Тема 1.5 Безопасность микропроцессорных систем

Некоторые встроенные системы находят массовое применение, например, устройства RFID. Встроенные системы являются привлекательной целью для создателей вредоносного кода из-за своей распространённости и относительной беззащитности. Постепенно злоумышленники пытаются создать вредоносный код для встроенных систем (например, RFID-вирус, Cabir). Этот процесс пока затрудняется разнородностью встроенных устройств, отсутствием доминирующего ПО и ограниченной функциональностью некоторых видов устройств. С другой стороны, задача антивирусных компаний и исследователей компьютерной безопасности осложнена теми же обстоятельствами, а также маломощностью встроенных систем, зачастую не позволяющей пользоваться распространённым антивирусным ПО.

При передаче сигнала через любой канал связи возможно возникновение ошибок, которые могут приводить к искажению переносимой информации. Существует много методов для исправления подобных ошибок, но прежде чем исправлять, необходимо эти ошибки обнаружить. Для этого также существуют определенные методы, основанные на избыточности передаваемой информации, что позволяет не только выявлять наличие факта искажения информации, но и в ряде случаев устранять эти искажения. Наиболее известные из методов обнаружения ошибок передачи данных являются:

1. Посимвольный контроль четности, называемый также поперечным, подразумевает передачу с каждым байтом дополнительного бита, принимающего единичное значение по четному или нечетному количеству единичных битов в контролируемом байте. Посимвольный контроль четности прост как в программной, так и в аппаратной реализации, но его вряд ли можно назвать эффективным методом обнаружения ошибок, так как искажение более одного бита исходной последовательности резко снижает вероятность обнаружения ошибки передачи. Этот вид контроля обычно реализуется аппаратно в устройствах связи.

2. Поблочный контроль четности, называемый продольным. Схема данного контроля подразумевает, что для источника и приемника информации заранее известно, какое число передаваемых символов будет рассматриваться ими как единый блок данных. В этой схеме контроля для каждой позиции разрядов в символах блока (поперек блока) рассчитываются свои биты четности, которые добавляются в виде обычного символа в конец блока. По сравнению с посимвольным контролем четности поблочный контроль четности обладает большими возможностями по обнаружению и даже корректировке ошибок передачи, но все равно ему не удается обнаруживать определенные типы ошибок.

3. Вычисление контрольных сумм. В отличие от предыдущих методов для метода контрольных сумм нет четкого определения алгоритма. Каждый разработчик трактует понятие контрольной суммы по-своему. В простейшем виде контрольная сумма — это арифметическая сумма двоичных значений контролируемого блока символов. Но этот метод обладает практически теми же недостатками, что и предыдущие, самый главный из которых — нечувствительность контрольной суммы к четному числу ошибок в одной колонке и самому порядку следования символов в блоке.

4. Контроль циклически избыточным кодом — CRC (Cyclical Redundancy Check). Это гораздо более мощный и широко используемый метод обнаружения ошибок передачи информации. Он обеспечивает обнаружение ошибок с высокой вероятностью. Кроме того, этот метод обладает рядом других полезных моментов, которые могут найти свое воплощение в практических задачах.

Пять шагов к обеспечению безопасности встраиваемых систем

В современных условиях растущей связности вычислительных устройств проблема информационной безопасности выходит на один уровень с проблемой сложности ПО. Статья описывает пять шагов по обеспечению информационной безопасности встраиваемых систем с учётом всех стадий их жизненного цикла – от проектирования до разработки, тестирования, развёртывания и обслуживания.

Первый «звоночек»

2010 год ознаменовался первым «червём», способным внедряться в промышленную инфраструктуру и позволять взломщику перехватывать управление критическими системами, – Stuxnet. Он был разработан для атаки на центрифуги, используемые в ядерной программе Ирана, и отбросил работы по программе предположительно на несколько лет назад. Stuxnet обладал способностью внедряться в программируемые логические контроллеры (ПЛК) и устанавливать на них вредоносный код, отключающий защитные блокировки и выводящий центрифугу из безопасного режима вращения.

Для определённых классов встраиваемых систем проблема безопасности не нова, в частности, информационная безопасность всегда ставилась во главу угла при разработке оборонных и правительственных систем. Однако наибольшую актуальность проблема безопасности сейчас представляет для коммерческих систем управления и критических встраиваемых систем, работающих в составе различных инфраструктур. Постоянно увеличивающаяся степень интеграции коммерческих встраиваемых систем с настольными и мобильными вычислительными платформами, Интернетом, облачными архитектурами и т.п. всё более обнажает неспособность классических ПЛК противостоять сетевым вторжениям, так как при их проектировании предполагалось, что изолированная сеть системы управления безопасна по умолчанию. Stuxnet развеял этот миф, продемонстрировав способность вторгаться в изолированные сети систем управления через подключаемые к ним компьютеры. Многие современные сетевые угрозы являются ещё более изощрёнными и могут исходить в том числе из косвенных источников, заставляя разработчиков встраиваемых систем учитывать всё большее количество требований к безопасности в своих продуктах.

