МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт Факультет приборостроения, информационных технологий и электроники	Кафедра «Информационная безопасность систем и технологий»
Специальность 100503 –	Информационная безопасность автоматизированных систем
Специализация –	Защищенные автоматизированные системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

МЕТОДИКА ВЫЯВЛЕНИЯ ОПАСНЫХ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ВСТРОЕННОГО ПРГРАММНОГО ОБЕСПЕЧЕНИЯ (BIOS) СРЕДСТВ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ, ПРИМЕНЯЕМЫХ В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ В ЗАЩИЩЕННОМ ИСПОЛНЕНИИ

Студент _____ Васин В.И.

Руководитель _____ Зефиров С.Л.

Консультант _____ Боровиков А.Ю.

Нормоконтролёр _____ Фатеев А.Г.

Рецензент ведущий инженер ПФ ФГУП „НТЦ „Атлас" _____ Пелин В.Н.

Работа допущена к защите (протокол заседания кафедры от 20.06.2016 № 14)

Заведующий кафедрой _____ Зефиров С.Л.

Работа защищена с отметкой отлично (протокол заседания ГЭК от 24.06.16 № 10)

Секретарь ГЭК _____ Иванов А.П.

Пенза 2016

РЕФЕРАТ

Отчет содержит: 77 страниц, 20 рисунков, 10 таблиц, 7 источников.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА В ЗАЩИЩЕННОМ ИСПОЛНЕНИИ, ВСТРОЕННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, МЕТОДИКА ВЫЯВЛЕНИЯ, ОПАСНЫЕ ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ, СРЕДСТВО ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ, УГРОЗА БЕЗОПАСНОСТИ ИНФОРМАЦИИ.

Цель работы – разработка методики, позволяющей выявлять опасные функциональные возможности встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

Объектом исследования является автоматизированная система в защищенном исполнении. Предметом исследования является встроенное программное обеспечения (BIOS) средств вычислительной техники автоматизированных систем в защищенном исполнении и его опасные функциональные возможности.

В процессе работы были рассмотрены существующие подходы к анализу встроенного программного обеспечения (BIOS) и методы тестирования программного обеспечения, рассмотрена типовая архитектура автоматизированной системы в защищенном исполнении и типовой состав средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении. Также было проведено моделирование угроз безопасности, разработана модель нарушителя, проведен анализ функциональных возможностей аппаратного обеспечения и встроенного программного обеспечения (BIOS), выявлен перечень опасных функциональных возможностей встроенного программного обеспечения (BIOS), разработан комплекс организационно-технических мер защиты от негативных действий опасных функциональных возможностей встроенного программного обеспечения (BIOS), а также проведены экспериментальные работы,

позволяющие обосновать достаточность данных организационно-технических мер.

В результате проведенной работы была разработана методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении. В соответствии с данной методикой разработаны меры защиты от негативных действий функциональных возможностей встроенного программного обеспечения (BIOS) на аппаратно-программную среду средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

СОДЕРЖАНИЕ

Введение 7

1 Обзор современного встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении 9

2 Обзор существующих подходов к анализу встроенного программного обеспечения (BIOS) 11

3 Обзор методов тестирования программного обеспечения 14

3.1 Полнота тестирования 14

3.2 Типы тестирования программного обеспечения 14

4 Концепция построения автоматизированной системы в защищенном исполнении .. 17

4.1 Типовая архитектура автоматизированной системы в защищенном исполнении 17

4.2 Типовой состав средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении 18

4.3 Модель угроз безопасности 23

4.4 Модель нарушителя 29

5 Методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах защищенного исполнения 31

5.1 Анализ функциональных возможностей аппаратного обеспечения 31

5.1.1 Общие сведения об аппаратных возможностях 31

5.1.2 Карта памяти 32

5.2 Анализ функциональных возможностей встроенного программного обеспечения (BIOS) 34

5.2.1 Описание секций встроенного программного обеспечения 34

5.2.2 Функциональные возможности UEFI BIOS 44

5.3 Выявление перечня опасных функциональных возможностей встроенного программного обеспечения (BIOS) 60

5.3.1 Определение перечня функциональных возможностей встроенного программного обеспечения (BIOS) 60

5.3.2 Определение перечня опасных функциональных возможностей встроенного программного обеспечения (BIOS) 61

5.4 Разработка комплекса организационно-технических мер защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS) 64

5.4.1 Определение перечня организационно-технических мер защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS) 64

5.4.2 Рекомендации по установке и настройке BIOS с целью защиты от негативных действий функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах защищенного исполнения 66

5.5 Экспериментальные работы, позволяющие обосновать достаточность принятых организационно-технических мер защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS) 73

Заключение 76

Список используемых источников 77

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей выпускной квалификационной работе применяют следующие обозначения и сокращения:

- AMI – american megatrends incorporated;
- BIOS – basic input output system;
- UEFI – unified extensible firmware interface;
- АРМ – автоматизированное рабочее место;
- АС – автоматизированная система;
- АСЗИ – автоматизированная система в защищенном исполнении;
- ВПО – встроенное программное обеспечение.
- ЛВС – локальная вычислительная сеть;
- МЭ – межсетевой экран;
- НСД – несанкционированный доступ;
- ОС – операционная система;
- ПАК – программно-аппаратный комплекс;
- ПО – программное обеспечение;
- СВТ – средство вычислительной техники;
- СЗИ – средство защиты информации;
- СОВ – система обнаружения вторжений.

ВВЕДЕНИЕ

В современном мире автоматизированная система является организационно-технической системой, которая позволяет вырабатывать решения на основе автоматизации информационных процессов в различных сферах деятельности. Важным моментом является то, что для обработки информации, необходимость защиты которой определяется законодательством Российской Федерации или решением ее обладателя, должны создаваться АСЗИ, в которых реализованы в соответствии с действующими нормативными правовыми актами требования о защите информации.

Процесс создания АСЗИ заключается в выполнении совокупности мероприятий, направленных на разработку и/или практическое применение информационной технологии, реализующей функции по защите информации согласно требованиям стандартов и нормативных документов по информационной безопасности. Важной особенностью является то что, инфраструктура АС так же определяется на этапе создания АСЗИ. Одним из основных элементов современной автоматизированной системы неотъемлемо является такое средство вычислительной техники как компьютер.

Актуальность настоящего дипломного проекта объясняется тем, что на данный момент компьютеры, входящие в состав АСЗИ, построены на системной логике иностранного производства. В связи с этим возникает потребность в проведении более детального аудита информационной безопасности таких средств вычислительной техники, где одной из основных составляющих является анализ встроенного программного обеспечения.

Целью настоящего дипломного проекта является разработка методики, позволяющей выявлять опасные функциональные возможности встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

В первом разделе проведен обзор современного встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

Во втором разделе проведен обзор существующих подходов к анализу встроенного программного обеспечения (BIOS).

В третьем разделе проведен обзор существующих методов тестирования программного обеспечения.

В четвертом разделе рассмотрена концепция построения автоматизированной системы в защищенном исполнении, проведено моделирование угроз безопасности и разработана модель нарушителя.

В пятом разделе проведен анализ функциональных возможностей аппаратного обеспечения и функциональных возможностей встроенного программного обеспечения (BIOS), по результатам которого выявлен перечень опасных функциональных возможностей встроенного программного обеспечения (BIOS) и разработан комплекс организационно-технических мер защиты от негативных действий выявленных опасных функциональных возможностей встроенного программного обеспечения (BIOS), а также проведены экспериментальные работы, позволяющие обосновать достаточность принятых мер защиты.

1 Обзор современного встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении

Одним из основных элементов современной автоматизированной системы (АС) неотъемлемо является такое средство вычислительной техники (СВТ) как компьютер. На данный момент популярные модели компьютеров, используемых повсеместно, построены на системной логике иностранного производства разработанной такими фирмами как Intel и AMD. Для правильного функционирования компьютера на протяжении всего времени была необходима такая система, которая позволяла бы правильно инициализировать все аппаратные компоненты и управлять питанием. Такой системой, и по настоящее время, является созданная более 30 лет назад базовая система ввода-вывода (BIOS).

Основными лидерами рынка производства BIOS все время являются два производителя BIOS, всем известные компании как: Phoenix и American Megatrends (AMI), так же на рынке присутствуют фирмы-разработчики материнских плат или сторонние компании, заключившие договор с производителями.

На данный момент BIOS в таком виде, в котором он был создан, практически изжил себя, так как система, созданная для IBM PC совместимых компьютеров в настоящий момент, считается устаревшей. На ее замену поступила новая система Unified Extensible Firmware Interface, созданная как загрузочная инициатива Intel в 1998 году. UEFI BIOS не имеет ограничений, присутствующих в созданным ранее BIOS, и тем самым не затормаживает развитие современных вычислительных систем.

UEFI BIOS имеет все особенности предшественника, такие как: проверку оборудования во время запуска компьютера, операции ввода-вывода, а так же обеспечение информационного взаимодействия между устройствами. Следует подчеркнуть, если предшественник в основном оставался недоступным для

пользователей, то политика UEFI обратная, так как пользователю доступно более функциональное меню настройки (BIOS Setup). UEFI не является монолитным как его предшественник, а наоборот это гибко программируемый интерфейс и он располагается так же в специальной микросхеме на материнской плате, но его размер значительно больше.

Современный UEFI можно назвать псевдо-операционной системой, способной выходить в Интернет или запускать специальные программы. Сегодня UEFI можно встретить почти на любом чипе с 32/64-битной архитектурой благодаря исходному коду написанном на высокоуровневом языке программирования [1].

UEFI имеет стандарт, который кроме огромного количества расширяемых возможностей UEFI определяет базовые функциональные возможности. Один из последних стандартов включает в себя возможность написания универсальных драйверов устройств, запуска специальных программ и возможность использования низкоуровневой криптографии.

2 Обзор существующих подходов к анализу встроенного программного обеспечения (BIOS)

Существующие подходы к анализу встроенного программного обеспечения (BIOS) включают в себя основные методы обнаружения ошибок и потенциальных уязвимостей в программном обеспечении. Для обнаружения ошибок и потенциальных уязвимостей программного обеспечения применяются следующие методы:

- статический анализ:
  - автоматизированный статический анализ бинарного кода;
  - автоматизированный статический анализ исходного кода;
  - ручной статический анализ;
- динамический анализ:
  - автоматизированный динамический анализ;
  - динамический анализ методом черного и белого ящика;
  - динамический анализ с ручной интерпретацией результатов;
  - динамический анализ с автоматической интерпретацией результатов;
  - фаззинг − тестирование;
- ручной анализ.

Автоматизированный статический анализ выполняется с помощью статических анализаторов программ. Это специальные программные средства, позволяющие производить сканирование исходных текстов программ и выявлять возможные ошибки, а также потенциальные противоречия. Для такого вида инструментов не требуется исполняемая программа, потому что выполняется синтаксический разбор текста программы, путем распознавания различных типов операторов. С их помощью можно проверить, правильно ли написаны операторы или сделать выводы о влиянии потока управления на поток данных в программе.

Автоматический статический анализ исходного кода выполняется с помощью декомпиляторов. Декомпилятор – это программное средство позволяющее транслировать исполняемый модуль в эквивалентный исходный код программы. В большинстве случаев современный статический анализ бинарного кода выполняется с помощью дизассемблеров, в силу того применяются защитные механизмы или особенности языка программирования. Дизассемблер – это транслятор позволяющий преобразовывать бинарный код в текст программы на языке ассемблера. По режиму работы с пользователем подразделяются на автоматические и интерактивные.

Ручной статический анализ выполняется методом инспекции кода. Данный способ заключается в внимательной инспекции исходного кода и формировании отчета о проведенном анализе. В процессе инспекции кода выявляются ошибки, расхождения в стиле написания кода, недочеты или участки кода с потенциальными ошибками.

Что касается динамического анализа кода, то это такой способ анализа выполняемый непосредственно при выполнении программы. Данный процесс можно разбить на несколько стадий: подготовка исходных данных для анализа, проведение тестового запуска программы и сбор необходимых параметров, анализ полученных данных в результате сбора.

Автоматизированный динамический анализ выполняется с помощью специальных анализаторов, которые в процессе работы автоматически генерируют необходимые данные для проведения проверки [2]. Анализ может производиться по принципам тестирования белого или чёрного ящика. Отличие данных методов только в том, что «белый ящик» — имеются данные о программном коде, а «черный ящик» – нет. Так же существует так называемый метод «серого ящика», это когда известна структура программы, но в самом тестировании эти данные не используются, а позволяют нам более точно подбирать входящие параметры и проверять выходящие.

Динамический анализ с автоматизированной и ручной интерпретацией результатов, например, фаззинг – выполняется методом ввода случайных значений и путем отслеживания поведения программы. При использовании ручного ввода случайных значений результаты испытания будут интерпретироваться соответственно вручную, и, наоборот, при автоматизированных тестах результаты интерпретируются автоматически.

Фаззинг-тестирование − методика тестирования, при которой вместо ожидаемых входных данных передаются случайные, невалидные данные. Основная идея данного подхода заключается в том, чтобы случайным образом влиять на поведение программы, изменяя входные данные в произвольном порядке. Большинство фаззеров разрабатываются внутри организаций для тестирования и анализа собственных продуктов. Важно отметить то, что использование фаззеров является обязательным этапом жизненного цикла безопасной разработки Microsoft.

3 Обзор методов тестирования программного обеспечения

3.1 Полнота тестирования

Тестирование программного обеспечения – это исследование, включающее все динамические и статические стадии жизненного цикла, затрагивающие его оценку с целью обнаружения дефектов.

Таким образом результативность решения задачи тестирования определяется выбранным набором тестовых ситуаций. Процесс тестирования эвентуально нескончаем, как с теоретической, но и фактической точки зрения. В связи с этим возникает необходимость в правилах, позволяющих определить набор подходящих тестовых ситуаций. Данные правила называют критериями полноты тестирования.

Распространённой практикой тестировщиков является разбиение всех ситуаций поведения программы на определенные классы эквивалентности, таким образом, что ситуации из близлежащих классов достаточно схожи друг с другом, а работа программного обеспечения в них фактически значимым образом не отличается. Полноту тестирования можно определять по доле «покрытых» в нем классов ситуаций. Следовательно, процент покрытых тестов классов ситуаций называется достигнутым тестовым покрытием, а сам критерий – критерий тестового покрытия [3]. Как уже отмечалось ранее, количество критериев полноты достаточно много, а тестирование, где результат покрытия 100% по одному критерию классов ситуаций, не является полным потому, что реальные системы не имеют ограниченное количество конечных состояний.

3.2 Типы тестирования программного обеспечения

Существующая классификация типов тестирования на данный момент достаточно сложна, потому что может подразделяться по различным аспектам. Далее приведены основные типы тестирования программного обеспечения, которые получили наибольшее распространение в современных условиях:

- тестирование черного ящика;
- тестирование белого ящика;

– модульное тестирование;
– интеграционное тестирование;
– функциональное тестирование;
– системное тестирование;
– регрессионное тестирование;
– нагрузочное тестирование;
– тестирование восстановления;
– тестирование безопасности.

Тестирование черного ящика – в данном типе тестирования не рассматривается внутренняя конструкция системы. Тесты основаны на выработанных требованиях и функциональных возможностях.

Тестирование белого ящика – данное тестирование основано на знании внутренней логики исходного кода программы. Для данного тестирования необходимо внутреннее программное обеспечение и исходный код программы, таким образом тесты строятся именно на основе путей выполнения, условий и ветвей программы.

Модульное тестирование – данное тестирование подразумевает собой тестирование отдельных программных компонентов или модулей. Обычно это делается программистом, а не тестировщиком потому, что для данного тестирования требуется детальные знания о внутренней структуре как всей программы, так и участка кода.

Интеграционное тестирование – данное тестирование основано на тестирование интегрированных модулей для проверки комбинированных функциональных возможностей после интеграции. Модулями, как правило, могут быть модули кода, отдельные приложения, клиентские и серверные приложения в сети и т.д. Этот тип тестирования особенно актуально для клиент–серверных приложений и распределенных систем.

Функциональное тестирование – этот тип тестирования игнорирует внутренние детали и сосредоточен на проверке соответствуют ли выходные данные требованиям или нет.

Системное тестирование – в данном типе тестирования вся система испытывается в соответствии с поставленными требованиями. Тестирование очень похоже на тестирование черного ящика, который основан на полных технических требованиях спецификаций и охватывает все комбинированные части системы.

Регрессионное тестирование – этот тип тестирования, как правило, применяется для обнаружения ошибок в протестированном модуле или функциональном участке, который будет подвержен модификации. Для данного тестирования, в основном, применяются средства автоматизации, так как охватить всю систему практически невозможно.

Нагрузочное тестирование – этот тип тестирования проводится для оценивания поведения программы под нагрузкой. Производится тестирование программы при больших нагрузках, такое как тестирование веб-сайта на предельных диапазонах запроса, чтобы определить, в какой момент времени отклик программы выйдет из строя или перестанет удовлетворять предъявленным требованиям.

Тестирование восстановления – этот тип тестирования применяется для того, чтобы проверить на сколько удовлетворительно касательно выработанных критериев программное обеспечение восстанавливается после аварий, сбоев оборудования или другие внешних факторов.

Тестирование безопасности – данное тестирование применяется для проверки подверженности программного обеспечения несанкционированному внутреннему или внешнему доступу.

4 Концепция построения автоматизированной системы в защищенном исполнении

4.1 Типовая архитектура автоматизированной системы в защищенном исполнении

В данном подразделе приводится типовая архитектура автоматизированной системы в защищенном исполнении, в которой осуществлено подключение двух локальных вычислительных сетей (ЛВС) обрабатывающих информацию различных категорий.

Взаимодействие между ЛВС осуществлено в соответствии с требованиями по безопасности информации. Для обеспечения требований по подключению ЛВС применяется межсетевой экран (МЭ) с системой обнаружения вторжения (СОВ), а для фильтрации данных используется специализированный протокол запроса-ответа информации и примененные специальные правила фильтрации, позволяющие пропускать только данные специализированного протокола, а остальные данные отбраковывать.

Архитектура АСЗИ представлена двумя ЛВС:

– ЛВС для служебного пользования (ДСП) – там, где циркулирует информация ограниченного доступа (защищаемая информация) представленная: сервером базы данных (БД), принтером, двумя автоматизированными рабочими местами (АРМ) и межсетевым экраном с системой обнаружения вторжений;

– ЛВС общедоступной информации (ОИ) – там, где циркулирует общедоступная информация (пакеты управления и т.д.) представленная: сервером управления, GSM модемом для приема и передачи управляющей информации и АРМ.

Далее на рисунке 1 приведена типовая архитектура АСЗИ. Типовая архитектура АСЗИ приведена в рамках данного дипломного проекта и в реальных условиях может отличаться от приведенной.

