А. А. Матросов
Московский инженерно-физический институт

ЗАЩИТА ПРИЛОЖЕНИЙ ПОСРЕДСТВОМ ВИРТУАЛИЗАЦИИ ЧАСТИ КОДА

Рассматривается проблема защиты интеллектуальной собственности для языков программирования, имеющих промежуточное представление. Предлагаемый способ защиты основывается на методе сокрытия части кода защищаемого приложения внутри «черного ящика» или альтернативной защищенной среды выполнения, которая позволяет существенно затруднить обратный анализ системы со стороны злоумышленника.

В последнее время все большую популярность среди разработчиков программного обеспечения приобретают языки программирования (.NET, JVM), имеющие промежуточное представление (байт-код). С точки зрения разработчика, это вполне оправданный выбор, так как эти технологии позволяют ускорить и упростить процесс разработки программного обеспечения, но здесь имеется такой существенный нюанс, как защита интеллектуальной собственности разработанного программного продукта. Анализировать код приложения в промежуточном представлении (байт-коде) гораздо проще, чем на машинном языке, отсюда возникает некоторая «простота» компрометации защитных механизмов такого приложения, так как модификация существующего кода тоже довольно несложный процесс. Усугубляет ситуацию и тот факт, что весь необходимый инструментарий для успешной компрометации можно без особых усилий найти в глобальной сети Интернет. Сгущают краски и сами поставщики средств разработки с промежуточным представлением, так как они включают в поставку дизассемблер, который может дизассемблировать приложение в гарантированно корректный код, после чего можно беспрепятственно вносить изменения и пересобрать листинг ассемблером, который также поставляется разработчиками.

В настоящей работе предлагается несколько иной подход к защите, чем в общепринятых средствах защиты программного обеспечения для платформ разработки с промежуточным представлением. Ключевая идея заключается в том, чтобы не предоставить для анализа злоумышленнику приложение с оригинальным промежуточным представлением (байт-кодом), для которого уже есть эффективные инструменты для компрометации, а защитить приложение таким образом, что ключевые алгоритмы для злоумышленника будут представлены в виде некоторого «черного ящика». Такая идея вполне реализуема, а основным фундаментом для ее реализации является перевод из уже существующего промежуточного представления в альтернативный вариант, который не будет знаком ни злоумышленнику, ни распространенным инструментам компрометации. В итоге при анализе злоумышленник будет всегда получать часть приложения в виде «черного ящика», так как альтернативное промежуточное представление будет уникальным для каждого защищаемого приложения. Среда выполнения для альтернативного промежуточного представления (защищенного байт-кода) обладает практически всеми свойствами «черного ящика» с точки зрения анализа и исследования механизмов ее работы злоумышленником. Таким образом, внутри «черного ящика» можно защитить ключевые алгоритмы, без которых программный продукт будет бесполезен. И такой подход позволит существенно затруднить исследование защищенного байт-кода, если предусмотреть в альтернативной среде выполнения (защищенное промежуточное представление) механизмы противодействия внешнему анализу и модификации со стороны злоумышленника.

Реализация идеи с «черным ящиком» строится на основе теории абстрактных стековых машин, на которых построены все системы разработки с промежуточным представлением. Основной идеей здесь является трансляция из одной системы команд в некоторую альтернативную систему команд, архитектурно очень близкую к оригиналу, абстрагируясь от микропроцессорной архитектуры путем использования универсальной теории абстрактных стековых машин. В итоге для каждого защищаемого приложения можно вырабатывать уникальную систему команд

альтернативного промежуточного представления, а также уникальную среду выполнения. После этого для компрометации предложенных выше защитных механизмов злоумышленнику для начала потребуется разобраться с альтернативной средой выполнения и реализовать самому весь необходимый инструментарий для ее исследования.

Сценарий защиты можно разделить на несколько основных этапов:

— трансляция защищаемого приложения в «родной» байт-код для среды выполнения, в которой выполняется оригинальное приложение;

— выделение участков кода, пригодных для защиты по описанной выше методике;

— трансляция оригинального байт-кода в уникальный альтернативный защищенный байт-код;

— генерация альтернативной среды выполнения для защищенного байт-кода.

Приведенный выше сценарий защиты является универсальным и подходит для защиты приложений под различные платформы разработки, имеющие возможность компиляции разрабатываемых приложений в промежуточный байт-код. При этом виртуальная среда выполнения, которая будет сгенерирована в процессе защиты, может быть двух видов:

— среда выполнения, реализованная посредством технологии Reflection.Emit (библиотека классов, существующая начиная со второй версии .NET Framework), которая сейчас активно применяется для построения среды выполнения средств разработки DSL (Domain Specific Language) языков;

— полностью замкнутая среда выполнения, реализованная целиком на языке программирования, компилирующемся в машинный код.

