Особенности использования ключей Guardant Sign / Time / Code

При работе с электронных ключами Guardant необходимо учитывать несколько важных особенностей, связанных с активным использованием ассиметричной криптографии, как в алгоритмах ключа, так и в защитных механизмах. Эти моменты необходимо учитывать при проектировании защиты.

1. **Безопасный обмен сеансовыми ключами**. При выполнении функции GrdLogin происходит взаимная аутентификация электронного ключа и Guardant API и безопасный обмен сеансовыми ключами. Слово «обмен» - это просто термин. На самом деле API и электронный ключ рассчитывают это значение независимо на основе безопасной ассиметричной схемы. В протоколе обмена его подсмотреть невозможно. Поэтому команда GrdLogin занимает существенно больше времени, чем остальные.

2. **Периодическая смена сеансовых ключей**. Если хакеру удастся как-то извлечь из защищенного псевдокодом Guardant API сеансовый ключ, то это тоже не принесет ему много пользы. Электронный ключ устроен так, что он работает на любом сеансовом ключе не больше нескольких минут. По его истечении, электронный ключ начинает возвращать специальный код ошибки в ответ на любую команду, полученную с использованием просроченного сеансового ключа, до тех пор, пока данная копия API опять не пройдет взаимную аутентификацию и не перегенерирует его заново. Соответственно, при проектировании защиты нужно учитывать тот факт, что эта перегенерация может занять какое-то дополнительное время. И если логика работы приложения не допускает таких задержек в выполнении, то лучше организовывать запросы к электронному ключу из параллельно работающих потоков.

3. **Использование GrdSign и GrdVerifySign**. Это очень мощный инструмент, который при правильном применении дает возможность создать защиту беспрецедентно высокого уровня. Для этого нужно во время работы защищенной программы аппаратно подписать (GrdSign) и программно проверить подпись (GrdVerifySign) от неких каждый раз разных случайных данных. Проверка подписи специально реализована чисто программно, так как это лишает хакера даже гипотетической возможности перехватить ее в эмуляторе на уровне драйвера или USB-шины. А ключ шифрования в ней используется публичный, который не может быть скомпрометирован.

4. **Атака на генератор случайных чисел**. Если хакер сможет сделать в электронный ключ запросы не случайными, а постоянными, то у него появится возможность сделать табличный эмулятор, который будет знать ответы на все вопросы, задаваемые программой. Чтобы посылаемые данные были действительно случайными, настоятельно рекомендуется использовать какой-либо хеш (например, SHA-256), от действительно случайных данных, которые использует программа и без которых она теряет смысл или львиную долю функциональности. Например, введенные пользователем данные, ID созданных потоков, адреса аллокированной памяти, и т. п. Иначе хакер сможет подсовывать приложению любые нужные ему значения в ответ на любые вызовы Guardant API.

5. **Атака на подмену публичного ключа**. Хотя публичный ключ ассиметричного алгоритма не является секретным, но, тем не менее, необходимо принять все меры по предотвращению его модификации и/или подмены.
   a. Не хранить его просто инициализированным в сегменте данных (на Си это инициализированная глобальная или статическая переменная или константа), так как там его проще всего подменить
   b. Рассчитывать его значение непосредственно перед использованием и сразу после использования затирать и удалять.
   c. Контролировать целостность кода, который работает с ним.

6. **Атака на подмену Guardant API**. Чтобы хакер не мог просто заменить код функций Guardant API на свой эмулятор, необходимо:
   a. Вызывать GrdVerifySign так, чтобы она в некой части возвратов возвращала другие коды ошибок, которые также надо проверять. Например, подавать на вход случайные данные в сообщении или подписи или неверный общий ECC ключ.
   b. Среди этих вызовов очень полезно время от времени вызывать GrdVerifySign с запросом на проверку валидности подписи одного из валидных сообщений, подписанного на этом же ключе, но не в процессе работы программы, а на этапе защиты. Для этого при защите приложения в нем необходимо сохранить какое-то количество подписанных случайных сообщений с подписями. Аналогично тому, как это делается с различными массивами вопросов и ответов, используемыми при работе с симметричными алгоритмами.
   c. Результат проверки подписи GrdVerifySign по сути один бит. Можно вместо того, чтобы ставить явную проверку кода возврата, составить из множества вызовов одну константу, которая будет использоваться в дальнейших вычислениях. Благо скорость работы подписи в ключе достаточно высока.