(В английском языке для обозначения безопасности используются два термина – "safety" и "security", имеющие совершенно разный смысл. Чтобы исключить неоднозначности, в настоящем переводе для разделения этих понятий используются термины «функциональная безопасность» и «информационная безопасность» соответственно. – *Прим. пер.*)

В поисках золотой середины

В последнее время информационная безопасность встраиваемых систем перешла в разряд обязательных (must-have) требований. Однако рост объёма требований увеличивает трудоёмкость разработки, а значит, и её стоимость. К тому же огромное число потенциально уязвимых встраиваемых систем, основанных на устаревших стратегиях безопасности, уже давно развёрнуты и эксплуатируются в современной тесно связанной сетевой среде. Потребности в поддержке облачных вычислений увеличивают степень связности устройств, тем самым ещё более усложняя ситуацию, так как незащищённые устройства в составе сети представляют собой потенциальные точки атаки.

Между различными аспектами бизнес-требований к устройству – функциональностью, производительностью, стоимостью, безопасностью и т.п. – должен соблюдаться разумный баланс, причём на каждом вертикальном рынке он может быть разным. Безопасность – важный компонент многих систем; однако обеспечение безопасности не должно ставить под угрозу сроки поставки или бюджет проекта. Безопасности часто уделяется недостаточно внимания в процессе разработки устройств, так как она считается чем-то подразумевающимся, а не отличительной особенностью.

При разработке безопасного устройства следует предварительно оценить такие моменты:

- степень важности аспектов продукта, предъявляющих требования к безопасности;
- среду эксплуатации и её предполагаемую степень подверженности угрозам;
- меры, которые необходимо принять для защиты выбранных аспектов и минимизации угроз.

Иными словами, следует определить требуемый уровень безопасности, адекватный характеру устройства, его целевому рынку и соответствующей среде эксплуатации. Обеспечение безопасности – непрерывный процесс, охватывающий весь жизненный цикл продукта, от проектирования до вывода из эксплуатации. Планирование и бюджетирование работ по обеспечению функциональной и информационной безопасности на протяжении всего жизненного цикла продукции, а также защита её от возможных будущих угроз являются важными и неотъемлемыми частями процесса разработки.

Сложность растёт, связность – тоже

Встраиваемые системы традиционно эксплуатировались в относительно изолированных средах, вследствие чего были защищены от широкого спектра угроз автоматически. Современные устройства, однако, часто бывают подключены к корпоративным сетям, вычислительным облакам, а то и напрямую к Интернету. Оборонные и разведывательные сети также развиваются в сторону использования облаков и мобильных платформ (смартфонов и т.п.), сохраняя при этом жёсткие требования к доменам безопасности. Всеобъемлющая связность сулит существенный выигрыш в функциональности и удобстве использования, однако, как уже упоминалось, также делает устройства более подверженными внешним атакам.

Ещё недавно безопасность была далеко не основным аспектом встраиваемых проектов, однако наступающая эра связанных устройств выводит её в разряд первоочередных системных требований. Часто получалось так, что аппаратное и программное обеспечение систем предыдущих поколений было разработано без учёта требований к безопасности коммуникаций и не содержало необходимых компонентов: межсетевых экранов, систем предотвращения вторжения и т.п. К тому же было бы опрометчиво предполагать, что сетевая среда может быть изолированной и защищённой, да и предсказать, как именно устройства в результате окажутся соединены и каких влияний окажут на них будущие устройства, тоже нельзя.

Самый эффективный путь обеспечения разумного баланса между функциональностью и безопасностью устройства – определять и ранжировать требования к безопасности изначально в контексте всей системы и среды эксплуатации, включая сетевую среду. При этом чем на более ранней стадии жизненного цикла начнётся этот процесс, тем эффективнее он будет работать.

Кибербезопасность и информационная безопасность

Большинство встраиваемых систем отличаются от корпоративных тем, что они непосредственно управляют процессами и оборудованием, входящим в состав ключевой инфраструктуры. С точки зрения безопасности основная проблема заключается в том, что злоумышленник потенциально может удалённо получить контроль над управляющими устройствами и либо отключить их, либо заставить их вести себя неправильно, в худшем случае приводя к отказам и физическим повреждениям.