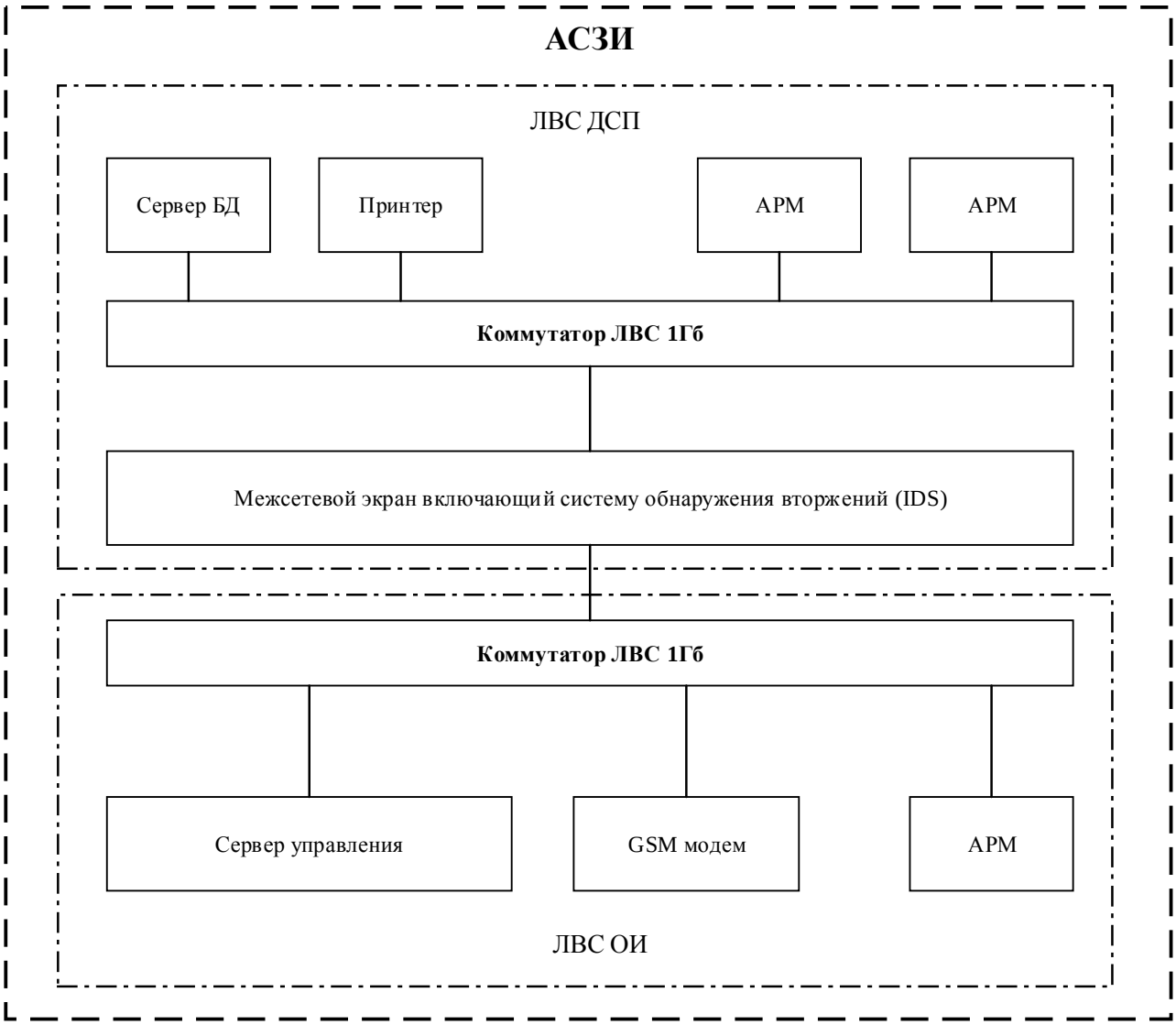


Рисунок 1 – Типовая архитектура АСЗИ

Необходимо учитывать, что при рассмотрении типовой архитектуры АСЗИ, целесообразно отмечать потребность в выявлении опасных функциональных возможностей встроенного программного обеспечения (BIOS) СВТ только в той части АС, в которой циркулирует критичная информация (ограниченного доступа), так как МЭ корректно настроен и обеспечивает обнаружение вторжений (воздействий) со стороны ЛВС ОИ.

4.2 Типовой состав средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении

Аппаратное обеспечение типового АРМ функционирует на современной серверной материнской плате SuperMicro X9SCA-F. Фотография платы представлена на рисунке 2.

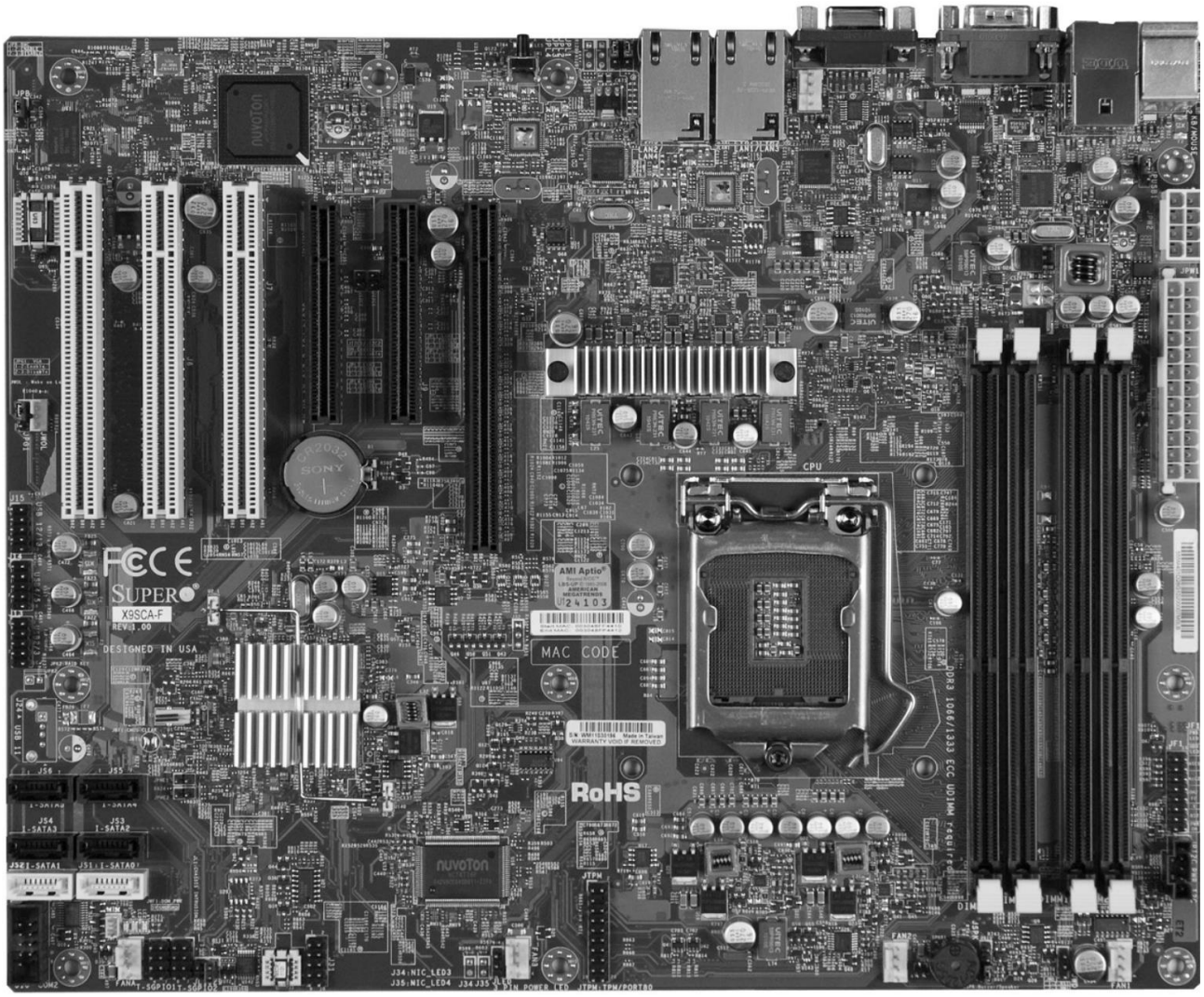


Рисунок 2 – Фотография материнской платы

Фотография выходной панели материнской платы с набором интерфейсов представлена на рисунке 3.

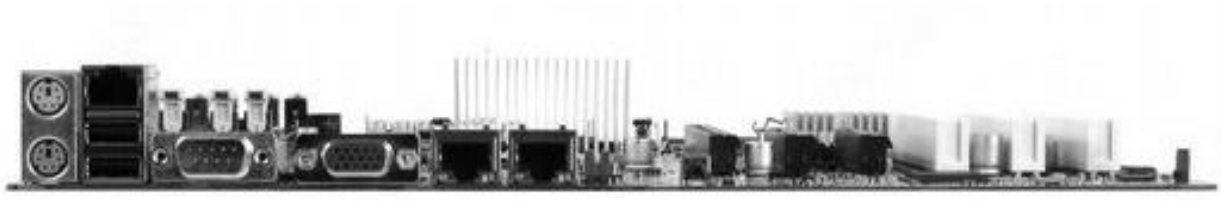


Рисунок 3 – Фотография выходной панели материнской платы

Более подробное размещение элементов на материнской плате (разъемов, перемычек и индикаторов) представлено на рисунке 4.

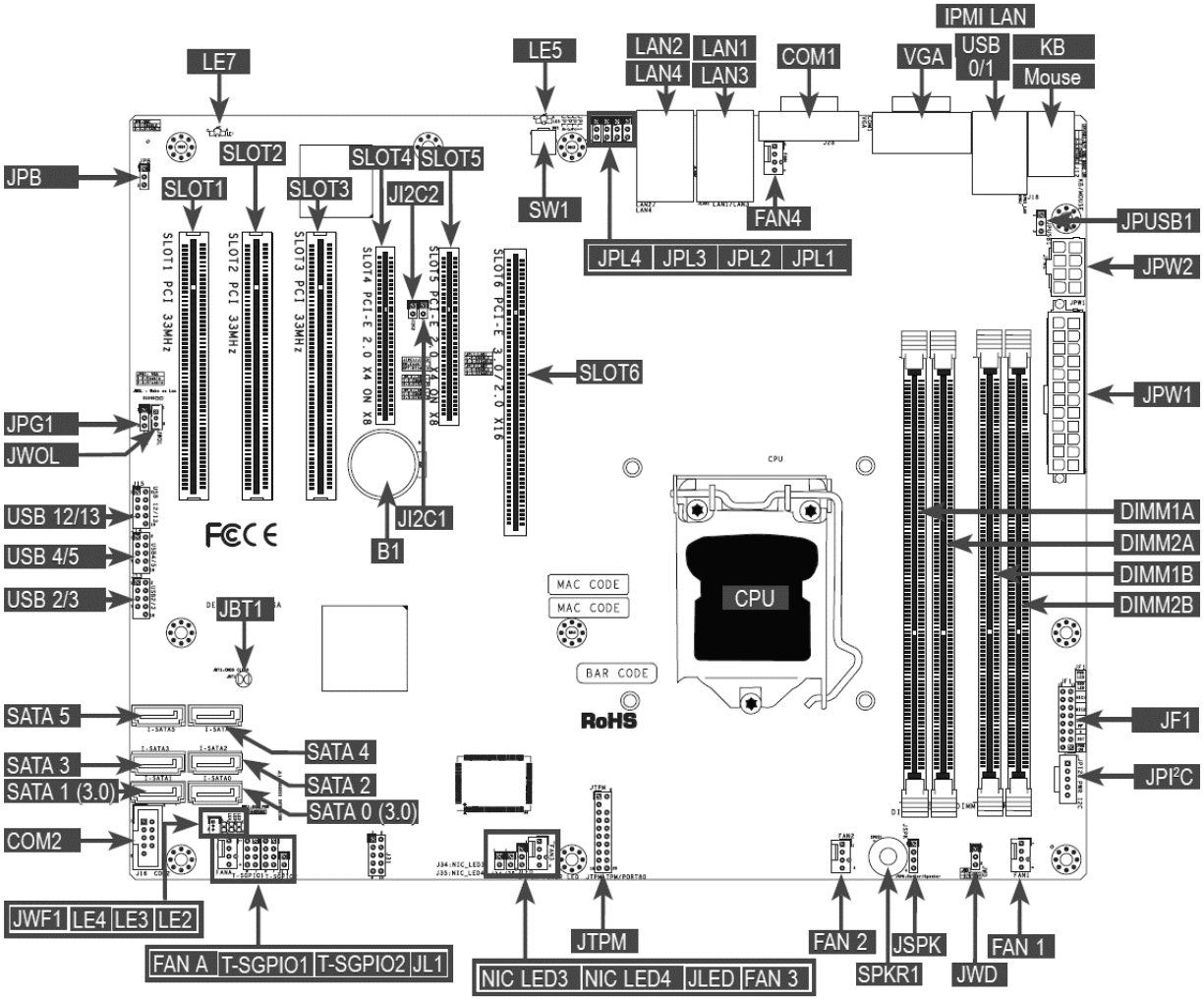


Рисунок 4 – Расположение элементов на материнской плате

Все элементы на материнской плате можно разделить соответственно на следующие группы: «Индикаторы», «Перемычки» и «Разъемы». Описание элементов группы «Индикаторы» представлено в таблице 1.

Таблица 1 – Индикаторы

Индикатор	Описание	Цвет/состояние	Статус
LE2	Индикатор работы системы	Зеленый	Включена
LE3	Индикатор установки неподдерживаемой памяти	Желтый: Моргает	Установлена неподдерживаемая память
LE4	Индикатор наличия питания	Зеленый: Горит постоянно	Подается питание на плату
LE5	Индикатор Unit ID (используется в сервисных	Синий: Горит постоянно	Индикатор включен

Продолжение таблицы 1

Индикатор	Описание	Цвет/состояние	Статус
	целях для идентификации платы, нуждающейся в обслуживании). Включается соответствующим переключателем (SW1) на задней панели.		
LE7	Индикатор работы IPMI	Зеленый: Моргает	IPMI работает в штатном режиме

Описание элементов группы «Перемычки» представлено соответственно в таблице 2.

Таблица 2 – Перемычки

Перемычка	Описание	По умолчанию
JBT1	Сброс CMOS	Разомкнуто
JI2C1/JI2C2	Включение SMB (System Management Bus) для слотов расширения PCI	Разомкнуто
JPB	Включение поддержки IPMI BMC	Включен (Контакты 1-2)
JPG1	Включение интегрированного графического адаптера	Включен (Контакты 1-2)
JPL1/JPL2	Включение сетевых интерфейсов LAN1/LAN2	Включен (Контакты 1-2)
JPUSB1	Разрешение выводить систему из режима сна с USB портов на задней панели ввода/вывода	Включен (Контакты 1-2)
JWD	Включение сторожевого таймера (Watch Dog Timer)	Сброшен (Контакты 1-2)

Описание элементов группы «Разъемы» представлено в таблице 3.

Таблица 3 – Разъемы

Разъем	Описание
B1	Гальванический элемент питания часов реального времени
COM1, COM2	COM порты
Fans 1-4, Fan A	Разъемы вентиляторов охлаждения системы/ЦП
JF1	Разъем для кнопок и индикаторов на передней панели
JL1	Разъем датчика вскрытия корпуса
JLED1	Разъем индикатора питания

Продолжение таблицы 3

Разъем	Описание
JPW1	24-выводной разъем основного питания
JPW2	+12V 8-выводной разъем питания процессора
KB, Mouse	Разъемы клавиатуры и мышки
LAN1/LAN2	Разъемы Gigabit Ethernet (RJ45)
IPMI LAN	Сетевой интерфейс IPMI
SATA 0/1	Разъемы SATA 3.0
SATA 2-5	Разъемы SATA 2.0
JPI2C	Разъем питания SMBus
JSPK	Разъем внешнего динамика
JTPM	Разъем TPM (Trusted Platform Module)
JWF1	Разъем питания SATA DOM (Disk On Module)
JWOL	Wake On LAN
SPKR1	Встроенный динамик
USB0/1	USB порты на задней панели
USB 2/3, USB 4/5, USB 12/13	Разъемы для USB портов на передней панели
VGA	VGA порт встроенной графики
DIMM 1A, 2A, 1B, 2B	DIMM слоты не буферизованной DDR3 памяти с поддержкой ECC Доступные частоты: 1333/1066 МГц
SW1	Переключатель Unit ID для включения индикатора UID (LE5) (используется в сервисных целях для идентификации платы, нуждающейся в обслуживании)
Slots 1-3	Разъемы PCI (33 МГц)
Slots 4-5	Разъемы PCI-Express 2.0 x4
Slot 6	Разъемы PCI-Express 2.0 x16

Структурная схема используемого процессорного модуля на материнской плате типового СВТ представлена на рисунке 5.

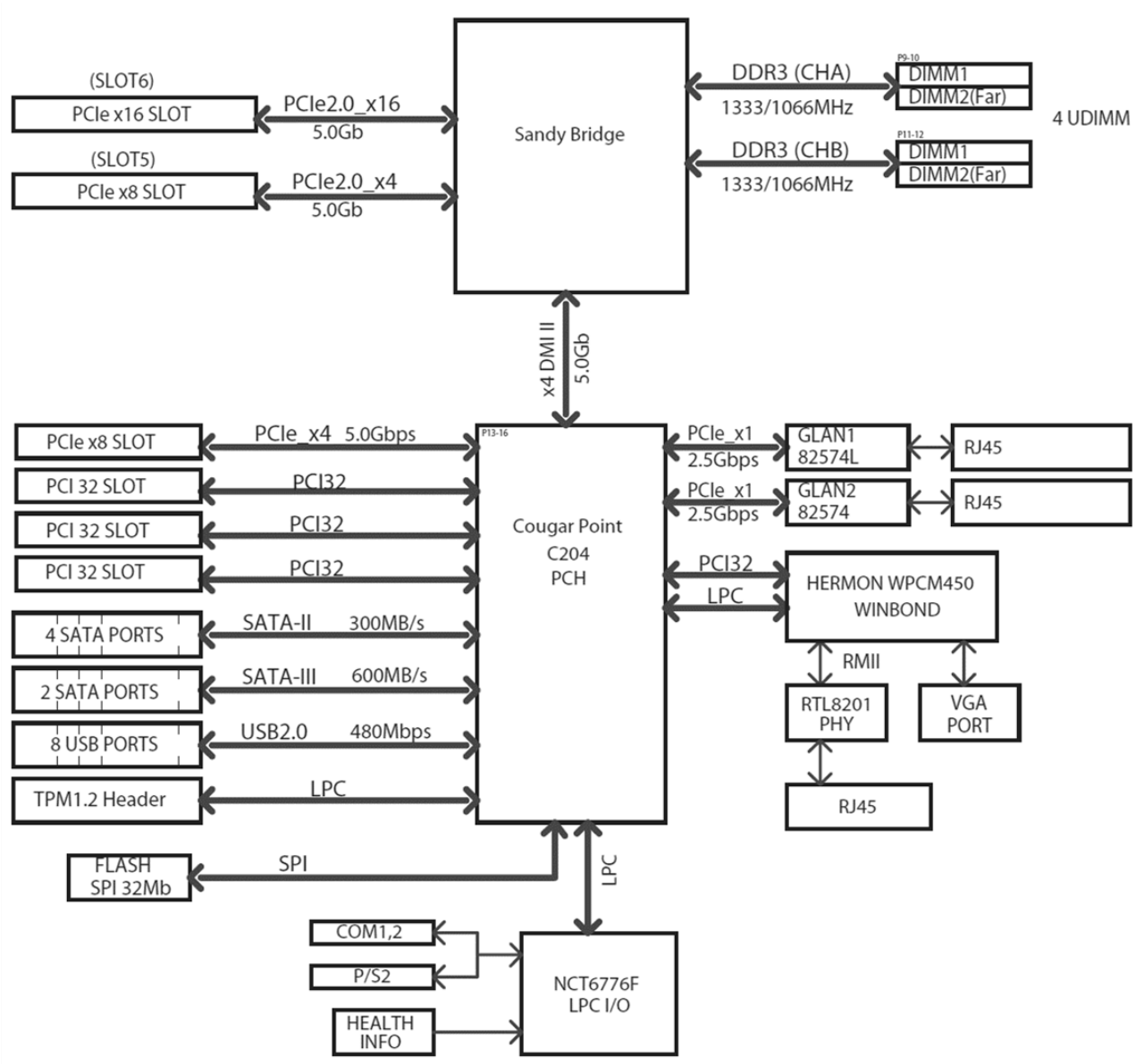


Рисунок 5 – Структурная схема процессорного модуля типового СВТ

4.3 Модель угроз безопасности

Идентификация угроз безопасности информации и составление их модели проводилось только для ЛВС ДСП, так как в ней циркулирует информация конфиденциального характера, нарушение свойств безопасности которой может нанести потенциальный ущерб. Угрозы для ЛВС ОИ далее не рассматривались, так как в ней циркулирует информация общедоступного характера, следовательно, нарушение её свойств безопасности не может нанести потенциальный ущерб.

