АППАРАТНАЯ ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ – КЛЮЧИ КОМПАНИИ SafeNet

А.Тархов

Согласно данным IDC (International Data Corporation), в 2005 году в России 83% используемого программного обеспечения (ПО) было нелицензионным, а убытки от компьютерного пиратства составили 1,63 млрд. долларов. Неудивительно, что средства защиты ПО от нелегального использования так востребованы в нашей стране. Одно из наиболее эффективных средств защиты – аппаратные ключи. О них и пойдет речь в настоящей статье.

В течение последних лет аппаратные средства защиты ПО – LPT-, а затем и USB-ключи, – неоднократно усовершенствовались. В своем развитии они прошли не один этап: от простейших элементов памяти до сложных микропроцессорных устройств, которые реализуют разнообразные алгоритмы проверок, основанные на современных методах шифрования.

Один из лидеров на рынке средств защиты ПО – компания SafeNet. Компания производит такие известные аппаратные средства защиты ПО, как USB-ключи SuperPro и UltraPro. SafeNet не остановилась на достигнутом и выпустила новый аппаратный ключ – Sentinel Hardware Key (SHK) (рис.1). Ряд принципиально новых решений, воплощенных в этом ключе, позволяет с уверенностью заявить: сегодня SHK – это наиболее надежное средство защиты ПО.

ЗАЩИТА ОБМЕНА ДАННЫМИ МЕЖДУ SHK И ПО

Одна из наиболее значимых инноваций SHK – реализация сразу двух алгоритмов, которые сегодня фактически являются стандартами в мире криптографии: AES (Advanced Encryption Standard) и ECC (Elliptic Curve Cryptography). В SHK эти алгоритмы применяются, в частности, для защиты всего потока данных между аппаратным ключом и приложением. Пересылаемые данные шифруются с помощью высокопроизводительного AES-алгоритма. Для каждой сессии обмена данными генерируется 128-битный сеансовый криптоключ. В этом случае применяют метод ECKAS-DH1, основанный на ECC-алгоритме. Защищенный коммуникационный туннель предотвращает все атаки типа "запись-воспроизведение". Суть таких атак сводится к записи "обращений" приложения к ключу и "ответов" ключа на них. Затем полученная информация используется для эмуляции ключа. Так как всякий раз поток данных шифруется с помощью нового сеансового криптоключа, запись этого потока просто бессмысленна. Во-первых, при попытке эмуляции ключа неизвестно, что именно запрашивает ПО в данный момент, а следовательно – что нужно "ответить". Во-вторых, подстановка даже "правильного" ответа не даст положительных результатов, так как расшифровка будет производиться с помощью нового сеансового криптоключа.

АТАКИ НА АППАРАТНЫЙ КЛЮЧ

Современные технологические решения, применяемые в SHK, предотвращают атаки низкого уровня, то есть атаки на сам аппаратный ключ. Внутреннюю прошивку ключа, задаваемую перед передачей ключа разработчику, невозможно не только считать, но и изменить. Таким образом удается избежать подмены ключей разных разработчиков. Кроме того, монтаж элементов SHK выполнен по технологии chip-on-board, которая затрудняет анализ внутреннего устройства ключа и таким образом препятствует реконструкции алгоритмов его работы.

Рис.1. Аппаратные ключи SHK

Атаки высокого уровня, то есть атаки на защищенное приложение, предотвращаются с помощью двух предлагаемых методов защиты: автоматической и интегрированой, а также их комбинации.

АВТОМАТИЧЕСКАЯ ЗАЩИТА – SENTINEL SHELL

Автоматическая защита реализована на базе технологии Sentinel Shell. Данная технология позволяет оперативно защитить приложение без изменения его исходного кода. Суть этой защиты сводится к сжатию исполняемого файла или динамической библиотеки из состава ПО. Сжатие защищаемого файла усложняет его декомпиляцию. С помощью комплекта разработчика (ToolKit) в защищаемое ПО встраиваются дополнительные команды для взаимодействия с ключом. Они проверяют наличие "правильного" ключа и выполняют распаковку файлов. В отсутствие SHK файлы и библиотеки не распаковываются и ПО не запускается. Кроме того, встроенные команды периодически проверяют наличие ключа и выполнение лицензионных ограничений (например, временных) в процессе работы защищенного приложения. Если ключ не обнаружен во время очередной проверки, приложение закрывается.

