ПОРТФЕЛЬ РЕДАКЦИИ

БИТ

S. S. Agafyin

Security Functions of Removable Key Carriers

Keywords: JavaCard, MULTOS, security functions

At this paper the author describes removable key devices platforms and their security functions.

С. С. Агафьин

ФУНКЦИИ БЕЗОПАСНОСТИ ВНЕШНИХ АППАРАТНЫХ МОДУЛЕЙ, ИСПОЛЬЗУЕМЫХ ДЛЯ ХРАНЕНИЯ КРИПТОГРАФИЧЕСКИХ КЛЮЧЕЙ

В ходе своего развития внешние аппаратные модули нашли применение практически во всех сферах человеческой деятельности: от электронных платежей до мобильной связи. Параллельно с аппаратной составляющей развивалась и программная часть, которая из низкоуровневого программного обеспечения, написанного на языке ассемблера, превратилась в серьезные вычислительные платформы, для которых создаются приложения на высокоуровневых Java-подобных языках. В данной статье приведено подробное описание особенностей существующих платформ внешних аппаратных модулей.

Существующие платформы внешних аппаратных модулей

Внешние аппаратные модули (ВАМ), предназначенные для хранения и обработки конфиденциальной информации (в том числе секретных ключей криптографических операций), строятся на множестве программных платформ.

Наиболее распространенной программной платформой для внешних аппаратных модулей является JavaCard, на базе которой функционируют 56 % современных модулей [1]. Около 28 % всех внешних аппаратных модулей реализованы на базе платформы MULTOS. Остальные 16 % распределены между нерасширяемыми программными платформами, для которых создавать приложения могут только разработчики системного программного обеспечения внешнего аппаратного модуля, которых можно считать доверенными, и, следовательно, исключить из дальнейшего рассмотрения.

MULTOS (Multiple Operating System) — операционная система, которая позволяет устанавливать на внешний аппаратный модуль несколько независимых приложений. Каждая программа изолируется операционной системой и не имеет интерфейсов взаимодействия с остальными программами. MULTOS позволяет устанавливать, обновлять и удалять приложения на любой стадии использования модуля. Приложения являются независимыми от аппаратной

платформы, поскольку исполняются на виртуальной машине. Программное обеспечение для MULTOS разрабатывается на языках MEL (MULTOS Executable Language) и C.

JavaCard предоставляет возможность разрабатывать программное обеспечение на языке, являющемся подмножеством языка Java. Данная платформа разрабатывалась для того, чтобы обеспечить совместное существование независимых приложений на одной аппаратной основе. В связи с этим уровень операционной системы модуля и уровень программного обеспечения отделены друг от друга.

Согласно докладу NIST [2], функции безопасности MULTOS являются подмножеством функций безопасности, реализованных в платформе JavaCard, поэтому далее можно рассматривать только функции безопасности JavaCard.

Жизненный цикл приложения JavaCard

Изначально приложение разрабатывается в виде набора Java-подобных файлов с расширением .java. Синтаксические конструкции языка JavaCard являются подмножеством синтаксических конструкций языка Java с дополнительными классами.

Основными блоками, которыми дополнена стандартная библиотека Java, являются: класс работы с APDU-интерфейсом, классы атомарной работы с массивами, дополнительные защищенные криптографические функции. Ряд методов, необходимых для работы с глобальными объектами, которые генерируются виртуальной машиной для взаимодействия с терминальной системой, самостоятельно обрабатываются исполняющей средой JavaCard.

Каждый исходный файл компилируется в промежуточный Java-код, хранимый в файлах .class. На данном этапе используется классический компилятор Java, который выполняет ряд проверок на соответствие типов, а также расширен набором классов, который определен в стандартной библиотеке JavaCard. Данный шаг разработки необходим для получения нормализованного вида программы, который затем будет передан в оптимизатор, специфицирующий программу для загрузки на внешний аппаратный модуль.

Совокупность .class-файлов, а также небольших информационных .exp-файлов, являющихся аналогами заголовочных файлов компилируемых языков программирования, передается на вход преобразователю. Он осуществляет дополнительные проверки, специфичные для внешних аппаратных модулей (например, использование только базовых типов byte и short), а также с помощью составных алгоритмов оптимизации объединяет скомпилированные файлы классов в единый CAP-файл, который может быть загружен на внешний аппаратный модуль.

Данный файл, хранящий полную информацию о разработанном приложении, с помощью команд, определенных в ГОСТ Р ИСО/МЭК 7816-4-2004 [3], загружается и устанавливается на внешний аппаратный модуль. Процесс установки может проводиться как средствами, включенными в состав пакета разработчика JavaCard, так и с помощью утилит, распространяемых разработчиками модулей. Эти утилиты могут осуществлять дополнительные проверки CAP-файла, что помогает избежать базовых ошибок, связанных с нарушением целостности приложения.

