Специальная публикация NIST 800-193

Руководства по отказоустойчивости встроенного микропрограммного обеспечения платформы

Andrew Regenscheid

Эта публикация доступна бесплатно в:
https://doi.org/10.6028/NIST.SP.800-193

КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ

National Institute of Standards and Technology
U.S. Department of Commerce

Специальная публикация NIST 800-193

Руководства по отказоустойчивости встроенного микропрограммного обеспечения платформы

Andrew Regenscheid
Отдел компьютерной безопасности
Лаборатория информационных технологий

Эта публикация доступна бесплатно в:
https://doi.org/10.6028/NIST.SP.800-193

Май 2018

Министерство торговли США
Wilbur L. Ross, Jr., министр

Национальный институт стандартов и технологий
Walter Copan, директор NIST и заместитель министра торговли по стандартам и технологиям

Полномочия

Эта публикация была разработана NIST в соответствии с его обязанностями, установленными согласно Закону о модернизации безопасности федеральной информации (FISMA) от 2014, 44 U.S.C. § 3551 *и далее,* Общественный закон (P.L). 113-283. NIST является ответственным за разработку стандартов и руководств по информационной безопасности, включая минимальные требования для федеральных информационных систем, но такие стандарты и руководства не должны применяться к системам национальной безопасности без специального санкционирования соответствующих федеральных должностных лиц, осуществляющих полномочия по таким системам. Это руководство непротиворечиво с требованиями Циркуляра A-130 Министерства управления и бюджета (OMB).

Ничто в этой публикации не должно использоваться в противоречие со стандартами и руководствами, определёнными Министром торговли в соответствие с его законными полномочиями как обязательные для федеральных агентств. Также, это руководство не должно быть интерпретировано как изменение или замена существующих полномочий Министра торговли, Директора OMB или какого-либо другого федерального должностного лица. Это руководство было подготовлено к использованию федеральными агентствами. Эта публикация может быть также использована на добровольной основе неправительственными организациями и это не попадает по действие авторского права. Однако упоминание приветствовалось бы NIST.

Специальная публикация Национального института стандартов и технологий 800-193
Natl. Inst. Stand. Technol. Spec. Publ. 800-193, 45 pages (May 2018)
CODEN: NSPUE2

Эта публикация доступна бесплатно в:
https://doi.org/10.6028/NIST.SP.800-193

Комментарии к этой публикации могут быть представлены в:

Национальный институт стандартов и технологий
Для: Отдел компьютерной безопасности, лаборатория информационных технологий
100 Bureau Drive (Mail Stop 8930), Gaithersburg, MD 20899-8930
Email: sp800-193comments@nist.gov

Отчёты по технологиям компьютерных систем

Лаборатория информационных технологий (ITL) в Национальном институте стандартов и технологий (NIST) продвигает американскую экономику и общее благосостояние, обеспечивая техническое лидерство для национальной инфраструктуры измерений и стандартов. ITL разрабатывает тесты, методы испытаний, справочные данные, осуществляет подтверждения концепций реализации и технический анализ, чтобы продвинуть разработку и продуктивное использование информационных технологий. Обязанности ITL включают разработку управленческих, административных, технических и физических стандартов и руководств для обеспечения рентабельной безопасности и приватности информации не связанной с национальной безопасностью в федеральных информационных системах. Специальные публикации 800-серии содержат информацию относительно исследований, руководств и усилий ITL, направленных на повышение безопасности информационных систем, и её совместных работ с отраслями, правительством и академическими организациями.

Резюме

Этот документ предоставляет технические руководства и рекомендации, поддерживающие отказоустойчивость встроенного микропрограммного обеспечения платформы и данных от потенциально разрушительных атак. Платформа - набор фундаментальных компонентов аппаратных средств и встроенного микропрограммного обеспечения, требуемых для загрузки и функционирования системы. Успешная атака на встроенное микропрограммное обеспечение платформы может перевести систему в неоперабельную, возможно навсегда, или потребовать перепрограммирования оригинальным производителем, что приводит к значительным потерям пользователей. Технические руководства в этом документе продвигают отказоустойчивость платформы, описывая механизмы безопасности для защиты платформы от несанкционированных изменений, обнаружения происходящих несанкционированных изменений и восстановления после атак быстро и безопасно. Разработчики, включая производителей укомплектованного оборудования (OEMs) и поставщиков компонентов/устройств, могут использовать эти руководства, чтобы встроить в платформы более стойкие механизмы защиты. Системные администраторы, специалисты по безопасности и пользователи могут использовать этот документ, чтобы формировать стратегии приобретения и приоритеты для будущих систем.

Ключевые слова

BIOS; подписание кода; встроенное микропрограммное обеспечение; дополнительное ROM; встроенное микропрограммное обеспечение платформы

Благодарности

Автор, Andrew Regenscheid из Национального института стандартов и технологий (NIST), хочет благодарить своих коллег, которые рассмотрели проекты этого документа и способствовали его техническому содержанию. В частности, NIST ценит содействие от экспертов от промышленности и правительства, которое помогло вести эту работу. Среди этих экспертов был Chirag Schroff от Cisco; Mukund Khatri от Dell; CJ Coppersmith, Gary Campbell, Shiva Dasari и Tom Laffey от Hewlett Packard Enterprise; Jim Mann от HP Inc.; Charles Palmer от IBM; Bob Hale, David Riss и Vincent Zimmer от Intel Corporation; Paul England и Rob Spiger от Microsoft и Shane Steiger.

NIST также хотел бы выразить признательность и поблагодарить Kevin Bingham, Cara Steib и Mike Boyle из Агентства национальной безопасности, которые предоставили существенные дополнения в этот документ, а также Jeffrey Burke от Noblis NSP.

Аудитория

Целевая аудитория для этого документа включает вендоров систем и устройств платформ компьютерных систем, включая производителей клиентских, серверных и сетевых устройств. Принципы защиты и рекомендации, содержавшиеся в этом документе, должны быть широко применимы и к другим классам систем с обновляемым встроенным микропрограммным обеспечением, включая устройства Интернета вещей, встроенные системы и мобильные устройства. Технические руководства предполагают, что читатели имеют экспертные знания в архитектурах платформ и предназначены, прежде всего, для разработчиков и инженеров, ответственных за реализацию технологий безопасности микропрограммного уровня в системах и устройствах.

Фирменная информация

Резюме

Архитектуры современных вычислительных систем можно представить в слоях. Верхние слои - *программное обеспечение*, состоявшее из операционной системы и приложений. Для предоставления большинства функциональных возможностей, используемых пользователями, они полагаются на функции и услуги, предоставляемые нижележащими слоями, которые в этом документе обобщённо именуются как *платформа*. Платформа включает аппаратные средства и микропрограммные компоненты необходимые, чтобы инициализировать компоненты, загрузить систему и предоставить услуги среды выполнения, реализуемые аппаратными компонентами.

Встроенное микропрограммное обеспечение платформы и связанные с ним данные конфигурации, очень важны для доверенности вычислительной системы. Большая часть этого встроенного микропрограммного обеспечения имеет высокую привилегию в архитектурах систем и потому, что это встроенное микропрограммное обеспечение необходимо для функционирования систем, является важным восстановление этого встроенного микропрограммного обеспечения. Успешная атака на встроенное микропрограммное обеспечение платформы может перевести систему в неоперабельную, возможно навсегда, или потребовать перепрограммирования оригинальным производителем, что приводит к значительным потерям пользователей. Другие изощрённые вредоносные атаки могут попытаться внедрить постоянное вредоносное программное обеспечение в это встроенное микропрограммное обеспечение, изменив критические сервисы низкого уровня, чтобы нарушить функционирование, эксфильтрировать данные или иначе повлиять на состояние безопасности компьютерной системы.

Более ранние публикации NIST учитывали угрозу атак на один конкретный тип встроенного микропрограммного обеспечения платформы - загрузочное встроенное микропрограммное обеспечение, обычно известное как Базовая система ввода-вывода (BIOS). Однако платформа состоит из многих других устройств со встроенным микропрограммным обеспечением и данными конфигурации. Эти устройства, включая контроллеры запоминающих устройств и сетевые контроллеры, графические процессоры и служебные процессоры, также имеют высокую привилегию и необходимы для систем, чтобы вести себя безопасно и достоверно.

Этот документ обеспечивает технические руководства, направленные на поддержание отказоустойчивости платформ против потенциально разрушительных атак. Эти руководства основаны на следующих трех принципах:

- **Защита:** Механизмы для обеспечения того, чтобы код Встроенного микропрограммного обеспечения Платформы и критические данные остались в состоянии целостности и были защищены от повреждения, такие как процесс для обеспечения аутентичности и целостности микропрограммных обновлений.
- **Обнаружение:** Механизмы для обнаружения того, когда код Встроенного микропрограммного обеспечения Платформы и критические данные были повреждены.
- **Восстановление:** Механизмы для восстановления кода Встроенного микропрограммного обеспечения Платформы и критических данных к состоянию целостности, если обнаружено, что некоторый такой микропрограммный код или критические данные являются поврежденными или, когда восстановление вызвано через санкционированный механизм. Восстановление ограничено способностью возвратить микропрограммный код и критические данные.

Эти руководства предназначены, чтобы учесть платформы в клиентских персональных компьютерах (PC), серверах и сетевых устройствах, но являются широко применимыми и к другим классам систем. Внедренцы, включая Производителей укомплектованного оборудования (OEMs) и поставщиков компонент/устройств, могут использовать эти руководства, чтобы встроить более сильные механизмы защиты в платформы. Системные администраторы, специалисты по безопасности и пользователи могут использовать этот документ, чтобы руководствоваться в стратегии приобретения и приоритетах для будущих систем.

Оглавление

Список приложений

Список иллюстраций

1 Введение

1.1 Назначение

Современное компьютерные системы и системы информационных технологий построены на множестве аппаратных компонентов, которые предоставляют фундаментальные возможности, требуемые для функционирования систем. У многих из этих аппаратных компонентов есть встроенное микропрограммное обеспечение и данные конфигурации, которые управляют их поведением и которые должны находиться в целостном состоянии для правильного функционирования систем. Один пример такого встроенного микропрограммного обеспечения обычно упоминается как Базовая система ввода-вывода (BIOS), которая используется, чтобы облегчить процесс инициализации аппаратного обеспечения и переход управления к операционной системе. В зависимости от системы, в ней могут быть десятки или сотни микроконтроллеров с другими видами программируемого встроенного микропрограммного обеспечения, которые поддерживают полную архитектуру системы. Этот набор аппаратных средств и микропрограммных компонентов, как правило, называют *платформой*.

Устройства, которые составляют платформу, крайне важны для целостности и доступности систем, построенных на платформе. Без этих устройств системы могут быть не в состоянии функционировать правильно или вообще могут не работать. Целевые атаки на некоторые устройства платформы могут значительно повлиять на состояние безопасности этих систем, понизив, возможно, до низкого уровня, осуществить постоянное вредоносное присутствие. У атак, которые стремятся повредить или удалить встроенное микропрограммное обеспечение платформы, есть потенциал, чтобы перевести системы в постоянный отказ, нанести существенный ущерб затронутым сторонам.

Назначение этого документа состоит в том, чтобы предоставить руководства безопасности по поддержке отказоустойчивости систем на уровне платформы. Как определено Международным советом по системной инженерии (INCOSE), устойчивость системы - “способность системы с конкретными характеристиками до, в течение и после нарушения функционирования, ликвидировать нарушения, восстановиться до допустимого уровня функционирования и поддерживать этот уровень в течение приемлемого промежутка времени”. [10] Применительно к информационным системам, кибер отказоустойчивость - “способность предвидеть, противостоять, восстанавливаться после и приспосабливаться к неблагоприятным условиям, усилиям, атакам или компрометациям систем, которые содержат кибер ресурсы”. Несмотря на то, что руководства по кибер отказоустойчивости на системном уровне описаны в проекте NIST Специальная Публикация 800-160, Том 2, эта публикация констатирует, что отказоустойчивость на системном уровне должна поддерживаться основополагающими средствами безопасности в компьютерных платформах. Руководства в этом документе поддерживают кибер отказоустойчивость, определяя механизмы, которые защищают встроенное микропрограммное обеспечение и данные конфигурации от атак, и то, как можно обнаружить и восстановиться после успешных атак.

1.2 Аудитория

Целевая аудитория для этого документа включает вендоров системных и платформенных устройств компьютерных систем, включая производителей клиентских, серверных и сетевых устройств. Технические руководства предполагают, что читатели имеют экспертные знания в архитектурах платформ и предназначены, прежде всего, для разработчиков и инженеров, ответственных за реализацию технологий безопасности уровня встроенного микропрограммного обеспечения в системах и устройствах.

Материал может быть также использован при разработке общекорпоративных стратегий приобретения и развертывания. Материал в этом документе технически ориентирован, и предполагается, что у читателей есть, по крайней мере, основное понимание принципов компьютерной безопасности и архитектур ЭВМ. Документ предоставляет справочную информацию, чтобы помочь таким читателям в понимании тем, которые обсуждаются.

1.2 Применимость и область

Цель этого документа состоит в том, чтобы предоставить принципы и руководства, которые могут поддержать отказоустойчивость платформы, прежде всего, против удаленных атак. Эти принципы и руководства непосредственно относятся к отдельным устройствам, которые составляют платформу (см. Раздел 2.1 для перечня примеров). Конкретно, они описывают механизмы защиты, нацеленные на защиту каждого устройства от несанкционированных изменений его встроенного микропрограммного обеспечения или критических данных и восстановления платформы до целостного состояния.

1.3 Структура документа

Остаток от этого документа организован в следующие основные разделы:

- Раздел 2 предоставляет информативные материалы, описывающие компоненты и архитектуры платформы.
- Раздел 3 описывает принципы защиты, которые формируют основу для руководств в этом документе, и описывает ключевые концепции для применения этих принципов к отказоустойчивости платформ.
- Раздел 4 содержит технические руководства безопасности по защите микропрограммного кода и критических данных, обнаружению санкционированных изменений и восстановлению до целостного состояния.
- Приложение A содержит список акронимов и сокращений в документе.
- Приложение B содержит глоссарий выбранных терминов из документа.
- Приложение C содержит список обозначений в документе.

2 Архитектура платформы

Гарантирование того, что микропрограммный код платформы и критические данные всегда находятся в целостном состоянии очень важно, чтобы гарантировать, что вычислительная система может функционировать свободной от вредоносного программного обеспечения. Современные клиент-серверные вычислительные системы могут считаться разделенными на две логических конструкции высокого уровня, *платформу* и *программное обеспечение*. Для назначений этого документа мы будем представлять комбинацию этих двух логических конструкций как систему. Обратите внимание на то, что рисунок 1 просто иллюстративен и не предназначен, чтобы представлять все возможные устройства в платформе, а также он не предназначен, чтобы представлять образцовую архитектуру для какого-то конкретного устройства. На высоком уровне, элементы в синим заштрихованных блоках - устройства, которые считаются частью платформы.

