). Требования, предъявляемые к библиотекам:

– высокая скорость работы;

– малый объем занимаемой памяти;

– платформонезависимость.

Вначале, была рассмотрена популярная библиотека mbedTLS. Она содержит поддержку для всех необходимых алгоритмов. Но тестирование выявило, что она занимает очень много места в памяти и проверка ЭЦП на базе ECDSA занимает несколько секунд. В связи с этим, было решено использовать менее популярные аналоги.

В качестве библиотеки для алгоритмов SHA256 и AES было решено использовать Cifra. В скомпилированном виде код для этих алгоритмов занимает всего несколько килобайт, при этом сохраняя относительно высокую скорость работы. Данная библиотека так же содержит опциональные методы защиты от раскрытия ключей шифрования с помощью анализа потребления энергии процессором, что является большим плюсом для загрузчика.

Для алгоритма ECDSA была выбрана библиотека micro-ecc. Так же имеет небольшой размер, но ее главное отличие — это скорость работы. Она выполняет операцию проверки цифровой сигнатуры всего за 8 миллисекунд. Эта библиотека также содержит механизмы противодействия анализу электропотребления, но так как функция создания ЭЦП в загрузчике не используется, то это не является важным критерием при выборе.

Так как загрузчик выполняется в среде без операционной системы, ему требуются специальные версии стандартных библиотек, способные работать в этих условиях. В загрузчике используются стандартные библиотеки для «C» и «C++» от Embedded Artistry.

В качестве аппаратной платформы был выбран микроконтроллер Cortex-M на базе архитектуры ARMv7M.

# 3 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

## 3.1 Реализация файловой системы

Для реализации и поддержки остальных функций были сформированы требования к функционалу файловой системы:

– Поддержка использования строчных путей файлов и папок;

– Поддержка выравнивания файлов по фиксированной границе;

– Поддержка шифрования;

– Поддержка аутентификации при помощи ЭЦП;

– Поддержка атрибутов файлов.

Для удобства доступа к файлам и удобства их организации было решено использовать древовидную файловую структуру с поддержкой указания путей при помощи строки формата ASCII. В качестве разделителя будет использоваться символ «/», он не может быть использован в именах файлов (как и NULL). Путь к файлу всегда должен начинаться с «/». Пример пути: «/sys/kernel.elf». Алгоритм поиска нужного файла был описан в разделе 2.4.2.

Для оптимизации размера, занимаемого путями файлов в образе файловой системы, была реализована оптимизация. Ее идея заключается в разделении всех путей на их составные компоненты и удалении дубликатов. Далее строиться древовидная секция, состоящая из специальных структур – нод имен файлов. В начале данной секции находиться корневая нода, она не имеет закрепленной за ней строки. Пример секции представлен на рис. 9.

Поля ноды имен файлов:

– nameOffset – сдвиг относительно начала секции, содержащей строку данной ноды;

– prevNode – индекс предыдущей ноды;

– nextNode – индекс следующей группы нод или индекс файла если данная нода указывает на файл;

– size_nodeCnt – размер следующей группы нод;

– isFile – флаг, показывающий на то, указывает ли данная нода на файл.

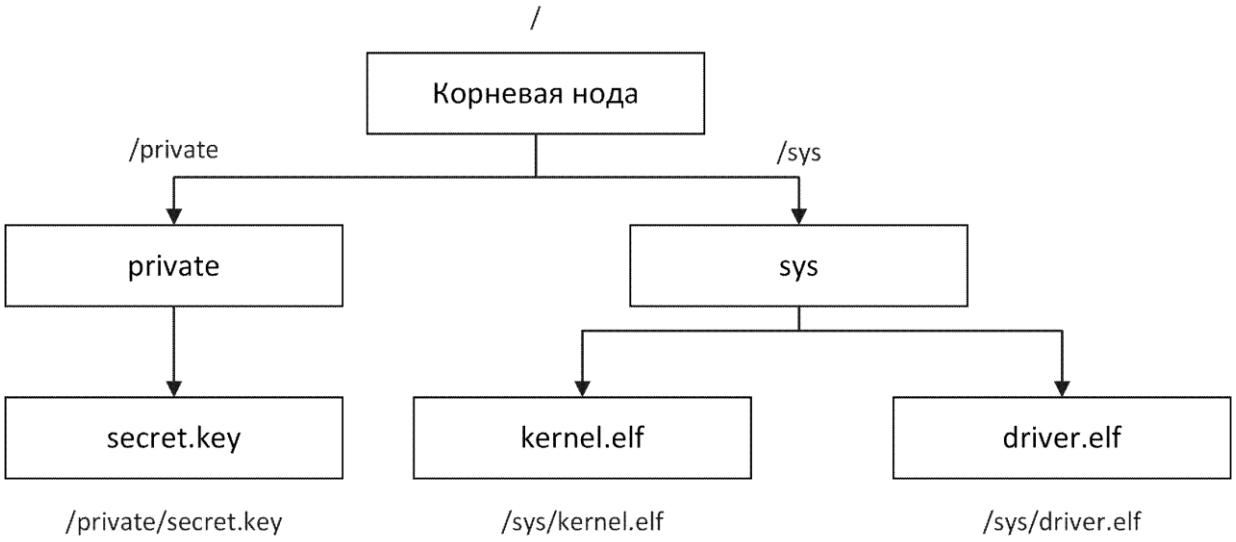


Рисунок 9 – пример организации нод имен файлов

Для хранения строк была выделена отдельная секция. Все строки имеют ненулевую длину и оканчиваются символом NULL. На одну строку может указывать одна и более нод. Строки расположены друг за другом без каких-либо промежутков.

Для описания параметра файла используется структура – дескриптор файла:

– lastModifTime – дата последней модификации, количество секунд с начала эпохи Unix;

– fileOffset – сдвиг относительно начала секции данных файлов;

– fileSize – размер файла;

– fileNameNodeIdx – индекс последней ноды имен файлов в цепочке пути данного файла;

– attr – атрибуты файла.

Поля «lastModifTime» и «attr» являются информационными. Если файл имеет атрибут «исполняемый», то его начало должно быть выравнено по границе 4096 байт. Это нужно для обеспечения требований выравнивания внутри исполняемых файлов при исполнении кода без копирования в оперативную память. Все дескрипторы помещаются в свою отдельную секцию в виде массива. «fileNameNodeIdx» не может быть равен нулю, так как это корневая нода. Размер файла ограничен 4 ГБайт.

Основной структурой, связывающей все секции воедино, является суперблок. Его задачей является определение основных параметров структур, используемых в файловой системе и указание местонахождения всех секций.

Поля структуры суперблок:

– fileCnt – количество файлов, которое содержится в данном образе файловой системы;

– fileTableEntrySize – размер структуры дескриптора файла;

– fileTablePtr – указатель на секцию дескрипторов файлов;

– fileNameNodeCnt – количество нод имен файлов;

– fileNameNodeSize – размер структуры ноды имени файла;

– fileNameNodeTablePtr - указатель на секцию нод имен файлов;

– fileNameSecIdx – индекс дескриптора, указывающего на секцию строк пути фалов;

– endBlkSize – размер конечного блока;

– endBlkPtr – указатель на конечный блок.