В корпоративных системах меры безопасности преимущественно ориентированы на предотвращение несанкционированного доступа к конфиденциальной информации. Однако для встраиваемых систем необходимо обеспечивать не только защиту хранимых и передаваемых данных, но и корректность и безопасность их функционирования. Термин «кибербезопасность» ещё относительно молод и часто используется как синоним термина «информационная безопасность» (information assurance), однако у информационной безопасности есть формальное определение – «совокупность мер, направленных на охрану и защиту информации и информационных систем путём обеспечения их готовности, целостности, конфиденциальности, достоверности и авторизованности доступа к ним» [1]. Термин «кибербезопасность» обычно используется в более общем смысле и определяется в словарях как совокупность мер, предпринимаемых для защиты компьютеров или вычислительных систем от несанкционированного доступа или атаки. В рамках данной статьи (а также в соответствии с тем, в каком контексте этот термин употребляется в индустрии встраиваемых систем) кибербезопасностью называется защита встраиваемого устройства от атак с целью обеспечения его корректного и безопасного функционирования в соответствующей среде при одновременной защите конфиденциальной информации (если таковая присутствует).

Для разработчиков систем информационной безопасности основной задачей в этом смысле является защита данных, которые устройство хранит и передаёт. Обычно эти данные являются конфиденциальными, а в оборонных и правительственных приложениях могут подпадать под гриф секретности. Встраиваемое устройство должно быть разработано таким образом, чтобы по возможности затруднить к себе доступ потенциальному злоумышленнику, таким образом поддерживая целостность хранимых и передаваемых данных. Система защиты при этом должна быть способной справиться с широким спектром различных угроз – от сетевых атак до физического доступа.

Никакой производитель не хочет, чтобы его устройства можно было легко вывести из строя или похитить из них конфиденциальные данные. Многие классы устройств не подразумевают работу с конфиденциальными данными вообще; однако корректность и безопасность их функционирования является первоочередной задачей. По мере развития межмашинных (M2M) коммуникаций в инфраструктуре («умные» сети энергоснабжения, облачные вычислительные структуры и т.п.) требования к безопасности встраиваемых устройств переходят в разряд критических.

Рекомендации по повышению безопасности встраиваемых систем

Разработчикам встраиваемых систем при решении вопросов безопасности следует иметь в виду пять основных шагов (рис. 1). Описываемая схема не является готовой методологией, но скорее призвана обрисовать подход, позволяющий смотреть на обеспечение безопасности как на последовательность решений, принимаемых на различных стадиях цикла разработки продукта.

Шаг 1: оценка угроз

Повышение безопасности встраиваемого устройства начинается с идентификации потенциальных угроз. Угрозы следует оценивать в контексте про-

Рис. 1. Пять шагов к обеспечению безопасности встраиваемых систем

изводителя устройства, операторов (если они предусмотрены) и конечных пользователей (включая соответствующую среду эксплуатации) и описывать в виде пар так называемых *векторов атаки* (то есть способов осуществления атаки) и эксплуатируемых ими *уязвимостей* (то есть незащищённостей или сбоев в программном или аппаратном обеспечении, позволяющих атаке достигнуть цели). В качестве примера вектора атаки можно привести кабельное Ethernet-соединение и предоставляемые через него типовые сервисы наподобие HTTP, FTP, SSH или отладочных агентов. Примером уязвимости может служить слишком простой или установленный по умолчанию пароль, или ошибка кодирования (например, отсутствие проверок переполнения стека), или даже концептуальная ошибка проектирования (например, некорректная последовательность начальной загрузки). Самая трудная часть здесь – это предсказать и предотвратить все возможные векторы атаки и уязвимости заранее.

Чтобы оценка принесла плоды, необходимо проанализировать устройство в очень широком спектре потенциальных угроз. Множество реальных современных угроз возникло из предположения, что устройство просто не может использоваться тем или иным способом. Stuxnet, в частности, атаковал ПЛК, расположенные в общей сети с инфицированными настольными и портативными компьютерами. Однако даже учитывая, что такая сеть изначально не подключена к Интернету, вполне вероятно, что к ней иногда будут подключаться диагностические или инструментальные компьютеры. При оценке угроз важно рассматривать не только само устройство, но и внешнюю среду, в частности, операторов и конечных пользователей.

В процессе оценки угроз для встраиваемого устройства необходимо сделать как минимум следующее.