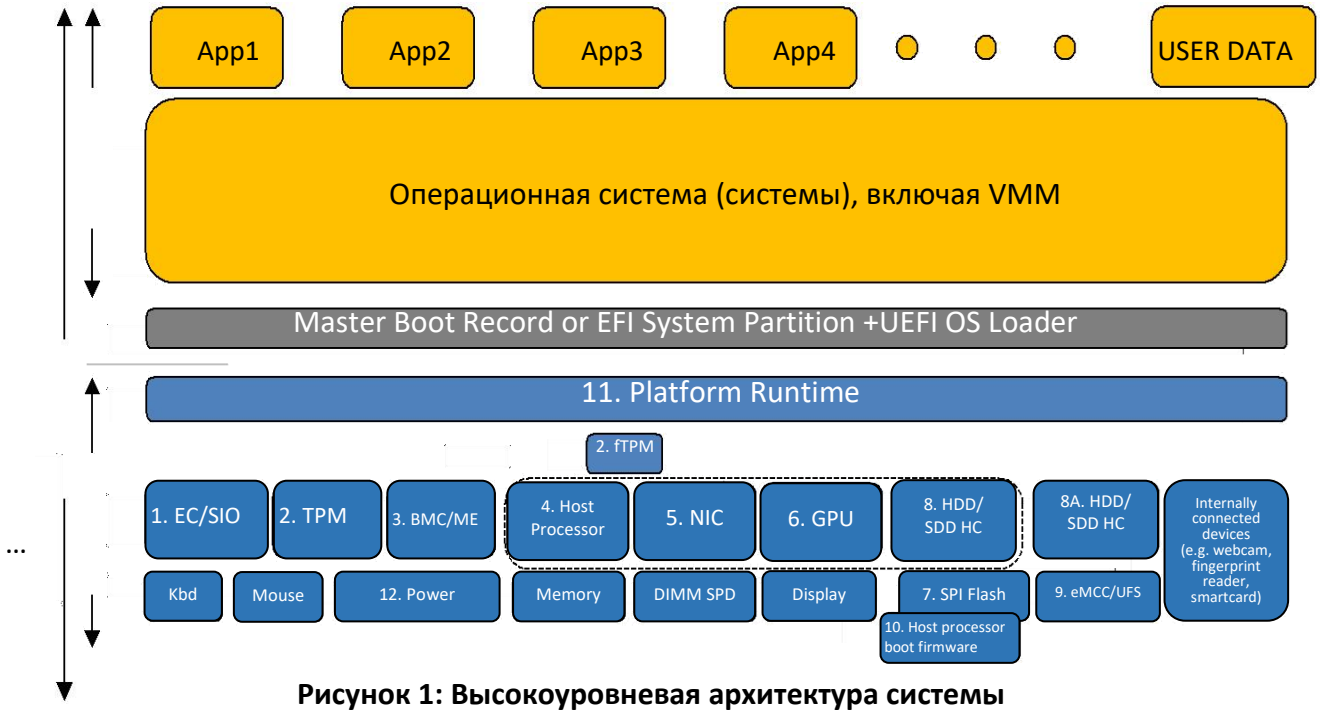

Рисунок 1: Высокоуровневая архитектура системы

Вообще говоря, *платформа* состоит из аппаратных средств и встроенного микропрограммного обеспечения, необходимого, чтобы загрузить систему к состоянию, в котором может быть загружено программное обеспечение или операционная система; *программное обеспечение* состоит из элементов, требуемых, чтобы загрузить операционную систему и все приложения и данные, впоследствии обрабатываемые операционной системой. Следует иметь ввиду, что некоторое встроенное микропрограммное обеспечение продолжает выполняться после того как программное обеспечение начнёт выполняться. Существующая наиболее успешная промышленная практика, а также публикации NIST, такие как NIST SP 800-147, *Руководства по защите BIOS* [1] и 800-147B NIST SP, *Руководства по защите BIOS для серверов* [2], уже рассматривали проблему защиты целостности загрузочного встроенного микропрограммного обеспечения главного процессора платформы (традиционно называемого BIOS, и позже UEFI [3])[1] и механизмы его обновления, но защита - только один из трех основных элементов кибер отказоустойчивости (другие два, являются обнаружением и восстановлением). Кроме того, устойчивость другого критического встроенного микропрограммного обеспечения на платформе еще не была учтена до уровня, как для загрузочного встроенного микропрограммного обеспечения главного процессора. Хотя это выходит за рамки этого документа, чтобы идентифицировать и определить каждую категорию и архитектуру устройств, которые содержат встроенное микропрограммное обеспечение, этот документ, применим к любому

устройству на платформе, которое содержит встроенное микропрограммное обеспечение, включая PC, сервера, сетевые устройства, смартфоны, планшеты, и т.д.

2.1 Устройства платформы

Как отмечено выше, платформа - набор устройств, которые предоставляют функциональные возможности и сервисы, необходимые операционной системе и приложениям. Хотя конечной целью является отказоустойчивость платформы в целом, важно понимать, что платформа состоит из многих различных устройств, часто разрабатываемых и производимых различными вендорами. По этой причине технические руководства в этом документе описаны с точки зрения руководств для отдельных устройств платформы.

Для описания отказоустойчивой платформы этот раздел определяет список устройств, которые часто очень важны для нормального и безопасного функционирования платформы. Эти устройства, как правило, содержат изменяемое встроенное микропрограммное обеспечение и покрываются обозначенной областью руководств безопасности в этом документе.

Однако, это не должно рассматриваться как исчерпывающий список всех устройств в каждой интересующей платформе. Вендоры платформ должны тщательно рассмотреть другие устройства, которые относятся к области их конкретной платформы.

В случае традиционной платформы на базе x86 (настольный компьютер, ноутбук, сервер, коммутатор), эти устройства представлены на рисунке 1 и определены ниже. Обратите внимание на то, что числа здесь ссылаются на используемые в качестве обозначений устройств на рисунке 1. Упорядочивание не предназначено, чтобы подразумевать любой приоритет или последовательность.

1. **Embedded Controller (EC) / Super I/O (SIO)**

   EC, как правило, ассоциируется с мобильными платформами (ноутбуки, конвертеры, планшеты), в то время как SIO, как правило, ассоциируется с настольными платформами (настольные компьютеры, настольные рабочие станции, моноблоки, тонкие клиенты). Это утверждение является не универсально верным, но в общем достаточно верно для установления типа клиентской системы, в которой присутствует EC или SIO. EC или SIO, как правило, контролируют функции в платформе, такие как клавиатура, светодиоды, вентиляторы, мониторинг/зарядка батареи, тепловой мониторинг, и т.д. Кроме того, это, как правило, - первое устройство системной платы в платформе, которое выполняет код ещё при нахождении главного процессора в отключенном состоянии, пока EC/SIO готовит главный процессор чтобы передать свою начальную строку кода встроенному микропрограммному обеспечению главного процессора.

2. **Trusted Platform Module (TPM)**

   TPM [4] - сопроцессор безопасности, способный безопасно хранить и использовать ключи шифрования и измерения состояния платформы. Эти возможности могут использоваться, среди прочего, чтобы обеспечить безопасность данных, хранимых в системе, обеспечить строгую идентичность устройства и подтвердить состояние системы. Хотя не все платформы включают или используют TPM, но в тех системах, в которых TPM включается и используется, его встроенное микропрограммное обеспечение должно быть защищено, учитывая его критичность в обеспечении гарантии доверенность платформы. TPMs также содержат энергонезависимую память, которая может содержать критические данные и, поэтому, должна быть защищена. TPM’s могут быть или дискретными устройствами или могут быть реализованными во встроенном микропрограммном обеспечении, выполняемом на хост-контроллере платформы или другом микроконтроллере (последние иногда упоминаются как микропрограммный TPM’s или fTPM).

3. **Контроллер управления платы (Baseboard Management Controller, BMC) / Механизм управления (Management Engine, ME)**
   BMC связан с серверными платформами, в то время как ME, как правило, связан с клиентскими платформами. В обоих случаях основной аспект их функциональности - служить внеполосным устройством управления, позволяющим администраторам платформы управлять платформой не требуя, чтобы работала хостовая операционная система. Несмотря на то, что они не всегда строго необходимы для основной вычислительной функции серверной или клиентской платформы, самые современные серверные и клиентские платформы включают BMC/ME, делая очень важным то, чтобы их встроенное микропрограммное обеспечение не влияло негативно на состояние целостности домена защиты главного процессора.

4. **Главный процессор (Host Processor) [иначе Центральный процессор (Central Processing Unit, CPU), иначе Процессор приложений (Application Processing Unit, APU)]**
   Главный процессор - основное устройство обработки в типовой платформе, традиционно называемое центральным процессором, а теперь также иногда называемое APU или Интегральной схемой (SoC). Это - устройство обработки, на котором работает основная операционная система (и/или гипервизор), а также приложения пользователей. Это - процессор, который отвечает за загрузку и выполнение встроенного микропрограммного обеспечения главного процессора.

5. **Сетевой контроллер (Network Interface Controller, NIC)**
   Может быть отдельным или интегрированным как часть SoC, самые современные клиентские и серверные платформы имеют хотя бы один NIC (проводной или радио), а могут иметь несколько, включая несколько типов (проводной, Wi-Fi, сотовый). Хотя, для загрузки платформы наличие NIC строго не требуется, в сегодняшнем взаимосвязанном мире важно иметь некоторую форму подключения в какой-то момент после того, как система загружена. Более важно то, что скомпрометированное встроенное микропрограммное обеспечение NIC может служить стартовой площадкой для других деяний в системе, использоваться, чтобы экс-фильтровать данные, служить посредником и т.д. В дополнение к встроенному микропрограммному обеспечению, запускаемому микроконтроллером, NIC может включать расширение встроенного микропрограммного обеспечения постоянной памяти (ROM), которое подгружается во время загрузки и выполняется главным процессором. Очень важно, что расширение встроенного микропрограммного обеспечения ROM NIC также защищено. Расширение встроенного микропрограммного обеспечения ROM, может находиться вместе с загрузочным встроенным микропрограммным обеспечением главного процессора (в случае интегрированного NIC), или находиться отдельно в самом NIC, как в случае платы расширения. NIC часто также содержит критические данные, например адреса Контроля доступа носителя информации (MAC), могут храниться в изменяемой памяти. Атака на это критические данные может привести к отказу в обслуживании (DoS) обеих из этих платформ, а также другой системы с соответствующим адресом MAC.

6. **Графический процессор (Graphics Processing Unit, GPU)**
   GPU - устройство, которое служит основным «внешним» устройством интерфейса пользователя (HID) в клиентских платформах. В некоторых случаях GPUs могут также использоваться в качестве сопроцессоров, чтобы поддерживать высокоэффективные вычисления. GPUs могут служить стартовой площадкой для других деяний в системе. В дополнение к встроенному микропрограммному обеспечению, запускаемому микроконтроллером, GPU может включать расширение встроенного микропрограммного обеспечения ROM, которое подгружается во время загрузки и выполняется главным процессором. Расширение встроенного микропрограммного обеспечения ROM, может находиться вместе с загрузочным встроенным микропрограммным обеспечением главного процессора (в случае интегрированного GPU), или может находиться отдельно в самом GPU, как в случае платы расширения.

7. **Последовательный периферийный интерфейс (Serial Peripheral Interface, SPI) флэш**
   Большинство современных платформ включает некоторый объём флэш SPI, чтобы хранить встроенное микропрограммное обеспечение, как правило, для загрузочного встроенного микропрограммного обеспечения главного процессора, хотя он может использоваться и для другого назначения.

8. **A) Хост-контроллер (Host Controller, HC) для запоминающих устройств**
   Для большинства современных платформ требуется некоторая форма хранения или в форме HDD или в форме SSD, чтобы загрузить операционную систему и поддержать приложения и данные пользователя. Для данных, которые будут храниться на запоминающем устройстве, используется Хост-контроллер (HC), чтобы переместить данные с оперативной памяти платформы на физический носитель по некоторой шине (например, SATA, SCSI, PCIe). У этого HC есть свой собственный микроконтроллер и связанное встроенное микропрограммное обеспечение. HC может быть или интегрирован в SoC или быть отдельным устройством или на плате расширения.

   B) Жесткий диск (Hard Disk Drive, HDD) / Твердотельный диск (Solid State Drive, SSD)
   HDD или SSD представляют текущее состояние возможностей для хранения больших объёмов данных в традиционной платформе. Эти устройства сопрягаются с Хост-контроллером. В HDD или SSD, микроконтроллер и связанное встроенное микропрограммное обеспечение используются, чтобы выполнить операцию по текущему хранению данных, посланных из оперативной памяти платформы на запоминающее устройство. Компрометация встроенного микропрограммного обеспечения HDD или SSD может также использоваться в качестве стартовой площадки для других деяний в системе или может использоваться, чтобы ставить под угрозу данные платформы и/или пользователя.

9. **Встроенная Карта MultiMedia (eMMC)/Universal Flash Storage (UFS)**
   eMMC и UFS появляются в качестве стандартных запоминающих устройств для мобильных систем. Каждая из них может включать их собственное расширение встроенного микропрограммного обеспечения ROM и/или микроконтроллер со связанным встроенным микропрограммным обеспечением.

10. **Загрузочное встроенное микропрограммное обеспечение главного процессора**
    В большинстве современных платформ загрузочное встроенное микропрограммное обеспечение главного процессора содержится во флэш устройстве SPI. BIOS и Unified Extensible Firmware Interface (UEFI) - примеры этого типа встроенного микропрограммного обеспечения.

11. **Встроенное микропрограммное обеспечение среды выполнения платформы**
    В дополнение к загрузочному встроенному микропрограммному обеспечению есть код среды выполнения платформы. Это - код, который остается резидентом в памяти и исполняется после того, как платформа загружена. Он является самым типичным для микроконтроллеров, где встроенное микропрограммное обеспечение требуется, чтобы осуществлять выполнение некоторой функции, в то время как система полностью функционирует. Примером встроенного микропрограммного обеспечения главного процессора, которое рассматривают как код среды выполнения, является код System Management Mode (SMM).

12. **Блок питания**
    Некоторые блоки питания имеют свой собственный микроконтроллер и встроенное микропрограммное обеспечение. Общие архитектуры батареи также включают внутреннюю логику и встроенное микропрограммное обеспечение, определяя заряд и уровень разряда батареи.

13. **Связующая логика (CPLD's, FPGA's) – *не изображена***
    Современные встроенные системы используют программируемые логические компоненты,

чтобы предоставить функциональность связующей логики. Есть два типа программируемых логических компонентов, Программируемая логическая матриц, известная как FPGAs и сложная программируемая логическая интегральная схема, известная как CPLDs. FPGAs, как правило, загружаются bitstream программами из подключенных флэш устройств. CPLDs, с другой стороны, программируются с bitstream однажды, а затем они сохраняют свою функцию, пока не будут перепрограммированы снова. Как правило, эта функциональность необходима для базового функционирования системы и, если будет повреждена, то это может привести к постоянному отказу в обслуживании платформы.