ЛВС ДСП является системой не имеющей подключения к сетям общего пользования, но имеет подключение к ЛВС ОИ через межсетевой экран с

системой обнаружения вторжения, в пределах контролируемой зоны. Объектом воздействия угроз информационной безопасности является информация конфиденциального характера, обрабатывающаяся на АРМ.

При обработке конфиденциальной информации на АРМ в ЛВС ДСП возможно наличие следующих уязвимостей:

– небезопасная конфигурация оборудования;

– неправильное распределение прав доступа;

– дефекты программных средств;

– отсутствие необходимых средств защиты, контролирующих аутентификацию и проверку целостности;

– дефекты встроенного программного обеспечения (BIOS).

Все потенциальные нарушители (источники угроз) делятся на две группы:

– внутренние нарушители, это физические лица, имеющие право пребывания на территории контролируемой зоны, в пределах которой размещается оборудование;

– внешние нарушители, это физические лица, не имеющие права пребывания на территории контролируемой зоны, в пределах которой размещается оборудование.

Перечень угроз безопасности информации составлялся с использованием возможных источников угроз и объектов воздействий, в результате чего были идентифицированы следующие угрозы:

– угроза несанкционированного доступа (НСД) к конфиденциальной информации, обрабатываемой на АРМ;

– угроза внедрения вредоносного ПО;

– угроза обхода функций средств доверенной загрузки;

– угроза утечки конфиденциальной информации по каналам взаимодействия с ЛВС ОИ через межсетевой экран.

Далее было составлено описание угроз, результаты которого представлены в таблице 4.

Таблица 4 – Описание угроз

Описание угрозы	Источник	Нарушаемые свойства	Объекты воздействия	Уязвимость
Угроза НСД к конфиденциальной информации, обрабатываемой на АРМ.	Внутренний нарушитель.	Конфиденциальность.	Информация, обрабатываемая на АРМ.	Дефекты программных средств; Отсутствие необходимых средств защиты, контролирующих аутентификацию и проверку целостности; Дефекты ВПО (BIOS); Неправильное распределение прав доступа.
Угроза внедрения вредоносного ПО.	Внутренний нарушитель	Конфиденциальность; Целостность.	Информация, обрабатываемая на АРМ.	Дефекты программных средств; Отсутствие необходимых средств защиты, контролирующих аутентификацию и проверку целостности; Небезопасная конфигурация оборудования; Неправильное распределение прав доступа.
Угроза обхода функций средств доверенной загрузки.	Внутренний нарушитель	Конфиденциальность; Доступность.	Информация, обрабатываемая на АРМ.	Дефекты ВПО (BIOS); Отсутствие необходимых средств защиты, контролирующих аутентификацию и проверку целостности.
Угроза утечки конфиденциальной информации по каналам взаимодействия с ЛВС ОИ через межсетевой экран.	Внутренний нарушитель	Конфиденциальность.	Информация, обрабатываемая на АРМ.	Дефекты программных средств; Небезопасная конфигурация оборудования; Неправильное распределение прав доступа; Отсутствие необходимых средств, защиты контролирующих аутентификацию и проверку целостности; Дефекты ВПО (BIOS).

В ходе анализа типовой архитектуры АСЗИ было определено, что для обеспечения защиты информации, обрабатывающийся в АРМ ЛВС ДСП, уже

применяются защитные меры и средства защиты информации, прошедшие оценку соответствия в форме обязательной сертификации на соответствие требованиям по безопасности информации.

Перечень мер защиты информации, уже реализованных в АСЗИ, обеспечивающих блокирование (нейтрализацию) одной или нескольких уязвимостей, включенных в модель угроз безопасности информации представлен далее:

– управление идентификаторами, в том числе создание, присвоение, изменение, уничтожение идентификаторов;

– реализация мандатного метода управления доступом и правил разграничения доступа;

– использование сертифицированной операционной системы и сертифицированного специального программного обеспечения;

– ограничение программной среды;

– опечатывание (опломбирование) АРМ и его разъемов для защиты от несанкционированного доступа к аппаратному обеспечению и подключения внешних устройств;

– использование сертифицированной антивирусной защиты;

– использование межсетевого экрана и системы обнаружение вторжений.

Для реализации некоторых мер защиты используются:

– СЗИ от НСД «Secret Net LSP» 1.2 имеет сертификат ФСТЭК №2790, подтверждающий соответствие требованиям руководящих документов по 5-му классу защищенности по СВТ и по 4-му уровню контроля отсутствия НДВ, таким образом может использоваться для защиты АС до класса защищенности 1Г включительно;

– ПАК «Соболь» версия 3.0 (релиз 3.0.8) имеет сертификат ФСТЭК №1967, подтверждающий соответствие требованиям руководящих документов, таким образом может применяться для защиты АС до класса защищенности 1Б включительно;

– «Антивирус Касперского» 8.0 для Linux File Servers (версия 8.0.2.256) имеет сертификат ФСБ № СФ/СЗИ-0060, подтверждающий соответствие требованиям руководящих документов;

– «Astra Linux Special Edition» имеет сертификат ФСТЭК № 2557, подтверждающий соответствие операционной системы требованиям руководящих документов ФСТЭК России по 3-му классу защищенности СВТ и 2-му уровню контроля отсутствия недекларированных возможностей, таким образом операционная система может использоваться при создании АС до класса защищенности 1Б включительно;

– МЭ и СОВ «Рубикон-К» имеет сертификат ФСТЭК №3290, подтверждающий, что комплекс является средством защиты информации, реализующим функции межсетевого экрана и системы обнаружения вторжений уровня сети, и соответствует требованиям документов «Средства вычислительной техники. Межсетевые экраны. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа (Гостехкомиссия России, 1997) - по 3 классу защищенности, «Требования к системам обнаружения вторжений» (ФСТЭК России, 2011) - по 4 классу защиты, «Профиль защиты систем обнаружения вторжений уровня сети четвёртого класса защиты ИТ.СОВ.С4.ПЗ» (ФСТЭК России, 2012) - по 4 классу защиты и «Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей» (Гостехкомиссия Россия, 1999) - по 4 уровня контроля при выполнении указаний по эксплуатации, приведенных в формуляре.

Данные меры защиты покрывают существующие уязвимости не в полной мере, поэтому на основе полученных результатов далее была составлена адаптированная модель актуальных угроз безопасности информации и представлена в таблице 5.

Таблица 5 – Адаптированная модель актуальных угроз

Описание угрозы	Источник	Нарушаемые свойства	Объекты воздействия	Уязвимость
Угроза НСД к конфиден-циальной информации, обрабатываемой на АРМ.	Внутренний нарушитель.	Конфиден-циальность.	Информация, обрабатыва-емая на АРМ.	Дефекты ВПО (BIOS).
Угроза обхода функций средств доверенной загрузки.	Внутренний нарушитель.	Конфиден-циальность; Доступность.	Информация, обрабатыва-емая на АРМ.	Дефекты ВПО (BIOS);
Угроза утечки конфиден-циальной информации по каналам взаимодей-ствия с ЛВС ОИ через межсетевой экран.	Внутренний нарушитель.	Конфиден-циальность.	Информация, обрабатыва-емая на АРМ.	Дефекты ВПО (BIOS).

При проведении адаптации модели угроз безопасности информации с учетом используемых защитных мер в АСЗИ было определено, что использование имеющихся защитных мер без нейтрализации (блокирования) уязвимости «Дефекты ВПО (BIOS)» является недостаточным.

Для встроенного программного обеспечения (BIOS) АРМ возникновение данных угроз информационной безопасности может быть обусловлено следующими факторами и условиями:

– сбоями и неисправностями аппаратного обеспечения, содержащего встроенное программное обеспечение АРМ;

– ошибочными или преднамеренными действиями оператора при настройке встроенного программного обеспечения АРМ;

– негативными действиями опасных функциональных возможностей, внедренных или изначально присутствующих во встроенном программном обеспечении АРМ, направленными на нарушение штатных алгоритмов его функционирования.

Данные факторы и условия могут возникать как преднамеренно, так и случайно. Для реализации преднамеренных и случайных угроз требуются целенаправленные действия нарушителя – атака.

Для создания перечня способов атак (перечня возможных атак) необходимо создание перечней конкретных субъектов, объектов, целей, каналов, средств атаки, а также конкретных сведений об встроенном программном обеспечении АРМ, которые могут быть доступны нарушителю. Совокупность данных перечней характеризуют потенциального нарушителя и приводятся далее в подразделе «Модель нарушителя».

4.4 Модель нарушителя

Все потенциальные нарушители могут быть разделены на внешних и внутренних нарушителей.

К внешним нарушителям относятся лица, осуществляющие попытки выполнения атак на встроенное программного обеспечение АРМ по каналам связи из-за пределов контролируемой зоны. К внутренним нарушителям относятся лица, осуществляющие попытки выполнения атак на встроенное программное обеспечение в пределах контролируемой зоны.

В связи с тем, что по результатам рассмотрения архитектуры АСЗИ сделан вывод об отсутствии в ЛВС ДСП незащищенных каналов связи, выходящих за пределы контролируемой зоны, а каналы связи, выходящие из ЛВС ДСП в ЛВС ОИ, защищены межсетевым экраном с системой обнаружения вторжений, то внешнего нарушителя можно исключить из дальнейшего рассмотрения в качестве потенциального нарушителя.

Внутренний нарушитель имеет возможность:

– использовать разъемы на системном блоке АРМ для подключения штатного оборудования, установленного в том же помещении или внесенного извне, с помощью которого возможна активизация служебных режимов ПЭВМ, предусмотренных производителем для расширенного тестирования, изменения настроек и перепрошивки встроенного программного обеспечения;

– манипулировать настройками встроенного программного обеспечения, записанными в область данных прошивки BIOS, пытаясь перевести АРМ в нештатный режим работы;

– воздействовать на АРМ, используя специализированное интеллектуальное оборудование, настроенное на возможности конкретного типа ПЭВМ с целью воздействия на процесс функционирования ПЭВМ в режиме POST для отключения процедуры самотестирования, а также с целью, удаленной перепрошивки встроенного программного обеспечения.

Внутренний нарушитель имеет сведения о всех особенностями архитектуры построения конкретной версии встроенного программного обеспечения и при наличии в нем скрытых функциональных возможностей может целенаправленно их использовать.

По результатам проведенного анализа конструкции АРМ установлено, что при хранении и в процессе эксплуатации системный блок АРМ опечатан (опломбирован) и не может быть бесконтрольно вскрыт нарушителем с целью изменения состава аппаратных средств и переключения перемычек, а также в системном блоке отсутствуют средства бесконтактного контроля или доступа к внутренним устройствам (радиочастотные метки, радиоинтерфейс и другие средства дистанционного контроля и связи).

Установка программного обеспечения (в том числе встроенного) происходит доверенными сотрудниками (администратором), не являющимися нарушителями.

5 Методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах защищенного исполнения

5.1 Анализ функциональных возможностей аппаратного обеспечения

5.1.1 Общие сведения об аппаратных возможностях

В данном пункте рассмотрены основные функциональные возможности аппаратного обеспечения типового АРМ, инициализируемые с помощью встроенного программного обеспечения UEFI (Unified Extensible Firmware Interface). UEFI – унифицированный интерфейс между операционной системой и микропрограммами, управляющими низкоуровневыми функциями оборудования.

Таким образом, по результатам рассмотрения на прошлых этапах дипломного проектирования аппаратных составляющих, а именно перемычек, разъемов, элементов, индикаторов материнской платы и структурной схемы процессорного модуля были определены основные функциональные возможности аппаратного обеспечения:

– центральный процессор (CPU) Intel Xeon E3-1240 v2 3,4GHz. Основные характеристики: количество ядер 4 шт., тактовая частота 3,4GHz, 8 МБ кэш-памяти;

– оперативная память (SDRAM) Kingston. Основные характеристики: 4 слота SDRAM на частоте 1333 МГц общим объемом – 16 ГБ;

– чипсет «Южный мост» Intel® C204. Основные характеристики: поддерживаемая шина PCI, поддерживаемая шина PCI Express, контроллер USB 2.0, контроллер SATA, контроллер IDE, интегрированный сетевой адаптер, технология виртуализации Intel для направленного ввода/вывода (VT-d);

– графический процессор (VGA) NVIDIA Quadro K200D. Основные характеристики: память 2 ГБ GDDR5, разрядность 128-бит;

– контроллер удаленного мониторинга и управления физическим состоянием сервера BMC Nuvoton WPCM450. Основные характеристики: модуль Realtek RTL8201N PHY для поддержки IPMI 2.0;

– базовая система ввода-вывода (UEFI BIOS). Основные характеристики: перепрограммируемое постоянное запоминающие устройство (ППЗУ) SPI AMI BIOS® SM, размер микросхемы памяти 8 МБ.

5.1.2 Карта памяти

Карта памяти – структура данных, которая указывает, как память распределена. Карта памяти – одна из основных частей, необходимых операционной системе для того чтобы знать, как распределяется память, чтобы правильно загрузить ядро, драйвера и СПО.

Карта памяти включается в себя хранилища, напрямую доступные процессору:

– физическая память (основная память и память интегрированного графического процессора);

– спроецированные порты ввода-вывода (Memory Mapped I/O) (конфигурационное пространство шины PCI Express, APIC (контроллер прерываний) центрального процессора, видеобуфер, спроецированная в память микросхема с UEFI BIOS).

Однако, карта памяти не включает в себя:

– кэш процессора;

– накопители (жесткие диски, оптические накопители и USB-накопители.

Под физической памятью подразумевается основная память ЭВМ (DRAM – dynamic random access memory) и она подразделяется на три основных диапазона:

– область совместимости (Legacy) (0 – 1МБ);

– основная память до 4ГБ (1 – 4ГБ);

– основная память после 4 ГБ (>4ГБ).

Область совместимости необходима для совместимости со старыми операционными системами. Она поделена на следующие диапазоны, представленные в таблице 6.

Таблица 6 – Диапазоны области совместимости

Размер	Смещение	Описание
0 – 640КБ	0 – 0xA0000	Область DOS
768 – 896КБ	0xC0000 – 0xE0000	Расширенная область для Legacy OpROM (Option ROM)
896 КБ – 1МБ	0xE0000 – 0x100000	Область BIOS. Данная область может быть системной памятью или спроецированной на микросхему памяти BIOS.

Область основной памяти между 1МБ и 4ГБ используется ОС Astra Linux. Однако, память под границей 4ГБ выделена так же для устройств ввода-вывода, таких как флэш-память, контроллер прерываний и т.д., поэтому существует специальный регистр TOLUD (Top of Low Usable DRAM) для обозначения этой границы. Данная память подразделяется на следующие области:

– основная область DRAM. Данная область используется ОС МСВС;

– ISA Hole (15 – 16МБ);

– защищенная область памяти (PMR). Данная область недоступна контролеру DMA и защищена;

– память SMM. Данная область доступна процессору только если центральный процессор находится в режиме SMM;

– память интегрированного графического адаптера. Данная область используется только при наличии активированного графического адаптера;

– память Intel Manageability Engine (ME). Данная область в случае если физической памяти больше 4ГБ располагается выше границы 4ГБ.

Область памяти выше границы 4ГБ также используется в ОС Astra Linux. Так как в физической памяти присутствует область между TOLUD и 4 ГБ, то она перераспределена чипсетом для отображения памяти выше границы 4 ГБ. Для этого существует набор регистров Reclaim BASE/Reclaim SIZE. Память выше границы 4ГБ делится на следующие области:

– основная область DRAM. Данная область используется ОС Astra Linux;

– защищенная область памяти (PMR). Данная область недоступна контролеру DMA и защищена;

– память Intel Manageability Engine (ME).

Так же как было описано выше для взаимодействия центрального процессора и периферии используются спроецированные в память порты ввода-вывода (Memory Mapped I/O). Далее представлены основные устройства способные спроецировать в память порты ввода вывода:

– спроецированная в память микросхема UEFI BIOS;

– APIC (контроллер прерываний) центрального процессора и портов ввода/вывода;

– MSI (прерывания, инициируемые сообщениями);

– конфигурационное пространство шины PCI-express;

– видеобуфер.

Карта памяти разделена на фиксированные и переменные диапазоны адресов. В разделе 9.4 документации на чипсет представлена таблица диапазонов адресов, декодированных чипсетом [4].

5.2 Анализ функциональных возможностей встроенного программного обеспечения (BIOS)

5.2.1 Описание секций встроенного программного обеспечения

5.2.1.1 Общие сведения о встроенном программном обеспечении

По результатам проведения работ с аппаратным обеспечением было установлено, что программное обеспечение, необходимое для инициализации и

правильного функционирования аппаратного обеспечения, а также дальнейшей загрузки системного ПО находится в микросхеме флэш-памяти.

На данной материнской плате используется микросхема 64 Mb SPI AMI BIOS SM Flash для хранения программных модулей и настроек, обеспечивающих работоспособность и функциональность системы. Версия прошивки, содержащейся в данной микросхеме – 2.20 от 20.02.2015 г., размер – 8 МБ. Данная прошивка имеет стандартный Intel формат, представленный на рисунке 6.