Количество файлов ограниченно 65535. Размеры структур должны совпадать с размерами для данной версии файловой системы. Секции не должны накладываться друг на друга. Конечный блок служит механизмом проверки целостности файловой системы и значений полей суперблока, в частности. Он должен находиться в самом конце и иметь всего одно поле – указатель на суперблок.

Для идентификации файловой системы, получения базовых параметров и обеспечения работы криптографического слоя был реализован статический заголовок.

Поля статического заголовка:

– magic – магическое число;

– fsVersion – версия файловой системы;

– fileLen – длинна расшифрованного файла;

– superBlkSize – размер структуры суперблока;

– isBigEndian – флаг, указывающий порядок следования байтов;

– isEncrypted – флаг, указывающий на то, зашифрован ли данный образ;

– readableID – произвольный идентификатор из 16 байт;

– fileHash – хэш образа, не включая этот заголовок;

– hash – хэш этого заголовка, с полем «signature» заполненным нулями;

– signature – ЭЦП на основе поля «hash».

Магическое число состоит из 4 символов – 'B', 'F', 'S', '_'. Данное поле служит для начальной идентификации типа файла и всегда должно содержать именно эти символы.

Версия файловой системы — это структура, состоящая из: мажорная версия, минорная версия, версия патча.

Поскольку в качестве алгоритма шифрования был выбран AES, который является блочным алгоритмом, то длинна зашифрованного файла может быть больше чем расшифрованного. В связи с этим, если требуется удлинить файл, то к ему в конец добавляется необходимое количество нулевых байт. Для того, чтобы отбросить эти байты после расшифровки используется поле «fileLen».

Структуры файловой системы могут быть определены в двух порядках байт: от большого к маленькому и наоборот. Это сделано для упрощения и ускорения драйвера файловой системы, работающем на устройстве. Статический заголовок должен быть выравнен по границе 4096 байт. Итоговая структура файловой системы показана на рисунке 10.


Рисунок 10 – общая структура файловой системы

Криптографическая система работает следующим образом. Сначала вычисляется хэш всего образа за исключением статического заголовка. Этот хэш помещается в статический заголовок и вычисляется его хэш. От этого хэша вычисляется цифровая подпись и так же помещается в статический заголовок. И последним шагом основная часть образа шифруется, по необходимости добавляя байты для получения целого числа блоков.

Такой алгоритм защиты позволяет предотвратить любые модификации образа файловой системы, так как их можно будет обнаружить и признать образ непригодным к использованию.

Слой шифрования в свою очередь охватывает только чувствительную к раскрытию часть файловой системы, что позволяет производить проверку аутентичности образа без предварительной расшифровки.

Алгоритм работы криптографической системы представлен на рисунке 11.

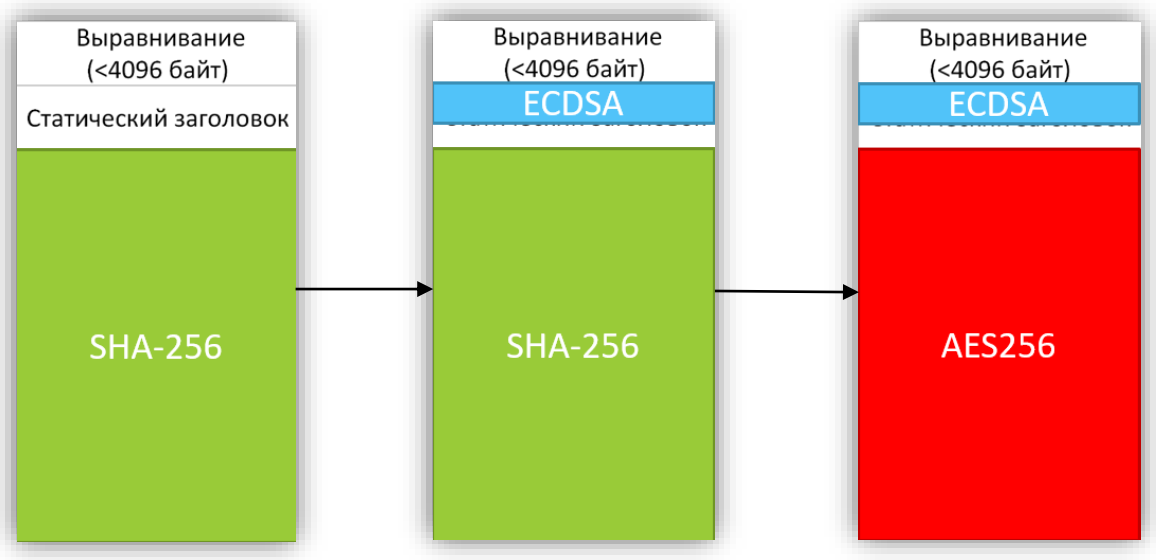


Рисунок 11 – схема работы криптографического слоя

Также для упрощения работы с файловой системой были созданы вспомогательные утилиты. Их задачей является создание и распаковка образа файловой системы. На рисунках 12 и 13 представлены окна помощи этих утилит. Так как предполагается, что они будут тесно интегрированы в процесс сборки, то было решено использовать интерфейс командной строки.

```
> ./bootfs2f -h

Unpack BootFS

Usage: bootfs2f [OPTIONS] [INPUT FILE] [OUTPUT DIR]

Options:
  -h, --help                  Print this message
  -v, --verify [public key]   Verify BootFS with ECDSA public key. [public key] is PEM file, that contains public key
  -d, --decrypt [key]         Decrypt BootFS contents with AES-256. [key] is text file, that contains key and IV in hex form on separate lines
  -w, --warn                  Display warnings
  --ignore-errors             Ignore hash and signature mismatch errors
  --check                     Do verification, but don't write files
```

Рисунок 12 – окно помощи утилиты boofs2f

```
> ./f2bootfs -h

Convert file structure into BootFS filesystem

Usage: f2bootfs [OPTIONS] [INPUT DIR] [OUTPUT FILE]

Options:
  -h, --help                  Print this message
  -b, --bigend                Generate BootFS in big endian mode
  -s, --sign [private key]    Sign BootFS with ECDSA private key. [private key] is PEM file, that contains private key
  -e, --encrypt [key file]    Encrypt BootFS contents with AES-256. [key file] is text file, that contains key and IV in hex form on separate lines
  -i, --id [readable id]      Readable id with max length of 16, must match value in bootloader (default: all zeros)
  -m, --manifest [file]       Generate manifest file, containing file addresses
  -w, --warn                  Display warnings
```

Рисунок 13 – окно помощи утилиты f2bootfs

Помимо основных функций, данные утилиты позволяют производить

проверку корректности уже созданного образа, создавать манифест хранимых файлов и выбор порядка байт.

### 3.2 Реализация модуля загрузки исполняемых файлов формата ELF

Для реализации возможности загрузки исполняемого кода приложения в произвольное место в памяти требуется поддержка загрузки позиционно-независимого кода. Для этого используются специальные форматы исполняемых файлов. Примерами служат COFF, ELF, MachO и т.д. Разработка своего формата быстро была признана нецелесообразной ввиду необходимости написания большого количества вспомогательных программ и внесений изменений в компилятор. Из доступных форматов наиболее подходящим оказался ELF. Он имеет богатый набор инструментов для работы с ним, открытую документацию и не перегружен функционалом, что делает его идеальным кандидатом. Общая структура формата представлена на рисунке 14.