14. **Вентиляторы – *не изображены***

    У некоторых вентиляторов есть свой собственный микроконтроллер и связанное встроенное микропрограммное обеспечение.

2.2 Код и данные в устройствах платформы

Устройства, описанные выше, как правило, содержат некоторый набор встроенного микропрограммного обеспечения и данных в энергонезависимой памяти, или находиться на самом устройстве или на общем запоминающем устройстве (например, флэш SPI). Этот раздел описывает микропрограммный код и данные и кратко обсуждает область документа, связанную с этими компонентами.

2.2.1 Код

Микропрограммный код - набор инструкций, используемых блоком обработки любого устройства, чтобы выполнять операции, требуемые от устройства. Исторически, встроенное микропрограммное обеспечение в устройствах платформы редко изменялось на месте, хотя вендоры систем или компонентов могут разработать микропрограммные обновления, которые исправляют уязвимости, ошибки или добавляют новую функциональность. По мере увеличения сложности этого встроенного микропрограммного обеспечения, микропрограммные обновления стало всё более распространяться с инструментами, предоставляемыми вендорами аппаратного обеспечения и операционной системы, чтобы помочь администраторам обновлять свое встроенное микропрограммное обеспечение.

Поскольку встроенное микропрограммное обеспечение в значительной степени стимулирует поведение устройства важно, чтобы оно осталось в доверенном состоянии на платформе. Атаки на микропрограммный код могут перевести устройство в неоперабельное или внести в устройство злонамеренную функциональность. Встроенное микропрограммное обеспечение должно загружаться только из санкционированного источника, как правило, производителя системы или устройства платформы.

Руководства в этом документе описывают механизмы защиты микропрограммного кода, путём проверки обновлений с использованием цифровых подписей. Они также описывают механизмы обнаружения несанкционированных изменений во встроенном микропрограммном обеспечении и безопасным методам восстановления.

2.2.2 Данные

Данные - сведения, которые использует код встроенного микропрограммного обеспечения платформы, чтобы осуществлять его функционирование, как определено кодом. Данные могут быть далее категорированы как критичные и некритичные. Критичные данные включают параметры конфигурации и политики, которые необходимы, чтобы устройство было способно поддерживать его состояние безопасности. Некритические данные включают все другие данные.

2.2.2.1 Критические данные

Критические данные, могут использоваться для различного назначения, включая:

- **Параметры конфигурации:** Данные, которые сообщают коду, как формировать эксплуатационные аспекты устройства

Пример: Предоставление периферии, которая запрещена правилами корпоративной безопасности.

Пример: Таблица нефункциональных секторов на жестком диске.

- **Политики:** Данные, которые сообщают коду, какой образом действовать или как реагировать

  Пример: Порядок загрузки системы описывает разрешённые устройства, с которых делается попытка загрузиться, а также порядок.

  Пример: Защищенная загрузка UEFI, набор безопасных конфигураций, контролирующих к какому стороннему коду BIOS будет применяться ручной контроль.

Критические данные трудно точно определить, потому что данные, которые могут быть очень важными для одного устройства могут быть не очень важными для другого. Однако общие характеристики критических данных включают:

- Они должны быть достоверными для надлежащей загрузки и функционирования устройства;
- Они сохраняются между включениями (например, сохраняются в энергонезависимой памяти)
- Они изменяют поведение или функции устройства
- Они должны быть достоверными, чтобы поддерживать защиту, обнаружение и/или восстановление встроенного микропрограммного обеспечения платформы и связанных данных.

Некоторые критические данные жёстко запрограммированы в коде и обновляются только посредством обновления образа встроенного микропрограммного обеспечения. Для целей этого документа жёстко запрограммированные данные считаются частью кода и защищаются согласно руководствам по защите, обнаружению и восстановлению микропрограммного кода.

У устройств платформы часто есть другие данные, которые конфигурируются во время нормального функционирования администраторами платформы, аппаратными средствами, встроенным микропрограммным обеспечением или программным обеспечением. Поскольку повреждение критических данных может повлиять на нормальное или безопасное функционирование системы, важно защитить критические данные от повреждения и быть в состоянии восстановить, когда обнаружены проблемы. Однако, надежная защита некоторых форм критических данных может быть архитектурно трудной, вследствие предположения о том, что у некоторых сущностей, таких как операционные системы и драйверы устройств, есть доступ, чтобы изменить эти настройки.

Некоторые данные конфигурации могут быть изменены только через определенные интерфейсы, контролируемые кодом уровня платформы. Например, в эту категорию попадают динамические переменные UEFI. Такой базовый уровень защиты ограждает от атакующих, непосредственно изменяющих данные конфигурации, и позволяет встроенному микропрограммному обеспечению платформы удостоверять входные данные прежде, чем передать изменения на хранение. Однако сущности могут быть в состоянии использовать эти определенные интерфейсы, чтобы сделать правильно построенные, но злонамеренные изменения конфигурации.

Чтобы принять меры против такого вмешательства, изменения к некоторым особенно чувствительным данным конфигурации могут требовать санкционирования, прежде чем будут использоваться определенные интерфейсы, описанные выше. В некоторых случаях, устройства платформы, такие как загрузочное встроенное микропрограммное обеспечение главного процессора или встроенное микропрограммное обеспечение служебного процессора, могут быть способны аутентифицировать администраторов платформы до разрешения им внести изменения. Другие опознавательные технологии могут позволить встроенному микропрограммному обеспечению платформы криптографически проверять источник и целостность изменений.

Некоторыми критическими данными управляет встроенное микропрограммное обеспечение без программного воздействия через внешние интерфейсы (например, данные о выравнивании степени износа) и их потеря или повреждение могут привести к постоянной потере сервиса устройства. Этот тип данных о состоянии должен быть защищен на высшем уровне и не может записываться от остальной части платформы.

2.2.2.2 Некритические данные

Некритические данные могут использоваться для различных целей, включая:

- **Информационные / UI:** Данные, которые являются просто информационными или используются в качестве части пользовательского интерфейса (UI) для конечного пользователя

  Пример: Имя тега актива “Свойства NIST” показывается во время загрузки

- **Штатные:** Штатные установки, которые не затрагивают целостность платформы

  Примеры: Состояние клавиши Num Lock при начальной загрузке системы; выполнение BIOS быстрой загрузки или стандартной загрузки

Некритические данные не должны быть критичными для безопасной загрузки или функционирования платформы. На практике, все данные, потребляемые встроенным микропрограммным обеспечением платформы, могут быть чувствительным к безопасности, включая некоторые данные, которые непосредственно не влияют на правильное и безопасное функционирование платформы. Ошибки или вредоносные атаки на любые данные, используемые встроенным микропрограммным обеспечением платформы, могут выявлять и эксплуатировать уязвимости в его коде. По сути, особое внимание должно быть уделено любому не доверенному входу или данным, используемым платформой.

3 Принципы и ключевые концепции

Этот раздел предоставляет краткое описание ключевых принципов отказоустойчивости платформы, которые обеспечивают основу руководств в этом документе. Он также обсуждает главные архитектурные принципы и рассмотрения, используемые в документе.

3.1 Принципы поддержания отказоустойчивости платформы

Руководства безопасности в этом документе основаны на следующих трех принципах:

- **Защита:**

  Механизмы для обеспечения того, чтобы код встроенного микропрограммного обеспечения платформы и критические данные остались в состоянии целостности и были защищены от повреждения, такие как процесс обеспечения аутентичности и целостности микропрограммных обновлений.

- **Обнаружение:**

  Механизмы для обнаружения того, что код встроенного микропрограммного обеспечения платформы и критические данные были повреждены или иначе изменены от санкционированного состояния.

- **Восстановление:**

  Механизмы для восстановления кода встроенного микропрограммного обеспечения платформы и критических данных к состоянию целостности, если обнаружено, что некоторый такой микропрограммный код или критические данные были повреждены или, когда требуется восстановление через санкционированный механизм. Восстановление ограничено способностью возвратить микропрограммный код и критические данные.

Технические руководства, представленные в Разделе 4, организованы вокруг этих принципов. Первый принцип, защита, подобен по области и назначению руководствам, представленным в NIST SP 800-147, *Руководства по защите BIOS* [1]. Основной принцип защиты расширен в этом документе, чтобы относиться к более широкому набору встроенного микропрограммного обеспечения и конфигурационных данных платформы.

Хотя механизмы защиты предназначены, чтобы предотвращать разрушительные или вредоносные атаки на встроенное микропрограммное обеспечение платформы и критические данные, эти механизмы, могут быть несовершенными или их невозможно реализовать на всех категориях устройств. Для этих случаев предназначены механизмы обнаружения и восстановления для обнаружения и ликвидации последствий атак, чтобы возвратить нормальное и безопасное функционирование устройства.

3.2 Свойства отказоустойчивости

Технические руководства в этом документе написаны с точки зрения руководств для отдельных устройств платформы, чтобы сделать их широко применимыми ко множеству устройств, платформ и систем. Несмотря на узкую нацеленность на устройства, намерение этого документа состоит в том, чтобы представить руководства, поддерживающие полную отказоустойчивость в системах против разрушительных атак, гарантировав, что базовая платформа отказоустойчива.

Платформы могут быть не в состоянии полностью обеспечить возможности защиты, обнаружения и восстановления по всем устройствам платформы. Потеря функциональности в даже одном устройстве, может быть достаточной, чтобы перевести всю систему в постоянно неоперабельную, если это конкретное устройство играет важную роль в загрузке или функционировании платформы. Для платформы в целом, чтобы требовать отказоустойчивости к разрушительным атакам, набор

устройств платформы, необходимых, чтобы минимально восстановить функционирование системы, и достаточных, чтобы восстановить обоснованную функциональность, должен сам быть отказоустойчивым. Мы называем этот набор устройств *критическими устройствами платформы*. Конкретные свойства отказоустойчивости могут изменяться от платформы к платформе.

Защищенная

Для платформы, которая считается *Защищенной*, все критические устройства платформы должны выполнять руководства защиты, представленные в Разделах 4.1 и 4.2, но могут не полностью обладать возможностями по восстановлению встроенного микропрограммного обеспечения устройств и/или критических данных.

Восстанавливаемая

Для платформы, которая считается *Восстанавливаемой*, все критические устройства платформы должны предоставлять средства обнаружения повреждений, как описано в Разделах 4.1 и 4.3 и предоставлять средства восстановления после этого повреждения в соответствии с руководствами в Разделах 4.1 и 4.4.

Отказоустойчивая

Для платформы, которая считается *Отказоустойчивой*, все критические устройства платформы должны выполнять все руководства в Разделе 4. Некритические устройства также должны отвечать этим требованиям или, по крайней мере, быть разработаны таким образом, что компрометация одного из этих устройств не повлияет на безопасность платформы в целом. Отказоустойчивые платформы должны пытаться предотвращать атаки, способные к разрушению правильного функционирования платформы, а также предоставлять механизмы для обнаружения и восстановления после злонамеренных или случайных проблем, которые происходят.

3.3 Roots of Trust и Chains of Trust

Механизмы защиты, описанные в этом документе, основаны на Roots of Trust (RoT). Root of Trust - элемент, который формирует основание для предоставления одной или более конкретных функций безопасности, таких как измерение, хранение, информирование, восстановление, проверка и обновление. RoT должен быть разработан так, чтобы всегда вести себя ожидаемом образом, потому что его надлежащее функционирование важно для обеспечения его функций безопасности и потому, что его неправильное функционирование не может быть обнаружено. RoT, как правило, просто первый элемент в Chain of Trust (CoT) и может служить основой в такой цепи, чтобы обеспечить более сложную функциональность. Ответственность и возможности RoT могут быть реализованы полностью в RoT или могут быть выполненный делегатом или агентом, порожденным RoT через цепь доверия, связянную с RoT. Например, RoT для восстановления (RTRec), при инициировании, начнет процесс восстановления, запуская другой элемент, который определяет соответствующую последовательность восстановления и начинает цепь последовательных элементов, которые выполняют действия по восстановлению. Рисунок 2 предоставляет высокоуровневое описание того, как доверенные цепи устанавливаются от начального RoT.

Обычно последующие элементы объединяются в поддержании цепи доверия, начатой RoT. Компонентам в цепи доверия дают привилегию, чтобы выполнить функции, критические по безопасности, при выполнении обновлений устройств, которые не доступны менее достоверному программному обеспечению. У RoTs и CoTs могут быть механизмы, чтобы отключить эти привилегии, как только функция безопасности выполнена, или если установлено, что функция безопасности не требуется. CoT может также отключить привилегии перед передачей управления к необъединённому элементу.

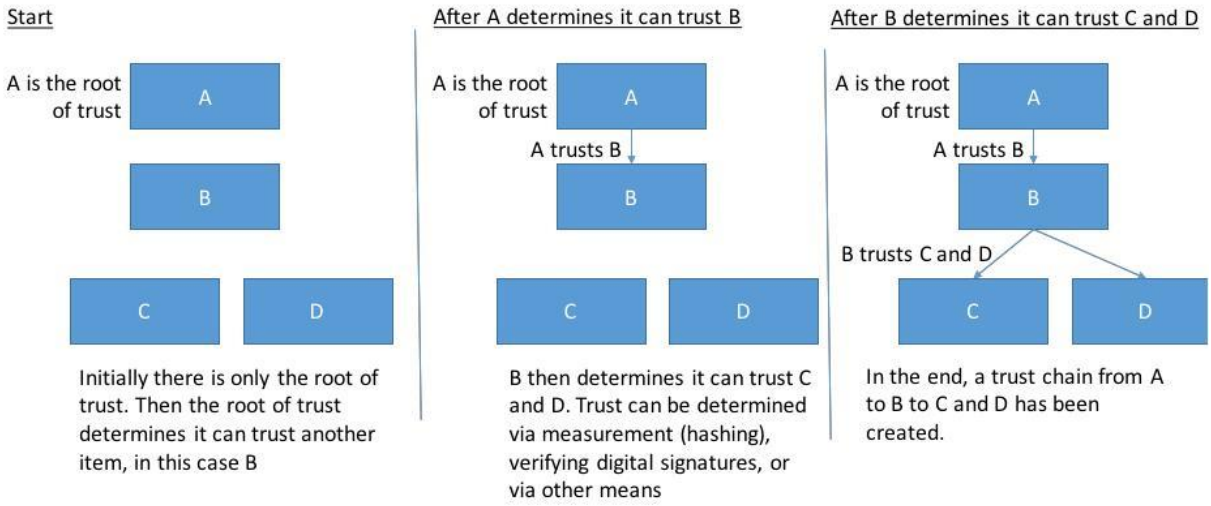

Рисунок 2: Roots of Trust

Поскольку RoTs важны для обеспечения критических функций безопасности, они должны безопасно проектироваться. Главными рассмотрениями для определения доверительности RoTs являются анализ области атак RoT, и оценка смягчения последствий, используемого для защиты от этой области атак. Ответственность обеспечения доверенности RoT находится на вендоре, который предоставляет Root of Trust. Вендоры, как правило, защищают RoTs или делая их неизменными, или гарантируя, что целостность и аутентичность любых изменений в RoTs проверены до выполнения таких обновлений. Часто RoTs работают в изолированных средах на более высоком уровне полномочий, чем что-нибудь, что может изменить его и/или завершить его функционирование, прежде чем что-либо сможет изменить его, чтобы гарантировать, что другие устройства не могут ставить под угрозу его поведение во время функционирования.