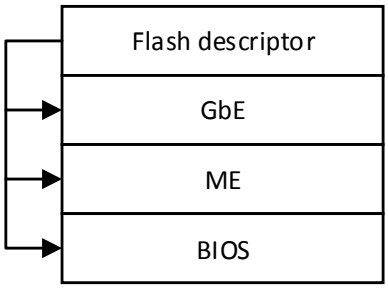


Рисунок 6 – Структура прошивки флэш-памяти

Представленная ранее структура прошивки флэш памяти включает в себя четыре основных раздела (региона):

– Flash descriptor. Данный раздел содержит информацию о микросхеме памяти, регионах, правах доступа и конфигурации аппаратной части материнской платы;

– GbE. Данный раздел содержит начальный MAC-адрес и код прошивки для интегрированного Ethernet адаптера;

– ME. Данный раздел содержит исполняемый код Intel Management Engine;

– BIOS. Данный раздел содержит исполняемый код и конфигурацию UEFI.

Далее более подробно рассмотрен каждый из четырех разделов прошивки флэш-памяти микросхемы 64 Mb SPI AMI BIOS SM Flash.

5.2.1.2 Секция Flash Descriptor

Данный раздел находится в микросхеме флэш-памяти по адресу 0x00000 и подразделяется на одиннадцать секций, представленных на рисунке 7.

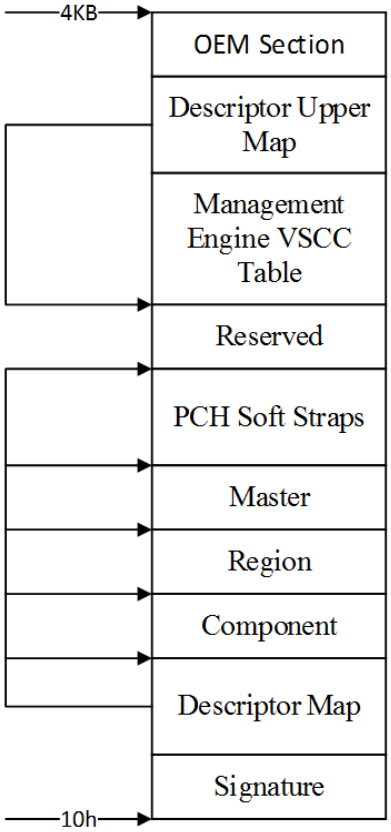


Рисунок 7 – Структура раздела Flash descriptor

Раздел Flash description содержит следующие секции:

– Signature. Данная секция включает в себя первые 16 байт, которые не используются и всегда равны 0xFF, за ними следует сигнатура 0x0FF0A55A;

– Description Map. Данная секция указывает начальные смещения пяти секций и их размеры;

– Component. Данная секция содержит информацию об используемых микросхемах флэш-памяти: их количество (1 или 2), плотность (от 512 Кб до 16 МБ), запрещенные команды и частоты чтения, быстрого чтения и стирания/записи;

– Region. Данная секция содержит начальные смещения пяти регионов и их размеры;

– Master. Данная секция содержит настройки доступа каждого из трех возможны мастеров (BIOS, ME, GbE) к пяти соответствующим возможным регионам;

– PCH Soft Straps. Данная секция содержит параметры конфигурации процессора и северного моста;

– Upper Map. Данная секция содержит смещение и размер таблицы VSCC;

– VSCC. Данная секция содержит идентификаторы JEDEC и данные VSCC всех поддерживаемых Management Engine микросхем флэш памяти;

– OEM. Данная секция содержит информацию, заполненную OEM – производителем по своему назначению.

Таким образом, на основании вышеизложенного установлено, что данная секция не содержит исполняемый код, а включает в себя только конфигурационную информацию и является неизменной.

5.2.1.3 Секция GbE

Данный раздел находится в микросхеме флэш-памяти по смещению 0x10000. Раздел присутствует только на платах со встроенными сетевыми картами Intel и используется для хранения данных конфигурации встроенного сетевого адаптера и его MAC адреса [5].

Таким образом, на основании вышеизложенного установлено, что данная секция не содержит исполняемый код, а включает в себя только конфигурационную информацию и при перепрограммировании стандартными средствами не обновляется.

5.2.1.4 Секция ME

Данный раздел находится в микросхеме флэш-памяти по смещению 0x30000 и содержит прошивку для Intel ME включая ее настройки. Intel ME – подсистема предназначенная для различных задач связанных с мониторингом и обслуживанием компьютера во время сна и загрузки. В начале данного региона

находится таблица, описывающая все разделы данной секции микросхемы памяти. Заголовок таблицы разделов располагается в памяти со смещением 16 байт от начала региона и описывается структурой, представленной в таблице 7.

Таблица 7 – Структура заголовка таблицы разделов региона

Смещение	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00																
10	«$FPT»				Entries				Ver	Type	Len	Chk	Life		Limit	
20	UMA size				Flags											
30+	Первая запись таблицы разделов															

Записи таблицы разделов начинаются со смещения 30h и описываются структурой, представленной в таблице 8.

Таблица 8 – Структура записей таблицы разделов региона

Смещение	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Name				Owner				Offset				Length			
10	StartTokens				MaxTokens				ScratchSectors				Flags			

Таким образом, на основании вышеизложенного установлено, что данная секция содержит конфигурационную информацию и исполняемый код для различных задач, связанных с мониторингом и обслуживанием компьютера во время сна и на этапе загрузки АРМ. Исполняемый код данной секции не выполняется на центральном процессоре, а необходим для инициализации оборудования мониторинга и обслуживания компьютера.

5.2.1.5 Секция UEFI BIOS

Секция UEFI BIOS представляет собой Firmware Volume. Firmware Volume – это логическое представление содержимого микросхемы флэш-памяти. Независимо от используемой внутри Firmware Volume файловой системы, заголовок Firmware Volume имеет следующий стандартизированный вид [6]:

```
typedef struct {
      UINT8 ZeroVector[16];
      UINT8 FileSystemGuid[16];
```

```
    UINT64 FvLength;
    UINT32 Signature;
    UINT32 Attributes;
    UINT16 HeaderLength;
    UINT16 Checksum;
    UINT16 ExtHeaderOffset;
    UINT8 Reserved[1];
    UINT8 Revision;
    EFI_FV_BLOCK_MAP BlockMap[];
} EFI_FIRMWARE_VOLUME_HEADER;

typedef struct {
    UINT32 NumBlocks;
    UINT32 Length;
} EFI_FV_BLOCK_MAP
```

Каждый элемент структуры EFI_FV_BLOCK_MAP рассмотрен более подробно:

– ZeroVector. Данный элемент располагается в начале Firmware Volume и резервирует 16 байт для совместимости с процессорами, а также содержит reset vector по нулевому адресу;

– FileSystemGuid. Данный элемент определяет используемую внутри этого Firmware Volume файловую систему;

– FvLength. Данный элемент содержит размер Firmware Volume с учетом всех заголовков;

– Signature. Данный элемент используется для поиска Firmware Volume и по стандарту всегда равен 0x4856465F (“_FVH”);

– Attributes. Данный элемент содержит атрибуты;

– HeaderLength. Данный элемент содержит размер заголовка без учета расширенного заголовка (описан далее);

– Checksum. Данный элемент содержит 16 битную контрольную сумму заголовка;

– ExtHeaderOffset. Данный элемент содержит смещение начала расширенного заголовка. В нем могут быть указаны GUID описываемого

Firmware Volume, список OEM типов файлов вместе с их GUID'ам, а также электронная подпись. Если дополнительного заголовка у Firmware Volume нет, то значение в этом поле 0x0000;

– Reserved. Данный элемент содержит зарезервированное поле, всегда 0x00;

– Revision. Данный элемент имеет значение всегда равное 0x02, так как стандарт PI описывает структуру только второй ревизии;

– BlockMap. Данный элемент содержит карту блоков, хранящуюся в виде списка структур EFI_FV_BLOCK_MAP, заканчивающегося такой же структурой с нулями в обоих полях. Весь список состоит всего из двух записей, так как все современные микросхемы флэш-памяти однородны (т.е. имеют блоки одинакового размера).

Firmware File System – это плоская файловая система без каталогов и иерархии, все файлы которой находятся в корневом каталоге, где получение списка файлов требует прохода по файловой системе от начала до конца. Все файлы имеют определенный стандартом заголовок. Файлы выровнены по восьми байтовой границе относительно начала файловой системы, для выравнивания по большим границам предусмотрен специальный файл-заполнитель. Также в стандарте описан специальный файл VTF, который присутствует в Firmware Volume образе BIOS и расположен так, что его последний байт является также последним во всей микросхеме. В нем находится код начальной загрузки, необходимый для фазы начальной загрузки (SEC).

Каждый файл в файловой системе (Firmware File System) имеет собственный заголовок, который представлен в виде структуры:

```
typedef struct {
      UINT8 Name[16];
      UINT8 HeaderChecksum;
      UINT8 DataChecksum;
      UINT8 Type;
      UINT8 Attributes;
      UINT8 Size[3];
```

```
    UINT8 State;
} EFI_FFS_FILE_HEADER;
```

Каждый элемент структуры EFI_FFS_FILE_HEADER рассмотрен более подробно:

– Name. Данный элемент содержит GUID файла, выступающий в роли имени, причем в одном Firmware Volume не может быть двух файлов с одинаковым GUID, если это не PAD файлы;

– HeaderChecksum. Данный элемент содержит восьмибитную контрольную сумму заголовка, исключая поле DataChecksum;

– DataChecksum. Данный элемент содержит восьмибитную контрольную сумму содержимого файла без учета заголовка;

– Type. Данный элемент содержит информацию о типе файла. Стандарт определяет 13 стандартных типов файлов (0x01 - 0x0D), 32 пользовательских типа для файлов OEM производителей (0xC0 - 0xDF), 16 пользовательских типов для отладки (0xE0 - 0xEF) и 16 типов, специфичных для текущей версии FFS (0xF0 - 0xFF), из которых используется только 0xF0 - EFI_FV_FILETYPE_FFS_PAD для файла-заполнителя. По стандарту файл пустой, т.е. все его биты, кроме битов заголовка, установлены в значение EFI_FVB_ERASE_POLARITY. Используется он для выравнивания следующего за ним файла по границе, большей стандартных 8 байт. Минимальный размер PAD файла равен размеру заголовка – 24 байта;

– Attributes. Данный элемент содержит атрибут FFS_ATTRIB_FIXED, указывающий на неперемещаемость файла внутри Firmware Volume и набор атрибутов FFS_ATTRIB_DATA_ALIGNMENT, указывающих на выравнивание данных (не заголовка) файла по какой-либо границе;

– Size. Данный элемент содержит информацию о размере файла вместе с заголовком, хранится как 24-битный UINT;

– State. Данный элемент хранит состояние файла. Это поле используется после загрузки Firmware Volume в память и при операциях с файлами внутри Firmware Volume. Состояние всех валидных файлов внутри образа BIOS — 0xF8.

Все файлы файловой системы, кроме RAW, разделены на секции, выравненные по границе 4 байта от начала области данных файла. Заголовок секции представлен в виде следующей структуры:

```
typedef struct {
     UINT8 Size[3];
     UINT8 Type;
} EFI_COMMON_SECTION_HEADER;
```

Каждый элемент структуры EFI_COMMON_SECTION_HEADER рассмотрен более подробно:

– Size. Данный элемент содержит размер секции в том же формате, что и в заголовке файла;

– Type. Данные элемент содержит информацию о типе секции.

Секции делятся на два подкласса — encapsulation и leaf. В первых содержатся секции других типов, а вторые содержат непосредственно данные.

Для encapsulation секций определено 3 типа содержимого:

– 0x01 — EFI_SECTION_COMPRESSION, указывает на то, что секция сжата по какому-либо алгоритму;

– 0x02 — EFI_SECTION_GUID_DEFINED, указывает на то, что содержимое секции должно рассматриваться пользователем файла в соответствии с GUID, записанным в нем;

– 0x03 — EFI_SECTION_DISPOSABLE, указывает на секцию, данные в которой не важны для работы файла и могут быть удалены при пересборке для экономии места.

Заголовок EFI_COMPRESSION_SECTION представлен структурой, которая выглядит следующим образом:

```
typedef struct {
     UINT8 Size[3];
     UINT8 Type;
     UINT32 UncompressedSize;
     UINT8 CompressionType;
} EFI_COMPRESSION_SECTION;
```

Каждый элемент структуры EFI_COMPRESSION_SECTION рассмотрен более подробно:

– UncompressedSize. Данный элемент содержит размер распакованных данных;

– CompressionType. Данный элемент содержит информацию о применяемом алгоритме сжатия (Tiano и LZMA).

Для leaf-секций определено 12 типов содержимого, представленных в таблице 9.

Таблица 9 – Типы содержимого leaf – секций

Название	Тип	Описание
PE32	0x10	64-битный исполняемый код в формате PE32+ со всеми его заголовками. Основной формат исполняемого кода в UEFI.
PIC	0x11	64-битный исполняемый код, не зависящий от позиции. Используется только некоторыми модулями PEI, формат совпадает с PE32+, только информация о relocation обрезана.
TE	0x12	64-битный исполняемый код, используемый модулями и ядром PEI. От PE32+ отличается заголовком, который уменьшен для экономии места в кэше процессора.
DXE_DEPEX	0x13	Секция, описывающая зависимости драйвера DXE, в котором она находится.
VERSION	0x14	Содержит версию файла и опционально Unicode строку с полной версией.
USER_INTERFACE	0x15	Содержит Unicode строку с именем файла.
COMPATIBILITY16	0x16	16-битный исполняемый код для совместимости со старыми системами.
FIRMWARE_VOLUME_IMAGE	0x17	Секция с образом Firmware Volume. Внутри этого Firmware Volume может быть еще файл с такой

Продолжение таблицы 9

Название	Тип	Описание
		секцией, и так далее, пока место в микросхеме не закончится.
FREEFORM_SUBTYPE_GUID	0x18	Секция содержит данные, интерпретация которых зависит от записанного ее начало GUID.
RAW	0x19	Секция с «сырыми» данными. Что с ними делать — определяет тот, кто этот файл открыл.
PEI_DEPEX	0x1B	Секция, описывающая зависимости модуля PEI, в котором она находится.
SMM_DEPEX	0x1C	Секция, описывающая зависимости драйвера SMM, в котором она находится, формат совпадает с DXE_DEPEX.

Таким образом, на основании вышеизложенного установлено, что данная секция содержит конфигурационную информацию и исполняемый код для корректной инициализации оборудования при включении компьютера и дальнейшей передачи управления загрузчику операционной системы. Исполняемый код данной секции исполняется на центральном процессоре.

Далее в следующих пунктах дипломного проекта более подробно будет рассмотрена данная секция с целью выявления функциональных возможностей UEFI BIOS.

5.2.2 Функциональные возможности UEFI BIOS

5.2.2.1 Графический интерфейс

5.2.2.1.1 Главное меню

Главное меню включает в себя следующие настройки:

- System Date (настройка системной даты);
- System Time (настройка системного времени).

На рисунке 8 приведен скриншот главного меню (Main) графического интерфейса Aptio Setup Utility.

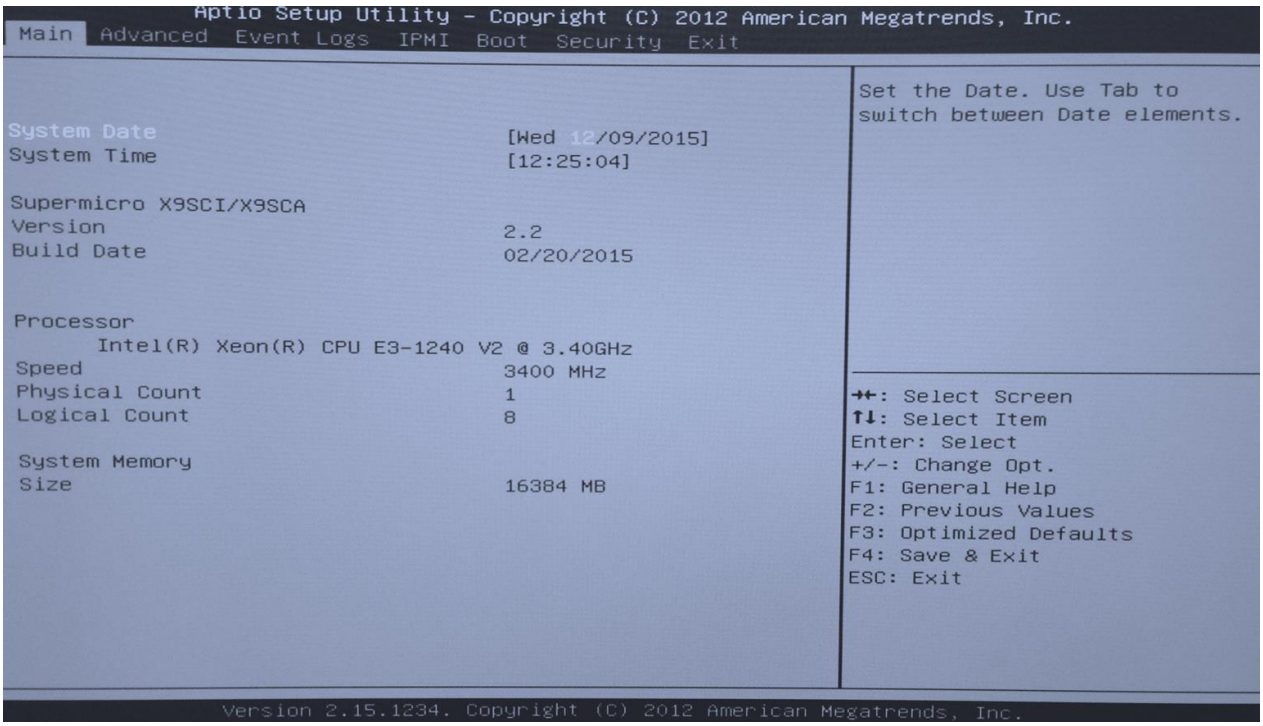


Рисунок 8 – Главное меню

5.2.2.1.2 Меню расширенных настроек

Расширенные настройки представлены как многоуровневое меню и включают в себя следующие подменю:

– Boot Feature (конфигурация особенностей загрузки);

– CPU Configuration (конфигурация процессора);

– Chipset Configuration (конфигурация чипсета);

– IDE/SATA Configuration (конфигурация IDE/SATA);

– Super IO Configuration (конфигурация последовательных портов);

– Serial Port Console Redirection (перенастройка последовательных портов);

– ACPI Configuration (конфигурация расширенного управления питанием);

– ME Subsystem (информация о версии Intel ME).

На рисунке 9 приведен скриншот меню расширенных настроек (Advanced) графического интерфейса Aptio Setup Utility.