- **Произвести полный анализ жизненного цикла продукта**, при этом необходимо учесть как разработчиков, так и изготовителей, дистрибьюторов, поставщиков и конечных пользователей, чтобы получить полную картину их влияния на безопасность. Необходимо также оценить приоритет кибербезопасности и защиты информации для данного устройства.
- **Определить и описать все возможные входные точки атаки**, при этом не следует зацикливаться на сетевом доступе, так как существуют и другие варианты, например, физический доступ через USB или последовательные порты. Когда входные точки определены, нужно оценить уязвимости для каждой из них. Защищено ли устройство от атаки через TCP-порт 80? Используется ли межсетевой экран? Если нет, какие порты TCP/UDP открыты? Аналогично для физического доступа: поддерживает ли устройство загрузку с внешнего USB-носителя? Необходим также анализ возможных комбинаций входных точек.
- **Построить матрицу рисков.** Поскольку возможных вариантов атаки может быть множество, требуется оценка вероятности атаки по каждому каналу и возможного урона от такой атаки.
- **Разработать стратегию сокращения рисков**, исходя из заданных приоритетов. Например, выигрышной стратегией может быть разбиение системы на разделы с разными уровнями безопасности. В ряде случаев это может вести к усложнению архитектуры, однако эта стратегия окупается за счёт уменьшения объёма тестирования и сокращения стоимости обновления на более поздних стадиях жизненного цикла.
- **Создать техническую спецификацию**, включающую в себя требования к безопасности, полученные на основе предыдущих действий. Это равноправная часть процесса разработки, но её следует расценивать как высокоприоритетную. План работ по проектированию, разработке, тестированию и сопровождению средств безопасности должен стать частью общего рабочего плана.

Шаг 2: проработка безопасной архитектуры

Ответом на усиление значения безопасности связанных устройств стало развитие ряда технологий и методологий разработки. Одной из важных парадигм разработки безопасных устройств является использование готовых коммерческих (commercial off-the-shelf – COTS) системных компонентов, которое позволяет обеспечивать безопасность, одновременно контролируя затраты. В качестве примеров можно привести как сертифицируемые ОС и связующее ПО, так и технологии виртуализации, разбиения на разделы и виртуальные среды исполнения, позволяющие увеличить уровень абстракции и разделения компонентов.

В последнее время набирает особую популярность виртуализация, так как она позволяет выполнять несколько ОС на общей разделяемой аппаратной платформе. Это даёт проектировщикам дополнительную гибкость, а также позволяет более эффективно использовать возможности оборудования по сравнению с дизайном на базе одной ОС. Виртуализация также предоставляет хорошую почву для распределения функциональности устройства по нескольким разделённым виртуальным средам, что позволяет разворачивать на одной физической платформе компоненты с повышенными требованиями к функциональной или информационной безопасности и некритические приложе-

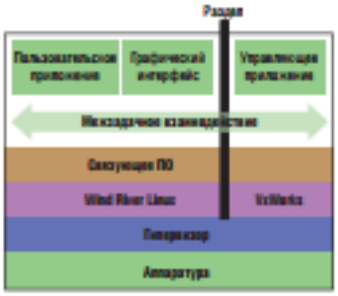


Рис. 2. Пример разбиения на разделы с использованием гипервизора

Рис. 3. Стек программных компонентов среды разработки и исполнения

ния. Например, в подобной виртуализированной среде можно развести ОС реального времени (ОС РВ) и ОС общего назначения по разным разделам, отвечающим за различные элементы функциональности (рис. 2). В этом случае возможные атаки на уязвимые места ОС общего назначения не смогут повлиять на раздел, в котором выполняется ОС РВ, – таким образом критическая функциональность устройства останется незатронутой.

Концепция повторно используемых компонентов применима не только к коду, разработанному внутри организации и затем многократно применяемому и различных проектах. На более высоком уровне это означает использование готовых модулей, гарантированно удовлетворяющих требованиям качества и безопасности, а это проверенный путь к сокращению себестоимости, срокам разработки и главное – к управлению рисками. Многие коммерческие программные продукты проходят исчерпывающее тестирование, валидацию и сертификацию и могут быть идеальными кандидатами на роль повторно используемых компонентов для устройств нового поколения. Разработчики могут сразу начать пользоваться необходимой функциональностью и готовой сертификационной документацией, в то время как разрабатывать и поддерживать всё это в рамках одного конкретного проекта было бы гораздо сложнее.

Шаг 3: выбор безопасной программной платформы

Выбор программной платформы для встраиваемой системы – решение ключевое. Реализация системы на основе компонентов, имеющих штатный комплект сертификационной документации, способна повысить уровень безопасности и сократить затраты; есть также и другие преимущества использования коммерческих компонентов вместо разработанных или портированных (и поддерживаемых) самостоятельно.

Вот неполный перечень коммерческих компонентов программной платформы, которые могут помочь в разработке устройства с повышенными требованиями к безопасности (рис. 3).