Раздел 4.1 этого документа предоставляет конкретные руководства по возможностям и свойствам RoT которые поддерживают отказоустойчивость платформы.

Платформы часто состоят из многочисленных устройств, часто с границами изоляции между устройствами и различных производителей. Платформе могут требоваться множество независимых RoTs и CoTs, чтобы обеспечить всесторонний охват отказоустойчивости. Например, у контроллера жестких дисков может быть различный с серверной платформой микроконтроллер и встроенное микропрограммное обеспечение. И контроллеру жестких дисков и серверной платформе, возможно, понадобятся их собственные независимые цепи доверия для восстановления, если их собственные критические данные будут повреждены.

3.4 Отношения устройства

Вследствие отсутствия возможностей или функциональности, у некоторых устройств платформы может не быть своего собственного root(s) of trust, чтобы выполнять обновление, обнаружение или восстановление. Мы именуем устройства, нуждающиеся в помощи, как симбионт устройства, а те, которые предоставляют помощь, как хост-устройства. Может быть установлена зависимость, каким образом хост-устройство и устройство симбионт совместно выполняют руководства по защите, обнаружению и/или восстановлению, которые устройство симбионт не может выполнить независимо. Такие зависимости могли бы усилить безопасный канал связи или другие технологии. Чтобы быть эффективным при предоставлении помощи, хост-устройство должно само выполнять руководства для механизмов его помощи, передаваемых устройству симбионту. Вместе, хост и

устройство симбионт предоставляют CoT, который реализует руководства безопасности для защиты, обнаружения и/или восстановления.

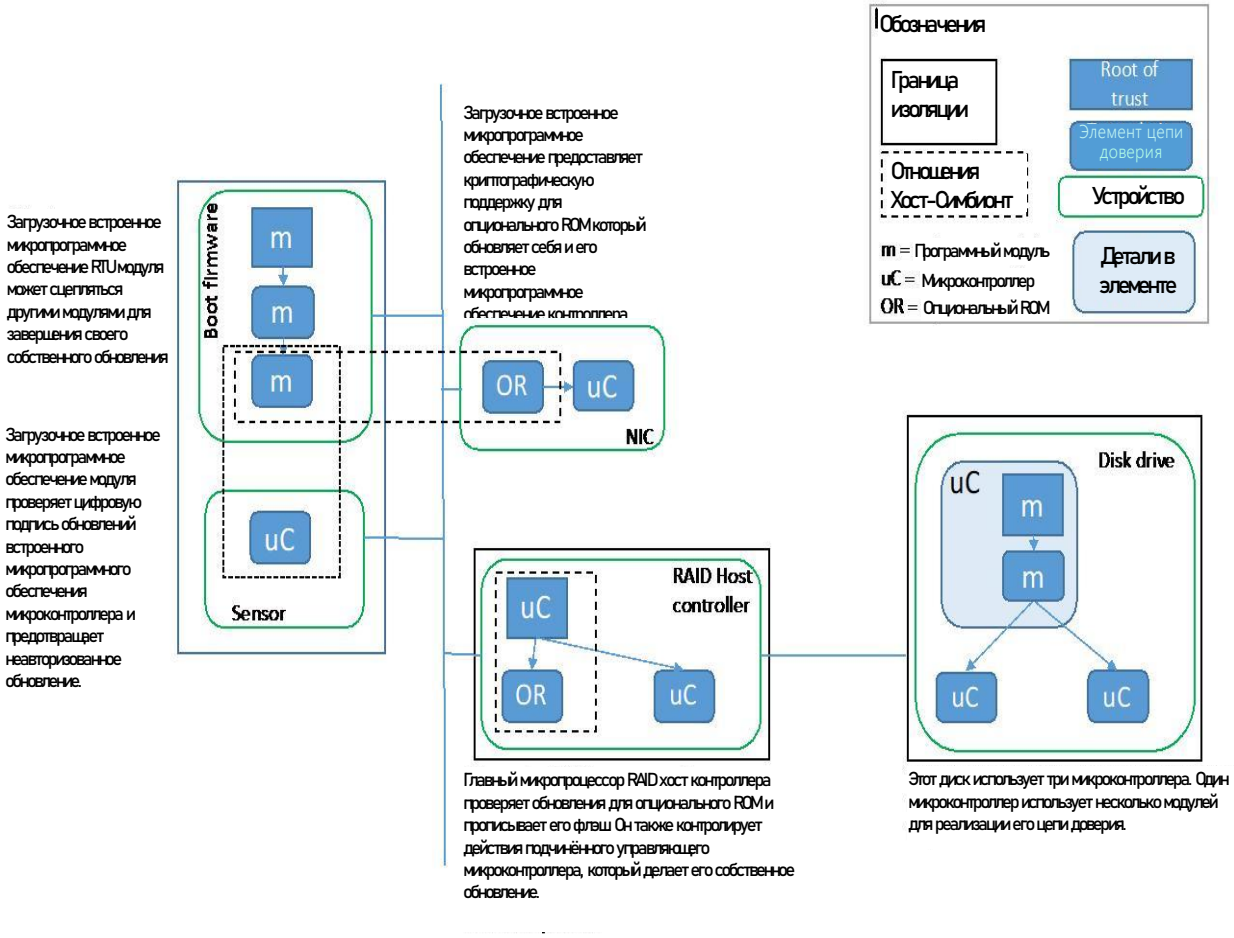

Рисунок 3: Цепи доверия

Могут быть отношения между устройствами, где доверие неявно — то есть, где доверие предоставляется архитектурой системы. Устройство может получить признак однозначного физического присутствия от устройства, когда отношения полного доверия уже существуют. То, что другое устройство послало сообщение через доверенный путь, означает, что устройство может доверять запросу.

Диаграмма на рисунке 3 показывает различные аспекты отношений между симбионтом и хостом; эти отношения могут быть в границах изоляции или за границами изоляции вне устройств. Она также показывает, как различные устройства сосуществуют вместе с несколькими roots of trust, несколькими chains of trust и каналами связи.

Могут также быть другие отношения, которые не подразумевают требования какого-либо уровня доверия. Считается, что устройство ответственно за получение обновлений. Это устройство может затем размножить эти обновления для других устройств. Так как каждое устройство (или устройство симбионт наряду с его хост-устройством) ответственны за подтверждение его собственных обновлений, нет требования для доверия между устройствами, предоставляющими обновления, и устройствами, которым предоставляются обновления.

3.5 Механизмы обновления встроенного микропрограммного обеспечения

Центральный принцип руководств защиты встроенного микропрограммного обеспечения – гарантия того, что только подлинные и авторизованные образы обновлений встроенного микропрограммного обеспечения, могут быть применены к устройствам платформы. Образ обновления подлинен, если источник (например, устройство, производитель системы или другая санкционированная сущность) и целостность могут быть успешно проверены. Технические процессы по проверке образов до применения обновлений называются *заверенными механизмами обновления*.

Санкционирование же является разрешением выполнить обновление. В то время как аутентификация, как правило, внедряется в устройство или систему производителем, санкционирование выполнить обновления, как правило, внедряется в устройство или систему владельцем.

3.5.1 Механизм заверенного обновления

Механизм заверенного обновления использует цифровые подписи, чтобы гарантировать аутентичность образа обновления встроенного микропрограммного обеспечения. Обновление образа встроенного микропрограммного обеспечения, использующее механизм заверенного обновления, полагается на Root of Trust для обновления (RTU), который содержит алгоритм проверки подписи и базу ключей, которая включает открытый ключ, необходимый для проверки подписи образа обновления встроенного микропрограммного обеспечения. База ключей и алгоритм проверки подписи хранятся защищенным способом в компьютерной системе и модифицируются только посредством использования механизма заверенного обновления или локального безопасного механизма обновления.

База ключей в RTU включает открытый ключ, используемый для проверки подписи [7] образа обновления встроенного микропрограммного обеспечения, или включает хеш [6] открытого ключа, если копия открытого ключа поставляется с образом обновления встроенного микропрограммного обеспечения. В последнем случае механизм обновления хеширует открытый ключ, предоставляемый с образом обновления встроенного микропрограммного обеспечения и гарантирует, что он соответствует хешу, который, представлен в базе ключей перед использованием предоставленного открытого ключа для проверки подписи образа обновления встроенного микропрограммного обеспечения.

Возможно, что закрытый ключ, соответствующий открытому ключу в базе ключей, может стать «скомпрометированным», например, закрытым ключом, украденным и раскрытым. Атакующий, который получает доступ к этому ключу, может подписать недействительное встроенное микропрограммное обеспечение, которое может повредить устройства платформы или внести в платформу вредоносное программное обеспечение. Поэтому, надлежащее использование подписей требует обеспечения того, чтобы восстановиться после компрометации ключей. Множество технологий может быть использовано для восстановления после этих ситуаций. Примеры варьируются от комплексных, включающих иерархии ключей, до более простых, включающих обновление базы ключей, когда восстанавливается (или обновляется) оставшаяся часть образа.

3.5.2 Санкционированный механизм обновления

Система и поддерживающее её программное обеспечение управления и встроенное микропрограммное обеспечение могут предоставить несколько санкционированных механизмов для того, чтобы законно обновить образ встроенного микропрограммного обеспечения. Они включают:

1. **Обновления, инициируемые пользователями**: Вендоры, как правило, снабжают конечных пользователей утилитами, способными к обновлению образа встроенного микропрограммного обеспечения. Выполнять эти обновления можно либо с внешнего носителя информации, либо через утилиты, которые могут обновлять образ встроенного микропрограммного обеспечения из операционной системы пользователя. В зависимости от механизмов защиты, реализованных в системе, эти утилиты могут или непосредственно обновлять образ встроенного

микропрограммного обеспечения, или они могут оставлять обновление до следующей перезагрузки системы. Обновленный код будет встречаться с критическими данными, написанными при различных пересмотрах кода. Обновленный код должен гарантировать, что платформа продолжает функционировать, оставаясь совместимой с критическими данными, обновляя критические данные для совместимости с обновленным кодом, или, по крайней мере, перезагружая критические данные к значениям по умолчанию.

2. **Управляемые обновления**: Компьютерная система может иметь аппаратные и программные агенты, которые позволяют системному администратору удаленно обновлять образ встроенного микропрограммного обеспечения без непосредственного участия пользователя.
3. **Откат**: Реализации, которые подтверждают подлинность обновлений прежде чем применить их, могут также проверять номера версий во время процесса обновления. В этих случаях у образа встроенного микропрограммного обеспечения может быть специальный процесс обновления для того, чтобы откатить установленное встроенное микропрограммное обеспечение до прежнего уровня к более ранней версии. Например, процесс отката может требовать физического присутствия пользователя. Этот механизм защищает от атакующих, устанавливающих старое встроенное микропрограммное обеспечение с известными уязвимостями.
4. **Ручное восстановление**: Для восстановления испорченного или неисправного встроенного микропрограммного обеспечения, компьютерные системы могут предоставлять механизмы, позволяющие пользователю при физическом присутствии во время процесса загрузки заменять образ встроенного микропрограммного обеспечения известной хорошей версией и конфигурацией.
5. **Автоматическое восстановление**: Некоторые компьютерные системы в состоянии обнаружить, когда образ встроенного микропрограммного обеспечения был поврежден и восстановить из резервного образа встроенного микропрограммного обеспечения, сохраненного в месте, отличающегося от места поврежденного образа (например, вторая карта флеш-памяти, защищенная область устройства хранения данных).

3.5.3 Безопасное локальное обновление

Хотя этот документ рекомендует, чтобы обновления встроенного микропрограммного обеспечения делались через заверенный механизм обновления, как описано в Разделе 3.5.1, некоторые устройства также могут поддерживать безопасный локальный механизм обновления. Эти механизмы авторизуют обновление встроенного микропрограммного обеспечения посредством процесса, демонстрирующего однозначное физическое присутствие. Безопасный локальный механизм обновления может использоваться, например, чтобы возвратить поврежденный образ встроенного микропрограммного обеспечения, который не может быть обновлен, используя заверенное обновление или автоматизированный механизм восстановления. Безопасный локальный механизм обновления может также использоваться физически существующим администратором, чтобы обновить до более раннего образа встроенного микропрограммного обеспечения устройство, которое не позволяет откат.

Чтобы защитить от удаленных атак, использующих безопасные локальные механизмы обновления, важно, чтобы эти механизмы проверяли, что пользователь физически авторизовал обновление. Удаленные механизмы, такие как взаимодействие с устройством или системой через удаленный терминал, не удовлетворяют этому требованию по физическому присутствию. Точно так же механизмы, которые могут быть обмануты вредоносным программным обеспечением, работающим на системе или устройстве, не удовлетворяют это требование. Дополнительную информацию см. в Разделе 3.6.2.

Обратите внимание на то, что устройства, которые реализуют безопасный локальный механизм обновления, потенциально уязвимы для нападений администраторов нарушителей или других атак с физическим доступом к устройству или системе. Дополнительные физические, внешние и технические меры безопасности важны для защиты этих устройств, но они выходят за рамки этого документа.

3.6 Другие рассмотрения для отказоустойчивости платформы

Этот документ не учитывает некоторые другие рассмотрения, когда покупатель, пользователь или системный администратор могут иметь отношение к отказоустойчивости кибер платформы. Далее следует неисчерпывающий перечень и обсуждение этих других рассмотрений.

3.6.1 Управление

Проектируя отказоустойчивые платформы, продавцы должны тщательно рассматривать своих целевых клиентов, чтобы гарантировать, что надлежащее управление и контроль политик, и параметры конфигурации можно администрировать способом, который лучше всего удовлетворяет потребности пользователей. Управление политиками и параметрами конфигурации может быть выполнено или локально, или удаленно. В зависимости от типа платформы клиенты могут ожидать возможность полностью безопасно управлять платформой из удаленного местоположения. Некоторые пользователи могут требовать, чтобы изменение в политике одобрял физически присутствующий пользователь. Другие клиенты могут хотеть быть в состоянии удаленно извлечь любые данные журналов, или они могут хотеть предотвращать экс-фильтрацию данных журналов кроме как через санкционированные локальные механизмы.

