ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЗАЩИТЫ ПРАВ ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Чеушев К.В.

Прохорчик Р.В. – магистр техн. наук, ассистент кафедры ПОИТ

Предложено оригинальное комплексное решение актуальной проблемы защиты Java-программ от несанкционированного использования. Комплексно применяется каскад средств защиты, которые в совокупности гораздо сложнее поддаются взлому, с переводом критических участков проверок на C, причём этот каскад встраивается автоматически в уже написанную программу, что уменьшает издержки её разработчика.

В условиях легкодоступности как копий программ, так и средств преодоления типовых защит их от несанкционированного использования для многих компаний-производителей ПО становится серьёзной проблемой обеспечить легальное, оплаченное использование своего продукта. Это особенно важно для компаний, чей продукт сложен и дорог в разработке, а тираж его невелик (продукт специфичен, и его рынок изначально мал). В таких условиях вырастает риск, что потребитель продукта может попытаться снять с него защиту (и это будет для него экономически оправдано), а для продавца каждая потерянная таким образом лицензия станет ощутимой утратой.

Наличие такого конфликта интересов породило соревнование, в котором одна сторона – злоумышленники – придумывает всё новые способы взлома, а другая – разработчики ПО – вынуждена изобретать всё более сложные механизмы защиты и противодействия взлому. Исходная позиция в этом соревновании состоит в том, что потребителю передаётся неким способом ключ, который он должен ввести в программу (или это делается автоматически), а код программы составлен так, что он работает некорректно без этого ключа. Но код программы злоумышленник может подвергнуть так называемой реверс-разработке, выяснить места учёта ключа и нейтрализовать защиту. Это особенно просто делается в случае написания программ, исходный код которых может быть восстановлен после компиляции, в частности это характерно для программ, написанных на языке программирования Java. Защита для программ на этом языке и обсуждается в работе.

При разработке средств защиты программ от несанкционированного использования важно учитывать следующее. Большие усилия по внедрению элементов защиты нередко могут быть нейтрализованы малыми усилиями злоумышленников, и наоборот. С другой стороны, насыщение программы элементами защиты может создать дополнительные сложности разработчику продукта, увеличить его издержки и снизить надёжность программы.

Поэтому главным критерием излагаемого подхода к решению задачи в конкретно поставленных условиях было соблюдение баланса. Учитывая не огромную, но и не самую скромную стоимость производимого и подлежащего защите продукта (а точнее класса продуктов, поскольку созданный инструментарий универсален в использовании), требовалось разработать такое решение, которое было бы просто в использовании, а затраты по взлому оказались бы невыгодными для потенциальных злоумышленников.

Исходя из позиционирования разрабатываемого инструмента было выдвинуто две ключевые идеи, в своей системности составляющие центральную идею решения:

– для противодействия реверс-разработке следует применить не просто несколько точек учёта ключа лицензии, а сложную сеть таких точек, и в целом каскад приёмов противодействия взлому;

– но поскольку такой каскад может создать ряд дополнительных проблем самим разработчикам продукта, изменения должны вноситься автоматически в уже созданный и отлаженный код продукта.

Таким образом, проектируется система защиты Java приложения от несанкционированного использования, такая, что элементы защиты будут автоматически вносится в защищаемую программу специальной утилитой. Предполагается, что при санкционированном использовании защищённой программы потребитель, оплативший лицензию, получает ключ. Введенный в программу, этот ключ переводит ее в состояние корректной работы.

Чтобы заставить программу работать без лицензии, злоумышленник может применить одну из двух стратегий:

– модифицировать программу таким образом, чтобы она корректно работала без ключа;

– создать собственный лицензионный ключ, каким-то образом определив структуру исходного ключа и способ его шифрования.

При реализации обеих стратегий он с большой вероятностью прибегнет к *реверс-разработке*, а именно будет анализировать код и логику программы, возможно, с применением специализированных инструментов. В первом случае он станет искать места условных переходов, зависящих от ключа, во втором – места трансформации ключа, которые позволили бы ему восстановить исходный ключ.

Соответственно, проектируя инструмент, нужно противодействовать обеим стратегиям потенциального взломщика: (1) предпринять усилия, чтобы взломщик не смог понять, как можно сгенерировать работающий ключ; (2) реализовать собственно защиту – условные переходы, которые запускают функциональность только при наличии ключа; (3) защитить проделанную работу от реверс-разработки.

Для решения первой задачи содержательные данные ключа – даты, контрольное значение и другие – перемежаются случайными данными и последовательно шифруются тремя ключами: идентификатором

устройства, идентификатором пользователя и закрытым ключом разработчика. Всё вместе это минимизирует риск того, что злоумышленник сможет сгенерировать корректный ключ.

