А.А. МАТРОСОВ
Московский инженерно-физический институт (Государственный Университет)

ЗАЩИТА ПРИЛОЖЕНИЙ ИМЕЮЩИХ ПРОМЕЖУТОЧНОЕ ПРЕДСТАВЛЕНИЕ (БАЙТ-КОД)

Целью данной работы является рассмотрение проблемы защиты интеллектуальной собственности для языков программирования имеющих промежуточное представление (.NET, JVM) и предложение одного из подходов ее решения. 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, однако в этом случае отдельной проблемой становится защита интеллектуальной собственности разработанного программного продукта. Дело в том, что анализировать код приложения в промежуточном представлении (байт-коде) гораздо проще, нежели 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. Ситуацию усугубляют сами поставщики средств разработки с промежуточным представлением, так как включают в поставку дизассемблер, который может дизассемблировать приложение в гарантировано корректный код, после чего можно беспрепятственно вносить изменения и пересобирать листинг ассемблером, который также поставляется разработчиками.

В данной статье предлагается к рассмотрению несколько иной подход к защите, чем в общепринятых средствах защиты программного обеспечения для платформ разработки с промежуточным представлением. Ключевая идея заключается в том, чтобы предоставить для анализа злоумышленнику не 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. Ключевой идеей здесь является трансляция из одной системы команд в некоторую альтернативную систему команд (архитектурно очень близкую к оригиналу) абстрагируясь от микропроцессорной архитектуры путем использования абстрактной стековой машины. В итоге мы можем для каждого защищаемого приложения вырабатывать уникальную систему команд альтернативного промежуточного представления, а так же уникальную среду выполнения. После этого для компрометации предложенных выше защитных механизмов злоумышленнику потребуется разобраться с альтернативной средой выполнения и реализовать самому весь необходимый инструментарий для ее исследования.

Сценарий защиты можно разделить на несколько основных этапов:

1) Трансляция защищаемого приложения в «родной» байт-код для среды выполнения, в которой выполняется оригинальное приложение.

2) Выделение участков кода пригодных для защиты по описанной выше методике.

3) Трансляция оригинального байт-кода в уникальный альтернативный защищенный байт-код.

4) 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.

Виртуальная среда выполнения, которая будет сгенерирована в процессе защиты, представлена в двух вариантах:

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

- второй вариант – это замкнутая среда выполнения, реализованная полностью на языке программирования компилирующегося 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 подхода есть один существенный недостаток, этот недостаток заключается 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 представляет собой существенные сложности в процессе 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. Кроме того в процессе реализации этого подхода существует проблема вызова библиотек среды выполнения .NET, так как вызывать их придется из машинного кода, а это является само по себе существенной проблемой, поскольку библиотеки среды

выполнения .NET реализованы полностью для управляемой (managed) среды выполнения. Но, несмотря на все недостатки этого подхода, построить защищенную среду выполнения применяя эту методику проще в том плане, что машинный код анализировать гораздо сложнее.

Данный подход к защите программного обеспечения обладает рядом преимуществ, связанных в первую очередь с повышенной стойкостью такой защиты относительно существующих 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.

В заключение хотелось бы отметить тот факт, что при защите приложений на базе современных платформ разработки с промежуточным представлением нужно разрабатывать новые концепции защиты. Механизмы защиты машинного кода подразумевают совсем другие принципы функционирования среды выполнения для кода защищенного приложения и не подходят 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. Выше были приведены основные концепции использования технологий виртуализации кода в процессе защиты приложений, оттранслированных в промежуточное представление. Опираясь на них можно построить довольно сложную систему защиты, которая составит серьезную преграду для злоумышленника, целью которого является компрометация защитных механизмов.

Список литературы:

1. 1. А. Ахо, Д. Ульман, Р. Сети. Компиляторы: принципы, технологии и инструментарий, Вильямс 2001.
2. Д. Хопкрофт, Р. Матвани, Д. Ульман «Введение в теорию автоматов, языков и вычислений – 2-е изд. » М.: Издательский дом «Вильямс», 2002.
3. В. Вольфенгаген «Категориальная абстрактная машина» М.: АО «Центр ЮрИнфоР», 2002.
4. Serge Lidin. Inside Microsoft .NET IL Assembler, Microsoft Press 2002.
5. Cristian Collberg, Clark Thomborson and Douglas Low. A taxonomy of obfuscating transformations. Technical Report 148, Departament of Computer Science, University of Auckland, July 1997.