3.6.2 Механизмы авторизации

Некоторые действия по восстановлению и администрированию могут внести существенные изменения или во встроенное микропрограммное обеспечение платформы или в программное обеспечение. Например, установки встроенного микропрограммного обеспечения могут контролировать порядок загрузки, а агент восстановления программного обеспечения может восстановить резервную копию, стирающую недавно созданные данные. Изменение этих установок может требовать авторизации уровня платформы, чтобы демонстрировать, что сущность, требующая изменения, уполномочена это сделать. Для некоторых сред, таких как крупные организации или дата-центры, профессиональный администратор платформы может авторизовать действия, удаленно используя учётные данные, предназначенные для управления платформой. В других средах, например, индивидуальных или малых предприятиях, может не быть удаленного администратора платформы. У некоторых систем, однако, могут быть учетные данные администратора платформы, которые могут использоваться локально. Кроме того, некоторые системы могут позволять пользователям утверждать санкционирование уровня платформы, гарантировав, что даёт команду или требует изменения физически присутствующий пользователь. На этих системах платформа должна однозначно проверить, что действие авторизовал физически присутствующий пользователь. Если всё сделано правильно, вредоносное программное обеспечение не сможет выдать от себя проверку санкционирования, которая включает подтверждение от физически присутствующего пользователя. Мы используем термин *однозначное физическое присутствие*, чтобы указать на локального пользователя, который не может быть сфальсифицирован вредоносным программным обеспечением.

Однозначное физическое присутствие допускает для утверждения авторизацию уровня платформы (или частичную авторизацию уровня платформы), демонстрирующую, что человек физически взаимодействует с устройством или платформой. Гарантируя, что действия по восстановлению или изменению критических данных авторизованы физически присутствующим человеком, однозначное физическое присутствие предоставляет путь управления, который предназначен, чтобы быть защищенным от влияния вредоносного программного обеспечения.

Создать платформы и устройства, которые правильно и достоверно проверяют подтверждение физически присутствующего человека сложно. Специальные физические кнопки или аппаратные перемычки могут предоставить относительно прямой и явный метод, которым можно продемонстрировать физическое присутствие. Конструктивные соображения платформы или рекомендации по развертыванию могут препятствовать тому, чтобы человек имел прямые физические механизмы, чтобы взаимодействовать с каждым устройством, поддерживающим функцию, которая полагается на однозначное физическое присутствие. В этих случаях должен быть доверенный путь между механизмами,

использующими для проверки однозначное физическое присутствие и устройством, которое выполнит действие от имени администратора. Чтобы обеспечить невозможность обхода руководств, найденных далее в этом документе, этот доверенный путь, который может включать устройства ввода-вывода (например, устройство интерфейса пользователя, видеокарты, и т.д.) и внутренние шины, должен быть защищен от манипуляции вредоносным программным обеспечением.

Есть различные технологии, которые могут предоставить доверенный путь между физическим механизмом, который проверяет однозначное физическое присутствие и платформой или устройством. Одним примером может быть утверждение или подтверждение команд от физически присутствующего человека только когда может быть доверие, что платформа находилась в целостном состоянии, прежде чем вредоносное программное обеспечение могло нарушить эти процессы, например, в начале процесса загрузки. В других случаях архитектуры системы могут предоставить доверенные пути между служебным процессором (например, EC, BMC) и другими устройствами платформы.

Устройства, которые полагаются на однозначное физическое присутствие вместо учетных данных администратора платформы, чтобы авторизовать административные действия, могут быть уязвимы для нападений людей с физическим доступом к устройству. По сути, его использование может не подойти в приложениях или средах, которые имеют недостаточно стойкую физическую безопасность.

3.6.3 Сетевое восстановление по сравнению с локальным

В большинстве случаев возможность локально восстановиться после повреждения будет самой целесообразной, предоставляя потребителю высший уровень удовлетворения, и может быть необходима, если нет сетевого соединения. Но признано, что это не всегда возможно, особенно при ограниченной памяти, которую имеют многие устройства. В случаях, когда локальное восстановление невозможно, может осуществляться сетевое восстановление, если делается в безопасным и доверенным способом, который может включать использование шифрования, цифровых подписей, безопасных методов передачи и т.д. Хотя и локальное и сетевое восстановление - приемлемые механизмы реализации, рекомендуется поддерживать для устройства возможность обоих механизмов, что предусматривает однородный верхний уровень отказоустойчивости.

3.6.4 Автоматизированное восстановление по сравнению ручным

Восстановление может проводиться одним из трех способов:

1. ***Полностью автоматизированное*** - не требует взаимодействия с пользователем для начала восстановления или во время процесса восстановления
2. ***Частично автоматизированное*** - восстановление, начинающееся автоматически, но требующее взаимодействия с пользователем в какой-то момент во время процесса восстановления
3. ***Ручное*** - взаимодействие с пользователем требуется для начала восстановления

Полностью автоматизированные механизмы восстановления, могут предпочитаться некоторыми пользователями, поскольку они могут допускать более быстрое восстановление в случае широко распространенной атаки.

Полностью автоматизированное восстановление не может быть поддержано всеми системами или желаемо всеми пользователями. Например, системы могут требовать административных учётных данных или санкционирования для продолжения процесса восстановления.

Ручное восстановление, может предпочитаться некоторыми пользователями потому, что сначала надо сообщить администратору платформы, что что-то ненормально, а затем ждать этого администратора, чтобы решить, какие шаги нужно сделать далее. Это может быть также полезно, если администратор платформы хочет получить информацию, чтобы помочь со следственным анализом.

Поведение и привилегии, требуемые для ручного восстановления, определяются, как правило, политиками, определяемыми администратором. Такие политики могут затронуть также автоматическое восстановление. Например, определенная администраторами политика может ограничить версии встроенного микропрограммного обеспечения, которые могут быть установлены во время восстановления. Те, кто устанавливает политики восстановления, должны делать это с осторожностью. Набор версий встроенного микропрограммного обеспечения в политике вполне может быть версией, которая была успешно восстановлена, подвергнувшись нападению. Просто переписывание уязвимой версии может привести к циклу атака/восстановления.

Сама политика может быть объектом атаки, поэтому проект внедрения системы восстановления должен обеспечивать возможность того, что политика недоступна. Кроме того, так как восстановление может быть многоступенчатым процессом, требование политики, которое должно быть выполнено в конце процесса восстановления, не должно встречаться во время промежуточных шагов.

Системы восстановления, которые переписывают образы встроенного микропрограммного обеспечения, могут в процессе разрушать свидетельства, которые будут полезны в анализе атаки. Системы восстановления, где это практично, должны предоставить методы, чтобы сохранить или записать подвергшихся нападению образы и другую информацию в случаях, когда восстановление встроенного микропрограммного обеспечения может потерять эту информацию.

3.6.5 Регистрация событий

Регистрация событий, связанных со встроенным микропрограммным обеспечением и с восстановлением часто может быть полезна для различных целей, включая, но не ограничиваясь:

- Следственный анализ, который позволяет администратору платформы в системе фиксировать информацию, которая, возможно, привела к атаке на платформу или фактической компрометации платформы. Это может быть полезно в определении того, может ли платформа содержать неизвестную уязвимость системы обеспечения безопасности, или в понимании того, может ли быть широко распространенная атака аналогичного характера.
- Предоставление журнала аудита, чтобы знать, когда событие имело место и, если обновление или восстановление были авторизованы, кто авторизовал его и когда.

Платформа и производители устройств должны определить, какой уровень регистрации событий может требоваться для их систем, принимая во внимание среду предполагаемых пользователей для этих систем. События, которые регистрируются, должны записываться способом, который предоставляет доверие к их целостности и допускает безопасное восстановление и передачу зарегистрированных событий. Необходимо соблюдать осторожность, чтобы гарантировать, что доступ к журналу событий контролируется. Лишенный полномочий персонал может использовать анализ данных журнала событий, чтобы расширить область атаки.

4 Руководства по безопасности встроенного микропрограммного обеспечения для устройств платформы

Этот раздел подробно излагает технические руководства по безопасности для устройств платформы для каждого из трех элементов отказоустойчивости: защита, обнаружение и восстановление. Устройства могут реализовать требования одного или более из этих разделов на основе свойств отказоустойчивости встроенного микропрограммного обеспечения, как определено в Разделе 3.2, которые они стремятся поддерживать. Раздел 4.1 предоставляет основополагающие руководства безопасности для Roots of Trust, которые поддерживают эти свойства. Раздел 4.2 предоставляет руководства по безопасности для защиты кода встроенного микропрограммного обеспечения и критических данных. Руководства для механизмов по обнаружению несанкционированных изменений во встроенном микропрограммном обеспечении и данных описаны в Разделе 4.3. Наконец, Раздел 4.4 определяет руководства по безопасности для механизмов встроенного микропрограммного обеспечения и восстановления данных.

Хотя руководства написаны с точки зрения применения к отдельным устройствам, устройство может реализовывать эти руководства при помощи со стороны другого устройства. Функциональность безопасности может быть реализована самим устройством (отдельным) или она может опираться на архитектуру безопасности, посредством чего другое устройство платформы предоставляет некоторые или все функции безопасности для этого устройства. Для уверенности в предоставлении другим устройством необходимой функциональности по безопасности требуются особые доверительные отношения между этими устройствами, как описано в Разделе 3.4. Эти руководства относятся к устройству, полагающемуся на функциональность безопасности от другого устройства, как к *симбионту,* и к устройству, предоставляющему эту функциональность для симбионта как к *хост-устройству*. В этих случаях симбионт и хост вместе формируют Root of Trust или Chain of Trust, ответственные за реализацию функций безопасности. Хост-устройство должно дополнительно отвечать всем требованиям для отдельного устройства.

Слова **shall, should, и may** используются, как определено в RFC 2119 [5].

4.1 Roots of Trust

Этот раздел предоставляет основополагающие руководства для Roots of Trust (RoT) и Chains of Trust (CoT), которые поддерживают последующие руководства для защиты, обнаружения и восстановления. Эти руководства организованы на основе логического компонента, ответственного за каждое из этих свойств безопасности:

- ***Root of Trust для обновления (RTU)*** ответственен за подтверждение обновлений встроенного микропрограммного обеспечения и изменений критических данных для поддержки возможностей платформы по защите.
- ***Root of Trust для обнаружения (RTD)*** ответственен за возможности обнаружения повреждения встроенного микропрограммного обеспечения и критических данных.
- ***Root of Trust для восстановления (RTRec)*** ответственен за восстановление встроенного микропрограммного обеспечения и критических данных, когда обнаружено повреждение, или когда получены инструкции администратора.

Обратите внимание на то, что это логические компоненты, которые не обязательно должны быть явными. Во многих случаях RoTs будет частью низкоуровневого встроенного микропрограммного обеспечения платформы и будет разделять много компонентов один от другого. Кроме того, хотя каждый RoT ответственен за функции, необходимые для поддержания данного свойства отказоустойчивости, в большинстве случаев он не реализует все эти функции в самом RoT. Как описано в Разделе 3.3, большинство этих функций будет осуществлено в CoT, привязанной к RoT. RoT - неотъемлемый доверенный компонент в этой цепи и передаёт доверие другим компонентам безопасным образом.

4.1.1 Roots of Trust (RoT) и Chains of Trust (CoT)

1) Механизмы защиты **должны** быть основаны на Roots of Trust (RoT).
2) Если будет использоваться Chains of Trust (CoT), RoT **должен** служить основой для CoT.
3) Все RoTs и CoTs **должны** быть или неизменяемыми, или защищены с использованием механизмов, которые гарантируют, что все RoTs, и CoTs остаются целостными.
4) Все элементы Chains of Trust для обновления, обнаружения и восстановления в энергонезависимой памяти **должны** быть реализованы во встроенном микропрограммном обеспечении платформы.

   Примечание: *Это руководство, что RoTs и CoTs реализуются как часть встроенного микропрограммного обеспечения платформы, применяется только к элементам, которые реализуют функции отказоустойчивости платформы, описанные в этом документе. Вендоры платформы поощрены поддерживать цепь доверия от загрузочного встроенного микропрограммного обеспечения до операционной системы, чтобы предоставлять отказоустойчивость против различных форм атак.*
5) Функции RoTs или CoTs **должны** быть стойкими к любому вмешательству, предпринятому программным обеспечением, работающим под, или как часть, операционной системы на главном процессоре.
6) Информацию, передаваемую от программного обеспечения на главном процессоре к встроенному микропрограммному обеспечению платформы**, нужно** рассматривать как не доверенную.
7) CoTs **может** быть расширена, чтобы включать элементы, которые не находятся в энергонезависимой памяти. Перед использованием эти элементы **должны** быть криптографически проверены более ранним элементом CoT.
8) RoTs и CoTs, которые пересекают границы устройств, или которые предоставляют услуги устройству симбионту, **должны** использовать безопасный канал связи между устройствами.

4.1.2 Root of Trust для обновления (RTU) и Chain of Trust для обновления (CTU)

1) Каждое устройство платформы с изменяемым встроенным микропрограммным обеспечением **должно** полагаться или на Root of Trust для Обновления (RTU) или на Chain of Trust для Обновления (CTU), которая связана с RTU, чтобы подтвердить подлинность обновлений встроенного микропрограммного обеспечения.
2) Если RTU или CTU изменяемые, то элементы RTU или CTU **должны** обновляться, используя заверенный механизм обновления, без физического вмешательства посредством безопасного локального обновления. Во время такого обновления RTU или CTU **должны** всегда быть функционирующими или восстанавливаться при последующей перезагрузке даже в случае неожиданного, катастрофического события (например, пропадания электроэнергии посреди операции записи флэша).
3) RTU или CTU **должны** включать базу ключей и одобренную реализацию алгоритма цифровой подписи по FIPS 186-4 [7] для проверки цифровой подписи обновлений образов встроенного микропрограммного обеспечения.
4) Если база ключей является обновляемой, то база ключей **должна** обновляться, используя заверенный механизм обновления, без однозначного физического присутствия посредством безопасного локального обновления.

   Примечание: *Обновляемые базы ключей предоставляют средство восстановиться после компрометации ключа подписания, но могут сделать базу ключей устройства более уязвимой для вмешательства. Разработчики, которые используют необновляемую базу ключей, поощрены проектировать механизмы смягчения и*

восстановления, которые учитывают угрозу потенциального раскрытия встроенного микропрограммного обеспечения, подписываемого ключами.

5) Заверенный механизм обновления, закрепленный в RTU**, должен** быть единственным средством для обновления встроенного микропрограммного обеспечения устройств, без однозначного физического присутствия посредством безопасного локального обновления.

4.1.3 Root of Trust для обнаружения (RTD) и Chain of Trust для обнаружения (CTD)

1) Каждое устройство платформы, которое реализует возможность обнаружения**, должно** полагаться или на Root of Trust для обнаружения (RTD) или на Chain of Trust для обнаружения (CTD), которая связана с RTD для его обнаружения.

2) RTD или CTD **должны** включать или иметь доступ к информации, необходимой, чтобы обнаружить повреждение кода и критических данных встроенного микропрограммного обеспечения.