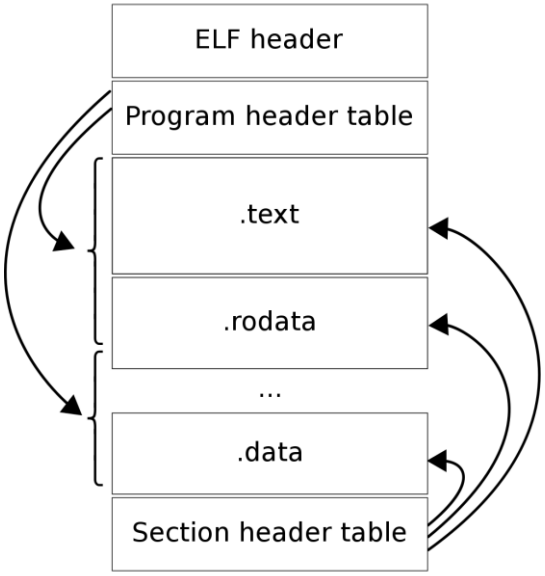


Рисунок 14 – структура ELF формата

Основой формата являются 4 компонента – главный заголовок, таблица программных заголовков, таблица заголовков секций и сами секции.

Главный заголовок определяет основные сведения об исполняемом файле: версия формата, разрядность, количество заголовков в таблицах, местоположение таблиц и точку начала исполнения.

Таблица программных заголовков является главной структурой при загрузке исполняемого файла. Она определяет куда, какие и сколько данных должны быть загружено, а также выравнивание необходимое для каждой загружаемой секции. Если несколько секции имеют одинаковые параметры загрузки и находятся рядом, то они объединяются в один программный заголовок.

Таблица заголовков секций определяет логические секции внутри загружаемых регионов. Не все секции являются загружаемыми. Каждая из них имеет тип, который определяет какие данные хранит секция. Это может быть код, данные, отладочная информация, дополнительная информация или данные релокации. Последний тип нужен для реализации возможности загрузки кода в произвольное место.

При загрузке ELF файла сначала происходит проверка его основных структур. Проверяется основной заголовок и обе таблицы. Это требуется для того, чтобы избежать загрузки ошибочных данных или приведения системы к катастрофической ошибке. Во время проверки так же происходит определения размеров итоговых секций. После этого система контролирующая данный модуль указывает адреса загрузки секции кода и данных. Они генерируются при помощи генератора случайных чисел, что является сутью ASLR. Модуль копирует секции и приступает к последнему этапу загрузки – обработке релокаций. Приложение, находящееся в загружаемом файле, располагает секции кода по виртуальным адресам 0x4000’0000-0x7FFF'FFFF, а секции данных по 0x8000'0000-0xBFFF'FFFF. Так как эти адреса виртуальные их нужно заменить на настоящие во время загрузки. Каждый объект релокации указывает место в загруженном коде, которое должно быть изменено, и тип изменения.

Для обращений к данным во время исполнения используется GOT таблица. Значения в ней должны быть так же изменены для корректной работы, но компилятор не генерирует для нее секцию релокаций. В связи с этим был написан скрипт, который на финальном этапе сборки добавляет эту секцию. Для уменьшения размера он так же удаляет все секции, которые не нужны при

загрузке и исполнении.

В результате файл готов к исполнению.

### 3.3 Реализация вспомогательных компонентов

Для обеспечения работы языка C++ были реализованы различные функции. Они включают в себя: инициализацию секций данных различных типов, установку указателя на стэк загрузчика, вызов функций статической инициализации, реализация функций проверки целостности стэка и удаление функций работы с кучей.

Для возможности конфигурации загрузчика на этапе компиляции в систему сборки были добавлены конфигурируемые переменные. Они позволяют включать или отключать опции шифрования, использования ЭЦП, отладочную версию и выбора процессора для компиляции. Список опций показан на рисунке 15.

```
> ./build.sh -h

ColdOS boot image assembler

Usage: build.sh [OPTIONS] [APP NAMES...]

Options:
    -h, --help                  Prints this message
    -b, --big                   Generate BootFS for big endian cpu
    --cpu [model]               Target cpu model (e.g. cortex-m4)
    -d, --debug                 Build in system binaries debug mode
    --appdebug                  Build apps in debug mode
    -e, --encrypt               Encrypt BootFS
    --sign                      Sign BootFS
    -t, --toolchain [path]      Path to alternative toolchain
    --id [bootfs id]            16 character readable ID
    --plat [platform name]      Destination platform
```

Рисунок 15 – опции скрипта сборки

Для взаимодействия приложения с загрузчиком был создан специальный программный интерфейс. Он состоит из двух частей: интерфейса передачи управления и интерфейса времени исполнения. Первый определяет данные и их формат, которые будут переданы в приложение при передаче управления. Они включают в себя три значения в соответствующей последовательности: адрес таблицы интерфейса времени исполнения, базовый адрес секции кода и

базовый адрес секции данных. Эти данные необходимы для начальной инициализации приложения и дальнейшего взаимодействия с загрузчиком. Второй интерфейс реализован в виде таблицы функций, расположенных в предопределенном порядке. Функции включают в себя:

– GetAPIVer – получение версии интерфейса;

– GetMemConfig – получение карты памяти устройства;

– GetFreeMem – получение распределения свободной памяти;

– InitCoreClock – инициализация системы тактирования устройства;

– DeinitCoreClock – функция обратная InitCoreClock;

– GetCoreFreq – получение текущей тактовой частоты;

– ResetBootRetryCntr – сброс счетчика попыток запуска;

– IncrBootRetryCntr – увеличение счетчика попыток запуска;

– ReqBootFSUpdate – запрос обновления образа файловой системы.

Счетчик попыток запуска служит защитой от бесконечных попыток загрузки из-за неисправного приложения. Для нормальной работы данной функции требуется взаимодействие загрузчика и приложения.

Карта свободной памяти сообщает приложению какую память можно использовать, а какая занята данными загрузчика, так как для работы данного интерфейса требуется хранить некоторое количество данных в оперативной памяти устройства.

Данный интерфейс также имеет систему версионирования. Ее использование позволяет вносить изменения в него без реализации более сложных механизмов проверки доступности новых функций.

Для удобства добавления различных аппаратных платформ был реализован интерфейс предоставления загрузчику сведений о конфигурации платформы и методах выполнений аппаратно-зависимых функций. Он включает в себя: определение карты памяти устройства пригодной к использованию; вывода отладочных сообщений; доступа к криптографическим ключам (для возможности реализации аппаратного хранилища); определение необходимости

использования подписанного и/или зашифрованного образа файловой системы; инициализации необходимых аппаратных ресурсов для загрузки и работа с флэш памятью. Для удобства сборки загрузчика для разных платформ, код для каждой из них помещается в отдельную папку и может быть выбран при сборке при помощи опции «--platform <название платформы>».

### 3.4 Расчет надежности ПО

Для расчета надежности были произведены 10 тестовых запусков загрузчика. Из них 10 запусков были удачными, соответственно 0 отказов. Итоговая вероятность безотказной работы (согласно упрощенной оценке по ГОСТ 28195-99) составляет:

$$P = 1 - \frac{n}{N} = 1 - \frac{0}{10} = 1 \quad (1)$$

где P – вероятность безотказной работы;

n –число зарегистрированных отказов;

N – число экспериментов.