- **Уровень поддержки оборудования.** Программные платформы (ОС, гипервизоры и т.п.) требуют наличия уровня поддержки оборудования, в состав которого входят драйверы конкретных аппаратных средств. Переход на коммерческие средства поддержки оборудования – существенный шаг на пути внедрения повторно используемых компонентов; однако здесь очень важно правильно выбрать поставщиков. Коммерческие средства поддержки оборудования (например BSP) оптимизированы для конкретных целевых устройств, обеспечены технической поддержкой, а в ряде случаев ещё и снабжены сертификационной документацией, позволяющей использовать их в составе систем, предъявляющих повышенные требования к функциональной и информационной безопасности.
- **Встраиваемые средства виртуализации.** Встраиваемый гипервизор способен обеспечить разбиение системы на разделы, параллельное выполнение нескольких ОС и поддержку многоядерных вычислений – всё это позволяет повысить степень интеграции. В сочетании с коммерческими средствами поддержки оборудования встраиваемые средства виртуализации, обеспеченные коммерческими пакетами сертификационной документации, помогают быстрее разрабатывать устройства, требующие реализации нескольких уровней безопасности одновременно.
- **ОС реального времени.** Многие встраиваемые системы предъявляют жёсткие требования к ресурсоёмкости и временным характеристикам ПО, а также требуют сертификации по различным стандартам функциональной/информационной безопасности. Хорошим фундаментом для таких приложений будет ОС реального времени (ОС РВ). (Объём кода ОС РВ обычно значительно меньше, чем объём кода ОС общего назначения, поэтому сертификация ОС РВ обходится существенно дешевле. – *Прим. ред.*). При выборе ОС РВ следует обратить особое внимание на следующее.
  - **Безопасность конфигурации по умолчанию.** Является ли безопасной конфигурация ОС РВ по умолчанию (например, отключены ли второстепенные сервисы, закрыты ли сетевые порты и т.п.)?
  - **Безопасность коммуникационных средств.** Поддерживает ли ОС РВ сервисы, реализующие стандарты защищённых коммуникаций? Содержит ли ОС РВ криптографические средства? Позволяет ли сетевой стек ОС РВ использовать безопасные сокеты, виртуальные частные сети (VPN), IPSec и т.п.?
  - **Сертификация.** Есть ли для данной ОС РВ сертификационные пакеты по стандартам безопасности, применимым к данному приложению? Проходил ли сетевой стек ОС РВ тестирование и валидацию на предмет безопасности?
  - **Приоритеты производителя в вопросах безопасности.** Имеют ли высокий приоритет вопросы безопасности у производителя ОС РВ? Есть ли у него выделенная рабочая группа по вопросам безопасности?
- **Встраиваемые ОС общего назначения.** Стандартные дистрибутивы ОС общего назначения наподобие Linux или Microsoft Windows не входят в класс ПО повышенной безопасности. Лучше всего использовать их в сочетании со встраиваемым гипервизором, который обеспечит их выполнение в выделенном защищённом разделе. Такой подход обеспечивает их изоляцию от критических частей си-

стемы, к которым предъявляются более жёсткие требования по безопасности.

- Связующее ПО. В зависимости от требований к системе уровень связующего ПО может включать в себя либо базовую, либо расширенную функциональность в области сетевых коммуникаций (включая безопасность и беспроводные технологии), аудио, графики и т.п. Выбор связующего ПО особенно важен, с точки зрения обеспечения защиты коммуникаций, так как именно оно содержит криптографические библиотеки и компоненты реализации защищённых протоколов наподобие SSL и IPSec. Дополнительную защиту можно обеспечить, применяя межсетевые экраны и авторизацию доступа. Связующее ПО, реализующее прочую функциональность (например графический интерфейс), тоже следует выбирать с учётом безопасности; в противном случае могут возникнуть непредвиденные уязвимости на системном уровне.
- Средства симуляции оборудования. Симуляция оборудования на уровне процессора, отдельных плат и системы в целом помогает ускорить процесс интеграции аппаратного и программного обеспечения, так как гарантирует доступность оборудования в нужном месте, в нужный срок и в достаточном количестве. (В процессе разработки встраиваемой системы это часто становится серьёзной проблемой, так как пока прототип не будет разработан, стабилизирован и произведён в достаточном количестве, ПО отлаживать будет физически не на чем. — *Прим. пер.*) К тому же средства симуляции позволяют использовать более эффективные методики отладки и тестирования, которые на реальном оборудовании реализовать физически невозможно (например, выполнение кода от момента ошибки в обратном порядке, чтобы восстановить цепочку событий. — *Прим. пер.*).
- Инструментарий. Инструменты, используемые для тестирования, отладки и статического анализа кода, играют критическую роль в выявлении ошибок проектирования и в предотвращении просачивания небезопасного кода в проект на ранних его стадиях. Средства автоматизированного тестирования и симуляции оборудования также значительно увеличивают эффективность процесса разработки.

Шаг 4: обеспечение безопасности приложений

Современные встраиваемые системы давно вышли за пределы традиционных узкоспециализированных устройств, выполняющих одну конкретную задачу. Сейчас в рамках одной встраиваемой системы могут сосуществовать несколько приложений, причём их функциональность может расширяться на всём протяжении жизненного цикла устройства посредством динамического обновления как аппаратуры, так и ПО.