3) Механизмы обнаружения, закрепленные в RTD**, должны** предоставлять возможности обнаружения, определенные в Разделе 4.3.

Примечание: *Этот документ предоставляет минимальные требования для возможностей обнаружения, внедренных в низкоуровневые аппаратных средства и встроенное микропрограммное обеспечение, чтобы предоставить отказоустойчивость против разрушительных атак. Однако, ничто в этом документе не должно быть истолковано как отвергание других возможностей обнаружения, которые имеются вне этой доверенной цепи.*

4.1.4 Root of Trust для восстановления (RTRec) и Chain of Trust для восстановления (CTRec)

1) Каждое устройство платформы, которое реализует возможность восстановления**, должно** полагаться или на Root of Trust для восстановления (RTRec) или на Chain of Trust для восстановления (CTRec), которая связана с RTRec для его восстановления.

2) RTRec или CTRec **должны** выполнять восстановление.

Примечание: *RTR не был выбран в качестве акронима для Root of Trust для восстановления, потому что RTR, как правило, используется, чтобы обозначить Root of Trust для сообщения. По сути, всюду по этому документу мы снимаем неоднозначность RTR при помощи RTRec, чтобы обозначить Root of Trust для восстановления.*

4.2 Защита

Хотя предыдущие публикации учитывали защиту BIOS (например, NIST SP 800-147 [1], NIST SP, 800-147B [2]), между тем остается другое, критическое в отношении безопасности встроенное микропрограммное обеспечение на платформе, которое не было учтено. Оно включает резидентное микропрограммное обеспечение в контроллерах управления, служебных процессорах, устройства хранения данных, сетевых контроллерах и графических процессорах. Защита должна распространяться также на критические данные, связанные с защищаемым встроенным микропрограммным обеспечением, поскольку некоторые из этих данных могут быть вектором атаки, которая может ставить под угрозу целостность платформы.

Все устройства платформы, которые обеспечивают защиту микропрограммного кода и критических данных, должны отвечать следующим требованиям.

4.2.1 Защита и обновление изменяемого кода

Этот раздел представляет руководство для защиты встроенного микропрограммного обеспечения на основе принципов заверенных микропрограммных обновлений, защиты целостности и невозможности обхода механизмов защиты. Механизмы заверения обновлений используют цифровые подписи, чтобы проверить целостность и аутентичность образов обновлений встроенного микропрограммного обеспечения. Защита целостности встроенного микропрограммного обеспечения предотвращает непреднамеренную или злонамеренную модификацию встроенного микропрограммного обеспечения вне процесса заверенного обновления встроенного микропрограммного обеспечения. Заключительный принцип, невозможность обхода, гарантирует, что не существует средств для обхода атакующим защищенных механизмов.

4.2.1.1 Заверенный механизм обновления

Один или более заверенных механизмов обновления, закрепленные в RTU**, должны** быть единственными средствами для обновления встроенного микропрограммного обеспечения устройств, без однозначного физического присутствия посредством безопасного локального обновления, как определено в Разделе 3.5.3. Заверенные механизмы обновления **должны** выполнять следующие руководства по аутентификации:

1) Образы обновления встроенного микропрограммного обеспечения должны быть подписаны, используя одобренный алгоритм цифровой подписи, как определено в FIPS 186-4 [7], *Стандарт цифровой подписи*, со стойкостью по безопасности по крайней мере 112 битов в соответствии с SP 800-57, *Рекомендации по управлению ключами – Часть 1: Основы* [8].
2) Каждый образ обновления встроенного микропрограммного обеспечения должен быть подписан санкционированной сущностью – обычно производителем устройств, производителем платформы или доверенной третьей стороной - в соответствии с SP 800-89, *Рекомендации по получению доверия для приложений цифровой подписи* [9].
3) Цифровая подпись нового или восстанавливаемого образа обновления встроенного микропрограммного обеспечения должна быть проверена RTU или CTU до завершения процесса обновления энергонезависимой памяти. Например, это может быть достигнуто, путём проверки содержания обновления в RAM и затем выполнения обновления активного флэша. В другом примере это может быть также достигнуто, путём загрузки обновления в область флэша, проверки его, а затем выбора этой области флэша как активной области.

4.2.1.2 Защита целостности

Чтобы предотвратить непреднамеренную или злонамеренную модификацию встроенного микропрограммного обеспечения, области энергонезависимой памяти, содержащие встроенное микропрограммное обеспечение устройств, должны быть защищены от таких модификаций за пределами санкционированного механизма обновления.

1) Области флэша, которые содержат встроенное микропрограммное обеспечение устройств**, должны** быть защищены так, чтобы оно было модифицируемым только через заверенный механизм обновления или безопасным локальным механизмом обновления, который гарантирует аутентичность и целостность микропрограммного образа обновления, требуя, чтобы авторизованный пользователь физически коснулся самой системы, чтобы провести обновление.

   Примечание: *чтобы гарантировать, что защита целостности не может быть обойдена, защита целостности всегда должна быть или задействована, или должна быть задействована до выполнения кода за пределами CTU. Механизмы целостности аппаратных средств могут предоставить более высокое доверие, чем программные механизмы или механизмы, основанные на встроенном микропрограммном*

обеспечении. Эти механизмы целостности защиты должны гарантировать, что встроенное микропрограммное обеспечение может быть изменено только как часть заверенного обновления или безопасного локального обновления.

4.2.1.3 Невозможность обхода

Принцип невозможности обхода состоит в том, что у атакующего не должно иметь возможности изменить встроенное микропрограммное обеспечение устройств вне заверенного механизма обновления или, если поддерживается, безопасного локального обновления. Любые намеренные или непреднамеренные механизмы, способные к обходу заверенного механизма обновления, могут создать уязвимость, позволяющую вредоносному программному обеспечению изменить встроенное микропрограммное обеспечение устройств со злонамеренного или недействительного образа. Они могут включать технологические или диагностические интерфейсы, которые позволяют получить доступ к областям флэш, структурные особенности, которые позволяют прямой доступ к памяти или уязвимости низкого уровня, которые позволяют манипуляцию памятью (например, rowhammer атаки).

Чтобы удовлетворить принципу невозможности обхода, эти потенциальные уязвимости нужно рассматривать везде при проектировании системы. Это может включать усилия по ограничению поверхности атаки устройств, тщательный анализ интерфейсов устройств и нестандартных наборов команд, и блокировку технологических и диагностических интерфейсов при производстве устройств.

1) Механизмы защиты должны гарантировать, что механизмы заверения обновлений невозможно обойти.
2) Заверенный механизм обновления должен быть способен к предотвращению несанкционированных обновлений встроенного микропрограммного обеспечения устройств к более ранней подлинной версии, которая имеет слабость безопасности, или обновлений версией с известной слабостью безопасности.

   ***Примечание**: Обновления до более ранних версий микропрограммного обеспечения, иногда названные «откатом», могут предоставить средство восстановиться после обновления встроенного микропрограммного обеспечения, которое функционирует неправильно. Однако несанкционированный откат может позволить атакующему восстановить уязвимый микропрограммный образ, который, в свою очередь, может позволить атакующему повредить устройство или ввести вредоносное программное обеспечение. По сути, устройства, которые поддерживают откат, должны включать соответствующие меры безопасности, чтобы гарантировать, что они не могут эксплуатироваться несанкционированной сущностью при атаке.*

4.2.2 Защита неизменяемого кода

Код может храниться в необновляемой памяти, такой как Постоянная память (ROM). Хотя защита для этого типа памяти стойкая, компромиссом является невозможность обновить код, чтобы исправить ошибки и уязвимости. Производители систем и устройств должны тщательно взвесить преимущества и недостатки использования энергонезависимой памяти, которая является не обновляемой.

1) Если это используется, то защита от записи необновляемой памяти **не должна** быть модифицируемой.

4.2.3 Защита критического встроенного микропрограммного обеспечения платформы при исполнении

Для удовлетворения принципу невозможности обхода, описанному в Разделе 4.2.1.3, важно, чтобы программное или аппаратное обеспечение контроллера шины, управляемые программным обеспечением, не были способны к вмешательству в целевую функцию *Критического встроенного микропрограммного обеспечения платформы*. Критическое встроенное микропрограммное обеспечение платформы - набор всего встроенного микропрограммного обеспечения платформы, которое либо (а) выполняет функции защиты, обнаружения, восстановления и обновления любого встроенного

микропрограммного обеспечения платформы, (b) поддерживают безопасность критических данных или (c) реализует интерфейсы для критических данных, которые не должны быть обойдены.

Устройства, которые должны соответствовать требованиям по защите, и полагаться на критическое встроенное микропрограммное обеспечение платформы, чтобы защитить образ встроенного микропрограммного обеспечения и/или критические данные в среде выполнения ОС, должны выполнять руководства этого подраздела. Цель этих руководств состоит в том, чтобы установить окружающую среду для критического встроенного микропрограммного обеспечения платформы, при выполнении в которой оно изолировано (защищено) от программного обеспечения. Такая изоляция (защита) может быть обеспечена или логически (например, использование System Management Mode в платформах на базе x86 или TrustZone в платформах, базирующихся на ARM), или физически (например, в RAM, приложенной к не главному процессору, которая физически или логически изолирована от главного процессора).

Этот подраздел не обязательно относится к встроенному микропрограммному обеспечению, которое классифицировано как некритическое (например, большинство BIOS на платформе типа PC, как правило, некритические).

1) Если код Критического встроенного микропрограммного обеспечения платформы в энергонезависимой памяти будет скопирован в RAM чтобы выполняться (для производительности, или по другим причинам) тогда программа встроенного микропрограммного обеспечения в RAM **должна** быть защищена от модификации программным обеспечением или **должна** закончить свою функцию, прежде чем программное обеспечение стартует.

2) Если Критическое встроенное микропрограммное обеспечение платформы использует RAM для временного хранения данных, то эта память **должна** быть защищена от программного обеспечения, работающего на Платформе, до завершения использования данных.

3) Программное обеспечение **не должно** быть в состоянии вмешаться в целевую функцию Критического встроенного микропрограммного обеспечения платформы. Например, запрещая выполнение, изменяя режим процессора или засоряя кэш.

   Примечание: *Эти руководства не препятствуют использованию RAM, которая перезаписываема программным обеспечением конкретно для связи со встроенным микропрограммным обеспечением или оборудованием устройства, включая использование памяти как промежуточной области для обновлений. Руководства предназначены, чтобы предотвращать несанкционированную модификацию выполняемого кода или приватного состояния, используемого Критическим встроенным микропрограммным обеспечением платформы.*

4.2.4 Защита критических данных

Несанкционированные изменения критических данных, хранимых и используемых устройствами, могут также серьезно повлиять на состояние безопасности устройств. Такие изменения могут изменять или отключать связанные с безопасностью важные функции, предоставляемые платформой, или препятствовать тому, чтобы устройство вообще функционировало. Хотя критические данные, возможно, должны быть модифицируемыми операционными системами и другими компонентами, руководства в этом подразделе стремятся предоставлять контролируемый интерфейс для этих изменений и принимать меры против изменений, которые перевели бы устройство в недействующее состояние.

1) Критические данные **должны** быть модифицируемы только через само устройство или определенные интерфейсы, предоставляемые встроенным микропрограммным обеспечением устройств. Примеры определенных интерфейсов включают собственные или общие прикладные программные интерфейсы (APIs), используемые встроенным микропрограммным обеспечением устройства, или интерфейсы, основанные на стандартах. Устройства симбионта могут полагаться на свои хост-устройства, чтобы отвечать этому требованию.

2) Обновления критических данных должны быть подтверждены или устройством, или хост-устройством симбионта до передачи изменений к критическим данным, чтобы гарантировать, что новые данные согласованы. Примеры подтверждения соответствия могут включать проверку размера или границ, проверку формата, и т.д.

3) Обновления критических данных должны быть авторизованы Администратором платформы или частью санкционированного механизма обновления встроенного микропрограммного обеспечения.

4) Обновления критических данных могут использовать механизмы, чтобы подтвердить подлинность критических данных, прежде чем они будут использоваться.

5) Устройство должно, по крайней мере, защищать свои заводские настройки, а также защищать свой код. Заводские настройки должны иметь возможность обновляться также как код.

4.3 Обнаружение

Руководства по обнаружению в этом разделе описывают механизмы, которые могут обнаружить несанкционированные изменения во встроенном микропрограммном обеспечении устройств и критических данных, прежде чем они будут выполняться или потребляться устройством. Когда несанкционированные изменения обнаружены, устройство может начать процесс восстановления, как описано в Разделе 4.4. Механизмы обнаружения особенно важны для устройств, которые испытывают недостаток в надежной защите их встроенного микропрограммного обеспечения или критических данных. Однако эти механизмы могут также предоставить средство обнаружить нарушения в защите встроенного микропрограммного обеспечения или критических данных для устройств, которые пытаются реализовать руководства в Разделе 4.2.

Все устройства, которые предоставляют обнаружение повреждения кода их встроенного микропрограммного обеспечения и критических данных, должны выполнять следующие руководства.

4.3.1 Обнаружение поврежденного кода

Выполнение неавторизованного или поврежденного встроенного микропрограммного обеспечения на устройстве может повредить устройство, внести вредоносное программное обеспечение в систему, или иначе повлиять на функции безопасности и возможности устройств или окружающих систем. Следующие руководства описывают механизмы проверки целостности встроенного микропрограммного обеспечения в процессе загрузки, используя Root of Trust для обнаружения (RTD), определенный в Разделе 4.1.3. Хотя криптографические проверки целостности самим устройством или хост-устройством предпочтительны, некоторые устройства (например, FPGAs или CPLDs) могут использовать другие механизмы, чтобы обнаружить повреждение в их коде и программируемой логике.

Чтобы эти механизмы обнаружения были эффективными, проект устройства должен гарантировать, что RTD остается доверенным в случае успешной атаки на само встроенное микропрограммное обеспечение.

1) Успешная атака, которая повреждает активные критические данные или образ встроенного микропрограммного обеспечения, или нарушает их механизмы защиты, **не должна** сама по себе приводить к результату в успешной атаке на RTD или информацию, необходимую, чтобы обнаружить повреждение образа встроенного микропрограммного обеспечения.

2) Одна или более следующих технологий **должны** использоваться RTD или CTD, чтобы подтверждать код встроенного микропрограммного обеспечения:

   a) Проверка целостности кода встроенного микропрограммного обеспечения устройства с использованием одобренного алгоритма цифровой подписи или криптографического хеша до выполнения кода вне RTD.