В исполняемый файл может быть также встроен механизм, препятствующий использованию отладчиков для "взлома" защиты. В этом случае ПО не будет запускаться в режиме отладки.

Sentinel Shell оснащена инструментарием Shell SDK. С его помощью можно указывать фрагменты кода, которые должны быть дополнительно зашифрованы. Приложение с "помеченными" фрагментами компилируется, а затем защищается Shell (с активированием опции Shell SDK). Во время защиты указанные фрагменты шифруются. Соответственно, реконструировать данные и алгоритмы по дампу памяти становится намного труднее.

ИНТЕГРИРОВАННАЯ ЗАЩИТА НА БАЗЕ API

Интегрированная защита реализуется с помощью API-вызовов (API – application programming interface), которые встраиваются в исходный код приложения. API-вызовы позволяют обращаться к различным API-элементам, записанным в ключ. В качестве API-элементов могут выступать, например, строковые (длиной до 256 символов), целочисленные (длиной до 32 бит) и булевы константы, счетчики, AES- и ECC-алгоритмы. Разработчик выбирает API-элементы и их параметры в соответствии с планируемой структурой защиты. Например, можно перенести различные константы из кода приложения в память ключа. Значения констант при использовании берутся не из самой программы, а из памяти ключа. В этом случае защищенное приложение не сможет работать без "нужного" ключа, так как ему не будет хватать требуемых данных.

Значения API-элементов, содержащих константы, с помощью соответствующего API-вызова могут быть не только считаны, но и перезаписаны, например, для привязки ПО к конкретному компьютеру. В SHK для каждого такого API-элемента разработчик может задать отдельный пароль разрешения записи. Это позволяет избирательно подходить к предоставлению прав на изменение значений различных элементов.

Рис.2. Задание параметров API-элемента типа ECC

В API-элементы, соответствующие AES- и ECC-алгоритмам, разработчик может записать криптоключи, используемые для шифрования и расшифровывания посылаемых в SHK данных с помощью соответствующих API-вызовов (рис.2). Можно также задавать количество разрешенных запусков защищаемого приложения, дату окончания его действия, время действия с момента первого обращения. Данные API-элементы могут быть неактивными, что позволяет организовать помодульное лицензирование приложения: для каждого модуля формируют отдельный AES- или ECC-элемент, который для неоплаченного модуля "программируется" как неактивный.

Усилить защиту ПО позволяет совместное применение защиты на базе API с защитой на основе Shell-технологии. В этом случае сначала в исходный код приложения встраивается набор API-вызовов, а затем ПО защищается с помощью Shell.

Каждый разработчик сам выбирает модель защиты и способы ее программной реализации. Поэтому избранная система становится фактически уникальной, что повышает ее надежность.

ШАБЛОНЫ И ГРУППЫ ЛИЦЕНЗИЙ

Допустим, приложение многомодульное и каждый модуль лицензируется отдельно. Каким образом в этом случае запрограммировать ключи для разных клиентов, учитывая, что им поставляются наборы лицензий с разными ограничениями по включенным модулям, количеству сетевых пользователей, времени действия лицензии и т.д.

Создатели SHK позаботились о том, чтобы формировать конфигурацию лицензий для многомодульных приложений было удобно. Сначала разработчик ПО создает *шаблоны лицензий* (например, SAPR на рис.3). Каждый элемент шаблона – это API- или Shell-элемент, который применяют для лицензирования и защиты модулей или отдельных функций внутри модулей ПО. Shell-элемент – это набор параметров, определяющих настройки защиты на базе Shell (например, Shell-элемент SimpleDraft, входящий в шаблон SAPR – см. рис. 3).