Как и в случае с настольными и серверными приложениями, для встраиваемых систем критично наличие средств безопасности, так как они становятся всё более открытыми для вредоносного кода и несанкционированного доступа к данным.

Встраиваемые системы, поддерживающие динамическое обновление, могут использовать для повышения безопасности технику белого списка (whitelisting). Использование белого списка позволяет устройствам загружать и запускать только те приложения, которые подтверждены как безопасные; всё остальное ПО, не входящее в белый список, системой принято не будет. Родственная техника чёрного списка, подразумевающая поддержку актуального списка известного вредоносного ПО и вирусов, используется для предотвращения загрузки и запуска содержимого, входящего в запрещённый список.

Поскольку чёрные списки гораздо объёмнее белых и гораздо чаще меняются, встраиваемым системам обычно недостаёт ресурсов для постоянной поддержки актуальности чёрных списков, так как для этого необходимо наличие постоянного хранилища, средств сетевой синхронизации, частых обновлений и т.п. Это делает белые списки более привлекательной стратегией для встраиваемых систем.

Ещё одна техника, которую можно использовать для повышения безопасности, — это оценка так называемой *репутации* источника данных. Если данные получены из источника с неизвестной или дурной репутацией, приложение может определить, какие шаги следует предпринять: выполнить проверку целостности, отвергнуть и т.п. Проверка репутации источников также способна помочь повысить производительность: если данные поступают из проверенных источников, то проверки безопасности будут занимать меньше процессорного времени.

Шаг 5: распространение мер безопасности на весь жизненный цикл продукции

Требования к безопасности постоянно меняются, так как постоянно меняются угрозы. По мере того как устройство набирает популярность (Stuxnet был ориентирован на широко распространённую модель ПЛК) и/или возраст на рынке, оно становится более подверженно атакам. В прошлом многие устройства не были рассчитаны на динамическое перепрограммирование и обновление в процессе эксплуатации (без существенных модификаций). Но это время прошло. Сегодня устройства обязаны поддерживать динамическое обновление, и не только для усовершенствования функциональности, но и для решения будущих вопросов безопасности.

Включение задач по обеспечению безопасности в процесс управления жизненным циклом изделия сегодня является первостепенной задачей. Мало того — важна ещё и скорость, с которой организация может реагировать на угрозы по мере их возникновения. Также не менее важным моментом является соответствие вашим стандартам безопасности продукции выбранных вами поставщиков.

В контексте безопасности разработчикам встраиваемых систем следует учитывать как минимум следующие аспекты управления жизненным циклом.

- Безопасность должна быть встроена во все стадии жизненного цикла. Безопасный дизайн должен присутствовать в системе изначально и быть основой всего жизненного цикла. Безопасную архитектуру нельзя добавить в случае необходимости.
- Будущие изменения должны быть предусмотрены заранее. Устройствам и системам требуется возможность динамического обновления, исправления ошибок и модернизации; будущие обновления и мониторинг безопасности необходимо включать в планы поддержки изначально.

- **Безопасность должна иметь приоритет в процессе разработки и тестирования.** Проблемы с безопасностью ПО фактически являются следствием ошибок в требованиях (будь то их формирование или реализация), и чем раньше в цикле разработки они будут обнаружены, тем дешевле обойдётся их исправление, а значит, и ликвидация уязвимостей. Тестирование безопасности должно включать в себя определение границ системы и методов эксплуатации слабых мест в их защите. Очень хороши здесь могут быть техники наподобие fuzz-тестирования (тестирование на основе заведомо некорректных или случайных входных данных, которое позволяет, в частности, выявлять ошибки в редко используемых участках кода. — *Прим. пер.*) или испытания на проникновение (симуляция векторов атаки потенциальных злоумышленников). Автоматизация тестирования безопасности вкупе с использованием средств симуляции оборудования способна существенно повысить эффективность процесса разработки и обеспечить гораздо более детальное исследование поведения продуктов, чем интуитивное (ad hoc) тестирование.
- **Устранение дефектов, связанных с безопасностью, должно иметь высокий приоритет.** Безопасности следует уделять особое внимание не только в процессе разработки, но и в процессе сопровождения. Когда уязвимость обнаруживается профессиональным сообществом, она быстро становится достоянием общественности. Производителям необходимо уметь оперативно реагировать на подобные ситуации по мере их возникновения.
- **Вопросами безопасности должна заниматься специальная рабочая группа.** В её задачи будут входить анализ уязвимостей, проработка возможных путей их устранения, внутренние и внешние коммуникации, планирование выпуска обновлений и контроль выполнения запланированных мероприятий. Подобные группы обычно кросс-функциональны и включают в себя представителей подразделений разработки аппаратуры и ПО, тестирования, технической поддержки, управления требованиями и документирования.