Примечание: *проверка целостности может также выполняться в среде выполнения. Эти механизмы могут быть или могут не быть закреплены в RTD.*

b) Устройства симбионта **могут** полагаться на хост-устройство для выполнения обнаружения. Если устройство симбионта загружается независимо от хост-устройства, проверка целостности встроенного микропрограммного обеспечения устройств симбионта **должна** быть выполнена до выполнения кода вне хоста CTD. В таких случаях применяются следующие дополнительные требования:

i) Встроенное микропрограммное обеспечение симбионта **должно** быть защищено согласно требованиям в Разделе 4.2.1.

ii) Хост-устройство **должно** быть способно немедленно инициализировать восстановления встроенного микропрограммного обеспечения симбионта, сопровождаемого перезапуском устройства, в случаях, когда повреждение обнаружено.

c) У некоторых устройств (например, FPGAs, CPLDs) может быть обновляемый на месте логический, а не микропрограммный код, часто называемый потоком битов конфигурации. Если эти устройства не имеют возможности поддержать криптографическую проверку или возможность измерять и сообщать в соответствии с a) или b), они должны использовать основанные на аппаратных средствах механизмы, чтобы обнаруживать отказы загрузочного устройства.

d) Другие технологии (например, сторожевые таймеры) могут использоваться в сочетании с криптографическими проверками целостности, чтобы обнаружить другие проблемы с процессом инициализации устройств платформы.

3) Если повреждение обнаружено, RTD или CTD **должны** быть способны начать процесс восстановления, чтобы вернуть код встроенного микропрограммного обеспечения устройств назад к подлинной версии.

4) Механизм обнаружения должен быть способен создавать уведомления о повреждении.

5) Механизм обнаружения должен быть способен регистрировать события, когда повреждение обнаружено.

6) Механизмы обнаружения могут быть способны использовать политики, установленные Администратором платформы, которые определяют меры, принятые для RTD/CTD в вышеупомянутых руководствах.

4.3.2 Обнаружение поврежденных критических данных

Этот раздел описывает механизмы для обнаружения недопустимых или поврежденных критических данных в устройствах платформы. Как отмечено выше, недопустимые критические данные могут перевести устройство в неоперабельное или отключить некоторую критическую функциональность безопасности. Подтверждение критических данных сложно, потому что данные часто предназначены, чтобы быть конфигурируемыми пользователем. Руководства в этой секции рекомендуют или непосредственно проверять содержание критических данных или реализовывать другие механизмы поиска признаков повреждения данных.

1) RTD или CTD **должны** выполнять проверки целостности критических данных до использования. Проверки целостности могут принимать форму, например, подтверждения данных по известным действительным значениям или подтверждения хеша хранения данных.

2) Любая, как альтернативная проверка целостности (для устройств, которые не могут поддержать такую способность), так и дополнительная к таким проверкам RTD или CTD **может** использовать сторожевые таймеры, чтобы обнаруживать потенциальное повреждение критических данных.

3) Если повреждение критических данных будет обнаружено, RTD или CTD **должны** быть способны начать процесс восстановления, чтобы восстановить критические данные устройства.
4) Механизм обнаружения **должен** быть способен регистрировать события, когда повреждение обнаружено.
5) RTD или CTD **должны** быть способны создавать уведомления о повреждении.
6) RTD или CTD**, могут быть** способны отправлять уведомления о повреждении.

4.4 Восстановление

Эта секция описывает механизмы для восстановления встроенного микропрограммного обеспечения платформы и критических данных к проверенному и санкционированному состоянию в случае, если обнаружены повреждения любого встроенного микропрограммного обеспечения или критических данных, или когда администратор начинает процесс ручного восстановления.

4.4.1 Восстановление изменяемого кода

Руководства по восстановлению встроенного микропрограммного обеспечения в этом разделе определяют механизмы по возвращению встроенного микропрограммного обеспечения к сохраненной локальной резервной копии или к восстановлению образа, загруженного с другого источника. В любом случае эти руководства определяют использование Заверенного механизма обновления (Раздел 4.2.1.1) для проверки целостности и аутентичности образа до восстановления.

1) Механизмы восстановления встроенного микропрограммного обеспечения **должны** сопротивляться атакам, которые повреждают активные критические данные или образ исходного встроенного микропрограммного обеспечения, или нарушают механизмы их защиты.
   a) RTRec, CTRec и подлинный образ восстанавливаемого встроенного микропрограммного обеспечения **должны** быть независимо защищены от исполняемого встроенного микропрограммного обеспечения.
2) RTRec или CTRec **должны** быть способны к получению подлинного образа встроенного микропрограммного обеспечения устройств.
   a) Если подлинный образ встроенного микропрограммного обеспечения будет сохранен локально в энергонезависимой памяти, то образ **должен** быть защищен от несанкционированной модификации.
3) Обновление локально сохраненным подлинным образом встроенного микропрограммного обеспечения **должно** осуществляться посредством Заверенного механизма обновления (Раздел 4.2.1.1) или безопасного локального обновления (Раздел 3.5.3).
4) Нелокальные механизмы восстановления **должны** использовать Заверенный механизм обновления (Раздел 4.2.1.1), чтобы проверить целостность и аутентичность образов восстановления до восстановления ими.
5) Если подлинный образ встроенного микропрограммного обеспечения сохранен удаленно, политики восстановления **должны** быть сконфигурированы с местоположением этого образа.
6) Если устройство (*устройство симбионта*) полагается на другое устройство платформы (*хост-устройство*), для получения его RTRec или CTRec, то RTRec или CTRec хост-устройства **должны** вызывать Заверенный механизм обновления хоста/симбионта во время операций по восстановлению.
7) Устройство **должно** или реализовывать свою собственные возможности по восстановлению, или это устройство (*устройство симбионта*) и другое устройство платформы (*хост-устройство*) **должно** вместе реализовывать возможности по восстановлению устройства симбионта.
8) Механизм восстановления **должен** быть способен к регистрации и сообщению о событиях, когда восстановление выполнено.

9) Механизм восстановления **должен** быть способен к предоставлению уведомлений о событиях и действиях по восстановлению.

10) Механизм восстановления**, может** быть способен к выполнению действия по восстановлению без уведомления или вмешательства пользователя или системного администратора.

11) Механизм восстановления **может** просить санкционирование от пользователя или системного администратора выполнить действие по восстановлению.

12) Администратор платформы **должен** быть в состоянии начать восстановление изменяемого кода. Устройства **должны** предоставить Администраторам платформы способ вызова восстановления. Устройства **могут** получить санкционирование уровня платформы чтобы начать восстановление через цепь одного или более доверенных устройств, первое из которых **должно** проверить санкционирование уровня платформы до того, как указать устройствам на начало восстановления.

13) Процесс восстановления **должен** быть защищен от несанкционированного восстановления к более ранней версии встроенного микропрограммного обеспечения, которая содержит недостаток в безопасности. Полный процесс восстановления **должен** облегчить восстановление к последней версии встроенного микропрограммного обеспечения. Это **может быть** реализовано как многоступенчатый процесс восстановления.

4.4.2 Восстановление критических данных

Этот раздел описывает механизмы восстановления критических данных устройств платформы, если у устройства или администратора есть причина полагать, что они были повреждены. Поскольку критические данные могут быть конфигурируемыми пользователем, восстановление требует доступности доверенных, известных как хорошие резервных копий критических данных. Эти резервные копии могут быть сохранены на самом устройстве или на некотором другом хост-устройстве. Поскольку эти резервные копии могут также быть уязвимы для атаки, это руководство определяет, что устройства также предоставляют средство, чтобы вернутся к известным как хорошие заводским настройкам.

1) Механизмы восстановления критических данных **должны** сопротивляться атакам, повреждающим активные критические данные, или Исходный образ встроенного микропрограммного обеспечения, или нарушают механизмы их защиты.
2) Устройство **должно** предоставить способ сделать копию известной как хорошая копии (или копий) активных критических данных для другого местоположения или местоположений. Защита в этих местоположениях **должна** быть, по крайней мере, такой же хорошей, как для активных критических данных. Защите **следует** быть лучше, чем таковой для активных критических данных.
    a) Устройство симбионта, которое не может сделать копию его собственных критических данных**, должно** сделать свои критические данные доступными для его хост-устройства. В этом случае хост-устройство должно сделать копию данных симбионта.
    b) Если симбионт предоставляет свои критические данные хост-устройству для того, чтобы хост-устройство сделало копию данных, то симбионт **должен** быть в состоянии потреблять восстановленные критические данные.
3) Устройство **может** определять, что его критические данные “известны как хорошие”, используя эти данные как часть успешной перезагрузки.
4) Устройство **должно** делать копию критических данных или автоматически, или по указанию пользователя или по указанию сделать это от другого доверенного устройства.
5) RTRec или CTRec **должны** быть способны к восстановлению критических данных к заводским настройкам.
6) RTRec или CTRec **должны** быть способны к восстановить последние известные как хорошие критические данные.

7) Устройство **не должно** использовать политики, сохраненные в качестве критических данных этим устройством, чтобы восстановить его собственные критические данные. Однако симбионт **может** полагаться на политики, предоставляемые хост-устройством.

8) Если доступны множественные резервные копии, RTRec или CTRec **могут** выбирать, какую резервную копию использовать.

9) Если обнаружение повреждения критических данных автоматическое, RTRec или CTRec **могут** получать санкционирование от хост-устройства или пользователя прежде, чем заменить текущие критические данные.

10) В отсутствие инициализации действий по восстановлению от RTD или CTD, администратор платформы **должен** быть в состоянии начать восстановление критических данных. Устройства **должны** предоставить Администраторам платформы способ для начала восстановления. Устройства, которые получают санкционированные запросы начать восстановление**, могут** затем давать указания другим устройствам, с которыми есть установленные доверительные отношения, чтобы начать восстановление или непосредственно, или через цепочки доверенных устройств.

Хотя это выходит за рамки этого документа, чтобы определить то, что составляет соответствующее состояние для платформы, чтобы восстановиться (кроме состояния целостности), примеры включают любое следующее:

- Восстановление к последнему состоянию, известному как хорошее
- Сброс к заводским настройкам
- Обновление до новейшего образа встроенного микропрограммного обеспечения
- Выполнение частичных 'ремонтных' операций
- Восстановление к определенной предприятием 'начальной точке'
- Любая комбинация вышеупомянутого

Обратите внимание на то, что процессы восстановления могут потребовать нескольких стадий, прежде чем нормальное функционирование будет восстановлено. Например, устройства могут первоначально восстановить данные конфигурации до заводских настроек прежде чем восстанавливать последние, известные как хорошие, данные конфигурации.
При рассмотрении того, как определить, какое из вышеупомянутых состояний целостности требуется, чтобы восстановить устройство, используется основанный на политике подход, требующий использования критических данных, чтобы хранить/поддерживать эти политики. Однако, если эти критические данные становятся поврежденными, то процесс восстановления может или быть не в состоянии выполниться или он может выполняться неправильно. Поэтому, вендоры должны тщательно рассматривать алгоритм, используемый для восстановления. К примеру, простейшим механизмом могло бы быть использование простого алгоритма Most Recently Used (MRU), например, алгоритм мог бы сначала попытаться восстановиться к последнему состоянию, известному как хорошее; если эти данные не валидны, то он мог бы затем попытаться восстановиться к предшествующему сохраненному состоянию; если это не доступно, то он мог бы попытаться восстановиться из удаленного местоположения хранения ресурсов предприятия; если это не доступно, то он мог бы попытаться перезагрузить к заводским настройкам. Использование алгоритмического подхода в этом способе избавляет от необходимости полагаться на критические данные, которые находятся в целостном состоянии во время операций по восстановлению.

Приложение A — акронимы

Отобранные акронимы и сокращения, используемые в этом документе, определены ниже.

BIOS	Базовая система ввода-вывода
CoT	Цепь доверия
CPLD	Сложное программируемое логическое устройство
CTD	Цепь доверия для обнаружения
CTRec	Цепь доверия для восстановления
CTU	Цепь доверия для обновления
FPGA	Программируемая пользователем вентильная матрица
ROM	Постоянная память
RoT	Root of Trust
RTD	Root of Trust для обнаружения
RTRec	Root of Trust для восстановления
RTU	Root of Trust для обновления
UEFI	Унифицированный расширяемый интерфейс встроенного микропрограммного обеспечения

Приложение B — глоссарий

Активные критические данные (Active Critical Data)	Копия критических данных, которая используется, чтобы инициализировать или конфигурировать устройство. *Примечание: активные критические данные не включают резервные копии критических данных.*
Плата расширения (Add-in Card)	Общий термин, используемый в отношении к любому устройству, которое может быть добавлено или удалено с платформы через подсоединение к шине, такое как PCI. Платы расширения, как правило, вставляются в корпус платформы, вместо того, чтобы находиться физически вне платформы. У платы расширения могут быть свои собственные устройства и связанное встроенное микропрограммное обеспечение, и может иметься своё собственное Встроенное микропрограммное обеспечение расширения ROM.
Заверенное обновление (Authenticated Update)	Обновление, которое использует заверенный механизм обновления.
Заверенный механизм обновления (Authenticated Update Mechanism)	Механизм обновления, гарантирующий, что обновление образа встроенного микропрограммного обеспечения было подписано в цифровой форме и что цифровая подпись может быть проверена с использованием одного из ключей в базе ключей Root of Trust для обновления (RTU) прежде, чем обновить образ встроенного микропрограммного обеспечения.
Санкционированное обновление (Authorized Update)	Обновление, которое использует санкционированный механизм обновления.
Санкционированный механизм обновления (Authorized Update Mechanism)	Механизм обновления, который проверяет санкционирование перед установкой обновления. Санкционирование может состоять из проверки обладания учётными данными, подтверждения физически присутствующим человеком или подобных средств.
Загрузочное встроенное микропрограммное обеспечение (Boot Firmware)	Общее обозначение для того, чтобы описать любое встроенное микропрограммное обеспечение на платформе, применяемое для загрузки (запуска, инициализации) устройства. Оно может включать, но не ограничено, инициализацию памяти устройства, инициализацию регистров устройства, инициализацию интерфейсов взаимодействия, проверку состояния устройства, и т.д. Общее назначение загрузочного встроенного микропрограммного обеспечения состоит в том, чтобы подготовить устройство или платформу для нормального использования.
Chain of Trust (CoT)	Chain of Trust (CoT) - последовательность совместимых элементов, связанных с Root of Trust (ROT), которые расширяют доверенную границу текущего элемента, передавая те же самые доверенные свойства следующему элементу, когда это переходит под его контроль. Результат - оба элементы одинаково в состоянии выполнить доверенную функцию, как будто они один доверенный элемент. Этот процесс может быть продолжен, дальнейшим распространением цепи доверия. Как только контроль передан коду, который не является, или не может быть проверенным, тогда Chain of Trust заканчивается. Это также называется передачей контроля к несовместимому элементу.
Код (Code)	Инструкции, непосредственно выполняемые процессором или подобным устройством (FPGA, CPLD, и т.д.). Включает также инструкции, которые интерпретируются программой. *Исходный код* - человекочитаемые инструкции, которые транслируются в код и затем выполняются.