# 4 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

При работе с любыми видами важной информации будь то хранение, обработка или передача, главное помнить, что от важной информации всегда зависят многие вещи, исходя из этого существуют и люди, которые всегда будут производить попытки незаконно перехватить, уничтожить или завладеть информацией, в этом случае возникает угроза информационной безопасности.

Информационная безопасность – это защита, а также сохранность информации, её важнейших элементов, в том числе систем и оборудования, которые предназначены для использования, сбережения и передачи информации.

Цель информационной безопасности – защита информации и данных, а также поддерживающей инфраструктуры от преднамеренного или случайного вмешательства, которое может повлечь за собой несанкционированное изменение информации или потерю данных.

Для обеспечения информационной безопасности необходимо придерживать трех основных принципов.

Целостность – подразумевает гарантию предотвращения случайных или преднамеренных искажений информации, а также разрушающих воздействий.

Доступность – подразумевает обеспечение надежного и эффективного доступа к информации авторизованным субъектам системы, также восстановление системы должно быть обеспечено так, чтобы при сбоях это не влияло отрицательно на работу.

Конфиденциальность – подразумевает, что информация известна только авторизованным субъектам системы, которые прошли проверку.

Для обеспечения безопасности информационной системы – загрузчика, так как он осуществляет обработку конфиденциальных данных, были предприняты следующие меры по обеспечению безопасности: аутентификация, шифрование данных и валидация форматов файлов.

### 4.1 Аутентификация

Аутентификация – под аутентификацией подразумевается подтверждение аутентичности образа прошивки, путем проверки ЭЦП с помощью публичного ключа, находящегося в загрузчике.

Для обеспечения высокого уровня защищенности, следует использовать алгоритм ECDSA с кривой SECP256K1 и хэш функцию с алгоритмом SHA256.

### 4.2 Шифрование данных

Так как в образе прошивки могут содержаться конфиденциальные данные организации или криптографические ключи, которые нужно сохранить в секрете, то она нуждается в защите. Основная угроза заключается в возможном перехвате данных, пока образ прошивки находится вне устройства. Для того, чтобы сохранить конфиденциальность передаваемых данных, должно использоваться их шифрование. В качестве алгоритма шифрования следует использовать AES-256. Данные должны быть зашифрованы все время пока они находятся за пределами организации и целевого устройства.

### 4.3 Модель нарушителя безопасности

Под нарушителем понимается субъект, который в результате умышленных или неумышленных действий может воздействовать на объект защиты.

Нарушители делятся по признаку принадлежности к ИС. Все нарушители делятся на две группы:

− внешние нарушители – физические лица, не имеющие права пребывания на территории контролируемой зоны, в пределах которой размещается оборудование ИС и не имеющие права доступа к информации хранящейся внутри прошивки;

− внутренние нарушители – физические лица, имеющие право пребывания на территории контролируемой зоны, в пределах которой размещается оборудование ИС.

К внутренним нарушителям могут относится:

− директор предприятия;

− разработчики;

− специалисты по управлению инфраструктурой;

− специалист по ИБ;

− обслуживающий персонал.

Предполагается, что возможность сговора внутренних нарушителей маловероятна ввиду принятых организационных и контролирующих мер. При этом предполагается, что внутренние нарушители могут, в силу складывающихся обстоятельств, действовать в одиночку.

В качестве внешнего нарушителя информационной безопасности, рассматривается нарушитель, который не имеет непосредственного доступа к техническим средствам и ресурсам системы, находящимся в пределах контролируемой зоны.

К внешним нарушителям могут относится:

− бывшие сотрудники – пользователи ИС;

− посторонние лица, пытающиеся совершить несанкционированный доступ к информации.

На таблице 1 представлена модель нарушителя информационной безопасности.

Таблица 1 – Модель нарушителя информационной безопасности

| Категория лиц (должности) | Тип (внутренний / внешний) | Мотивы действия | Квалификация нарушителя | Техническая оснащённость | Характер возможных действий |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| Директор предприятия | Внутренний | Реализация угроз безопасности ИС по неосторожности | Низкая | Компьютер в сети | Нарушение конфиденциальности информации, предоставление доступа к ИС некомпетентным людям |
| Разработчики | Внутренний | Реализация угроз безопасности ИС по неосторожности или же из корыстных побуждений | Высокая | Компьютер с доступом к ИС | Нарушение целостности и доступности информации, нарушение работоспособности системы, искажение информации |
| Специалисты по управлению инфраструктурой | Внутренний | Реализация угроз безопасности ИС по неосторожности или же из корыстных побуждений | Средняя | Компьютер с доступом к ИС | Нарушение целостности и доступности информации, нарушение работоспособности системы, искажение информации |
| Специалист по ИБ | Внутренний | Реализация угроз безопасности ИС по неосторожности или же из корыстных побуждений | Высокая | Компьютер с доступом к ИС | Нарушение целостности и доступности информации, нарушение работоспособности системы, искажение информации |
| Обслуживающий персонал | Внешний | Реализация угроз безопасности ИС по неосторожности или же из корыстных побуждений | Низкая | Компьютер в сети | НСД к информации, извлечение и искажение информации |

Продолжение таблицы 1

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| Пользователи | Внешний | Реализация угроз безопасности ИС по неосторожности или же из корыстных побуждений | Низкая | Компьютер, программное обеспечение | НСД к информации, извлечение информации |
| Посторонние лица высокой квалификации | Внешний | Реализация угроз безопасности ИС по неосторожности или же из корыстных побуждений | Высокая | Компьютер, программное обеспечение, специализированное оборудование | НСД к информации, извлечение информации |

В ходе анализа информационной безопасности предприятия были рассмотрены методы защиты информации, разрабатываемой ИС, используемые методы защиты информации, которые используются на предприятии, рассмотрена модель нарушителя информационной безопасности. Исходя из проведенного анализа можно сделать вывод, что разрабатываемая ИС хорошо защищена как от внешних, так и внутренних угроз, а также обеспечивает целостность, доступность и конфиденциальность информации.

# 5 БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ

## 5.1 Безопасность

Безопасность труда [2] на предприятии отображает состояние, при котором оператор вычислительной техники защищен от несчастных случаев и профессиональных заболеваний в рабочем процессе. Для того чтобы максимальной снизить риски получения травм и заболеваний операторов, а также обеспечить комфортные условия для эффективного труда, необходимо следовать различным требованиям для рабочих мест, оборудованных ПЭВМ.

