Использование аппаратных алгоритмов ключей Guardant

Если вы остановили свой выбор на электронных ключах Guardant, в вашем распоряжении есть целый ряд эффективных приемов повышения стойкости защиты к взлому.

1. **Задействуйте аппаратные алгоритмы**. Это обязательное условие для построения по-настоящему стойкой защиты. Правильное использование ответов аппаратных алгоритмов не только делает практически невозможным написание универсальных эмуляторов ключей Guardant. Удаление из защищенного приложения вызовов функций API также теряет смысл – ведь если аппаратный алгоритм не был запущен, то и важные для приложения данные не были декодированы.

2. **Задавайте ключу больше вопросов**. Если для защиты используется преобразование только одного массива данных (т. е. если приложение задает ключу всегда один и тот же вопрос), есть вероятность, что хакер все же подсмотрит верный ответ ключа и создаст подпрограмму-«заглушку», которая вместо функции API «подсовывает» приложению этот ответ. Поскольку в данном случае ключ возвращает всегда один и тот же ответ, такая подпрограмма-«заглушка» может свести роль аппаратного алгоритма в защите данного приложения «на нет». Чтобы избежать этого, нужно задавать аппаратному алгоритму большое количество разных вопросов. Создайте массив различных вопросов (т. е. блоков кодированных данных), и в разных местах приложения декодируйте аппаратным алгоритмом разные кодированные блоки. Кстати, наряду с действительно важными для приложения данными в состав такого массива могут входить и «лишние» данные, на самом деле не нужные приложению – это только дезориентирует хакера. Очень хорошо было бы организовать процесс случайной выборки вопроса – тогда в одном и том же месте приложения, но в разные моменты времени будут обрабатываться разные вопросы. Это сделает практически невозможным для хакера создание, как подпрограммы-«заглушки», так и эмулятора ключа.

3. **Используйте разные вопросы в разных версиях приложения**. Если в разных приложениях или разных версиях одного и того же продукта будут использоваться разные вопросы к алгоритму (т. е. разные блоки кодированных данных), это даст гарантию того, что хакер не сможет создать универсальный инструмент для взлома всех продуктов (или всех их версий). Даже если хакер исхитрится-таки создать эмулятор для приложения, выход его новой версии заставит хакера производить свою работу заново – ведь новая версия приложения работает уже с другими кодированными данными

4. **Задействуйте разные аппаратные алгоритмы** в разных версиях приложения. В электронном ключе создайте сразу несколько различных алгоритмов (например, 4). Затем, в первой версии программного продукта кодируйте данные, скажем, 1-м и 3-м алгоритмом, во второй – 2-м и 4-м, в третьей – 1-м и 2-м, и т. п. Эффект будет аналогичен рассмотренному выше, да и в самом электронном ключе ничего перепрограммировать будет не нужно – все необходимые аппаратные алгоритмы будут в нем присутствовать с самого начала.

5. **Вносите изменения в определители алгоритмов**. По своей сути определители алгоритмов аналогичны паролям или цифровым сертификатам, поэтому их тоже полезно время от времени менять. Это очень хорошая и распространенная во всем мире практика, повышающая защищенность системы. В ключах с часами реального времени полезно использовать для этого **возможность автоматического изменения определителя каждые N дней**. Алгоритм смены определителя подробно описан в документации, разработчику лишь остается подготовить массивы кодированных данных на несколько месяцев или лет вперед. Как и в предыдущих примерах, однажды взломанная программа может внезапно перестать работать т. к. определитель алгоритма сам изменился в какой-то момент времени.

6. **Воспользуйтесь алгоритмом цифровой подписи**. Алгоритм ECC160 позволяет подписывать произвольные данные с помощью функции GrdSign. Лучше всего, чтобы это были данные, которые возникают в приложении естественным образом (так называемая естественная энтропия): данные, которые пользователь вводит в программу с клавиатуры, движения мышки, значения из баз данных, полученные по запросу пользователя. Тогда простая табличная эмуляция станет невозможной. Проверять правильность подписи можно с помощью функции GrdVerifySign. Однако не следует использовать простое сравнения возвращаемого ею значения, т. к. сравнение по принципу «да/нет» легко сломать с помощью т. н. битхака (исправляется всего 1 бит в программе). Лучше всего собирать возвращаемые значения (например, из битов возврата GrdVerifySign делать 32-битные числа), проверять подпись случайных данных (создавать ложные цели для анализа хакером) и на основе большого числа вызовов GrdSign/ GrdVerifySign создавать значения, которые в дальнейшем можно использовать в работе защищенного приложения.

7. **Не храните ответы аппаратных алгоритмов в защищенном приложении**. Сам принцип использования аппаратных алгоритмов основан на предпосылке, что хакер не знает заранее ответов алгоритма. В противном случае алгоритмы не имеют особого смысла: хакер, зная все его ответы, может создать и эмулятор ключа, и подпрограмму-«заглушку». Поэтому ни в коем случае не следует хранить ответы ключа в приложении или в файлах данных! Просто используйте ответы ключа по назначению, без предварительной проверки правильности их декодирования алгоритмом, а сразу после использования – по возможности удаляйте из памяти. Данные могут быть декодированы неверно лишь в одном случае – если либо с защищенным приложением, либо с электронным ключом производились какие-то сомнительные манипуляции. Но, в таком случае, неверная работа (и даже «подвисание») приложения – событие весьма естественное. В самом деле, чего еще можно ожидать от приложения, «ломаемого» хакером, особенно если защита «надета» на приложение качественно? Но если проверка правильности ответов все же необходима, используйте для этого функцию подсчета CRC. На этапе установки защиты, перед тем как закодировать данные, вычислите их CRC, а в процессе работы приложения подсчитайте CRC этих же данных после их декодирования алгоритмом ключа. Если эти два значения совпали, то данные декодированы верно.

Важная информация

В связи с постоянно растущими вычислительными возможностями современных компьютеров настоятельно рекомендуется задавать размер вопроса/ответа и определителя аппаратных алгоритмов ключей Guardant не менее чем 8 байт. Это позволит минимизировать шансы атаки методом brute force (полный перебор).