После установки приложение доступно для выбора и работы с ним. Одновременно активным (выбранным) может быть только одно приложение, так как платформа является однопоточной. Из каждого приложения неявным образом экспортируется функция *process*, в которую производится передача управляющих командных сообщений протокола APDU.

Архитектура платформы JavaCard

Общая архитектура внешнего аппаратного модуля, поддерживающего JavaCard, может быть описана в виде многоуровневой модели.

Основными ее компонентами являются домен карт, домен безопасности, JavaCard API, API операционной системы, JavaCard Virtual Machine, операционная система, микропроцессор, память и сами приложения.

Домены безопасности являются совокупностью средств аутентификации на внешнем аппаратном модуле, которые должны быть соотнесены с каждым приложением, устанавливаемым на него. Они используются для получения повышенных привилегий доступа в процессе загрузки и установки приложения путем проверки соответствия секретных ключей подписи, с помощью которых генерируется электронная подпись приложения перед его установкой на модуль, ключам проверки подписи, хранимым в постоянной памяти устройства.

Домен модуля является специфическим доменом безопасности, задачи которого:
- проведение сбора информации о системных изменениях модуля;
- предотвращение потенциально небезопасных операций с приложениями модуля;
- аутентификация новых приложений через соответствующие им домены безопасности.

JavaCard Virtual Machine (JCVM) является основным компонентом платформы JavaCard, который обеспечивает взаимодействие приложений между собой, осуществляет проверку и выполнение всех действий, которые определены в байт-коде приложений, и предоставляет интерфейс для написания приложений (JavaCard API).

Микропроцессор и операционная система внешних аппаратных модулей, поддерживающих платформу JavaCard, не обладают какими-либо особенностями, которые могут повлиять на ее моделирование, однако стоит отметить, что операционная система во многих случаях предоставляет разработчику приложений программный интерфейс, который является расширением JavaCard API.

Внешний аппаратный модуль обладает несколькими разнородными видами памяти, предоставляемыми следующими устройствами:
- постоянное запоминающее устройство (ПЗУ, ROM);
- перепрограммируемое ПЗУ (ППЗУ, EPROM);
- электрически стираемое ППЗУ (ЭСППЗУ, EEPROM);
- оперативное запоминающее устройство (ОЗУ, RAM).

Первые два вида памяти используются для хранения постоянных значений и не могут быть использованы для записи, что позволяет исключить их из дальнейшего рассмотрения.

ОЗУ используется для хранения временных значений в рамках одной сессии работы с картой. При «холодном» или «теплом» восстановлении либо полном отключении питания (например, при отсоединении карты от считывателя) информация на этом запоминающем устройстве будет гарантированно удалена. Данная область памяти делится на две логические части: «стек» (JStack) и «временную кучу» (Transient Heap).

ЭСППЗУ также делится на две части: первая (Bytecodes & Applets Structs Sector) используется для хранения кода промежуточного представления приложения, вторая, «постоянная куча» (Persistent Heap), — для хранения данных, которые должны быть доступны в разных сессиях работы с картой, например криптографических ключей.

«Стек» используется для передачи значений в вызываемые методы, получения возвращаемых значений вызываемых методов, хранения локальных переменных базовых типов (boolean, byte, short, object ref).

«Временная куча» и «постоянная куча» используются для хранения массивов данных и объектов пользовательских классов. Доступ к «временной куче» предоставляется с помощью ключевого слова «new», доступ к «постоянной куче» — с помощью методов makeTransientObject() и makeTransientArray() системного абстрактного класса JCSystem.

Будем считать, что эти две области памяти являются эквивалентными, благодаря чему можно не исключить из рассмотрения предположение о корректности механизма очистки временной памяти при «холодном» или «теплом» восстановлении. Обобщим оба этих типа памяти названием «куча».

Виртуальная машина

Виртуальная машина интерпретирует байт-код Java и переводит его в инструкции, которые могут корректно исполняться на аппаратном обеспечении. Каждая аппаратная платформа реализует собственную виртуальную машину JavaCard, тем не менее в случае, когда она соответствует спецификациям, это позволяет абстрагироваться от аппаратной специфики при разработке прикладного программного обеспечения.

Другим преимуществом использования виртуальной машины является повышение защищенности. Интерпретатор следит, чтобы перед выполнением каждой инструкции промежуточного представления были выполнены предварительные условия, а также выполняет проверку зоны воздействия команды, что позволяет исключить влияние одного приложения на другое. Иными словами, любая попытка одного приложения воздействовать на другое приведет к генерации исключения времени исполнения.