### 5.1.1 Требования к помещению для работы с ПЭВМ

Правильное оборудование помещения с рабочим местом является важным фактором при работе с ПЭВМ. К помещениям, оборудованным вычислительными машинами, определены следующие требования:

– Естественное и искусственное освещение должно соответствовать требованиям действующей нормативной документации. Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток;

– Оконные проемы должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.;

– Помещения для эксплуатации ПЭВМ должны иметь естественное и искусственное освещение. Эксплуатация ПЭВМ в помещениях без естественного освещения допускается только при соответствующем обосновании и наличии положительного санитарно-эпидемиологического заключения, выданного в установленном порядке;

– Минимальная площадь для одного рабочего места жидкокристаллическим или плазменным экраном – 4,5 м$^2$;

– Для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно-отражающие материалы с коэффициентом отражения для потолка - 0,7 - 0,8; для стен - 0,5 - 0,6; для пола - 0,3 - 0,5;

– Полимерные материалы используются для внутренней отделки интерьера помещений с ПЭВМ при наличии санитарно-эпидемиологического заключения;

– Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации;

– Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

5.1.2 Требования к освещению на рабочих местах с ПЭВМ

При обустройстве рабочего места необходимо обращать внимание на организацию освещения так, как правильная [2] его организация снижает нагрузку на глаза пользователя. Для того чтобы обеспечить безопасность операторам на рабочих местах с ПЭВМ необходимо соблюдать следующие нормы:

– Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева;

– Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях следует применять системы комбинированного освещения;

– Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300 - 500 лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк;

– Следует ограничивать прямую блесткость от источников освещения, при этом яркость светящихся поверхностей (окна, светильники и др.), находящихся в поле зрения, должна быть не более 200 кд/м$^2$;

– Следует ограничивать отраженную блесткость на рабочих поверхностях (экран, стол, клавиатура и др.) за счет правильного выбора типов светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при этом яркость бликов на экране ПЭВМ не должна превышать 40 кд/м$^2$ и яркость потолка не должна превышать 200 кд/м$^2$;

– Яркость светильников общего освещения в зоне углов излучения от 50 до 90 градусов с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м$^2$, защитный угол светильников должен быть не менее 40 градусов;

– Светильники местного освещения должны иметь не просвечивающий отражатель с защитным углом не менее 40 градусов;

– Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ, при этом соотношение яркости между рабочими поверхностями не должно превышать 3:1 - 5:1, а между рабочими поверхностями и поверхностями стен и оборудования 10:1;

– В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы. При устройстве отраженного освещения в производственных и административно-общественных помещениях допускается применение металлогалогенных ламп. В светильниках местного освещения допускается применение ламп накаливания, в том числе галогенные;

– Коэффициент пульсации не должен превышать 5 %;

– В помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

5.1.3 Требования к уровням шума и вибрации

Интенсивный шум и вибрации плохо сказываются на здоровье оператора, также ослабляется его внимание, снижается производительность труда. При длительном воздействии они могут вызвать заболевания периферической

нервной системы. Поэтому в СанПиН 2.2.2/2.4.1340-03 устанавливаются следующие нормы, которым необходимо следовать чтобы избежать негативных последствий:

– В производственных помещениях при выполнении основных или вспомогательных работ с использованием ПЭВМ уровни шума на рабочих местах не должны превышать предельно допустимых значений, установленных для данных видов работ в соответствии с действующими санитарно-эпидемиологическими нормативами;

– При выполнении работ с использованием ПЭВМ в производственных помещениях уровень вибрации не должен превышать допустимых значений вибрации для рабочих мест в соответствии с действующими санитарно-эпидемиологическими нормативами;

– Шумящее оборудование (печатающие устройства, серверы и т.п.), уровни шума которого превышают нормативные, должно размещаться вне помещений с ПЭВМ.

5.1.4 Требования к микроклимату рабочего места с ПЭВМ

Микроклимат [1] является важной частью организации рабочего места ПЭВМ так как от него зависит создание зоны комфорта пользователя. Зона комфорта - оптимальное для организма человека сочетание температуры, влажности, скорости движения воздуха и т.д. Для обеспечения комфортной и безопасной работы необходимо соблюдать следующие нормы:

– В производственных помещениях, в которых работа с использованием ПЭВМ является вспомогательной, температура, относительная влажность и скорость движения воздуха на рабочих местах должны соответствовать действующим санитарным нормам микроклимата производственных помещений;

– В производственных помещениях, в которых работа с использованием ПЭВМ является основной и связана с нервно-эмоциональным напряжением, должны обеспечиваться оптимальные параметры микроклимата для категории работ 1а и 1б в соответствии с действующими санитарно-эпидемиологическими нормативами микроклимата производственных помещений;

– В помещениях, оборудованных ПЭВМ, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПЭВМ.

– Уровни положительных и отрицательных аэроионов в воздухе помещений, где расположены ПЭВМ, должны соответствовать действующим санитарно-эпидемиологическим нормативам;

– Содержание вредных химических веществ в воздухе производственных помещений не должна превышать предельно допустимых концентраций вредных веществ в воздухе рабочей зоны в соответствии с действующими гигиеническими нормативами.

5.1.5 Требования к организации рабочих мест с ПЭВМ

Организация рабочего пространства [1] является очень важным вопросом, так как при неправильной ее организации у операторов ПЭВМ могут произойти нарушения обменных процессов в костно-мышечной системе и искривлению позвоночника, иногда с образованием тромбов, способных привести к закупорке сосудов. Чтобы избежать негативных факторов, сохранить здоровье и повысить производительности труда оператора необходимо следовать следующим рекомендациям:

– Высота рабочей поверхности стола для пользователей должна регулироваться по высоте в пределах 680 - 800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм;

– Модульными размерами рабочей поверхности стола для ПЭВМ, на основании которых должны рассчитываться конструктивные размеры, следует считать: ширину 800, 1000, 1200 и 1400 мм, глубину 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм;

– Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм;

– Конструкция рабочего стула должна обеспечивать: ширину и глубину поверхности сиденья не менее 400 мм; поверхность сиденья с закругленным

передним краем; регулировку высоты поверхности сиденья в пределах 400-550 мм и угол наклона вперед до 15 градусов, и назад до 5 градусов; высоту опорной поверхности спинки 300 +-20 мм, ширину – не менее 380 мм и радиус кривизны горизонтальной плоскости - 400 мм; угол наклона спинки в вертикальной плоскости в пределах +-30 градусов; регулировку расстояния спинки от переднего края сиденья в пределах 260 - 400 мм; длина стационарных или съемных подлокотников должна составлять не менее 250 мм и шириной – 50 - 70 мм;

– Рабочее место пользователя ПЭВМ следует оборудовать подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20 градусов;

– Клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к оператору или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

5.1.6 Требования К ПЭВМ

ПЭВМ должны соответствовать следующим требованиям:

– ПЭВМ должны соответствовать требованиям настоящих санитарных правил и каждый их тип подлежит санитарно-эпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке;

– Концентрации вредных веществ, выделяемых ПЭВМ в воздух помещений, не должны превышать предельно допустимых концентраций (ПДК), установленных для атмосферного воздуха;

– Конструкция ПЭВМ должна обеспечивать возможность поворота корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана дисплея. Дизайн ПЭВМ должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света. Корпус ПЭВМ, клавиатура и другие блоки и

устройства ПЭВМ должны иметь матовую поверхность с коэффициентом отражения 0,4 - 0,6 и не иметь блестящих деталей, способных создавать блики;

– Временные допустимые уровни электромагнитных полей (ЭМП), создаваемых ПЭВМ, не должны превышать значений, представленных в таблице 2;

Таблица 2 – Временные уровни ЭМП, создаваемых ПЭВМ

| Наименование параметров ВДУ ЭМП | | ВДУ ЭМП |
|---|---|---|
| 1 | 2 | 3 |
| Напряженность электрического поля | в диапазоне частот 5 Гц-2 кГц | 25 В/м |
| | в диапазоне 2 кГЦ-400 кГц | 2,5 В/м |
| Плотность магнитного потока | в диапазоне частот 5 Гц-2 кГц | 250 нТл |
| | в диапазоне 2 кГЦ-400 кГц | 25 нТл |
| Плотность потока энергии | в диапазоне 300 МГц-300 ГГц | 10 мкВт/см$^2$ |
| Электростатический потенциал видеомонитора | | 500 В |

– Конструкция дисплея должна предусматривать регулирование яркости и контрастности.