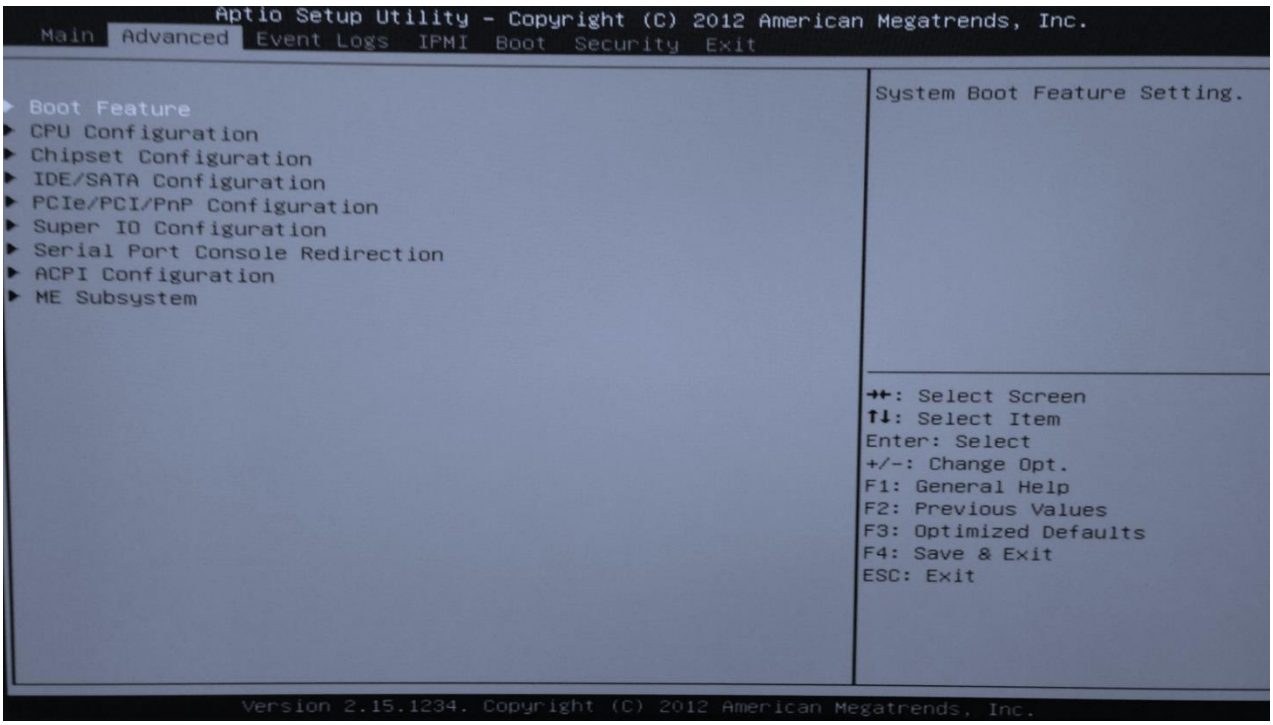


Рисунок 9 – Меню расширенных настроек

5.2.2.1.3 Меню регистрации событий

Регистрация событий представлена как многоуровневое меню и включает в себя следующее подменю и настройки:

– Change Smbios Event Log Settings (подменю настройки регистрации событий Smbios);

– View Smbios Event Log (просмотр зарегистрированных событий).

5.2.2.1.4 Меню мониторинга событий

Мониторинг состояний представлен как многоуровневое меню и включает в себя следующие подменю:

– IPMI Firmware Revision (информация о версии ПО системы мониторинга);

– IPMI STATUS (информация о статусе системы мониторинга);

– System Event Log (подменю событий системы мониторинга);

– BMC network configuration (конфигурация контроллера удаленного мониторинга).

На рисунке 10 приведен скриншот меню мониторинга состояний (IPMI) графического интерфейса Aptio Setup Utility.

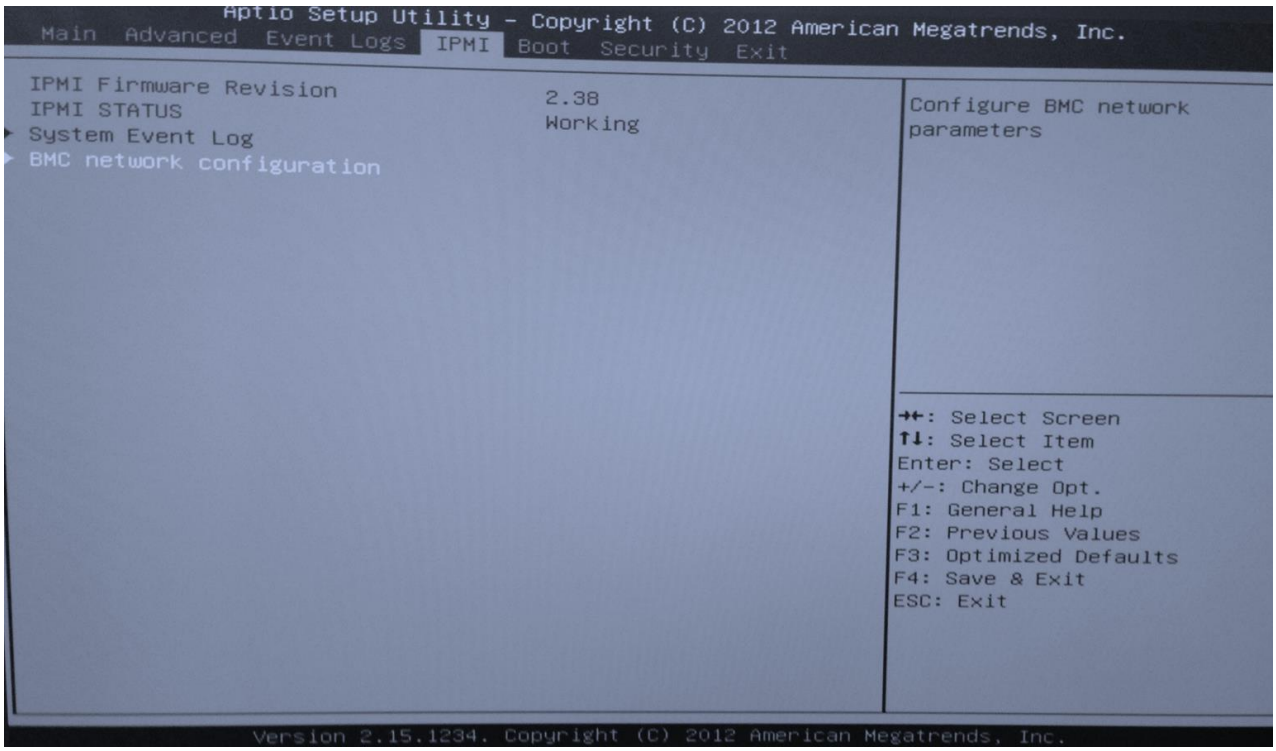


Рисунок 10 – Меню мониторинга состояний

5.2.2.1.5 Меню загрузки

Меню загрузки включает в себя следующие настройки:

– Setup Prompt Timeout (установка времени задержки);

– Retry Boot Devices (установка повторной загрузки с устройств);

– Boot option filter (настройка вариантов загрузки);

– Boot Option Priorities (установка приоритета загрузки с устройств);

– CD/DVD ROM Drive BBS Priorities (индивидуальная настройка приоритета загрузки с CD/DVD устройств);

– Hard Drive BBS Priorities (индивидуальная настройка приоритета загрузки с жестких дисков);

– Network Device BBS Priorities (индивидуальная настройка приоритета загрузки по сети).

На рисунке 11 приведен скриншот меню загрузки (Boot) графического интерфейса Aptio Setup Utility.

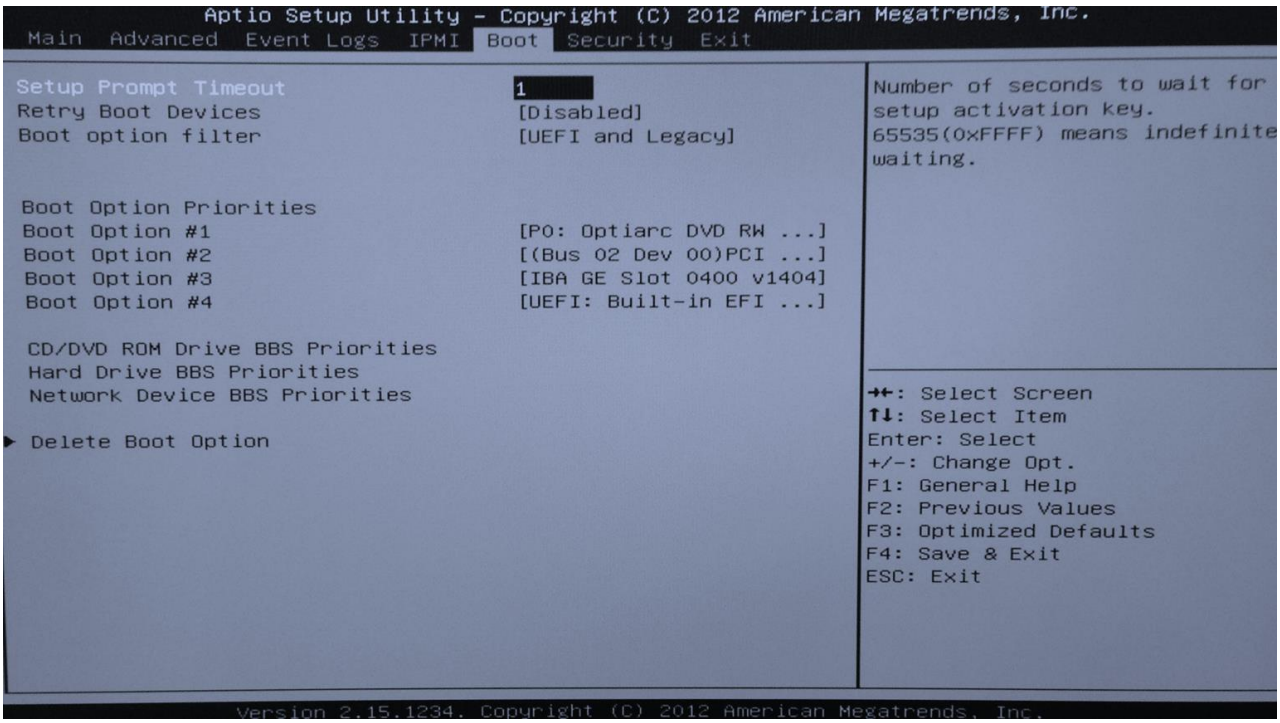


Рисунок 11 – Меню загрузки

5.2.2.1.6 Меню безопасности

Меню безопасности включает в себя следующие настройки:

– Password Check (настройка проверки пароля);

– Administration Password (настройка пароля администратора).

На рисунке 12 приведен скриншот меню безопасности (Security).

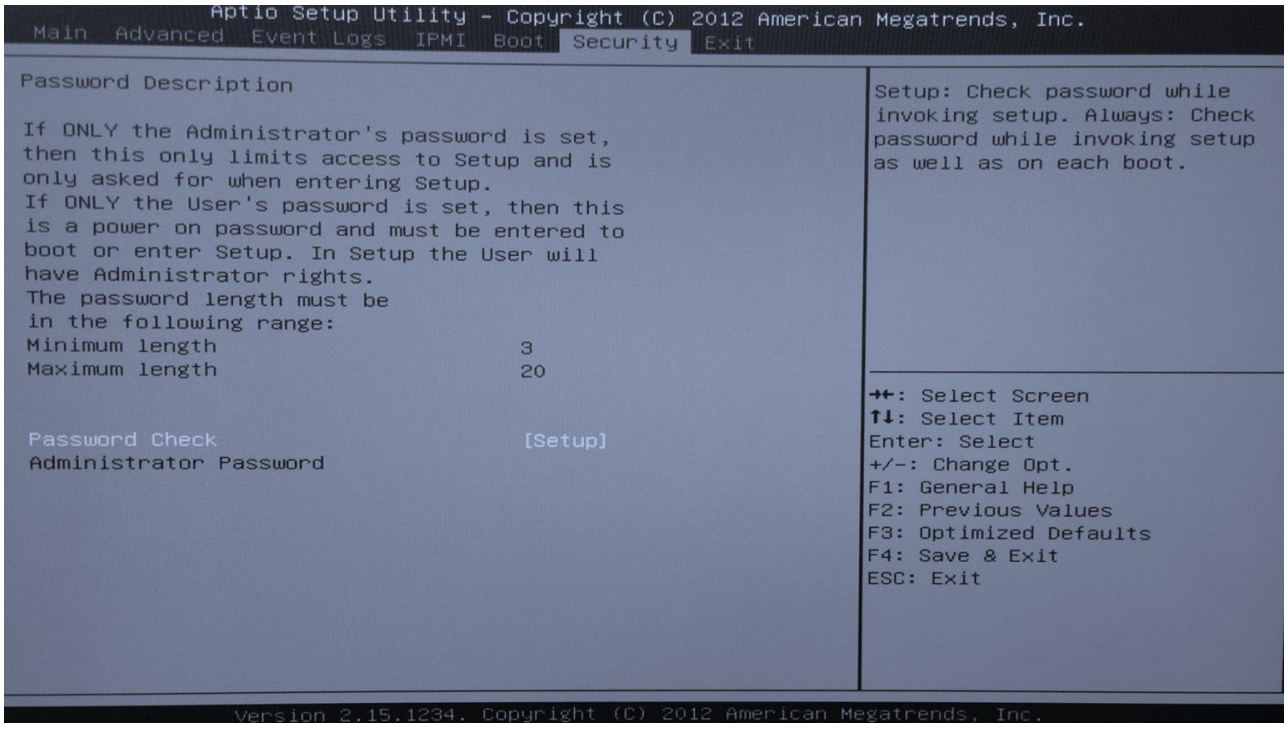


Рисунок 12 – Меню безопасности

5.2.2.1.7 Меню выхода

Меню выхода включает в себя следующие настройки:

– Save Changes and Reset (сохранить изменения и перезагрузится);
– Discard Changes and Exit (отменить изменения и выйти);
– Discard Changes (отменить изменения);
– Restore Defaults (восстановить исходные настройки);
– Save as User Defaults (сохранить пользовательские настройки);
– Restore User Defaults (восстановить пользовательские настройки).

На рисунке 13 приведен скриншот меню выхода (Exit) графического интерфейса Aptio Setup Utility.

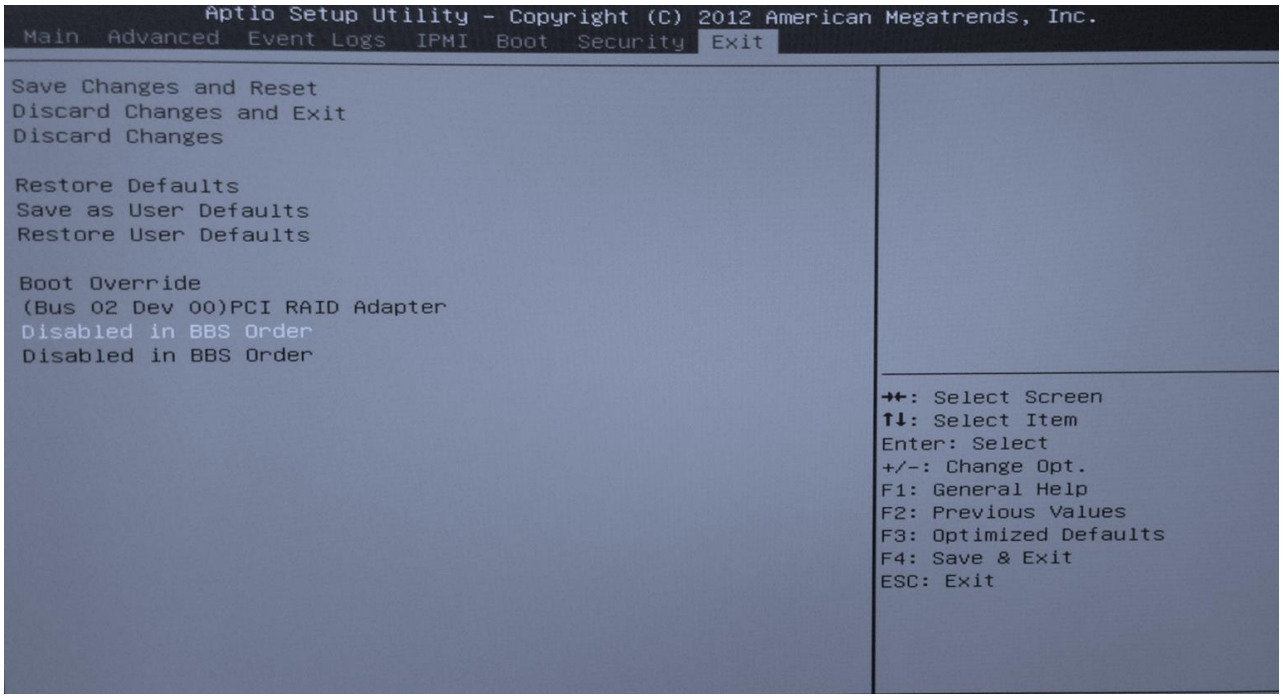


Рисунок 13 – Меню выхода из настроек

5.2.2.2 Процесс загрузки UEFI BIOS

5.2.2.2.1 Общие сведения процесса загрузки

После нажатия кнопки включения источник питания выполняет самотестирование. Если все напряжения соответствуют номинальным, источник питания выдает на материнскую плату сигнал PowerGood, а специальный триггер, вырабатывающий сигнал Reset, получив его, снимает сигнал сброса с соответствующего входа процессора. Сигнал Reset устанавливает сегментные

регистры и указатели команд в следующее состояние: CS = 0xFFFF; IP = 0x0; DS = SS = ES = 0x0, сбрасывает все биты управляющих регистров, а также обнуляет регистры арифметико-логического устройства.

С момента снятия сигнала Reset процессор начинает работу в реальном режиме и в течение нескольких циклов синхронизации приступает к выполнению инструкций по адресу FFFF:0000, считываемых из флэш-памяти. Размер области флэш-памяти от этого адреса до конца равен 16 байтам, и в ней, начиная с указанного адреса, хранится последовательность команд, осуществляющая переход на исполняемый код UEFI (фазу начальной инициализации).

Далее на рисунке 14 приведен процесс загрузки UEFI от момента передачи управления в фазу начальной инициализации (Security – SEC) до загрузки ОС [7].