Заключение

Описанные пять шагов позволяют организациям достичь фундаментального прогресса в области защиты связанных встраиваемых систем от угроз. Учёт вопросов безопасности в процессе разработки встраиваемых систем сегодня является ключевым требованием и предусматривает вовлечение всех уровней и подразделений организации в создание всеобъемлющей защитной структуры, жизненно необходимой в современных связанных вычислительных средах.

Литература

1. National Information Assurance (IA) Glossary : CNSS Instruction No. 4009. – 26 April 2010. – Committee on National Security System.

Автор – вице-президент компании Wind River по решениям в области жизненного цикла
Перевод Николая Горбунова, сотрудника фирмы ПРОСОФТ
Телефон: (495) 234-0636
E-mail: info@prosoft.ru

Циклический избыточный код (англ. Cyclic redundancy code, CRC) — алгоритм вычисления контрольной суммы, предназначенный для проверки целостности передаваемых данных. Алгоритм CRC обнаруживает все одиночные ошибки, двойные ошибки и ошибки в нечетном числе битов. Понятие циклических кодов достаточно широкое, однако на практике его обычно используют для обозначения только одной разновидности, использующей циклический контроль (проверку) избыточности. В связи с этим в англоязычной литературе CRC часто расшифровывается как Cyclic Redundancy Check.

CRC некоторой последовательности вычисляется на основании другой (исходной) битовой последовательности. Главная особенность (и практическая значимость) значения CRC состоит в том, что оно однозначно идентифицирует исходную битовую последовательность и поэтому используется в различных протоколах связи, а также для проверки целостности блоков данных, передаваемых различными устройствами. Благодаря относительной простоте алгоритм вычисления CRC часто реализуется на аппаратном уровне.

Основная идея вычисления CRC заключается в следующем. Исходная последовательность битов, которой могут быть и огромный файл, и текст размером несколько слов и даже символов, представляется единой последовательностью битов. Эта последовательность делится на некоторое фиксированное двоичное число (полином, CRC-полином, генераторный полином, англ. generator polinomial). Интерес представляет остаток от этого деления, который и является значением CRC. Все, что теперь требуется, — это некоторым образом запомнить его и передать вместе с исходной последовательностью. Приемник данной информации всегда может таким же образом выполнить деление и сравнить его остаток с исходным значением CRC. Если они равны, то считается, что исходное сообщение не повреждено, и т. д.

Степенью CRC-полинома W называют позицию самого старшего единичного бита. Например, степенью полинома 10011_2 равна 4.

Для вычисления CRC используют специальную т.н. полиномиальную арифметику. Вместо представления делителя, делимого (сообщения), частного и остатка в виде положительных целых чисел, можно представить их в виде полиномов с двоичными коэффициентами или в виде строки бит, каждый из которых является коэффициентом полинома.

Например, десятичное число 23 в шестнадцатеричной системе и двоичных системах будет иметь вид: $23_{10} = 17_{16} = 10111_2$, что совпадает с полиномом: $1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$ или упрощенно $x^4 + x^2 + x^1 + x^0$.

И сообщение, и делитель могут быть представлены в виде полиномов, с которыми можно выполнять любые арифметические действия.

Предположим, что надо перемножить числа 1101_2 и 1011_2. Это можно выполнить, как умножение полиномов: $\left(x^3 + x^2 + x^0\right) \cdot \left(x^3 + x^1 + x^0\right) = \left(x^6 + x^5 + x^3\right) + \left(x^4 + x^3 + x^1\right) + \left(x^3 + x^2 + x^0\right) =$ $1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 3 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$.

Поскольку перемножались двоичные числа ($x = 2$), то в выражении $3 \cdot x^3$ возможен перенос бита от этого члена суммы в старший разряд ($3 \cdot x^3 = x^4 + x^3$), поэтому окончательное выражение будет иметь вид: $x^7 + x^3 + x^2 + x^1 + x^0$.

Если рассматривать коэффициенты полинома как изолированные друг от друга и для вычисления каждого из них использовать свои собственные правила, то можно получить различные виды полиномиальной арифметики. В частности широко используется т.н. «полиномиальная арифметика по модулю 2», когда коэффициенты складываются по модулю 2 без переноса – то есть коэффициенты могут иметь значения лишь 0 или 1.

Тогда вышерассмотренный пример в полиномиальной арифметике по модулю 2 будет иметь вид: $\left(x^3 + x^2 + x^0\right) \cdot \left(x^3 + x^1 + x^0\right) = 1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 3 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0 =$ $x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0$.

Правила полиномиальной арифметики по модулю 2 можно использовать и при обычной арифметике, так как действия, выполняемые во время вычисления CRC, являются арифметическими операциями без учета переносов.

