Выбор кода для размещения в ключе

Это самый ответственный и нетривиальный этап разработки загружаемого кода. Суть его заключается в том, что разработчику нужно решить, какой именно код будет выполняться в электронном ключе.

Код должен быть устроен таким образом, чтобы выполнять определенную конечную задачу, которая в общем виде выглядит так:

- Получение буфера с входными данными
- Вычисления над данными из этого буфера
- Возврат буфера с выходными данными

Существует целый ряд требований к этому коду, налагающих достаточно серьезные ограничения на выбор. Требования условно можно разделить на несколько типов:

Требования по безопасности

Загружаемый код должен быть достаточно сложным, чтобы брутфорс или иные (более эффективные и продвинутые) методы анализа черных ящиков не сделали возможным создания эмулятора в короткое время.

Этот код должен отсутствовать в более ранних версиях приложения. Несоблюдение этого условия делает возможным сравнение версий приложения и нахождение перенесенного кода для внедрения его в эмулятор.

Требования по производительности

Контроллер ключа обладает достаточно большими вычислительными возможностями, однако мощность его все же гораздо ниже, чем у современных процессоров. Поэтому важно, чтобы код не был слишком ресурсоемким, в противном случае время его выполнения может возрасти до неприемлемого уровня.

Кроме того, код, загружаемый в ключ, не должен вызываться слишком часто, например, в цикле. Если при единичном вызове задержка при выполнении не будет значительной, то при циклическом вызове она может оказаться очень существенной.

Требования по реализуемости

Код, размещаемый в электронном ключе, не должен:

- Содержать вызовов API, которые нельзя было бы перенестив электронный ключ, или иметь внешние зависимости
- Использовать потоки ввода-вывода
- Использовать вывод на консоль
- Использовать динамическое распределение памяти

Коду, исполняемому внутри ключа, доступны лишь стандартные библиотеки C и функции для работы с электронным ключом.

Ключи Guardant Code позволяют исполнять алгоритмы до 20 тысяч строк кода на C (до 60 тысяч строк в моделях ключей с увеличенным размером памяти). Соответственно, размер кода должен укладываться в эти пределы.

Основные рекомендации

1. Основная угроза для систем защиты, основанных на электронных ключах – это эмуляторы. Всегда существует шанс, что в результате анализа «черного ящика», которым является электронный ключ, злоумышленник сможет построить его эмулятор. Это может быть либо табличный эмулятор, либо универсальный эмулятор алгоритма.
   При анализе черного ящика злоумышленник будет пытаться определить, каким образом изменения входных данных повлияют на выходные данные. Если этих днных мало, либо если алгоритм слишком простой, с высокой вероятностью можно построить алгоритм, функционально эквивалентный «черному ящику».
   Поэтому алгоритм загружаемого кода должен быть нетривиальным и с достаточно большой сложностью. Он должен принимать на вход и давать на выход достаточно много данных.

2. Что делать, если функционально алгоритму требуется мало данных? Можно ввести дополнительные данные, не связанные напрямую с функциональностью алгоритма. По сути, это будут «мусорные» данные, однако их использование будет затруднять анализ «черного ящика». Аналитику придется разбираться, какие данные и для чего нужны, и нужны ли вообще.

3. Существует возможность неявного вызова аппаратных алгоритмов через загружаемый код. Например, можно возвращать данные из загружаемого кода зашифрованными на симметричном алгоритме AES128. Перед использованием они должны быть расшифрованы с

использованием того же секретного ключа. Это увеличивает разнообразие данных, проходящих через интерфейс электронного ключа, и затрудняет эмуляцию. Для большего усложнения можно периодически менять ключ шифрования AES128 изнутри загружаемого кода.

4. Защищенные ячейки можно использовать для неявного обмена данными между загружаемым кодом и приложением. В них можно помещать как часть входных данных для загружаемого кода, так и принимать через них данные от загружаемого кода.

5. Загружаемый код должен контролировать данные, принимаемые от приложения, по размеру и корректности. Это поможет избегать определенного класса атак.

6. При возможности можно использовать сохранение данных в ключе между вызовами, так как размер таблицы вопросов-ответов в этом случае получается на порядок больше. Очевидно, такой подход сильно затруднит процесс построения эмуляторов.

7. Если в защищаемом приложении нет фрагментов со значительными вычислениями (только такие разумно размещать в Guardant Code), можно использовать идею трудности анализа большого количества простых множеств. Т.е. построить защиту на том, что размещаемые в ключе фрагменты небольшие, они делают простые вещи, но зато их много. Правда, это, по-прежнему, должны быть фрагменты, которые используются не слишком часто. Количество фрагментов должно быть действительно большим. Их точное количество зависит от сложности вычислений в них. Например, если во фрагментах только простые операции сложения или вычитания – их должно быть около пары сотен. Если есть умножение или деление – достаточно несколько десятков. Если функции типа логарифмов или синусов /косинусов, достаточно 10. Данные цифры действительны на ближайшие несколько лет (стойкость всегда оговаривается сроками).

8. Поскольку загрузка кода в Guardant Code – операция настолько безопасная, что может осуществляться у конечного пользователя, и быстрая даже для больших кусков загружаемого кода, можно создать защищенную программу, которую будет так же легко обновлять, как и незащищенную программу. Для этого не потребуется что-либо специальное делать с электронным ключом при обновлении – достаточно просто отправить клиенту обновление.
   Более того, существует возможность даже не записывать в Guardant Code перед продажей загружаемый код! Это можно сделать в момент 1-го запуска программы, проанализировав значения, возвращаемые функцией**GrdGetCodeInfo()**:

   - Перед продажей программы записать в ключ маску без загружаемого кода
   - В защищенной программе добавить проверку – загружен ли код в Guardant Code (точнее какой именно код загружен – существуют поля версии загруженного кода). Если кода нет, или он не тот, то загрузить его в момент инициализации приложения. Это произойдет только один раз при 1-м запуске программы. Поскольку загружаемый код зашифрован и подписан ЭЦП, в него невозможно внести изменений и невозможно подменить другим кодом или кодом от другой программы
   - Поскольку бывают программы разной стоимости, и обновления тоже могут быть платными или бесплатными, нужно предусмотреть политику ограничения вариантов загрузки кода. То есть, надо сделать так, чтобы Update, предназначенный для дорогой программы, нельзя было загрузить в ключ, который был куплен для дешевой программы. Сделать это просто: ключи Guardant Code для разных программ должны иметь разные пары закрытых ключей для проверки ЭЦП загружаемого код

Нужно обратить внимание, что контроль версии загружаемого кода самим ключом не осуществляется. Это может сделать только программа, которая загружает код. Поэтому данный метод не годится для платных обновлений, в отличие от бесплатных. Не годится потому, что обновление можно загрузить бесплатно в любой ключ, который содержит необходимые ключи шифрования и подписи для загрузки кода. Бесплатные обновления традиционно служат для исправления ошибок в ПО. Платные обновления применяются чаще всего тогда, когда пользователь программы получает новые возможности. Для платных обновлений надо пользоваться механизмом TRU, т. к. он настроен для работы с конкретным ключом и позволяет реализовать процедуру адресного обновления.