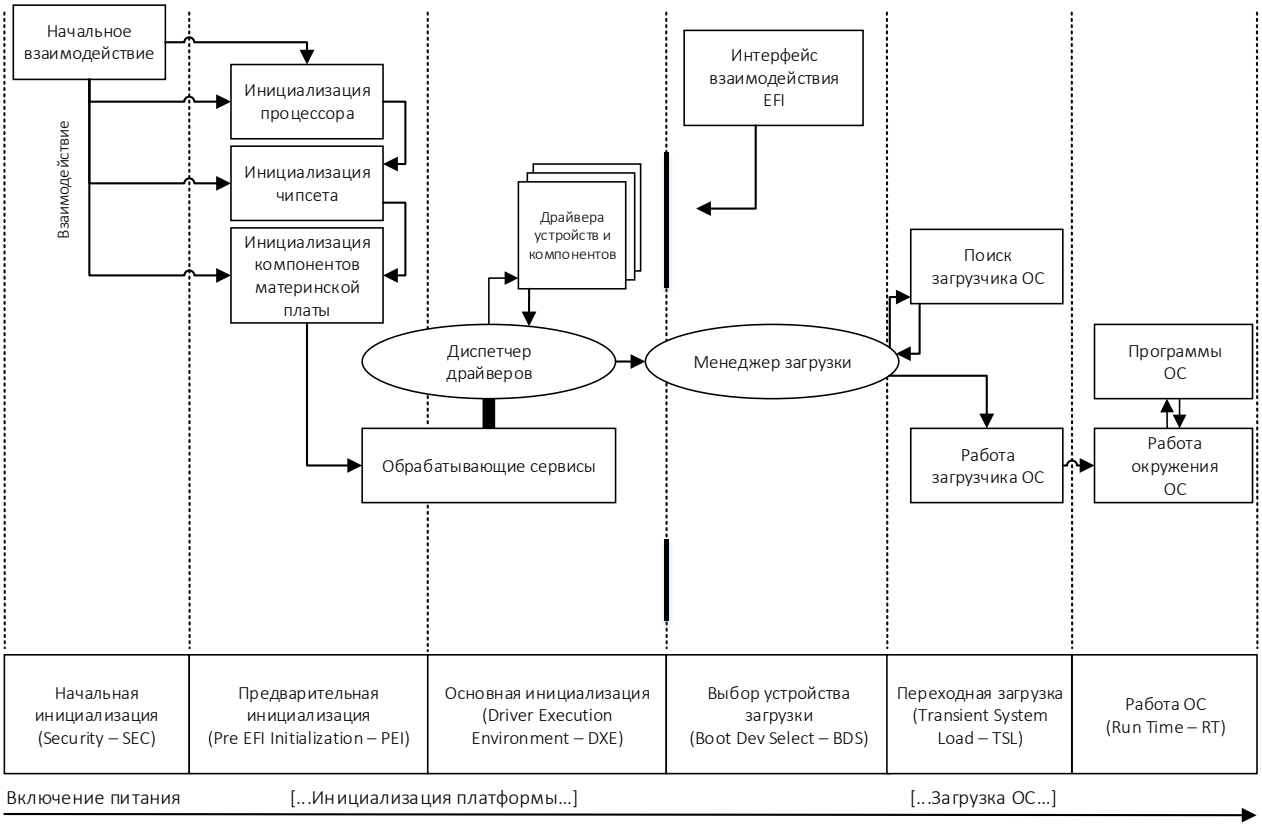


Рисунок 14 – Процесс загрузки UEFI

Как видно на рисунке, процесс загрузки UEFI состоит из: начальной инициализации, предварительной инициализации, основной инициализации, выбора устройств загрузки, переходной загрузки и работы ОС.

5.2.2.2.2 Начальная инициализация

Фаза начальной инициализации (Security - SEC) является первой из трех фаз процесса инициализации аппаратного обеспечения. Данная фаза реализует следующие функциональные задачи:

– обработка событий старта системы (включения питания, перезагрузка, выход из режима сна);

– подготовка временной памяти для дальнейшего ее использования в фазе предварительной инициализации;

– подготовка необходимых структур данных для дальнейшей передачи их в фазу предварительной инициализации;

– передача управления на ПО предварительной инициализации.

Схема алгоритма работы данной фазы представлена на рисунке 15.

Рисунок 15 – Схема алгоритма работы фазы начальной инициализации

Далее приведено описание алгоритма работы фазы начальной инициализации:

– сброс кэша и переход на главную процедуру инициализации. На данном этапе процессор работает в реальном режиме, а регистры сегмента имеют следующий вид CS:IP = F000:FFF0 и CS.BASE = FFFF_0000h;

– переключение в защищенный режим процессора. На данном этапе в регистры процессора загружается необходимая для работы в защищенном режиме управляющая информация и осуществляется его переключение;

– инициализация диапазонных регистров типа памяти (MTRR) для пакета драйверов, реализующих поддержку аппаратной платформы (Board Support Package – BSP);

– обновление микрокода процессора. На данном этапе происходит обновление/добавление микроинструкций процессора;

– пометка свободного кэша как не сбрасываемого. На данном этапе кэш становится доступен для использования как временная память до инициализации основной;

– начальная инициализация процессора. На данном этапе всем ядрам процессора посылается серия стандартных прерываний инициализации и запуска на базовой частоте;

– передача в фазу предварительной инициализации (PEI) сформированных на предыдущих этапах данных, таких как местоположение и размер временной памяти, местоположение стека, местоположение и объем встроенного программного обеспечения UEFI (ВПО), а также передача управления на ПО предварительной инициализации.

Таким образом, результатом выполнения фазы является обновленный микрокод процессора, переход в защищенный режим и передача управления на ПО предварительной инициализации. Также на данном этапе инициализации из ВПО считываются хранящиеся в нем патчи для микрокода процессора, информация о стартовом адресе и размере файла ВПО, при этом ПО начальной инициализации хранится и выполняется из микросхемы ПЗУ и по завершении выполнения управление на данную фазу больше не передается.

5.2.2.2.3 Предварительная инициализация

Фаза предварительной инициализации (Pre EFI Initialization – PEI) является второй из трех фаз инициализации аппаратного обеспечения. Данная фаза реализует следующие функциональные задачи:

– подготовка и инициализация основной оперативной памяти для запуска фазы основной инициализации;

– подготовка и передача данных об обнаруженных устройствах, таких как процессор, системная плата, память и т.д., в фазу основной инициализации для их дальнейшей правильной инициализации.

Схема алгоритма работы фазы предварительной инициализации представлена на рисунке 16.

Рисунок 16 – Схема алгоритма работы фазы предварительной инициализации

Далее приведено описание алгоритма работы фазы предварительной инициализации:

– перенос данных из микросхемы ПЗУ в временную память. На данном этапе данные, необходимые для работы фазы, переносятся в подготовленную временную память;

– инициализация процессора. На данном этапе загружаются и запускаются модули установки всех кэшей процессора и его рабочей частоты;

– начальная инициализация аппаратных средств. На данном этапе инициализируются встроенные интерфейсы платформы (SMBUS, Reset и т.д.), системный контроллер чипсета и хаб ввода-вывода (южный мост);

– инициализация основной оперативной памяти. На данном этапе основная память инициализируется и в нее копируются данные из временной памяти для дальнейшего использования;

– передача управления в фазу основной инициализации. На данном этапе данные передаются через основные независимые контейнеры (HOB), предназначенные для хранения и пересылок данных, в фазу основной инициализации.

Данные передаваемые через основные независимые контейнеры в фазу основной инициализации представлены на рисунке 17.

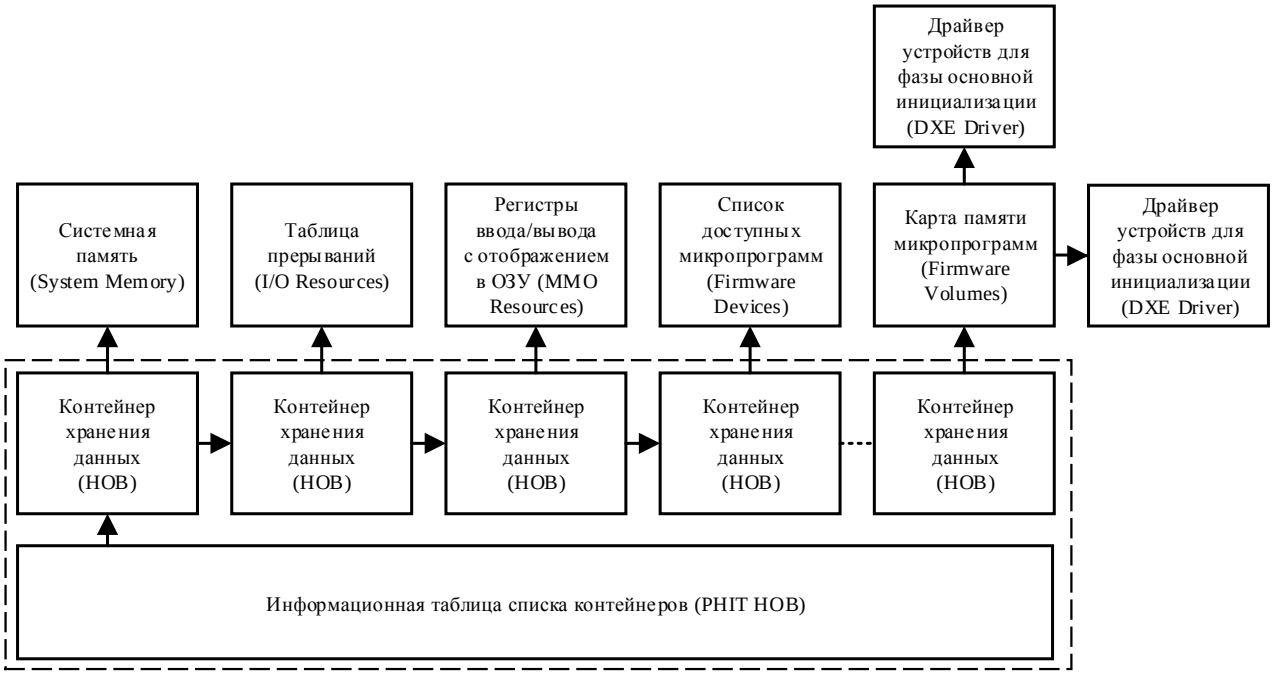


Рисунок 17 – Данные передаваемые в независимых контейнерах

Таким образом, но основании вышеизложенного установлено, что результатом выполнения предварительной инициализации является передача данных через основные независимые контейнеры в фазу основной инициализации. Данные передаваемые в контейнерах, содержат информацию о

системной памяти, наборе команд процессора и информацию об обнаруженных микропрограммах устройств.

Информация о микропрограммах устройств включает информацию об адресном пространстве памяти устройств, ее объемах, а также может содержать информацию о ресурсах ввода/вывода и их отображение в памяти. Также необходимо отметить, что на данном этапе из ВПО считывается код ядра фазы предварительной инициализации и модули для выполнения начальной инициализации оборудования, при этом ПО предварительной инициализации хранится и выполняется из оперативной памяти и при передачи управления в следующую фазу все еще находится в памяти.

5.2.2.2.4 Основная инициализация

Фаза основной инициализации (Driver Execution Environment – DXE) является последней из трех фаз процесса инициализации аппаратного обеспечения. Данная фаза реализует следующие функциональные задачи:

– обработка полученного списка контейнеров для дальнейшей инициализации оборудования системы драйверами;

– основная инициализация драйверами всего оборудования системы для дальнейшего его функционирования;

– передача управления на ПО выбора устройства загрузки (BDS).

Процесс передачи данных через основные независимые контейнеры хранения данных (HOB) в фазу основной инициализации представлен далее на рисунке 18.

После подготовки всех контейнеров хранения данных к передаче ядру фазы основной инициализации, передается их список. Ядро обрабатывает список и далее связывается через архитектурный протокол с компонентами материнской платы и необходимыми драйверами устройств для выполнения основной инициализации.

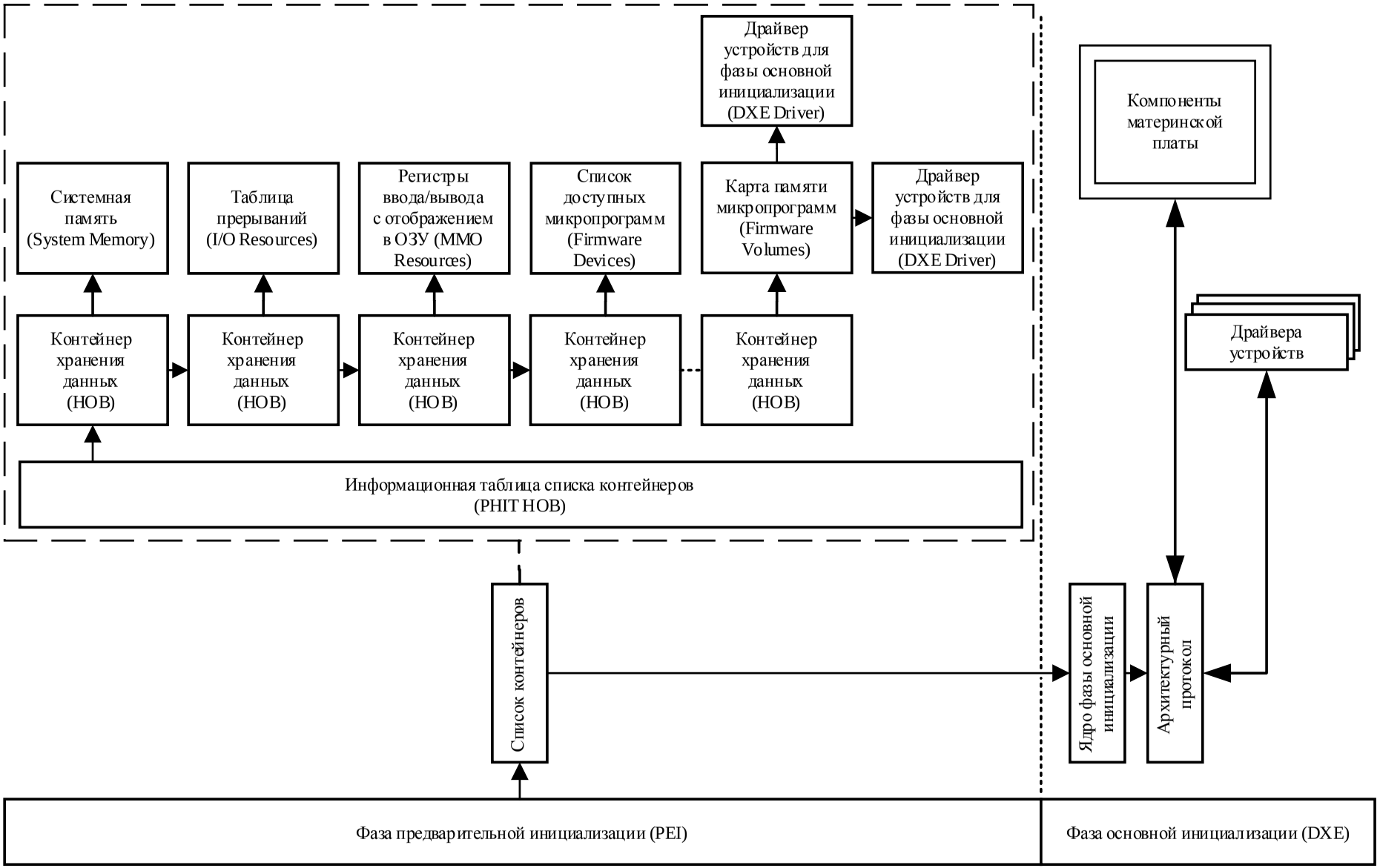


Рисунок 18 – Схема передачи данных в фазу основной инициализации

Схема алгоритма работы фазы основной инициализации представлена на рисунке 19.

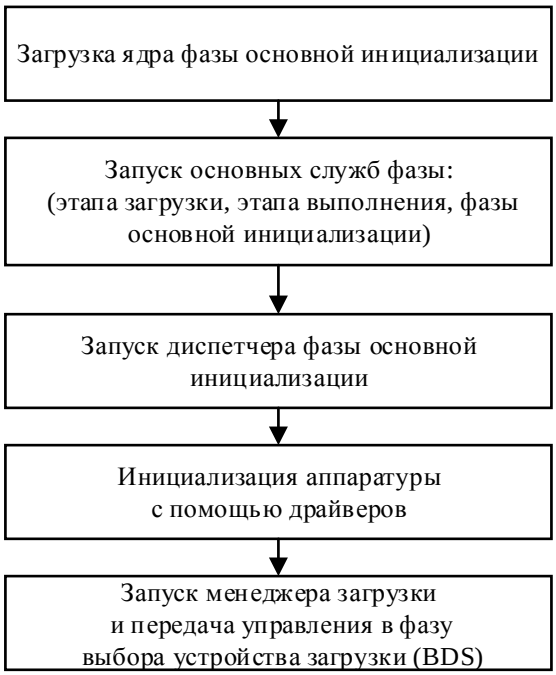


Рисунок 19 – Схема алгоритма работы фазы основной инициализации

Далее приведено описание алгоритма работы фазы основной инициализации:

– загрузка ядра фазы основной инициализации. На данном этапе загружается ядро фазы и обрабатывает полученный список контейнеров;

– запуск основных служб фазы. На данном этапе производится запуск сервисов – этапа загрузки (Boot Services), этапа выполнения (Runtime Services) и этапа основной инициализации (DXE Services);

– запуск диспетчера фазы основной инициализации. На данном этапе диспетчер на основании переданного списка контейнеров определяет доступные карты памяти микропрограмм, находит в них драйвера и запускает их соблюдая зависимости;

– инициализация аппаратуры с помощью драйверов. На данном этапе драйверы проводят окончательную инициализацию аппаратуры и предоставляют аппаратную абстракцию для системных служб и загрузочных устройств (в этот момент производится активация остальных компонентов, причем одновременно нескольких);

– запуск менеджера загрузки. На данном этапе после завершения работы всех драйверов управление передается на ПО выбора устройства загрузки (BDS).

Таким образом, на основании вышеизложенного установлено, что результатом выполнения основной инициализации является законченный процесс инициализации аппаратного обеспечения, и передача управления на ПО фазы выбора устройства загрузки. Необходимо также отметить, что на данной фазе из ВПО считывается код ядра данной фазы и модули для выполнения основной инициализации оборудования, при этом ПО основной инициализации выполняется в оперативной памяти и выгружается по завершению выполнения фазы.

5.2.2.2.5 Переходная загрузка

Фаза переходной загрузки реализует функциональную задачу переходной загрузки системы, где загрузчиком ОС Astra Linux для данной системы является GRUB2. Далее на рисунке 20 приведена схема процесса переходной загрузки от момента передачи управления в фазу до загрузки ОС.

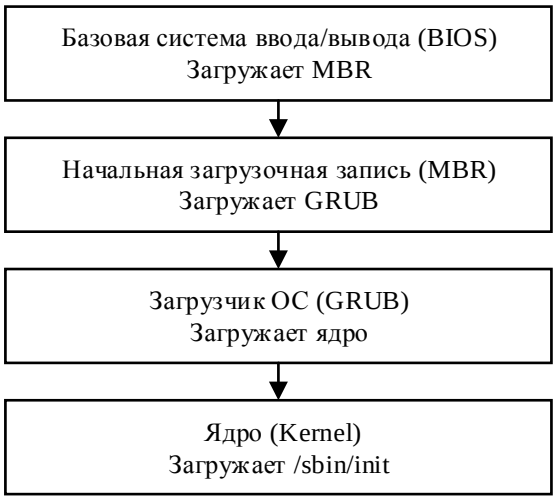


Рисунок 20 – Схема алгоритма работы фазы переходной загрузки

Далее приведено описание алгоритма работы фазы переходной загрузки:

– базовая система ввода/вывода (BIOS). На данном этапе BIOS, после того как нашел загрузочный диск, загружает и выполняет загрузочную запись (MBR);

– начальная загрузочная запись. MBR – главная загрузочная запись, хранящаяся на жестком диске в 1-м секторе размером в 512 байт, и содержащая информацию о GRUB’е. На данном этапе после передачи управления на MBR, загружается и выполняется загрузчик GRUB;

– загрузчик ОС. На данном этапе загрузчик ОС (GRUB) в зависимости от файла конфигурации загружает ядро ОС, расположенное в файловой системе, и передает на него управление;

– ядро. На данном этапе ядро (Kernel) монтирует файловую систему в соответствии с настройкой в файле grub.conf (файл конфигурации) и выполняет программу /sbin/init, которая запускает ОС и управляет системой используя несколько уровней исполнений (runlevels).