На этапе создания шаблона задают значения лицензионных ограничений, которые будут использоваться по умолча-

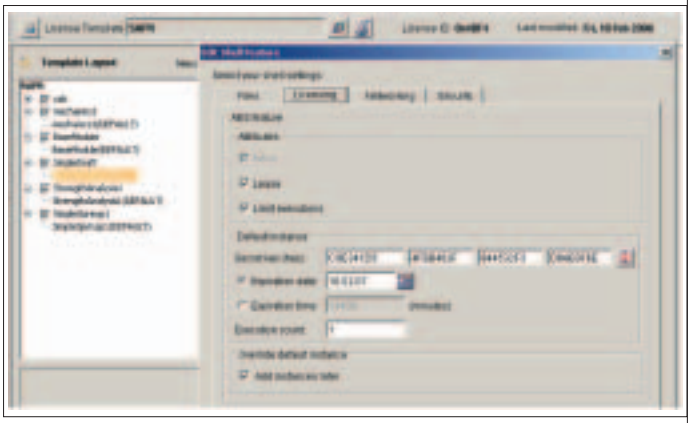

Рис.3. Формирование шаблона лицензий. Описание параметров Shell-элемента

нию. Стоит отметить, что количество сетевых пользователей задается на весь шаблон. Это означает, что элементы, входящие в один шаблон, будут "расходовать" сетевые лицензии совместно. Так, шаблон SAPR содержит 6 элементов, и каждый из них лицензирует отдельный модуль ПО. Если для этого шаблона количество сетевых пользователей равно, скажем, 10, то они смогут одновременно запустить не более 10 копий приложений, лицензируемых этим шаблоном, например: Calc, Calc, Mechanics1, BaseModule, BaseModule, SimpleDraft, SimpleDraft, StrengthAnalisys, StrengthAnalisys, SimpleSprings.

Затем формируют *группы лицензий*, в которые входят требуемые шаблоны. Например, *группа* Enhanced Configuration (рис.4) состоит из трех *шаблонов*: CAM System, SAPR, SAPR Advanced Module. Группирование позволяет задействовать один SHK для защиты и лицензирования нескольких программных пакетов, соответствующих шаблонам. При этом на каждый шаблон может быть задано свое собственное максимальное количество сетевых пользователей.

Когда группа создана, в нее может быть добавлен новый шаблон или исключен внесенный ранее. Кроме того, введенные в группу шаблоны, а также элементы шаблона можно исключать из записи в ключ. Для этого достаточно просто снять флаг напротив соответствующего элемента или шаблона при программировании конкретного ключа.

На этапе формирования группы можно также вносить изменения в параметры лицензионных ограничений (дата окон-

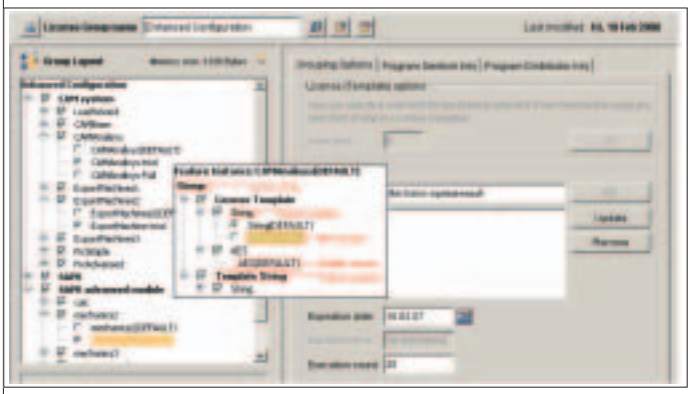

Рис.4. Формирование группы лицензий из нескольких шаблонов

чания действия лицензии и т.д.) для каждого элемента шаблона. Для этого создают альтернативный вариант настроек элемента. Такие варианты позволят менять параметры лицензий без изменения самого шаблона. Например, в шаблоне SAPR Advanced Module задано две альтернативы настроек элемента mechanics2: mechanics2(default) со стандартными настройками (заданными при формировании шаблона) и mechanics-временный с другим набором параметров (рис. 4). Альтернативные варианты настроек сохраняются и могут быть использованы при последующем программировании ключей.