Однако, несмотря на все преимущества, виртуальная машина JavaCard обладает рядом недостатков. Главной их причиной является то, что для реализации полноценной виртуальной машины, аналогичной Java Virtual Machine (JVM), требуются значительные ресурсы, которыми в большинстве случаев внешние аппаратные модули не обладают. Так, по сравнению с JVM, в реализациях виртуальной машины JavaCard отсутствуют:

- динамическая загрузка классов;
- менеджер безопасности;
- потоки;
- сборщик мусора;
- многомерные массивы;
- операции с плавающей точкой.

Для повышения защищенности конфиденциальной информации, обрабатываемой и хранимой на внешних аппаратных модулях, виртуальная машина JavaCard может быть расширена до защищенной виртуальной машины. Используя дополнительные метаданные, хранимые в специальной области памяти, защищенная виртуальная машина может осуществлять проверки типов данных перед выполнением команд и оценивать допустимость запрашиваемой информации. Однако данный подход имеет ряд существенных недостатков. Прежде всего, для выполнения большого числа проверок требуется существенное количество вычислительных ресурсов, что заметно снижает производительность внешнего аппаратного модуля. Помимо этого, для хранения данных, на основе которых будет проводиться анализ, требуется дополнительное выделение памяти. В силу серьезного ограничения ресурсов отсечение заметной их части для выполнения проверок может оказаться критичным и неприемлемым для прикладной разработки.

Взаимодействие с модулем

Согласно ГОСТ Р ИСО/МЭК 7816-4-2004, для взаимодействия внешнего аппаратного модуля и терминальной системы используется прикладной протокол, сообщениями в котором являются блоки данных прикладного протокола (Application Protocol Data Unit — APDU).

Шаг в прикладном протоколе состоит из посылки команды, обработки ее принимающей стороной и посылки ответа в обратном направлении. Таким образом, конкретный ответ соответствует конкретной команде; вместе они именуются парой «команда — ответ». В этой паре командное и ответное сообщения могут содержать данные.

Длина командного и ответного сообщения не должна превышать 256 байт. Командное сообщение состоит из двух основных частей: заголовка и данных. Заголовок представляет собой последовательность из 5 байт, каждый из которых имеет назначение, описанное в ГОСТ Р ИСО/МЭК 7816-4-2004:

- CLA — байт класса команды, указывает на наличие или отсутствие защищенного обмена сообщениями, а также обобщает все команды одного приложения;
- INS — байт инструкции, конкретизирует команду, которая должна быть выполнена на внешнем аппаратном модуле при обработке данного командного сообщения;
- P1/P2 — байты параметров, уточняют режим выполнения команды, также могут хранить смещение, начиная с которого производится чтение или запись;
- Lc/Le — байт длины, который в зависимости от типа командного сообщения определяет либо длину передаваемых данных, либо размер данных, которые ожидает получить терминал, взаимодействующий с модулем.

Остальные байты сообщения предназначены для хранения данных, которые могут передаваться в любом формате, определенном разработчиком. Обработка командного сообщения в программном обеспечении внешнего аппаратного модуля полностью определяется разработчиком и может не соответствовать принятым соглашениям.

Ответное сообщение также состоит из двух основных частей: ответных данных и байтов состояния. Ответные данные являются опциональной частью, соответственно, эта часть сообщения присутствует только в случае, когда переданная команда подразумевает ответ. Байты состояния являются обязательным полем и показывают состояние выполнения команды, а также номер исключения, которое было сгенерировано при неуспешном выполнении работы. Так, например, слово (последовательность 2 байт) состояния 0x9000 означает успешное выполнение команды, 0x6Cxx — неверно переданный байт Lc, а 0x6F00 — необработанное неизвестное исключение. Некоторые реализации программных платформ позволяют каждому приложению генерировать собственные слова состояния, однако в большинстве случаев разрешены только байты, специфицированные в ГОСТ Р ИСО/МЭК 7816-4-2004.

Верификация байт-кода

В спецификации платформы JavaCard определен ряд средств защиты программного обеспечения, которые позволяют обнаружить и предотвратить потенциально небезопасные действия приложения.

Существует ряд атак, которые путем использования уязвимостей платформы позволяют получить доступ к конфиденциальной информации, не обращаясь к ней напрямую. Эти способы нарушения безопасности приложений практически невозможно выявить методами, основанными на анализе журналов корректного доступа к информации.

Для предотвращения подобных атак, связанных с модификацией приложения после преобразования в CAP-формат, используется встроенный верификатор байт-кода. Этот механизм является обязательным для реализации в модулях, поддерживающих платформу JavaCard. Его основная задача — расширение функциональности JCVM, связанной с проверкой исполняемого кода. Верификатор проводит проверку соответствия типов параметров сигнатур методов и аргументов, передаваемых в процессе исполнения, обеспечивает различие доступа к объектам и массивам, а также проверяет корректность запрашиваемых данных.