7. **Использование однонаправленных симметричных алгоритмов шифрования AES и GSII64**. Можно, например, сделать так, чтобы в электронном ключе для защиты приложения некий AES или GSII64 алгоритм работал в обоих направлениях, а в попадающем к конечным пользователям ключе он мог только расшифровывать. Тогда можно хранить в некой защищенной ячейке (или файле) какую-либо конфигурационную или лицензионную информацию, зашифрованную на этом алгоритме при продаже софта не боясь, что хакер сможет подменять ее. Ведь для подмены нужно записать в эту ячейку информацию, зашифрованную на этом алгоритме. А зашифровать ее на ключе пользователя, на этом же алгоритме хакер просто не сможет, из-за его однонаправленности. Аналогичные механизмы можно применять с распространением зашифрованных обновлений, которые хакер, имея ключ конечного пользователя с однонаправленным алгоритмом, не сможет зашифровать, модифицировать и подсунуть программе.

> Однонаправленные алгоритмы AES128 можно использовать только в блочных режимах ECB и CBC. Поточные режимы CFB и OFB использовать нельзя.

8. **Атака на замену GrdAPI.DLL**. Обычно настоятельно рекомендуется всем нашим пользователям по возможности использовать статическую линковку, т. е. по сути, использовать Guardant API в виде OBJ вместо DLL. Если же это невозможно из-за ограничений средств разработки, то крайне рекомендуется сделать следующее:
   a. Перенести часть функциональности во внешнюю DLL, к которой статически прилинкована библиотека Guardant API (OBJ).
   b. Использовать традиционные механизмы работы с симметричными алгоритмами. Массивы вопросов и ответов, случайный выбор таблицы и запроса. Проверки при выполнении какой-либо функциональности.
   c. Время от времени вызывать GrdVerifySign с запросом о проверке валидности подпись одного из валидных случайных сообщений, подписанного на этом же ключе, но не в процессе работы программы, а на этапе защиты. Если хакер даже перехватит все вызовы Guardant API, то он не будет знать, что вернуть на такой вопрос.

9. **Time-ключи**. Появилась возможность надежно задавать ограничение работы, как всего электронного ключа, так и любого конкретного алгоритма. Это дает возможность сдавать ПО в аренду. Однако рекомендуется уведомлять конечного пользователя о близости прекращения работа программы заранее, для того, чтобы он успел продлить лицензию до того, как программа перестанет работать. Для этого в Guardant API есть все необходимые функции типа GrdPI. См. документацию на Guardant API. Аналогичные опции предусмотрены в новом мастере автозащиты.

10. **Счетчик запусков**. Аналогично, в Sign-ключах появилась возможность получить оставшееся значение счетчика запусков алгоритма. Соответственно, о приближении его завершения нужно информировать конечного пользователя заранее. Для этого есть специальная функция типа GrdPI_. См. документацию на Guardant API. Эти же опции предусмотрены в новом мастере автозащиты.

11. **Используйте драйвер Guardant**. Хотя использования ключей в HID-режиме очень удобно для распространения защищенной программы, все же наличие драйвера Guardant увеличивает защищенность, и по возможности мы рекомендуем все же использовать ключи не в HID-режиме. Тем не менее, надо обратить внимание, что программа, защищенная ключами в HID-режиме, в любом случае обеспечивает защиту гораздо более высокую, чем предыдущие поколения электронных ключей.

12. **Используйте Time-ключи не только для ограничения времени работы, но и для повышения уровня защищенности**. В отличие от других типов Time-ключей, электронные ключи Guardant Time можно использовать для повышения уровня защищенности. Такая уникальная возможность как BirthTime позволяет создавать такие системы защиты, которые активируются спустя время и анализировать которые хакер никак не может заранее. А технология FlipTime позволяет создавать алгоритмы, определители которых меняются через заданное количество дней, что опять затруднит анализ защиты заранее. Утилита для генерации определителей FlipTime также включена в комплект разработчика вместе с исходным кодом.

13. **Используйте высокую скорость работы новых ключей**. Guardant Sign / Time / Code работают до 10 раз быстрее, чем их предшественники. Поэтому с их помощью можно шифровать и расшифровывать гораздо больше информации, и это не отразится на быстродействии защищенной программы.

14. **Защита баз данных**. Можно реализовать вариант защиты, при котором база данных шифруется с помощью программного AES. При продаже база комплектуется электронным ключом, содержащим аппаратный алгоритм AES с теми же параметрами, как у программного алгоритма, на котором база данных была зашифрована. Отличием алгоритма в электронном ключе будет то, что он работает только в режиме расшифрования (без зашифрования) и расшифровывает не всю базу данных, а некую ее часть, нужную пользователю. В этом случае, содержимое базы данных нельзя будет модифицировать.

    Этот вариант подходит для случаев, когда у конечного пользователя нет необходимости пополнять базу данных.