Отметим, что при программировании ключей автоматически формируется отчет в стандартном XML-формате. В отчет вносится вся информация об операциях программирования ключа: дата и время программирования, серийный номер ключа, названия группы, шаблонов и элементов, а также используемые альтернативы настроек. Такой отчет можно импортировать в офисные приложения, используемые для управления взаимодействием с заказчиками.

ЗАЩИТА ОТ ПОДМЕНЫ СИСТЕМНОГО ВРЕМЕНИ

При выдаче временных лицензий или сдаче ПО в аренду необходимо предотвратить подмену системного времени. Сделать это можно с помощью технологии V-Clock. В память SHK заносится значение параметра LKDT (last known date and time) –

последние корректные дата и время. С этим параметром связан счетчик подмен системного времени cheat counter, значение которого задается разработчиком (оно может быть и нулевым, что вообще запрещает сдвигать назад системное время). При работе ПО системное время сверяется с LKDT. Если системное время опережает LKDT, то этот параметр обновляется. Если же системное время отстает от LKDT в интервале от 1 секунды до 30 дней, то значение счетчика cheat counter уменьшается на 1. Как только значение cheat counter достигает нуля, алгоритмы AES и ECC блокируются. Обе технологии защиты (автоматическая и интегрированная) основаны на этих алгоритмах. Поэтому защищенное ПО перестает работать или может быть переведено в режим с ограниченным функционалом. Если же системное время отстает более чем на 30 дней, то алгоритмы блокируются независимо от состояния счетчика cheat counter. Такой подход позволяет предотвратить подмену системного времени без дополнительных затрат на встроенные часы и элементы питания.

Существует и другая модификация Sentinel Hardware Key – SHK RTL (Real Time Clock) со встроенными часами и элементом питания. Такой ключ позволяет проверять время и дату на основе независимых показаний внутренних часов.

УДАЛЕННОЕ ОБНОВЛЕНИЕ

Использование демонстрационных лицензий, лицензий с ограничением на количество запусков, а также продажа ПО с частично неактивными модулями связаны с вопросом: как изменить параметры лицензии? Как продлить ее действие, увеличить количество разрешенных запусков, активизировать дополнительные модули, увеличить число сетевых пользователей и т.д.? При работе с SHK такие действия можно выполнять удаленно. Для этого применяют утилиту SecureUpdate, которая поставляется вместе с защищенным ПО. При выполнении удаленного обновления эта утилита считывает информацию о лицензии из ключа и сохраняет ее в зашифрованном виде, формируя код запроса. Код запроса передается разработчику, например по электронной почте, и загружается в ToolKit, где генерируется код обновления, который возвращается к пользователю. Код обновления загружается в SecureUpdate, и на основе кодовой информации изменяется лицензионная информация в ключе. Таким образом, лицензия может быть обновлена буквально за считанные минуты без физической передачи ключа разработчику.

Надежность процедуры удаленного обновления обеспечивается рядом технических решений:

- Коды запроса и обновления шифруются с помощью AES-алгоритма, криптоключ которого уникален для каждого разработчика. Он записывается во все SHK перед их передачей разработчику и не может быть считан и изменен даже самим разработчиком.
- Пользовательский SHK устроен таким образом, что каждый код обновления может быть использован только один раз.

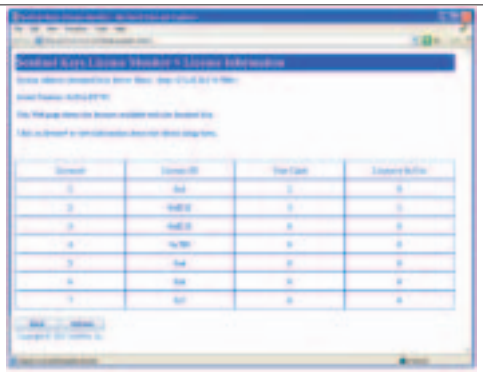

Рис.5. Мониторинг сервера лицензий

- Код запроса и код обновления являются парными. Следовательно, код обновления может быть использован только для SHK, с которого был сформирован его "парный" код запроса.