**5.2 Экологичность**

Производство электроники – это глобальная сфера деятельности, которая серьезно влияет на окружающую среду особенно, если учесть то, что многие компоненты создаются на спецзаводах по всему миру. Кроме того, для их производства часто необходимо прибегать к применению интенсивных химических процессов. Вследствие этого, выделяется множество негативных компонентов, которые со временем превращаются в опасные отходы. Например, используемые для вывода изображений ПЭВМ, ЖК-экраны – один из источников трехфтористого азота (NF3), парниковых газов, которые намного вреднее диоксида углерода. По сравнению с диоксидом углерода (CO2) NF3 является в 17 000 раз более активным парниковым газом, а его атмосферное время полураспада может составлять от 550 до 740 световых лет (у CO2 - от 30 до 40

лет). Поэтому правильная утилизация отходов компьютерного оборудования очень важная задача. Федеральный закон №89 «Об отходах производства и потребления» от 24.06.1998 является основным в вопросах регулирования обращения с отходами производства и потребления с целью предотвращения вредного воздействия отходов человека на окружающую среду. Всего определено 5 классов опасности:

– I класс – чрезвычайно опасные отходы;

– II класс – высокоопасные отходы;

– III класс – умеренно опасные отходы;

– IV класс – малоопасные отходы;

– V класс – практически неопасные отходы.

В категорию V класса опасности относят пищевые остатки, бумагу, керамику, необработанную древесину, текстиль из натурального волокна, и прочие продукты органического происхождения, которые не требуют специальных условий обращения. Отходами IV класса опасности относят называют утильсырье, срок разложения которого не превышает 3 лет. Если источник загрязнения исключен, то ущерб для природы незначителен. Отходы, который в природе переработается за 10 лет, относят к III классу опасности. Сюда включены: дизельное топливо; цементная пыль; моторные смазки и т.д. Под категорию II класса опасности отходов попадает утильсырье, который представляет серьезную угрозу для экосистемы – урон от его воздействия может быть восстановлен минимум через 30 лет. Это при условии своевременной ликвидации, иначе, вызванные его попаданием в воздух и почву негативные процессы будут необратимыми. К этому классу опасности принадлежат литий, фенол, хлороформ, серную кислоту, селен, сероводород, барий, формальдегид, сурьму, стирол, все нитриты, мышьяк, молибден и другие вещества. В отходах I класса опасности содержатся вещества, наносящие непоправимый вред здоровью людей, а также приводящие к разрушению экосистемы. Такие отходы могут представлять собой трансформаторы, конденсаторы, креозол и его остатки, ртутные термометры (отработанные или бракованные) и другие

ртутьсодержащие приборы, асбестовую пыль, синтетические и минеральные масла, отходы солей мышьяка В связи с этими классами юридические лица и или индивидуальные предприниматели обязаны обращаться к специализирующим фирмам, которые имеют лицензию на деятельность по сбору, транспортированию, обработке, утилизации обезвреживанию, размещению отходов разных классов. Утилизация отходов компьютеров, оргтехники и другой электроники поэтапный процесс. Порядок утилизации вычислительной техники состоит из следующих шагов:

– создание комиссии на предприятии, имеющем технику, подлежащей утилизации;

– составление экспертного заключения о том, что техника действительно должна быть списана. В качестве эксперта может выступать как независимый специалист, так и сотрудник компании, имеющий диплом, подтверждающий его компетентность в работе с данной техникой;

– составление акта технической экспертизы подтверждающего что техника уже вышла из строя и не подлежит ремонту;

– составление акта списания компьютерной техники с обязательным отображением в бухгалтерском учете предприятия;

– утилизация техники на соответствующем предприятии, имеющем право на переработку вычислительной техники;

– получение официального подтверждения в виде документа, сообщающего о том, что техника должна быть утилизирована в соответствующем порядке и опасные отходы не будут загрязнять окружающую среду.

### 5.3 Чрезвычайные ситуации

5.3.1 Действия при пожаре

Помещения, оборудованные ПЭВМ, являются одними из мест потенциальной опасности возникновения пожара. Основными факторами возгорания ПЭВМ являются:

− короткое замыкание комплектующих вычислительной машины;

− перегрев комплектующих.

Для предотвращения чрезвычайной ситуации персональный компьютер должен иметь доступ к свежему воздуху для охлаждения. Также ПЭВМ необходимо своевременно обслуживать и чистить от пыли, а также проверять провода на наличие повреждений изоляции. При возникновении пожара [1] необходимо выполнять следующие действия:

− немедленно сообщить об этом по телефону в пожарную часть (при этом необходимо четко назвать адрес учреждения, место возникновения пожара, а также сообщить свою должность, фамилию и номер своего телефона);

− необходимо оперативно организовать оповещение об этом всех находящихся в здании людей, независимо от размеров и места пожара или загорания, равно как и при обнаружении хотя бы малейших признаков горения;

− задействовать систему оповещения людей о пожаре, приступить самому и привлечь других сотрудников к эвакуации из здания в безопасное место согласно плану эвакуации;

− известить о пожаре руководителя;

− организовать встречу пожарных подразделений, принять меры по тушению пожара имеющимися в организации первичными средствами пожаротушения;

− Соблюдать осторожность. Ни в коем случае не пытаться тушить электроприборы водой. Нельзя прятаться в шкафах или подсоках. При возгорании одежды необходимо сразу же упасть на пол и делать перекаты. Не пытаться спуститься самостоятельно через окно при расположении такового на большой высоте;

− При возможности помочь пострадавшим. При термическом ожоге как можно быстрее охладить место ожога. Если ожоговая рана открыта, то промывать её водой нельзя. Нельзя смазывать ожоги маслом, вскрывать пузыри, срывать одежду.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был разработан загрузчик с функциями проверки аутентичности и целостности кода и возможностью хранения различных файлов в прошивке устройства. Процесс разработки был разделен на следующие этапы:

– этап анализа предметной области, в котором было рассмотрено состояние рынка интернета вещей, и состояние безопасности прошивок устройств. Также были рассмотрены основные подходы к обеспечению безопасности и существующие решения: MCUboot, ZEPboot и wolfBoot.

– этап проектирования загрузчика, во время которого были спроектированы 7 функциональных модулей и алгоритмы основных действий загрузчика: загрузки, работы с файловой системой и загрузки исполняемого файла. В качестве средств разработки были выбраны: язык программирования «C++», компилятор gcc, система сборки Make, криптографические библиотеки Cifra и microecc.

– этап разработки загрузчика, во время которого был написан демонстрационный вариант программного продукта и все необходимые утилиты.

В настоящий момент загрузчик имеет весь необходимый функционал для полной реализации заявленных требований. Реализованы программные интерфейсы, позволяющие легко переносить его между контроллерами разных производителей.

# БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1 Булгаков А.Б., Безопасность жизнедеятельности: учебное пособие /А.Б. Булгаков. – Благовещенск: Изд-во АмГУ, 2013. – 627 с.