Вторая и третья задачи решаются комплексно. Ключевой здесь была исходная посылка, что код, написанный на языке С, существенно труднее подвергается реверс-разработке, что в описанных выше условиях применения сводит к минимуму шанс злоумышленника довести взлом до конца. Поэтому в целом можно сказать, что средства защиты и противодействия взлому перенесены с Java на С каскадом из трёх основных решений.

1. Собственно элементы защиты спроектированы на языке С и утилита автоматически рассредотачивает их по коду методов Java. Но если бы в код Java просто вставлялись вызовы функций С, то такой способ защиты был бы легко обезврежен взломщиком. Поэтому в инструмент встроены средства перевода кода Java на С (не любого кода, а некоторого числа композиций). Утилита находит участки кода Java, которые она в состоянии перевести на С и делает это. И вот уже в этот участок вставляется код защиты и весь фрагмент оформляется как новая С-функция, а в Java-код вставляется её вызов. Теперь уже злоумышленник не может устранить этот вызов, потому что кроме проверки условий соблюдения лицензии функция делает полезную работу. Для реализации описываемой идеи пришлось проделать большую работу по сопоставлению синтаксического разбора для процессоров Java и С и реализовать собственный переводчик между этими языками для ограниченного числа допустимых фрагментов программы.

2. Вторая идея минимизирует риски отключения защиты даже если злоумышленник достиг стадии разбора С-фрагментов. Проверка условий соблюдения лицензии включает в себя несколько проверок параметров. И вместо того, чтобы реализовать их однотипно одной функцией (что допускало бы её нейтрализацию при глубоком погружении взломщика в код), для каждой проверки реализовано много вариантов её программирования, а затем все эти варианты ещё и автоматически перемешиваются, в итоге число возможных комбинаций достигает многих десятков, и в конкретное место (в переведенный с Java участок) будет подставлен фрагмент кода, реализующий один случайный из этих десятков вариантов. При таком способе взломщику придётся выискивать и нейтрализовывать все точки проверки условий.

3. В дополнении к вышеперечисленному выполняется ещё и стандартная защита от реверс-разработки – лексическая обфускация. На практике широко используется обфускатор Java ProGuard, который помимо обфускации также выполняет оптимизацию и удаление неиспользуемого кода, так что его применение было бы оправдано даже и без подмены имён, которые он осуществляет для затруднения реверс-разработки. Но он работает только с именами Java, а для применения в описываемом решении было бы надёжнее, если бы подменялись имена в вызовах С и в перекрёстных. Поэтому для совместного использования данного обфускатора и создаваемого инструмента, выполняется генерация конфигурационного файла программы ProGuard, в котором указаны новые имена для методов, вызываемых из С и функций, вызываемых из Java.

Предложено сбалансированное комплексное решение защиты Java-программ от несанкционированного использования. Сбалансировано оно в двух аспектах:

– решение достаточно сложное, для этого применяется каскад приёмов, которые в совокупности гораздо сложнее поддаются взлому, включая автоматический перевод участков защиты на язык С; но в то же время, оно не создаёт проблем разработчику продукта, поскольку этот каскад автоматически вносится созданным инструментом в уже готовый код продукта; это баланс в аспекте «сложность защиты – создаваемые этим проблемы»;

– второй аспект – баланс в координатах «стоимость защиты – стоимость взлома»; нельзя утверждать, что созданную защиту невозможно взломать, но стоимость такой операции окажется высокой, что при не очень высокой стоимости защищаемого продукта резко снижает риск того, что потребитель прибегнет к противозаконным действиям по нарушению прав интеллектуальной собственности.

Описанные решения реализованы в виде решений первой итерации, когда они ещё не составляют законченного инструмента, но пригодны для проведения экспериментов. Были проведены испытания и эксперименты с различными продуктами и проанализированы те материалы, которые в результате предстали для реверс-разработки.

Результаты этих испытаний и экспериментов подтверждают правильность исходных посылок и продуктивность предложенных решений. Действительно, в результате внесения в код программы каскада описанных выше средств защиты её взлом становится настолько трудоёмким, что это сводит на нет число потребителей, которые были бы готовы им воспользоваться. В то же время, все элементы защиты вносятся в Java-программу автоматически, и в ходе испытаний не зафиксировано ни одного отклонения защищённой программы от обычных параметров её функционирования.

На основе предложенного и проверенного экспериментами комплекса решений планируется создание универсального многокомпонентного программного средства защиты Java-приложений от несанкционированного использования.

Список использованных источников:

1. Керниган, Б. Язык программирования С / Керниган Б. У., Ритчи Д. М. – 2-е изд.– М.: «Вильямс», 2007 – 304 с.
2. Лав, Р. Linux. Системное программирование – СПб.: Питер, 2008 – 416 с.
3. Ретабоуил,С. Android NDK. Разработка приложений под Android на C/C++ – ДМК Пресс, 2014. - 496 с.
4. Панов А., Реверсинг и защита программ от взлома – БХВ-Петербург, 2006 – 256 с
5. Эккель, Б.Философия Java – Питер, 2011. – 640 с.