Для обеспечения надежной работы встроенный верификатор байт-кода не должен отличаться по функциональности от верификатора, который можно использовать на классических ЭВМ. Однако так как внешние аппаратные модули являются устройствами с серьезными ресурсными ограничениями, то либо это невозможно реализовать в полной мере, либо использование модуля

будет крайне затруднено, так как большая часть его вычислительных ресурсов будет уходить не на обработку команд, а на их проверку.

Анализ взаимодействия между приложениями

Поскольку JavaCard является платформой, которая позволяет устанавливать на внешний аппаратный модуль несколько приложений, обеспечивается разграничение доступа методов одного приложения к данным другого.

В большинстве случаев приложения не связаны между собой, поэтому JCVM по умолчанию полностью запрещает доступ приложений друг к другу. Однако в ряде использований требуется обеспечение совместной работы двух и более приложений, которые связаны друг с другом по данным. Для этой функциональности в JavaCard реализована поддержка разделяемых интерфейсов.

Объект класса, реализующего разделяемый интерфейс, называется объектом разделяемого интерфейса. Для приложения, в котором создан этот объект, он не обладает никакой спецификой и его поля и методы доступны для других объектов только в зависимости от модификаторов доступа. Из другого же контекста он представляется в виде объекта разделяемого интерфейса, и доступ предоставляется лишь к методам, которые были определены в разделяемом интерфейсе, остальные методы и поля объекта защищены средством анализа взаимодействия.

Разделяемый интерфейс должен быть реализован симметрично, то есть если одно приложение запрашивает через него данные из другого, то в обоих эти данные должны быть переданы в функции, организовывающие интерфейс.

Таким образом, корректное встраивание платформы JavaCard позволяет исключить атаки, связанные с установкой приложения, которое пытается получить доступ к конфиденциальной информации. Однако если предполагается, что нарушителем может быть разработчик приложения, то требуется дополнительная проверка данных, которые передаются с помощью этого механизма.

Криптографический интерфейс JavaCard

В случае, когда конфиденциальной информацией являются секретные криптографические ключи, для ограничения доступа к ним можно использовать встроенный криптографический интерфейс JavaCard.

Его основной особенностью является то, что секретные ключи хранятся в специальной выделенной области памяти, к которой нет внешнего доступа, а также то, что отсутствует возможность получения этих ключей с помощью каких-либо допустимых функций — генерация и использование объектов происходят с использованием инкапсуляции.

В некоторых реализациях платформы JavaCard для выполнения криптографических операций выделяется специализированная область памяти (CryptoRAM), доступ к которой возможен только через API системы и которая может быть реализована на отдельном чипе для того, чтобы обеспечить защиту от атаки типа «полное чтение».

Данный подход обладает рядом недостатков. Первый состоит в том, что количество криптографических алгоритмов, поддерживаемых платформой JavaCard, крайне мало и включает только DES, 3DES, AES и RSA, что не позволяет разрабатывать с помощью встроенных криптографических интерфейсов сертифицированное программное обеспечение. Другим недостатком является то, что ограничение интерфейса серьезно затрудняет реализацию различных криптографических протоколов, которые могут состоять не только из базовых криптографических операций, но и из их элементов, взаимодействующих с секретным ключом.

Заключение

Описанные архитектура и функции безопасности позволяют сделать вывод, что платформы внешних аппаратных модулей, являющихся отчуждаемыми носителями ключевой информации, не

имеют полноценных средств защиты, позволяющих считать их априорно стойкими к различным видам атак.

СПИСОК ЛИТЕРАТУРЫ:

1. Java Card Classic Platform Specification 3.0.4. URL: http://www.oracle.com/technetwork/java/ (дата обращения: 01.12.2013).
2. *McKeon B., Makewell R.* MULTOS. Technical Paper. URL: http://csrc.nist.gov/groups/STM/cmvp/documents/ (дата обращения: 01.12.2013).
3. ГОСТ Р ИСО/МЭК 7816-4-2004. Информационная технология. Карты идентификационные. Карты на интегральных схемах с контактами. Часть 4. Межотраслевые команды для обмена. Введ. 2005-01-01. М.: Изд-во стандартов, 2005. — 78 с.

REFERENCES:

1. Java Card Classic Platform Specification 3.0.4. URL: http://www.oracle.com/technetwork/java/ (data obrashcheniya: 01.12.2013).
2. *McKeon B., Makewell R.* MULTOS. Technical Paper. URL: http://csrc.nist.gov/groups/STM/cmvp/documents/ (data obrashcheniya: 01.12.2013).
3. GOST R ISO/MEK 7816-42004/ Informatsionnaya tekhnologiya. Karti identifikatsii. Karti na integralnikh skhemakh s kontaktami, Chast 4. Mezhotraslevie komahdi dlya obmena. Vved.2005-01-01/ M.: Izd-vo standartov, 2005. — 78 p.