2 Симакова Н.Н. Организация рабочих мест с персональными электронно-вычислительными машинами (ПЭВМ): учебное пособие / Симакова Н.Н. — Новосибирск: Сибирский государственный университет телекоммуникаций и информатики, 2013. — 78 с. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/55486.html. 02.06.2022

3 Internet of Things statistics for 2022 [Электронный ресурс] / URL: https://dataprot.net/statistics/iot-statistics/. – 02.06.2022

4 The State of IoT Security [Электронный ресурс] / URL: https://www.iot-forall.com/the-state-of-iot-security-enterprises-are-leaving-their-front-doors-unlocked. – 02.06.2022

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Документация Cifra [Электронный ресурс]: офиц. сайт – Режим доступа: https://cifra.readthedocs.io/en/latest/. – 02.06.2022

2 Документация C++ [Электронный ресурс]: офиц. сайт – Режим доступа: https://en.cppreference.com/w/. – 02.06.2022

3 Документация Make [Электронный ресурс]: офиц. сайт – Режим доступа: https://www.gnu.org/software/make/manual/make.html. – 02.06.2022

4 Кардаш, Т. А. Эргономика рабочих мест служащих и инженернотехнических работников, оснащенных ПЭВМ [Текст]: учеб. пособие / Т. А. Кардаш; АмГУ, ИФФ. - Благовещенск: Изд-во Амур. гос. ун-та, 2002. – 60 с.

5 Майкл Ф., Эффективная работа с унаследованным кодом. / Майкл Ф.: , 2018. – 412 с.

6 Мартин Р. Чистая архитектура. / Р. Мартин. – М.: Альфа-Книга, 2015. – 436 с.

7 Мартин Р. Чистый код. Создание, анализ и рефакторинг. / Р. Мартин. – М.: Альфа-Книга, 2013. – 465 с.

8 Нурмухаметов А.Р., Жаботинский Е.А., Курмангалеев Ш. Ф., Гайсарян С. С., Вишняков А. В. Мелкогранулярная рандомизация адресного пространства программы при запуске [Текст] / Нурмухаметов А.Р., Жаботинский Е.А., Курмангалеев Ш. Ф., Гайсарян С. С., Вишняков А. В. // Труды ИСП РАН – 2017. – том 29, выпуск 6 – С. 163 - 182.

9 Описание формата ELF [Электронный ресурс]: офиц. сайт – Режим доступа: https://www.cs.cmu.edu/afs/cs/academic/class/15213-f00/docs/elf.pdf. – 02.06.2022

10 Пособие по безопасной работе на персональных компьютерах [Текст] / разраб. В. К. Шумилин. - М.: НЦ ЭНАС, 2005. - 28 с.

11 Прата, Стивен Язык программирования C++. Лекции и упражнения / Стивен Прата. – М.: Вильямс, 2015. – 1248 с.

12 СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания». 28 января 2021.

13 Седжвик, Роберт Алгоритмы на C++ / Роберт Седжвик. – М.: Вильямс, 2014. – 1056 с.

14 Симакова Н.Н. Организация рабочих мест с персональными электронно-вычислительными машинами (ПЭВМ): учебное пособие / Симакова Н.Н. — Новосибирск: Сибирский государственный университет телекоммуникаций и информатики, 2013. — 78 с. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/55486.html. 02.06.2022

15 Томас К. Алгоритмы: построение и анализ. / Томас Кормен, Чарльз Лейзерсон: Альфа-Книга, 2018. – 302 с. (дата обращения: 01.05.2021).

16 Уильямс, Энтони Параллельное программирование на C++ в действии. Практика разработки многопоточных программ / Энтони Уильямс. – М.: ДМК Пресс, 2012. – 672 с.

17 Уоррен Г., Алгоритмические трюки для программистов. / Г. Уоррен. - М.: Вильямс, 2014. – 512 с.

18 Фаулер М. Рефакторинг. Улучшение проекта существующего кода. / М. Фаулер. – М.: Альфа-Книга, 2017. – 448 с. (дата обращения: 04.05.2021).

19 Шумилин, В.К. ПЭВМ. Защита пользователя [Текст] / Шумилин В.К. - М.: Охрана труда и социальное страхование, 2001. – 214 с.

20 Юрий, Магда Использование ассемблера для оптимизации программ на C++ (+ CD-ROM) / Магда Юрий. – М.: БХВ-Петербург, 2013. – 496 с.

21 ARMv7-M Architecture Reference Manual [Электронный ресурс]: офиц. сайт – Режим доступа: https://developer.arm.com/documentation/ddi0403/latest. – 02.06.2022

22 gcc.gnu.org [Электронный ресурс]: офиц. сайт – Режим доступа: https://gcc.gnu.org/onlinedocs/gcc/ARM-Options.html. – 02.06.2022

23 Internet of Things statistics for 2022 [Электронный ресурс] / URL:

https://dataprot.net/statistics/iot-statistics/. – 02.06.2022

24 The State of IoT Security [Электронный ресурс] / URL: https://www.iot-forall.com/the-state-of-iot-security-enterprises-are-leaving-their-front-doors-unlocked. – 02.06.2022

ПРИЛОЖЕНИЕ А

Техническое задание

1 ОБЩИЕ СВЕДЕНИЯ

**1.1 Полное наименование системы**

Защищенный загрузчик для систем интернета вещей.

**1.2 Область применения**

Данный программный продукт должен позволить повысить защиту встраиваемых систем, которые подключены к системам коммуникаций за счет использования криптографических алгоритмов и дополнительных мер, усложняющих проникновение внутрь системы.

**1.3 Наименование предприятий разработчика и заказчика системы**

Разработчик: студент группы 855-об, факультета математики и информатики, Амурского государственного университета Олейник Илья Антонович.

Заказчик: Федеральное государственное бюджетное образовательное учреждение высшего образования «Амурский государственный университет»

Фактический адрес: 675028, Амурская область, г. Благовещенск, ул. Игнатьевское шоссе, 21, Телефон: +74162234500

**1.4 Перечень документов**

Документы, на основании которых создается ИС:

– ГОСТ 34.601-89 – техническое задание на проектирование автоматизированной системы;

– РД 50-34.698-90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.

**1.5 Плановые сроки начала и окончания работы**

Срок начала работ: 10.02.2022г.

Срок окончания работ: 20.06.2022г.

В процессе разработки сроки могут быть уточнены.

**1.6 Сведения об источниках и порядке финансирования работ**

Так как разработка программного приложения выполняется в рамках учебного проекта, финансирование не осуществляется.

**1.7 Порядок оформления и предъявления заказчику результатов работ по созданию системы**

Результаты работы предъявляются Заказчику в виде:

– функционирующей ИС, представляющей собой загрузчик и вспомогательные утилиты;

– исходных файлов проекта;

Результаты предоставляются Исполнителем в сроки, установленные Заказчиком. Приемка системы осуществляется Заказчиком в установленном порядке. Порядок предъявления системы, ее испытаний и окончательной приемки определен в разделе 6 настоящего технического задания.

2 НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

**2.1 Назначение системы**

Проектируемая система должна выполнять следующие функции:

– аутентификацию кода и данных;

– возможность обновления;

– поддержка хранения файлов в прошивке;

– шифрование данных;

– поддержка загрузки исполняемых файлов ELF формата.

**2.2 Цели создания системы**

Целью создания программного продукта является улучшение безопасности встраиваемых систем интернета вещей, добавление функционала, обеспечивающего более простое создание прошивки устройства.