Повреждение (Corruption)	Повреждение - потеря целостности кода встроенного микропрограммного обеспечения или ошибка или непредусмотренное значение в критических данных, которые могут быть результатом одной из многих различных причин, включая, но не ограничиваясь, злонамеренные действия (например, атакующего), плохо написанный код (например, переполнение буфера, алгоритмическая ошибка), случайные события (например, неправильное действие пользователя), отказ установить патч безопасности, несанкционированные изменения, или вызванные аппаратными средствами (например, целостность сигнала, альфа-частицы, пропадание электроэнергии).
Критические данные (Critical Data)	Критические данные - изменяемые данные, которые сохраняются при выключении энергии и должны быть в актуальном состоянии, которое авторизовано Администратором платформы, для восстановления и/или загрузки платформы для безопасного и правильного функционирования.
Критическое встроенное микропрограммное обеспечение платформы (Critical Platform Firmware)	Набор встроенного микропрограммного обеспечения платформы, которое либо: (a) выполняет функции защиты, проверки, восстановления и обновления любого встроенного микропрограммного обеспечения платформы, (b) поддерживает безопасность критических данных или (c) реализует интерфейс для критических данных, которые не должны быть обойдены.
Устройство (Device)	Общий термин, используемый в отношении к любому вычислительному или запоминающему элементу или набору вычислительных или запоминающих элементов на платформе. Примеры устройств включают центральный процессор (CPU), блок обработки приложений (APU), встроенный диспетчер (EC), контроллер соединительной платы (BMC), доверенный модуль платформы (TPM), графический процессор (GPU), контроллер сетевого интерфейса (NIC), жесткий диск (HDD), твердотельный диск (SSD), Память (ROM), флэш-ROM, и т.д.
Встроенное микропрограммное обеспечение устройства (Device Firmware)	Набор встроенного микропрограммного обеспечения нехостового процессора и встроенного микропрограммного обеспечения расширения ROM, которое используется только конкретным устройством. Это встроенное микропрограммное обеспечение, как правило, предоставляется производителем устройств.
Встроенное микропрограммное обеспечение расширения ROM (Expansion ROM Firmware)	Взаимосвязь периферийных компонентов (PCI) - термин для встроенного микропрограммного обеспечения, выполняемого на главном процессоре, которое используется дополнительным устройством во время процесса загрузки. Оно включает Встроенное микропрограммное обеспечение опционального ROM, приложения UEFI и драйверы UEFI. Встроенное микропрограммное обеспечение расширения ROM может быть поставлено как часть Загрузочного встроенного микропрограммного обеспечения главного процессора, или может быть отдельным (например, из платы расширения). В этом документе, мы будем использовать термин Встроенное микропрограммное обеспечение расширения ROM в отношении либо к Встроенному микропрограммному обеспечению опционального ROM либо к драйверам и приложениям UEFI в общем.
Встроенное микропрограммное обеспечение (Firmware)	Общий термин для описания любого кода хранимого в чипе, который либо находится в векторе сброса (или эквивалентном) соответствующего процессора или который предоставлен как расширение к другому встроенному микропрограммному обеспечению (такому как Встроенное микропрограммное обеспечение расширения ROM). Операционные системы общего

	назначения, которые хранятся в чипе, в общем не считаются встроенным микропрограммным обеспечением в области этого документа.
Хост-устройство (Host Device)	Устройство, которое помогает устройству симбионта образовывать roots of trust и chains of trust, требуемые для выполнения защиты, обнаружения и/или восстановления руководствами в этом документе. Распределение функциональности между устройствами хоста и симбионта зависит от реализации. Хост-устройство должно непосредственно выполнять руководства для его собственного встроенного микропрограммного обеспечения самостоятельно. Обратите внимание на то, что это нельзя путать с Главным процессором. Главный процессор может служить Хост устройство, но Хост-устройство - не обязательно Главный процессор.
Главный процессор (Host Processor)	Главный процессор - основной блок обработки на платформе, традиционно названный Центральным процессором (CPU), теперь также иногда называемый Application Processing Unit (APU) или System on Chip (SoC). Это - блок обработки на котором работают основная операционная система (и/или гипервизор), а также пользовательские приложения. Это процессор, который ответственен за загрузку и выполнение Загрузочного встроенного микропрограммного обеспечения Главного процессора.
Загрузочное встроенное микропрограммное обеспечение Главного процессора (Host Processor Boot Firmware)	Общий термин, используемый для описания встроенного микропрограммного обеспечения, загружаемого и выполняемого Главным процессором, которое предоставляет основные загрузочные возможности платформе. Этот класс встроенного микропрограммного обеспечения включает Устаревший BIOS, Системный BIOS и UEFI, а также другие реализации. Где различие между Устаревшим BIOS и UEFI не важно, будет использоваться термин Загрузочное встроенное микропрограммное обеспечение Главного процессора. Где различие важно, это будет упомянуто соответственно. Встроенное микропрограммное обеспечение Расширения ROM можно также рассматривать как часть Загрузочного встроенного микропрограммного обеспечения Главного процессора. Встроенное микропрограммное обеспечения расширения ROM, может быть включено как часть Загрузочного встроенного микропрограммного обеспечения Главного процессора, или может быть отделено от Загрузочного встроенного микропрограммного обеспечения Главного процессора (например, загружаться из платы расширения). Загрузочное встроенное микропрограммное обеспечение Главного процессора включает встроенное микропрограммное обеспечение, которое может быть доступным во время среды выполнения.
Постоянный (Immutable)	Неизменный. В контексте этого документа это относится только к невозможности делать изменения на месте через штатные механизмы производителя и/или определенные интерфейсы. Отметьте, что производитель платформы или устройств может, однако, быть в состоянии внести изменения через инструменты производства или сервиса, непосредственно в локально (физически) представленную платформу или устройство.
Устаревший BIOS (Базовая система ввода-вывода) (Legacy BIOS (Basic Input/Output System))	Одна форма Загрузочного встроенного микропрограммного обеспечения Главного процессора, используемая на x86 платформах, которая использует устаревшую структуру x86 BIOS. Эта форма Загрузочного встроенного микропрограммного обеспечения Главного процессора была или заменяется UEFI.

Переменный (Mutable)	Изменяемый. В контексте этого документа это относится только к возможности делать изменения в области через штатные механизмы и/или определенные интерфейсы производителя. Такие механизмы могут требовать криптографических механизмов или конкретного физического присутствия.
Некритические данные (Non-critical Data)	Некритические данные - изменяемые данные, которые сохраняются при выключении, но не очень важны для загрузки, функционирования или восстановления устройства.
Неглавный процессор (Non-Host Processor)	Неглавный процессор - общий термин, используемый, чтобы описать любой блок обработки на платформе, который не является главным процессором (например, микроконтроллер, сопроцессор, и т.п.).
Встроенное микропрограммное обеспечение неглавного процессора (Non-Host Processor Firmware)	Встроенное микропрограммное обеспечение неглавного процессора - общий термин, используемый чтобы описать встроенное микропрограммное обеспечение, используемое любым блоком обработки на платформе, который не является главным процессором.
Встроенное микропрограммное обеспечение опционального ROM (Option ROM Firmware)	Устаревший термин для загрузочного встроенного микропрограммного обеспечения, обычно исполняемого на главном процессоре, которое используется устройством во время процесса загрузки. Встроенное микропрограммное обеспечение опционального ROM может быть включено в загрузочное встроенное микропрограммное обеспечение главного процессора или может содержаться в отдельном устройстве (таком как плата расширения).
Периферия (иначе внешнее устройство) (Peripheral (aka external device))	Периферия (также известная как внешнее устройство) является устройством которое находится физически вне платформы и связано с платформой или по проводам, или беспроводно. Периферия состоит из её собственных устройств, у которых может быть их собственное встроенное микропрограммное обеспечение. Хотя концептуально принципы и руководства в этом документе могут применяться также и к периферии, она выходит за рамки этого документ.
Платформа (Platform)	Платформа состоит из одного или нескольких устройства, собранных и работающих вместе, чтобы предоставить конкретную вычислительную функцию, и не включающая ни какое другое программное обеспечение кроме встроенного микропрограммного обеспечения как части устройств платформы. Примеры платформ включают: ноутбук, настольный компьютер, сервер, сетевой переключатель, лезвие, и т.п.
Привилегии Администратора платформы (Platform Administrator Privilege)	Привилегии, требуемые для управления встроенным микропрограммным обеспечением и критическими данными в устройствах платформы. В частности, эти привилегии могут быть необходимы для авторизации обновлений встроенного микропрограммного обеспечения, изменения его параметров конфигурации и операций по его восстановлению.
Администратор платформы (Platform Administrator)	Сущность с правами администратора платформы.
Встроенное микропрограммное обеспечение	Набор встроенного микропрограммного обеспечения всех устройств на платформе. В этом документ, термин встроенное микропрограммное обеспечение платформы может использоваться для указания на набор

платформы (Platform Firmware)	всего встроенного микропрограммного обеспечения, используемого устройствами на платформе.
Исходный образ встроенного микропрограммного обеспечения (Primary Firmware Image)	Исполняемый код, хранимый в устройстве. Различные части исходного образ встроенного микропрограммного обеспечения могут защищаться по-разному.
Память только для чтения (ROM) (Read Only Memory (ROM))	Запоминающее устройство, которое является таким, как было первоначально установлено, и не может быть переписано через любой механизм, делая эту память постоянной (неизменной).
Отказоустойчивость (Resilient)	Способность системы с конкретными характеристиками до, в период и после воздействия поглощать воздействие, восстанавливаться к допустимому уровню работы, и поддерживать этот уровень на приемлемом промежутке времени.
Root of Trust (RoT)	Элемент, который формирует базис для обеспечения одной или более функции безопасности, таких как измерение, хранение, сообщение, восстановление, проверка, обновление, и т.д. RoT доверен, чтобы всегда вести себя ожидаемым образом, потому что его неправильное поведение не может быть обнаружено, и потому что его надлежащее функционирование важно для обеспечения его конкретные безопасностью функции.
Исполняемое Встроенное микропрограммное обеспечение (Runtime Firmware)	Общий термин для описания любого встроенного микропрограммного обеспечения на платформе активного/функционирующего или доступного для использования во время периода выполнения (после того, как загрузка завершена).
Программное обеспечение (Software)	Программное обеспечение состоит из элементов, требуемых для загрузки операционной системы, операционной системы и всех пользовательских приложений и пользовательских данных, впоследствии обрабатываемых операционной системой. Обратитесь к рисунку 1 для графического представления.
Устройство симбионт (Symbiont Device)	Устройство симбионт - устройство, которое полагается полностью или частично на другое устройство (хост-устройство), чтобы установить roots and chains of trust, требуемые руководствами в этом документе для обеспечения защиты, обнаружения и восстановления. Распределение функциональности между хост-устройством и устройством симбионт зависит от реализации. Например, хост-устройство проверяет обновления и поддерживает критические данные в то время как устройство симбионт ответственно за выполнение всех других руководств. Свойство симбионта может быть переходным: устройство может будьте устройством симбионт к хост-устройству, в то время как эти два могут служить как хост для другого устройства симбионт и так далее.

Система (System)	Система - совокупная вычислительная сущность, включающая все элементы на платформе (аппаратные средства, встроенное микропрограммное обеспечение) и программное обеспечение (операционная система, пользовательские приложения, пользовательские данные). Система может представляться как логическая конструкция (например, программный стек) или физическая конструкция (например, ноутбук, настольный компьютер, сервер, сетевой переключатель, и т.д.). Обратитесь к рисунку 1 для графического описания.
UEFI (Унифицированный расширяемый микропрограммный интерфейс) (UEFI (Unified Extensible Firmware Interface))	Одна форма Загрузочного встроенного микропрограммного обеспечения Главного процессора, которая использует структуру Унифицированного расширяемого микропрограммного интерфейса (UEFI) (как определено Комиссией UEFI).
Драйверы UEFI (UEFI Drivers)	Автономные бинарные исполняемые файлы, которые загружаются во время процесса загрузки, чтобы взаимодействовать с конкретными частями аппаратных средств.
Непосредственное физическое присутствие (Unambiguous Physical Presence)	Указывает на санкционирование местным человеком, который не может быть выдан за вредоносное программное обеспечение.

Приложение C — Ссылки

[1] D D. Cooper, W. Polk, A. Regenscheid, and M. Souppaya, *Руководства по защите BIOS,* Специальная публикация (SP) NIST 800-147, Национальный институт стандартов и технологий, Gaithersburg, Maryland, апрель 2011, 26pp. https://doi.org/10.6028/NIST.SP.800-147

[2] A. Regenscheid., *Руководства по защите BIOS для серверов,* Специальная публикация (SP) NIST 800-147B, Национальный институт стандартов и технологий, Gaithersburg, Maryland, август 2014, 32pp. https://doi.org/10.6028/NIST.SP.800-147B

[3] Спецификации, Комиссия по Унифицированному расширяемому микропрограммному интерфейсу [веб-сайт], http://www.uefi.org/specifications [доступен с 5/2/18]

[4] Спецификация библиотеки TPM, Trusted Computing Group [веб-сайт], https://trustedcomputinggroup.org/tpm-library-specification/[доступен с 5/2/18]

[5] S. Bradner, *Ключевые слова для использования в RFCs, чтобы указать на уровни требований*, RFC 2119, Специальная комиссия по интернет-разработкам, март 1997, 2pp, https://doi.org/10.17487/RFC2119

[6] Американское Министерство торговли. *Стандарт безопасного хеша*, Федеральные стандарты обработки информации (FIPS) публикация 180-4, август 2015, 36pp. https://doi.org/10.6028/NIST.FIPS.180-4

[7] Американское Министерство торговли. *Стандарт цифровой подписи*, Федеральные стандарты обработки информации (FIPS) публикация 186-4, июль 2013, 130pp. https://doi.org/10.6028/NIST.FIPS.186-4

[8] E. Barker, *Рекомендации по управлению ключами, часть 1: Основы,* Специальная публикация (SP) NIST 800-57 часть 1 пересмотр 4, Национальный институт стандартов и технологий, Gaithersburg, Maryland, январь 2016, 160pp. https://doi.org/10.6028/NIST.SP.800-57pt1r4

[9] E. Barker, *Рекомендации по получению доверия для цифровой подписи Приложений,* Специальная публикация (SP) NIST 800-89, Национальный институт стандартов и технологий, Gaithersburg, Maryland, ноябрь 2006, 38pp. https://doi.org/10.6028/NIST.SP.800-89

[10] Международный совет по проектированию систем, “Устойчивость работы систем Чартер группы”, ноябрь 2011.

[11] R. Ross, R. Graubart, D. Bodeau, and R. McQuaid, *Разработка безопасных систем: Рассмотрения кибер отказоустойчивости для разработки доверенных защищенных систем*, Специальная публикация (SP) NIST 800-160 том 2 (ПРОЕКТ), Национальный институт стандартов и технологий, Gaithersburg, Maryland, март 2018, 158pp. https://csrc.nist.gov/CSRC/media/Publications/sp/800-160/vol-2/draft/documents/sp800-160-vol2-draft.pdf