Таким образом, но основании вышеизложенного установлено, что результатом выполнения переходной загрузки является загрузка ОС. На этом все ПО фазы выгружается, и работа UEFI BIOS прекращается, а за дальнейшее управление отвечает ОС Astra Linux.

5.2.2.3 Взаимодействие UEFI BIOS с ОС Astra Linux

В данном подпункте рассмотрены основные способы взаимодействия UEFI BIOS и ОС Astra Linux, с помощью интегрированных сервисов. Для этого в UEFI BIOS имеется системная таблица, которая описывает структуру двух возможных видов сервисов, позволяющих осуществлять информационное взаимодействие: загрузочные (boot) и рабочие (runtime) сервисы. Первые работают только до загрузки ОС и обеспечивают взаимодействие с графическими и текстовыми терминалами, шинами и блочными устройствами, а вторые обеспечивают взаимодействие непосредственно с ОС (пробуждение из сна, прошивка секции UEFI BIOS и системная перезагрузка).

Как было отмечено ранее ОС Astra Linux поддерживает загрузку только в режиме совместимости UEFI Legacy Support. В данном режиме рабочие (runtime) сервисы не доступны и не предоставляют интерфейс взаимодействия для его

дальнейшего использования в ОС. В дополнение к этому, ядро ОС Astra Linux не поддерживает UEFI режим.

Таким образом, взаимодействие UEFI BIOS с ОС Astra Linux сводится к вызову функции ExitBootServices() и передаче управления на загрузчик ОС.

5.3 Выявление перечня опасных функциональных возможностей встроенного программного обеспечения (BIOS)

5.3.1 Определение перечня функциональных возможностей встроенного программного обеспечения (BIOS)

В данном пункте определен перечень функциональных возможностей встроенного программного обеспечения (BIOS), выявленных на предыдущих этапах в ходе проведения работ с документацией и со встроенным программным обеспечением UEFI BIOS:

- инициализация аппаратного обеспечения АРМ;
- тестирование аппаратного обеспечения АРМ;
- перепрошивка UEFI BIOS;
- установка параметров UEFI BIOS;
- ведение журнала событий UEFI BIOS;
- удаленный мониторинг функционирования АРМ;
- аппаратная виртуализация;
- загрузка операционной системы с внутренних носителей;
- загрузка операционной системы с внешних носителей;
- загрузка операционной системы по сети «Ethernet».

Ввиду того, что в рамках данного дипломного проекта работы с ВПО UEFI BIOS по восстановлению исходного кода путем дизассемблирования бинарного кода не проводились, вышеприведенный перечень функциональных возможностей UEFI BIOS не является полным. Функциональные возможности UEFI BIOS, которые не были выявлены в ходе дипломного проектирования отнесены к «Недокументированным функциональным возможностям».

5.3.2 Определение перечня опасных функциональных возможностей встроенного программного обеспечения (BIOS)

При определении перечня опасных функциональных возможностей встроенного программного обеспечения (BIOS) любое деструктивное воздействие на аппаратно-программную среду выполнения системы рассматривается как несанкционированный доступ к некоторому ресурсу, содержащему критические данные. Под несанкционированным доступом со стороны UEFI BIOS понимается самостоятельно инициированный доступ к ресурсам системы, непредусмотренный штатным алгоритмом работы системы. Критическими данными считаются данные или исполняемый код в системе, позволяющий непосредственно или косвенно влиять на процесс обработки информации.

Далее с целью определения перечня опасных функциональных возможностей встроенного программного обеспечения (BIOS), для каждой функциональной возможности встроенного программного обеспечения (BIOS) были определены опасные последствия, к которым возможно могут приводить данные функциональные возможности UEFI BIOS.

Для инициализации аппаратного обеспечения используется исполняемый код секции UEFI BIOS. В процессе процедуры инициализации в результате искажения команд и данных исполняемого кода UEFI BIOS возможно появление отклонений алгоритма функционирования от штатного, приводящих к некорректной инициализации аппаратного обеспечения и как следствие ухудшения его характеристики надежности, что является опасным последствием.

Для тестирования аппаратного обеспечения используется исполняемый код секции UEFI BIOS. В процессе процедуры тестирования в результате искажения команд и данных исполняемого кода UEFI BIOS возможно появление отклонений алгоритма функционирования от штатного, приводящих к невозможности определить неисправность аппаратного обеспечения и как

следствие ухудшаются его характеристики надежности, что является опасным последствием.

Для перепрошивки UEFI BIOS АРМ используются специализированные утилиты с сайта производителя материнской платы. Штатными средствами UEFI BIOS и средствами ОС Astra Linux перепрошивку выполнить невозможно. Перепрошивка UEFI BIOS версией 2.20 осуществляется доверенным лицом (администратором) перед началом эксплуатации АРМ. В процессе процедуры перепрошивки в результате искажений команд и данных специализированной утилиты возможно некорректное перепрограммирование микросхемы флэш-памяти, приводящее к искажению команд и данных исполняемого кода UEFI BIOS и как следствие к его нештатному алгоритму функционирования, что является опасным последствием.

Установка параметров UEFI BIOS осуществляется доверенным лицом (администратором) перед началом эксплуатации АРМ в соответствии с руководством по настройке UEFI BIOS.

Для ведения журнала используется исполняемый код секции UEFI BIOS. В процессе процедуры журналирования в результате искажения команд и данных исполняемого кода UEFI BIOS возможно появление отклонений алгоритма функционирования от штатного, приводящих к невозможности зарегистрировать событие, произошедшее при функционировании UEFI BIOS и как следствие затруднит дальнейший мониторинг событий во время сервисного обслуживания, что не является опасным последствием.

Использование удаленного мониторинга АРМ может приводить к несанкционированному доступу к аппаратно-программной среде функционирования, позволяющему получить доступ к критичным данным, что является опасным последствием.

Использование аппаратной виртуализации АРМ может приводить к несанкционированному доступу к аппаратно-программной среде

функционирования, позволяющему получить доступ к критичным данным, что является опасным последствием.

Для загрузки операционной системы с внутреннего носителя используется исполняемый код секции UEFI BIOS. В процессе процедуры загрузки в результате искажения команд и данных исполняемого кода UEFI BIOS возможно появление отклонений алгоритма функционирования от штатного, приводящих к ошибочному выбору устройства для дальнейшей загрузки ОС Astra Linux, и как следствие невозможность загрузить ОС и перевести в рабочий режим АРМ, что не является опасным последствием.

Использование процедуры загрузки операционной системы с внешних носителей АРМ может приводить к несанкционированному доступу к аппаратно-программной среде функционирования, позволяющему получить доступ к критичным данным, что является опасным последствием.

Использование процедуры загрузки операционной системы АРМ по сети «Ethernet» может приводить к несанкционированному доступу к аппаратно-программной среде функционирования, позволяющему получить доступ к критичным данным, что является опасным последствием.

Функциональные возможности UEFI BIOS, которые не были выявлены в ходе дипломного проектирования и отнесены к «недокументированным функциональным возможностям» являются потенциально опасными и могут содержать программные конструкции, оказывающие деструктивное влияние на аппаратно-программную среду функционирования АРМ, и могут приводить к следующим опасным последствиям:

– условному (безусловному) перенаправлению (копированию) части или всего потока критичных данных;

– беспрепятственной модификации прошивки UEFI BIOS;

– переназначение или изменение штатных режимов работы встроенных ресурсов системы, порождающих сбои и неисправности аппаратно-программного обеспечения;

– изменение собственного исполняемого кода и кода других программ в зависимости от изменения установок таймера и настроек UEFI BIOS.

Таким образом по результатам проведенного анализа был определен и представлен перечень опасных функциональных возможностей встроенного программного обеспечения (BIOS):

– инициализация аппаратного обеспечения АРМ;
– тестирование аппаратного обеспечения АРМ;
– перепрошивка UEFI BIOS;
– установка параметров UEFI BIOS;
– удаленный мониторинг функционирования АРМ;
– аппаратная виртуализация;
– загрузка операционной системы с внешних носителей;
– загрузка операционной системы по сети «Ethernet»;
– недокументированные функциональные возможности ВПО.

5.4 Разработка комплекса организационно-технических мер защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS)

5.4.1 Определение перечня организационно-технических мер защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS)

В данном пункте приведен комплекс организационно-технических мер защиты, позволяющий обеспечить невозможность деструктивного влияния опасных функциональных возможностей на аппаратно-программную среду функционирования АРМ с целью создания факторов и условий для представленных угроз информационной безопасности.

Перечень организационно-технических мер защиты информации представлен далее:

а) ежесуточный контроль целостности ВПО UEFI BIOS по алгоритму хеширования «MD5»;

б) сравнение значения доверенным лицом (администратором), полученного в результате расчета целостности на прошитый во флэш-память исполняемый код UEFI BIOS, с эталонным значением, указанным на носителе с образом UEFI BIOS;

в) опечатывание (опломбирование) АРМ и его разъемов для защиты от несанкционированного доступа к аппаратному обеспечению и подключения внешних устройств;

д) установка параметров UEFI BIOS для защиты от несанкционированной работы пользователей с внешними носителями и загрузки по сети «Ethernet»;

е) установка пароля администратора для защиты от несанкционированного изменения параметров UEFI BIOS;

ж) обеспечение отсутствия в АРМ специализированных утилит, позволяющих осуществить несанкционированную перепрошивку UEFI BIOS;

и) опечатывание (опломбирование) разъема IPMI LAN для защиты от несанкционированного подключения для осуществления удаленного мониторинга;

к) установка на материнской плате перемычки JPB в положение «отключено» для аппаратной блокировки функций удаленного мониторинга.

л) установка параметров UEFI BIOS для блокировки функций аппаратной виртуализации;

м) установка параметров UEFI BIOS для защиты от несанкционированной работы пользователей с внешними носителями;

н) установка параметров UEFI BIOS для блокировки функций загрузки операционной системы по сети «Ethernet» и разрешения загрузки только с внутренних носителей;

п) подключение АРМ только к защищенным каналам связи в пределах контролируемой зоны;

р) доработка протокола взаимодействия АРМ с МЭ, после закупки материнских плат с исследуемой версией UEFI BIOS 2.20, в части введения поля контрольной суммы (CRC-16) на область данных, передаваемых в пакете с

ограничительной пометкой. В случае случайной или преднамеренной записи в данный пакет конфиденциальной информации, данный пакет будет гарантированно отбракован МЭ.

Далее была произведено сопоставление опасных функциональных возможностей встроенного программного обеспечения (BIOS) и разработанных организационно-технических мер защиты, результаты которого представлены в таблице 10.

Таблица 10 – Результат сопоставления

Опасная функциональная возможность	Организационно-технические меры защиты
Инициализация аппаратного обеспечения АРМ	а)
Тестирование аппаратного обеспечения АРМ	а)
Перепрошивка UEFI BIOS	б), в), д), е), ж)
Установка параметров UEFI BIOS	в), е)
Удаленный мониторинг функционирования АРМ	и), к)
Аппаратная виртуализация	г), л)
Загрузка операционной системы с внешних носителей	в), е), м)
Загрузка операционной системы по сети «Ethernet»	е), н), п)
Недокументированные функциональные возможности ВПО	а), в), п), р)

5.4.2 Рекомендации по установке и настройке BIOS с целью защиты от негативных действий функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах защищенного исполнения

5.4.2.1 Прошивка UEFI BIOS

Для обновления прошивки микросхемы 64 Mb SPI AMI BIOS SM Flash материнской платы SuperMicro X9SCA-F АРМ до исследуемой версии UEFI BIOS 2.20. используется фирменная утилита с сайта производителя материнской

платы. Далее представлен перечень файлов, необходимых для прошивки UEFI BIOS:

– AFUDOSU.EXE – фирменная утилита с сайта производителя материнской платы, предназначенная для прошивки UEFI BIOS;

– FLASH.BAT – пакетный файл, содержащий последовательность команд с определенными ключами для правильной прошивки UEFI BIOS;

– X9SCA5.220 – файл образа прошивки UEFI BIOS версии 2.20;

– BIOS.220.MD5 – файл с эталонной контрольной суммой, полученной с помощью утилиты md5sum на секцию UEFI BIOS.

Для обновления прошивки UEFI BIOS АРМ необходимо выполнить следующие действия:

– создать загрузочный USB-носитель с ОС DOS;

– скопировать вышеперечисленные файлы на созданный загрузочный USB-носитель;

– на начальном этапе загрузки АРМ нажать клавишу «Delete» для активации средства Aptio Setup Utility, позволяющего изменять настройки UEFI BIOS;

– перейти в меню «Загрузка» (Boot) и в пункте «Приоритет вариантов загрузки» (Boot Option Priorities) выбрать первым по приоритету созданный USB-носитель;

– перейти в меню «Выход» (Exit) и выбрать пункт «Сохранить изменения и перезагрузиться» (Save Changes and Reset);

– дождаться перезагрузки АРМ и загрузки ОС DOS с USB-носителя;

– запустить пакетный файл FLASH.BAT для запуска процедуры прошивки UEFI BIOS;

– дождаться уведомления об успешном обновлении прошивки UEFI BIOS и перезагрузить АРМ для дальнейшей настройки параметров UEFI BIOS.

5.4.2.2 Настройка параметров UEFI BIOS

После прошивки UEFI BIOS версией 2.20 перед настройкой его параметров необходимо применить настройки по умолчанию, для чего перейти в меню «Выход» (Exit) и выбрать пункт «Восстановить настройки» (Restore Defaults).

Для настройки параметров UEFI BIOS необходимо произвести следующие действия:

– отключить аппаратную виртуализацию. Для отключения аппаратной виртуализации необходимо:

  – перейти в меню «Расширенная настройка» (Advanced);

  – перейти в подменю «Конфигурация процессора» (CPU Configuration);

  – установить параметр Intel Virtualization Technology в положение «отключено» (disabled);

  – вернуться в меню «Расширенная настройка» и перейти в подменю «Конфигурация чипсета» (Chipset Configuration);

  – перейти в подменю «Конфигурации интегрированного контроллера ввода-вывода» (Integrated IO configuration);

  – установить параметр Intel VT-d в положение «отключено» (disabled);

– отключить поддержку USB-носителей в режиме совместимости. Для отключения USB-носителей в режиме совместимости необходимо:

  – перейти в меню «Расширенная настройка» (Advanced);

  – перейти в подменю «Конфигурация чипсета» (Chipset Configuration);

  – перейти в подменю «Конфигурация южного моста» (South Bridge Configuration);

  – установить параметр Legacy USB support в положение «отключено» (disabled);

– отключить загрузку ВПО интегрированного сетевого адаптера. Для отключения загрузки ВПО интегрированного сетевого адаптера необходимо:

– перейти в меню «Расширенная настройка» (Advanced);

– перейти в подменю «Конфигурация PCIe/PCI/PnP» (PCIe/PCI/PnP Configuration);

– установить параметры Load Onboard LAN 1 Option ROM и Load Onboard LAN 2 Option ROM в положение «отключено» (disabled);

– включить поддержку режима Legacy BIOS. Для включения поддержки режима Legacy BIOS необходимо:

– перейти в меню «Загрузка» (Boot);

– перейти в меню «Вариант загрузки» (Boot option filter);

– выбрать из предлагаемых вариантов Legacy only;

– задать приоритет загрузки ОС с внутренних носителей. Для задания приоритета загрузки ОС с внутренних носителей необходимо:

– перейти в меню «Загрузка» (Boot);

– перейти в подменю Boot Option #1;

– выбрать внутренний носитель с ОС Astra Linux;

– установить параметры Boot Option #2 и Boot Option #3 в положение «отключено» (disabled);

– задать пароль администратора. Для задания пароля администратора необходимо перейти в меню «Безопасность» (Security) и в подменю «Пароль администратора» (Administrator Password) ввести пароль. Пароль администратора не должен содержать слова administrator или его часть, должен состоять не менее чем из 8 символов и в нем должны присутствовать символы трех категорий – цифры, прописные и строчные буквы английского языка;

– отключить функции удаленного мониторинга. Для включения аппаратной блокировки функции удаленного мониторинга необходимо выключить АРМ и установить перемычку JPB на материнской плате в положение «отключено» (контакты 2-3).

5.4.2.3 Настройка контроля целостности секции UEFI BIOS и создание файла-сценария для получения образа секции UEFI BIOS

После прошивки и настройки параметров UEFI BIOS необходимо выполнить настройку процедуры контроля целостности секции UEFI BIOS, для чего включить АРМ и дождаться загрузки ОС Astra Linux. После успешной загрузки ОС необходимо выполнить вход под учетной записью администратора (root).

Для настройки подсистемы контроля целостности необходимо выполнить следующие действия:

– создать исполняемый файл-сценарий для получения образа секции UEFI BIOS;

– получить образ секции UEFI BIOS;

– выполнить сравнение контрольной суммы полученного образа с эталонной контрольной суммой, указанной в файле BIOS.220.MD5;

– добавить файл-сценарий в автозагрузку;

– настроить подсистему контроля целостности.

Для создания исполняемого файла-сценария необходимо выполнить следующие действия:

– в директории /etc/init.d создать файл-сценарий (например, biosdump);

– в файл-сценария необходимо поместить код для получения и сохранения секции UEFI BIOS в файл (например, /etc/biosdump.bin). Секция UEFI BIOS расположена в памяти с адреса 0xFFC60000 по 0xFFFFFFFF. Пример содержимого файла-сценария представлен далее:

```
#!/bin/bash
#
# chkconfig: 2345 01 01
# description: Dump BIOS region from flash to filesystem
# Source function library.
. /etc/init.d/functions
```

```
OUTPUT_FILE=/etc/biosdump.bin
FLASH_SIZE=0x3A0000
FLASH_MEMORY_BASE=0xFFFFFFFF
dump()
{
     rm -f $OUTPUT_FILE
dd if=/dev/mem of=$OUTPUT_FILE bs=64K skip=$(( ($FLASH_MEMORY_BASE
- $FLASH_SIZE + 1) / 1024 / 64 )) > /dev/null 2>&1
}
start()
{
     touch /var/lock/subsys/biosdump
}
stop() {
     rm -f /var/lock/subsys/biosdump
     dump
}
case "$1" in
  start)
     start
        ;;
  stop)
     stop
        ;;
  dump)
dump
        ;;
  *)
        echo $"Usage: $0 {start|stop|dump}"
        exit 1
esac
exit 0
```

В зависимости от используемой подсистемы контроля целостности допускается создавать образ секций UEFI BIOS либо во время загрузки ОС для дальнейшего контроля целостности средствами ОС, либо во время завершения

работы ОС для дальнейшего контроля целостности при следующем включении АРМ средствами аппаратно-программными модулями доверенной загрузки.

5.4.2.4 Настройка подсистемы контроля целостности

Для получения образа секции UEFI BIOS необходимо запустить созданный на предыдущем этапе файл-сценарий (например, cd /etc/init.d/ && ./biosdump dump), в результате выполнения которого образ секции UEFI BIOS будет сохранен в файл, расположенный в заданной в файле-сценарии директории (например, /etc/biosdump.bin).