Первый способ хорош тем, что на его основе проще всего генерировать уникальную среду выполнения, так как он основывается на технологиях, которые являются общепринятыми для построения компиляторов среды выполнения .NET. Кроме того, при использовании первого способа отпадает проблема вызова библиотек среды выполнения .NET непосредственно из защищенного кода, так как в технологии Reflection.Emit такие механизмы заложены на уровне архитектуры. Но у этого, казалось бы, замечательного подхода есть один существенный недостаток, заключающийся в простоте обратного анализа реализованной по этому принципу защищенной среды выполнения.

Второй способ имеет существенные преимущества с точки зрения защищенности от обратного анализа защищенной среды выполнения, но он привносит значительные сложности в процесс реализации, так как здесь придется все реализовывать с нуля. Если в первом случае у нас уже имеется некоторый каркас для реализации, то здесь все придется реализовывать самим. Также в процессе реализации этого подхода есть концептуальная проблема вызова библиотек среды выполнения .NET, так как вызывать их придется из машинного кода. Это само по себе является существенной проблемой, так как библиотеки среды выполнения .NET реализованы полностью для управляемой (managed) среды выполнения. Но, несмотря на все недостатки этого подхода, построить защищенную среду выполнения, применяя подобную методику, проще в том плане, что машинный код анализировать гораздо сложнее.

Предлагаемый подход к защите программного обеспечения обладает рядом преимуществ, связанных в первую очередь с повышенной стойкостью такой защиты относительно предлагаемых на рынке решений. Еще одним преимуществом является возможность выполнения защищенного байт-кода на внешнем аппаратном модуле, так как эта особенность предусмотрена на уровне архитектуры защищенного байт-кода. Устроено это следующим образом: по требованию защищенный байт-код может быть оттранслирован в «родной» машинный код для аппаратной платформы, на которой будут выполняться защищенные участки кода. Следовательно, защищенный байт-код представляет собой некоторое языковое подмножество современных ассемблеров для процессоров RISC (Reduced Instruction Set Computing), но являющееся более абстрактным относительно аппаратной архитектуры. Это, в свою очередь, позволяет при необходимости осуществлять трансляцию непосредственно в ассемблерный код процессора, на котором будет происходить выполнение. На данный момент

предусмотрена трансляция в команды процессоров семейства ARM7 и ARM9, так как на их основе существует множество уже готовых конструктивов печатных плат для построения внешнего аппаратного модуля, выполняющего части кода защищаемого приложения.

В заключение хотелось бы отметить тот факт, что при защите приложений для современных платформ разработки с промежуточным представлением нужно разрабатывать новые концепции защиты, поскольку все то, что было придумано для защиты машинного кода, подразумевает совсем другие принципы функционирования среды выполнения для кода защищенного приложения и не подходит для защиты программ, использующих среду выполнения с промежуточным кодом. Поэтому при защите приложений с промежуточным представлением стоит особое внимание уделить непосредственно свойствам среды выполнения промежуточного кода и уже с учетом всей специфики среды выполнения разрабатывать защитные механизмы. Опираясь на приведенные выше основные концепции, можно построить довольно сложную систему защиты, ставящую серьезную преграду для злоумышленника, целью которого является компрометация защитных механизмов.

СПИСОК ЛИТЕРАТУРЫ

1. *Матросов А. А.* Защита приложений, имеющих промежуточное представление (байт-код), от несанкционированного использования и распространения // Проблемы информационной безопасности в системе высшей школы. М., 2006. С. 98–99.
2. *Хогланд Г., Мак-Граф Г.* Взлом программного обеспечения: анализ и использование кода. М., 2005.
3. *Ахо А., Ульман Д., Сети Р.* Компиляторы: принципы, технологии и инструментарий. М., 2001.
4. *Хопкрофт Д., Матвани Р., Ульман Д.* Введение в теорию автоматов, языков и вычислений. 2-е изд. М., 2002.
5. *Вольфенгаген В.* Категориальная абстрактная машина М., 2002.
6. *Крупский В.* Введение в сложность вычислений. М., 2006.
7. *Lidin Serge.* Inside Microsoft .NET IL Assembler, Microsoft Press 2002.
8. *Krall Andreas.* Efficient JavaVM just-in-time compilation. In Proceedings of the 1998 International Conference on Parallel Architectures and Compilation Techniques. Paris, France, October 1998.
9. *Collberg Cristian, Thomborson Clark and Low Douglas.* A taxonomy of obfuscating transformations. Technical Report 148, Department of Computer Science, University of Auckland. July 1997.