Сложение двух чисел в CRC арифметике полностью аналогично обычному арифметическому действия за исключением отсутствия переносов из разряда в разряд. Это означает, что каждая пара битов определяет результат своего разряда вне зависимости от результатов других пар. Например:

$$\begin{array}{r} 10011011 \\ +\ 11001010 \\ \hline 01010001 \end{array} \qquad \begin{array}{r} 10011011 \\ -\ 11001010 \\ \hline 01010001 \end{array}.$$

Т.е. и при сложении и при вычитании над каждым битом в отдельности выполняется операция, аналогичная операции *XOR*, поэтому в CRC арифметике две арифметические операции (сложение и вычитание) заменяются операцией *XOR*.

Умножение, как и в обычной арифметике, считается суммой значений первого сомножителя, сдвинутых в соответствии со значением второго сомножителя. Причем при суммировании так же используется CRC сложение. Например:

$$\begin{array}{r} 1101 \\ \times\ 1011 \\ \hline 1101 \\ 1101. \\ 0000.. \\ 1101... \\ \hline 1111111 \end{array}$$

Деление в CRC арифметике определяется аналогично с учетом того, что вычитание выполняется по правилам CRC арифметики. Например:

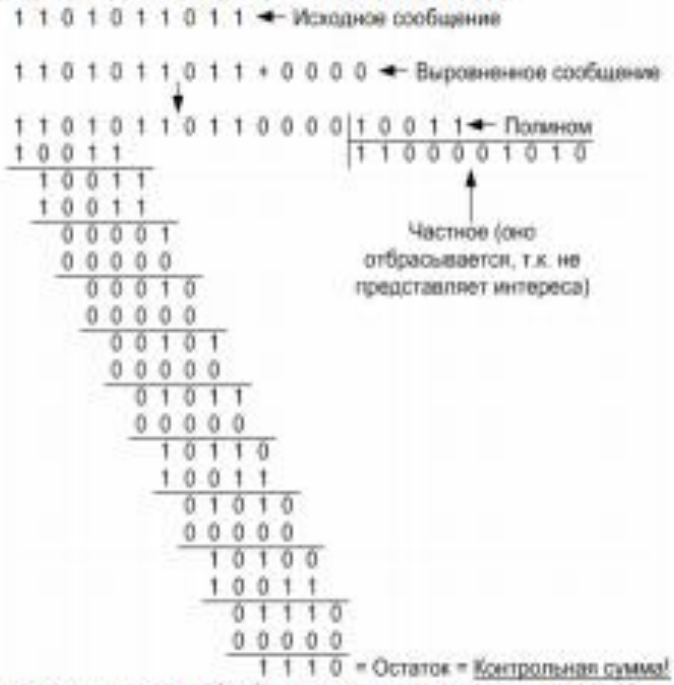
```
1 1 0 1 0 1 1 0 1 1 0 0 0 0 | 1 0 0 1 1
1 0 0 1 1                   | 1 1 0 0 0 0 1 0 1 0
  1 0 0 1 1
  1 0 0 1 1
    0 0 0 0 1
    0 0 0 0 0
      0 0 0 1 0
      0 0 0 0 0
        0 0 1 0 1
        0 0 0 0 0
          0 1 0 1 1
          0 0 0 0 0
            1 0 1 1 0
            1 0 0 1 1
              0 1 0 1 0
              0 0 0 0 0
                1 0 1 0 0
                1 0 0 1 1
                  0 1 1 1 0
                  0 0 0 0 0
                    1 1 1 0 = Остаток
```

В CRC арифметике считается что число *A* делится на число *B*, если его можно получить из нуля путем некоторого числа сложений сдвинутого числа B. Например, пусть $A=0111010110_2$ и $B=11_2$, тогда

```
= 0 1 1 1 0 1 0 1 1 0
+ . . . . . . . 1 1 .
+ . . . . 1 1 . . . .
+ . . . 1 1 . . . . .
+ . 1 1 . . . . . . .
```

Перед началом вычисления CRC исходное сообщение следует дополнить *W* нулями справа и выполнить деление по правилам CRC-арифметики. Рассмотрим пример:

Исходное сообщение: 1101011011
Генераторный полином: 10011
Сообщение, дополненное W битами: 11010110110000



Как видно, контрольная сумма (CRC) в этом примере равна 1110. Как правило, контрольная сумма добавляется к исходному сообщению (в нашем примере передаваемое сообщение будет равно 11010110111110) и полученное расширенное сообщение передается через канал связи.

На другом конце канала приемник может сделать одно из возможных действий (оба варианта совершенно равноправны):

1. Выделить текст собственно сообщения, вычислить для него контрольную сумму (не забыв при этом дополнить сообщение W битами), и сравнить ее с переданной.
2. Вычислить контрольную сумму для всего переданного сообщения (без добавления нулей), и посмотреть, получится ли в результате нулевой остаток.