Для сравнения контрольной суммы полученного образа с эталонной необходимо рассчитать контрольную сумму полученного образа UEFI BIOS с помощью утилиты md5sum (например, md5sum –b /etc/biosdump.bin) и сравнить полученную контрольную сумму с эталонной контрольной суммой, находящейся в файле BIOS.220.MD5. Если контрольные суммы совпадают, то перейти к дальнейшей настройке, иначе еще раз произвести процедуру обновления прошивки UEFI BIOS.

Для добавления файла-сценария в автозагрузку необходимо выполнить следующие действия:

– добавить созданный файл-сценарий в автозагрузку с помощью утилиты chkconfig с ключом add (например, chkconfig –-add biosdump);

– проверить наличие файла сценария в автозагрузке с помощью утилиты chkconfig с ключом list (например, chkconfig --list).

Таким образом, в зависимости от конфигурации файла-сценария, создание образа секции UEFI BIOS будет производиться автоматически либо во время загрузки ОС (при использовании init-скриптов – ветка «start»), либо во время завершения работы ОС (при использовании init-скриптов – ветка «stop»).

Для настройки подсистемы контроля целостности, в зависимости от используемой подсистемы контроля целостности (Astra Linux, АПМДЗ и т.д.), необходимо штатными средствами подсистемы добавить путь к полученному

образу секции UEFI BIOS и выполнить переинициализацию базы данных подсистемы контроля целостности.

5.5 Экспериментальные работы, позволяющие обосновать достаточность принятых организационно-технических мер защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS)

Для экспериментальной части работ с ВПО UEFI BIOS были проведены следующие эксперименты:

– нагрузочное тестирование аппаратного обеспечения. В ходе данного эксперимента для АРМ были установлены базовые оптимизированные параметры UEFI BIOS для инициализации аппаратного обеспечения. В ОС Astra Linux в течении суток был запущен нагрузочный тест (Stress) оперативной памяти и процессора, в результате работы которого ошибок не выявлено;

– искажение файла с образом секции UEFI BIOS. В ходе данного эксперимента перед завершением работы АРМ был преднамеренно искажен файл с образом секции UEFI BIOS. При следующей загрузке АРМ подсистема контроля целостности обнаружила искажение в файле с образом секции UEFI BIOS, и дальнейшая загрузка АРМ была прекращена;

– перепрошивка UEFI BIOS с использованием штатных средств ПО или специализированного ПО АРМ. В ходе данного эксперимента штатными средствами ПО или специализированным ПО перепрошить UEFI BIOS не удалось, ввиду их отсутствия в АРМ;

– перепрошивка UEFI BIOS с внешнего носителя. В ходе данного эксперимента были созданы: загрузочный USB-носитель и CD-диск с необходимым ПО, и подключены к АРМ. Однако загрузиться с данных носителей не удалось, ввиду отсутствия возможности загрузки с внешних носителей в АРМ, которая была отключена настройками UEFI BIOS, а выполнить вход в графическую оболочку UEFI BIOS для изменения настроек или вход в ОС Astra Linux для запуска специализированной утилиты

перепрошивки UEFI BIOS оказалось невозможным, ввиду наличия пароля администратора для входа в систему;

– изменение настроек UEFI BIOS. В ходе данного эксперимента изменить настройки *UEFI BIOS* не удалось, ввиду наличия пароля администратора для входа в систему;

– подбор пароля администратора UEFI BIOS. В ходе данного эксперимента подобрать пароль администратора не удалось, ввиду использования парольной политики;

– к разъему IPMI LAN был подключен ПЭВМ для осуществления удаленного мониторинга функционирования АРМ. Однако получить информацию от АРМ не удалось, ввиду отсутствия возможности удаленного мониторинга функционирования АРМ, которая была отключена перемычкой JPB на материнской плате;

– в АРМ была установлена и настроена виртуальная среда функционирования гостевой ОС. Однако корректно развернуть и настроить данную ОС не удалось, ввиду отсутствия возможности использования аппаратной виртуализации, которая была отключена настройками UEFI BIOS, а выполнить вход в графическую оболочку UEFI BIOS для изменения настроек оказалось невозможным, ввиду наличия пароля администратора для входа в систему;

– были созданы: загрузочный USB-носитель и CD-диск с необходимым ПО, и подключены к АРМ. Однако загрузится с данных носителей не удалось, ввиду отсутствия возможности загрузки с внешних носителей для АРМ, которая была отключена настройками UEFI BIOS, а выполнить вход в графическую оболочку UEFI BIOS для изменения настроек оказалось невозможным, ввиду наличия пароля администратора для входа в систему;

– к встроенному сетевому интерфейсу АРМ был подключен ПЭВМ для загрузки с него операционной системы по сети «Ethernet». Однако с данной ПЭВМ загрузиться не удалось, ввиду отсутствия возможности загрузки по сети

«Ethernet» в АРМ, которая была отключена настройками UEFI BIOS, а выполнить вход в графическую оболочку UEFI BIOS для изменения настроек оказалось невозможным, ввиду наличия пароля администратора для входа в систему;

– передача пакетов с АРМ на МЭ, не содержащих контрольной суммы на область данных. В ходе данного эксперимента генератором трафика с АРМ были отправлены пакеты произвольного вида на МЭ, т.е. воспроизведено случайное или преднамеренное перенаправление информации. Однако данные пакеты были отбракованы МЭ, ввиду несоответствия формату и отсутствия в них поля контрольной суммы (CRC-16) на область данных;

– передача искаженных пакетов с АРМ на МЭ, содержащих контрольную сумму на область данных. В ходе данного эксперимента в СПО АРМ были внесены искажения в область данных корректно сформированных пакетов и отправлены на МЭ, т.е. воспроизведено случайное или преднамеренное попадание конфиденциальной информации. Однако данные пакеты были отбракованы МЭ, ввиду обнаружения неверной контрольной суммы (CRC-16) на область данных.

Таким образом, результаты экспериментальных работ подтверждают эффективность и достаточность разработанного комплекса организационно–технических мер, направленных на обеспечение защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS).

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта были рассмотрены существующие подходы к анализу встроенного программного обеспечения (BIOS) и методы тестирования программного обеспечения, рассмотрена типовая архитектура автоматизированной системы в защищенном исполнении и типовой состав средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении. Далее было проведено моделирование угроз безопасности, разработана модель нарушителя, проведен анализ функциональных возможностей аппаратного обеспечения и встроенного программного обеспечения (BIOS), и выявлен перечень опасных функциональных возможностей встроенного программного обеспечения (BIOS).

В результате проведенной работы была разработана методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении. В соответствии с данной методикой были разработаны меры защиты от негативных действий функциональных возможностей встроенного программного обеспечения (BIOS) на аппаратно-программную среду средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

Таким образом, все пункты технического задания на выпускную квалификационную работу выполнены в полном объеме.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Васин В.И., Любезнов А.П. Встроенное программное обеспечение UEFI в современных средствах вычислительной техники // Сборник статей III ежегодной межвузовской научно-практической конференции «Информационные технологии в науке и образовании. Проблемы и перспективы» / под ред. Фионовой Л.Р., Дурина А.В. - Пенза: Изд-во ПГУ, 2016. – С. 148-151.

2 Динамический анализ кода. [Электронный ресурс]: Режим доступа: http://www.viva64.com/ru/t/0070/, свободный.

3 Кулямин В.В. Методы верификации программного обеспечения // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению "Информационно-телекоммуникационные системы", 2008. - 117 с..

4 Intel 6 Series Chipset and Intel C200 Series Chipset Datasheet. [Электронный ресурс]: Режим доступа: http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/6-chipset - c200-chipset-datasheet.pdf, свободный.

5 Intel 82579 Gigabit Ethernet PHY Datasheet v.2.1. [Электронный ресурс]: Режим доступа: http://www.intel.com/content/www/us/en/embedded/ products/networking/82579-gbe-phy-datasheet-vol-2-1.html, свободный.

6 Unified Extensible Firmware Interface Specification Version 2.5. [Электронный ресурс]: Режим доступа: http://www.uefi.org/sites/ default/files/resources/UEFI%202_5.pdf, свободный.

7 Hacking the Extensible Firmware Interface - Black Hat. [Электронный ресурс]: Режим доступа: https://www.blackhat.com/presentations/bh-usa-07/Heasman/Presentation/bh-usa-07-heasman.pdf, свободный.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

УДК 004.056.53

Политехнический институт
Факультет приборостроения, информационных технологий и электроники
Выпускающая кафедра:
Информационная безопасность систем и технологий
Учебная группа 11ПИ1

УТВЕРЖДАЮ
Зав. кафедрой ИБСТ
к.т.н., доцент С.Л. Зефиров
«27» 04 2016 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

студента группы 11ПИ1 Васина Вячеслава Ивановича

Специальность 100503 – Информационная безопасность автоматизированных систем

Тема дипломного проекта: Методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

Руководитель ВКР

к.т.н., заведующий кафедрой ИБСТ Зефиров С. Л.

Нормоконтролер

к.т.н., доцент Фатеев А.Г.

1 Цели и задачи ВКР

Цель работы – разработка методики, позволяющей выявлять опасные функциональные возможности встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

По результатам проведения работ со встроенным программным обеспечением (BIOS) в соответствии с данной методикой разработаны меры защиты от негативных действий функциональных возможностей встроенного программного обеспечения (BIOS) на аппаратно-программную среду средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении (АСЗИ).

2 Тактико-технические требования к выполнению ВКР

Технические требования:

- методика должна быть применена для встроенного программного обеспечения (BIOS) содержащего секцию UEFI BIOS;
- методика должна содержать:
  - типовую архитектуру АСЗИ;
  - модель угроз безопасности;
  - модель нарушителя;
  - типовой состав СВТ, применяемых в АСЗИ;
  - анализ встроенного программного обеспечения (BIOS) СВТ, применяемых в АСЗИ, по документации;
  - рекомендации по установке и настройке BIOS с целью защиты от негативных действий функциональных возможностей встроенного программного обеспечения (BIOS) СВТ, применяемых в АСЗИ;

– в методике анализ встроенного программного обеспечения (BIOS) СВТ, применяемых в АСЗИ, должен включать:

– анализ функциональных возможностей аппаратного обеспечения;

– анализ функциональных возможностей встроенного программного обеспечения (BIOS);

– выявление перечня опасных функциональных возможностей встроенного программного обеспечения (BIOS);

– разработку комплекса организационно-технических мер защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS).

В ВКР должны быть рассмотрены следующие вопросы:

– обзор современного встроенного программного обеспечения (BIOS) СВТ, применяемых в АСЗИ;

– обзор существующих подходов к анализу встроенного программного обеспечения (BIOS);

– обзор методов тестирования программного обеспечения;

– разработка методики выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) СВТ, применяемых в АСЗИ;

– экспериментальные работы, позволяющие обосновать достаточность принятых организационно-технических мер защиты от опасных функциональных возможностей встроенного программного обеспечения (BIOS).

3 Требования к разрабатываемой документации

Отчет по ВКР выполняется в соответствии с требованиями ГОСТ 7.32.

Перечень и содержание графической части ВКР:

– инфраструктура типовой АСЗИ - 1л., ф. А1 (плакат);

– структура типового современного СВТ, используемого в АСЗИ - 1л., ф. А1 (плакат);

– структура прошивки флэш-памяти встроенного программного обеспечения (BIOS) - 1л., ф. А1 (плакат);

– схема процесса загрузки UEFI BIOS - 1л., ф. А1 (плакат);

– список функциональных возможностей UEFI BIOS - 1л., ф. А1 (плакат);

– список мер защиты от негативных действий функциональных возможностей - 1л., ф. А1 (плакат).

Электронная копия отчета и графической части ВКР должна быть представлена на магнитном или оптическом носителе.

4 Сроки выполнения ВКР

Тема утверждена приказом ректора ПГУ № 491 а/о от «25» 04 2016 г.

Дата выдачи задания «25» марта 2016 г.

Время дипломного проектирования с 18.04.2016 г. по 10.06.2016 г.

Задание к исполнению принял «25» марта 2016 г.

Исполнитель ВКР _____ Васин В.И.

Ректору Пензенского
государственного университета
А. Д. Гулякову
от студента 5 курса ФПИТЭ
очной формы обучения
Васина Вячеслава Ивановича

заявление.

Я, Васин Вячеслав Иванович, студент 5 курса очной формы обучения специальности 10.05.03 – «Информационная безопасность автоматизированных систем», прошу провести проверку с использованием системы «Антиплагиат» дипломного проекта на тему «Методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении», выполненной мной самостоятельно, на содержание элементов плагиата.

Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим Стандартом университета «Выпускная квалификационная работа обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры», согласно которому обнаружение плагиата является основанием для недопуска ВКР к защите и отчисления из университета.

_____ /В.И. Васин/
31.05.2016

ВКР представлена на проверку 31.05.2016.

Руководитель ВКР _____ /С.Л. Зефиров/

Протокол

проверки на оригинальность в системе «Антиплагиат.ВУЗ» выпускной квалификационной работы на тему «Методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении» студента 5 курса В.И. Васина специальности 10.05.03 – «Информационная безопасность автоматизированных систем».

Руководитель ВКР, к.т.н., заведующий кафедрой ИБСТ Зефиров Сергей Львович.

Отчет о проверке № 1

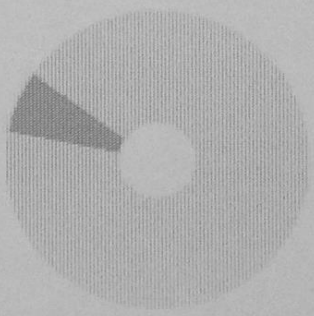
ФИО: Иванов Алексей Петрович
дата выгрузки: 03.06.2016 00:18:15
пользователь: ap_ivanov@mail.ru / ID: 1267386
отчет предоставлен сервисом «Антиплагиат»
на сайте http://www.antiplagiat.ru

Информация о документе

№ документа: 38
Имя исходного файла: ВКР_Васин_ВИ_11ПИ1.pdf
Размер текста: 2277 кБ
Тип документа: Не указано
Символов в тексте: 103914
Слов в тексте: 12236
Число предложений: 410

Информация об отчете

Дата: Отчет от 03.06.2016 00:18:50 - Последний готовый отчет
Комментарии: Не указано
Оценка оригинальности: 92.86%
Заимствования: 7.14%
Цитирование: 0%



Оригинальность: 92.86%
Заимствования: 7.14%
Цитирование: 0%

15.06.2016 г. С.Л. Зефиров

Секретарь ГЭК А.П. Иванов

Отзыв

на выпускную квалификационную работу студента 5 курса

Васина Вячеслава Ивановича на тему:

Методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении

Специальность 10.05.03

В рамках дипломного проекта Васину В.И. была поставлена задача разработать методику, позволяющую выявлять опасные функциональные возможности встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

В ходе выполнения дипломного проекта были рассмотрены существующие подходы к анализу встроенного программного обеспечения (BIOS) и методы тестирования программного обеспечения, рассмотрена типовая архитектура автоматизированной системы в защищенном исполнении и типовой состав средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении. Было проведено моделирование угроз безопасности, разработана модель нарушителя, проведен анализ функциональных возможностей аппаратного обеспечения и встроенного программного обеспечения (BIOS), и выявлен перечень опасных функциональных возможностей встроенного программного обеспечения (BIOS).

По результатам проведения работ со встроенным программным обеспечением (BIOS) в соответствии с данной методикой были разработаны меры защиты от негативных действий функциональных возможностей встроенного программного обеспечения (BIOS) на аппаратно-программную среду средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

По теме дипломного проекта Васиным В.И. была написана статья «Встроенное программное обеспечение UEFI в современных средствах вычислительной техники», опубликованная в сборнике статей III ежегодной межвузовской научно-практической конференции «Информационные технологии в науке и образовании. Проблемы и перспективы».

За время дипломного проектирования Васин В.И. показал отличные навыки в освоении нового материала, работы с научно-технической литературой, умение применять полученные знания на практике и самостоятельно решать технические проблемы.

В заключении необходимо отметить, что дипломный проект выполнен в полном соответствии с техническим заданием, может быть допущен к защите в ГАК и заслуживает оценки «отлично», а дипломник Васин Вячеслав Иванович достоин присвоения квалификации специалиста по информационной безопасности по специальности 10.05.03 «Информационная безопасность автоматизированных систем».

Количество баллов – 15.

Руководитель ВКР,
к.т.н., заведующий
кафедрой ИБСТ  Зефиров Сергей Львович

Рецензия

на выпускную квалификационную работу студента 5 курса

Васина Вячеслава Ивановича на тему:

Методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении

Специальность 10.05.03

На рецензию представлена выпускная квалификационная работа студента факультета приборостроения, информационных технологий и электроники Пензенского государственного университета Васина Вячеслава Ивановича, содержащая пояснительную записку в объёме 77 страниц, 20 рисунков, 10 таблиц, 7 источников.

Предложенная для дипломного проекта тема: «Методика выявления опасных функциональных возможностей встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении» является актуальной, и результаты исследования представляют практический интерес для дальнейшей работы.

Целью дипломного проектирования является разработка методики, позволяющей выявлять опасные функциональные возможности встроенного программного обеспечения (BIOS) средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении.

В процессе выполнения дипломного проекта были рассмотрены существующие подходы к анализу встроенного программного обеспечения (BIOS) и методы тестирования программного обеспечения, рассмотрена типовая архитектура автоматизированной системы в защищенном исполнении и типовой состав средств вычислительной техники, применяемых в автоматизированных системах в защищенном исполнении. Было проведено моделирование угроз безопасности, разработана модель нарушителя, проведен анализ функциональных возможностей аппаратного обеспечения и

встроенного программного обеспечения (BIOS), и выявлен перечень опасных функциональных возможностей встроенного программного обеспечения (BIOS).

В результате рассмотрения представленных материалов следует отметить следующее:

– содержание дипломного проекта полностью соответствует техническому заданию на дипломное проектирование;

– проект выполнен в соответствии с заданием в полном объеме;

– материалы пояснительной записки изложены технически грамотно.

К недостатку выпускной квалификационной работы можно отнести следующее: не проведено исследование рынка на предмет наличия современных материнских плат, имеющих соответствующие разрешительные документы на встроенное программное обеспечение (BIOS), для применения в АСЗИ.

Несмотря на указанный недостаток считаю, что все пункты технического задания были выполнены, и выпускная квалификационная работа может быть допущена к защите в ГАК и заслуживает оценки «отлично», а дипломник Васин Вячеслав Иванович достоит присвоения квалификации специалиста по информационной безопасности по специальности 10.05.03 «Информационная безопасность автоматизированных систем».

С учетом актуальности темы, трудоемкости и сложности выполнения данную выпускную квалификационную работу оцениваю в 9 баллов.

Рецензент,
ведущий инженер
ПФ ФГУП «НТЦ «Атлас» Пелин Владимир Николаевич

ОТДЕЛ КАДРОВ

Зам. нач. нач. начальника ОК
Т.И. Бубнова