Продолжение ПРИЛОЖЕНИЯ А

Техническое задание

# 3 ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

## 3.1 Краткие сведения об объекте автоматизации

Под объектом автоматизации понимаются вычислительные модули встраиваемых систем.

# 4 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ПРОДУКТУ

## 4.1 Требования к приложению

### 4.1.1 Требования к структуре и функционированию

Загрузчик должен реализовывать следующие функции:

– аутентификацию кода и данных;

– возможность обновления;

– поддержка хранения файлов в прошивке;

– шифрование данных;

– поддержка загрузки исполняемых файлов ELF формата.

### 4.1.2 Требования к квалификации и численности персонала, режиму его работы

Пользователю программного обеспечения, необходимо ознакомиться с руководством пользователя, чтобы иметь возможность обновления прошивки и управление режимом загрузки.

### 4.1.3 Требования к интерфейсу пользователя

К интерфейсу пользователя предъявляются следующие требования:

– взаимодействие с системой должно происходить в текстовом формате;

– текстовое описание команд должно хорошо читаться и иметь приятный дизайн;

– текстовое описание должно быть понятным и однозначно интерпретируемым;

– управление в ПО должно быть интуитивно понятным и удобным.

4.1.4 Перспективы развития, модернизация системы

При модернизации программного обеспечения могут вноситься изменения или осуществляться дополнения приложения. К ним относятся:

– улучшение качества имеющихся функций;

– добавление новых режимов загрузки;

– добавление поддержки новых аппаратных платформ;

– добавление новых функций файловой системы;

– добавление новых функций.

4.1.5 Требования к защите информации от несанкционированного доступа

В приложении необходимо реализовать защиту от несанкционированного доступа, с помощью методов шифрования и аутентификации кода и данных.

**4.2 Требования к видам обеспечения**

4.2.1 Требования к информационному обеспечению и программной документации

Состав программной документации, предъявляемой на этапе внедрения:

– ГОСТ 19.402-78 – описание программы;

– ГОСТ 19.401-78 – тестирование программы.

4.2.2 Требования к лингвистическому обеспечению

Для разработки и поддержки данного загрузчика необходимы знания языка программирования C++, python, ассемблера, а также умение пользоваться инструментами gcc, gdb, make.

4.2.3 Требования к программному обеспечению

Для разработки программного продукта необходимо иметь следующее программное обеспечение:

– 64-разрядная операционная система Linux;

– компилятор gcc 10 или выше;

– интерпретатор python 3.7 или выше;

– система сборки Make.

4.2.4 Требования к техническому обеспечению

Для разработки необходимо иметь персональный компьютер или ноутбук, к которым предъявляются следующие минимальные требования:

– процессор с архитектурой x86-64 и поддержкой набора инструкций SSE2;

– 30 ГБайт свободного места на жестком диске или SSD;

– 4 ГБайт ОЗУ.

Для эксплуатации ПО необходимо иметь микроконтроллер с архитектурой процессора ARMv7M.

4.2.5 Требования к условиям эксплуатации, и характеристика окружающей среды

Помещения, в которых предполагается использование ПО, должны соответствовать согласованным показателям температуры, влажности и освещённости.

Условия эксплуатации должны соответствовать нормальным климатическим условиям, определённым в ГОСТ 27201-87 и иметь следующие значения:

– искусственное освещение в помещениях эксплуатации компьютеров должно осуществляться системой равномерного освещения;

– температура воздуха в помещении от 15 °С до 25 °С;

– относительная влажность воздуха в помещении от 40% до 60% при температуре 25 °С;

– атмосферное давление от 630 мм. Рт. Ст. до 800 мм Рт. Ст.

4.2.6 Требования к организационному обеспечению

Для корректной эксплуатации системы следует разработать руководство пользователей и провести инструктаж.

5 СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ

**5.1 Перечень стадий и этапов работ по созданию системы**

Этапы создания ПО, которые необходимо выполнить:

1 этап – разработка технического задания, определение требований к приложению, стадий, этапов и сроков разработки программы, согласование и утверждение технического задания;

2 этап – анализ процессов деятельности организации. В конце этого этапа будут разработаны диаграммы организационной структуры организации, а также диаграммы ее внешнего и внутреннего документооборота;

3 этап – анализ предметной области и средств разработки. В конце данного этапа будет изучена технологии защиты кода встраиваемых систем и их реализации, выбраны программно-технические средства для разработки приложения;

4 этап – разработка программного продукта. В конце этого этапа будет разработан прототип загрузчика;

5 этап – тестирование программного продукта. В конце этого этапа будут выявлены недостатки и ошибки в работе приложения;

6 этап – доработка программного продукта;

7 этап – согласование созданного приложения с требованиями заказчика;

8 этап – внедрение и сопровождение загрузчика, обучение пользователей работе с системой, устранение обнаруженных неполадок.

### 5.2 Состав организации исполнителя работ

Все работы выполняются студентом факультета математики и информатики Амурского государственного университета Олейник Ильей Антоновичем.

## 6 ТРЕБОВАНИЯ К ПРИЕМКЕ-СДАЧЕ ПРОЕКТА

### 6.1 Виды, состав, объем и методы испытаний загрузчика

Должны быть проведены следующие виды испытаний:

– предварительные испытания;

– опытная эксплуатация;

– приёмочные испытания.

На этапе предварительных испытаний проводится тестирование загрузчика, проверка его работоспособности при взаимодействии с подготовленными образами прошивки.

На этапе опытной эксплуатации проверяется работоспособность приложения на реальных занятиях, при работе учащихся с рабочей тетрадью. В ходе этого этапа устраняются выявленные недостатки системы.

Приемочные испытания проводят для определения соответствия системы техническому заданию, оценки качества опытной эксплуатации и решения вопроса о возможности приемки системы в постоянную эксплуатацию.

### 6.2 Общие требования к приемке работ по стадиям

Предварительные испытания и эксплуатация проводятся на аппаратных средствах Исполнителя.

По результатам испытаний возможны доработки и исправления. Выявленные в ПО и документации недостатки Исполнитель исправляет в специально оговоренные после проведения испытаний сроки.

Сдача-приёмка работ производится поэтапно, в соответствии с календарным планом.

Все создаваемые в рамках настоящей работы программные изделия передаются Заказчику, как в виде готового файла для установки приложения, так и в виде исходных кодов, представляемых в электронной форме на стандартном машинном носителе.

## 7 ТРЕБОВАНИЯ К СОСТАВУ И СОДЕРЖАНИЮ РАБОТ ПО ПОДГОТОВКЕ ОРГАНИЗАЦИИ ЗАКАЗЧИКА К ВВОДУ ПРИЛОЖЕНИЯ В ДЕЙСТВИЕ

Перед вводом в эксплуатацию готового продукта разработчик должен договорится с руководителем организации о временном промежутке, в течении которого он обязан внедрить разработанный программный продукт. Под внедрением понимается комплекс мероприятий, включающий обучение пользователей, установку приложения для дальнейшего использования, предоставление необходимой документации к загрузчику.

### 7.1 Организационные мероприятия

Под организационными мероприятиями понимаются ознакомление пользователей с «Руководством пользователя», а также предоставление инструкций по установке приложения на встраиваемые системы пользователей.

## 8 ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ

При вводе программы в эксплуатацию пакет сопроводительных документов должен включать:

– техническое задание;

– описание программного продукта;

– руководство пользователя.