ЗАЩИЩЕННАЯ ДИСТРИБУЦИЯ

Обычно крупные софтверные компании имеют разветвленную дистрибьюторскую сеть с многочисленными региональными представительствами. В этом случае возникает вопрос, каким образом организовать поставку защищенного ПО через партнеров. Классический вариант – передача дистрибьютору нескольких комплектов дистрибутивов и запрограммированных пользовательских ключей. Однако такой вариант крайне неудобен, так как дистрибьютор не может формировать конфигурацию ПО в соответствии с требованиями конкретного заказчика. Второй вариант – передать дистрибьютору незапрограммированные пользовательские ключи, ToolKit и проект защиты. Этот вариант неприемлем, так как дистрибьюторов невозможно проконтролировать. Никто не может поручиться, что пользовательский ключ будет запрограммирован с тем набором модулей, который заявлен дистрибьютором в отчетности. С SHK эту ситуацию разрешить очень просто с помощью уникальной технологии дистрибьюторского ключа.

Суть концепции заключается в следующем. Разработчик формирует дистрибьюторский проект – файл, содержащий группу лицензий для ПО. Группа может состоять из разных шаблонов и описывать множество модулей или отдельных приложений. В дистрибьюторский ключ записывается криптоключ AES-алгоритма, с помощью которого зашифрован передаваемый дистрибьютору проект. Это гарантирует невозможность использования проекта без дистрибьюторского ключа. При расширении продуктовых линеек разработчик может передать дистрибьютору дополнительные группы лицензий, зашифрованные с помощью того же AES-алгоритма.

Дистрибьюторский ключ содержит также дистрибьюторские лицензии, число которых определяет разработчик. Каждая такая лицензия позволяет запрограммировать пользовательский ключ на одного сетевого пользователя (например, программирование ключа на 17 пользователей расходует 17 дистрибьюторских лицензий). Таким образом, дистрибьюторская сеть находится под контролем: каждый дистрибьютор получает свой набор лицензий и отчитывается за него (в том числе и финансово) перед разработчиком. Эта модель удобна еще и потому, что дистрибьюторский ключ может быть удаленно обновлен с использованием утилиты SecureUpdate – точно так же, как и пользовательский ключ. Таким образом может быть изменено количество дистрибьюторских лицензий.

АППАРАТНЫЕ МОДИФИКАЦИИ SHK

Ключи SHK имеют три не взаимозаменяемые аппаратные реализации:

- Пользовательский ключ предназначен для конечного пользователя. Он программируется разработчиком или дистрибьютором и защищает ПО.
- Дистрибьюторский ключ используется разработчиком для контроля дистрибьюторов и дистрибьюторами при программировании пользовательских ключей.
- Ключ разработчика – применяется разработчиком защищаемого ПО. Он содержит уникальный криптоключ. Криптоключ используется при программировании пользовательских и дистрибьюторских ключей и при выполнении удаленных обновлений. Не имея доступа к ключу разработчика, невозможно запрограммировать ни один ключ, сгенерировать код обновления и изменить шаблоны лицензий.

Ключи SHK могут быть двух типов: сетевые и локальные. Локальные ключи выдают только одну лицензию и должны быть подсоединены к тому компьютеру, на котором запускается защищенное ПО. Сетевые ключи могут одновременно выдавать лицензии нескольким компьютерам по локальной сети. В этом случае один компьютер выбирается как сервер лицензий, на котором устанавливается программа Sentinel Keys Server и подключается SHK.

Мониторинг сервера лицензий выполняют с помощью web-браузера (рис.5). При этом выдается полная информация о выбранном сервере лицензий: общее количество доступных лицензий, число используемых лицензий, серийный номер ключа, а также детализация по каждой лицензии и информация о пользователях. Системный администратор имеет полномочия на отключение выбранных пользователей, перераспределяя таким образом ресурсы ПО между сотрудниками компании.

В заключение отметим, что в 2006 году номинантами 21-й ежегодной премии CODiE в категории за лучшее DRM (Digital Rights Management) решение стали два продукта SafeNet – Sentinel RMS и Sentinel Hardware Keys. Организатором этой премии является ассоциация Software & Information Industry Association (SIIA). Премия присуждается за достижения в сфере программных, информационных и образовательных технологий.