Эссе студента 018 гр. Улькина Григория Александровича

Достоинства и недостатки систем аппаратной защиты (на примере ключей HASP)

Вводная часть

Данное эссе посвящено теме защиты компьютерных программ от нелегального копирования с использованием программно-аппаратных средств защиты. Почти каждый человек знает, что проблема пиратства программного обеспечения с каждым годом обостряется всё больше. Связано это в первую очередь с тем, что доходы от сферы написания программных продуктов растут из года в год. Более того, не смотря на серьёзное развитие рынка программных продуктов, некоторые компании просто понятия не имеют о том, что защита программных продуктов действительно необходимая вещь.
Ни для кого не секрет, что большинство атак на программные продукты связано с изменением программного кода времени выполнения программы. Иными словами, при помощи специальной утилиты – отладчика – можно отследить в коде ассемблера любые моменты «жизненного цикла» программы. В том числе и в те моменты, в которые происходит выявление принадлежности пользователя к числу авторизованных для использования программы пользователей (например, ему выдаётся специальный регистрационный номер после покупки программы, который он вводит в поля одной из форм программы). В этом случае процедуру сверки ключа и прочее можно исследовать на предмет содержимого регистров и после «смоделировать» успешный исход авторизации, занося в регистры или в память по нужному адресу заведомо верное значение и делая безусловный переход в коде ассемблера в обход процедуры авторизации. Этот метод используется повсеместно, и, к сожалению, перед ним достаточно сложно устоять. Всё, что в этом случае нужно злоумышленнику – это как максимум – одна легальная копия программы + хороший отладчик. Некоторое время можно было этому противостоять при помощи специальных программ – упаковщиков. Они всего лишь изменяли ассемблерный код с целью минимизации размера программы, однако, как потом выяснилось, это ещё и неплохой метод защиты. Читать такой код – сложнее, соответственно и анализировать. Но и это не могло остановить человека, который желает взломать целевой программный продукт.

Именно поэтому стали применять качественно новый метод защиты программ - при помощи аппаратных ключей. В качестве таковых средств - здесь рассматриваются аппаратные ключи HASP. Аббревиатура HASP: Hardware Against Software Piracy (HASP). HASP – это современная система защиты программного обеспечения от незаконного использования, построенная на аппаратной основе и предотвращающая несанкционированные доступ к защищенным программам и их исполнение.

Основой ключей HASP является специализированная заказная микросхема (ASIC – Application Specific Integrated Circuit), имеющая уникальный для каждого ключа алгоритм работы. Принцип защиты состоит в том, что в процессе выполнения защищённая программа опрашивает подключённый к компьютеру ключ HASP. Если HASP возвращает правильный ответ и работает по требуемому алгоритму, программа выполняется нормально. В противном случае, по Вашему усмотрению, она может завершаться, переключаться в демонстрационный режим или блокировать доступ к каким - либо функциям. Таким образом, всё «в руках» программиста.

Метод защиты состоит в “привязывании” защищаемой программы к уникальному ключу, распространяемому вместе с программой. Ключи, как правило, подключаются к LPT-, USB-, PCMCIA портам компьютера пользователя, и работоспособность программы зависит от присутствия этого ключа. Различают ключи с памятью и без, сетевые версии и версии с встроенным таймером. Существует довольно большое число разновидностей таких ключей. В нашей стране наибольшей популярностью пользуются ключи HASP компании Aladdin Software Security.

Следует выделить достоинства технологии HASP:

- **Специализированное аппаратное обеспечение**
  Основу всех ключей HASP (исключая USB-HASP) составляет патентованная специализированная микросхема (ASIC – Application-Specific Integrated Circuit). Чип, выполненный по 1,2-микронной технологии, содержит 2800 логических элементов, что делает практически невозможным обратный инжиниринг и “взлом” аппаратной части HASP. Чип имеет сложную внутреннюю организацию и нетривиальные алгоритмы работы. Логику работы чипа практически невозможно реализовать с помощью стандартных наборов микросхем PAL, GAL или PEEL, его очень сложно воспроизвести, а содержащийся в его памяти микрокод - считать, расшифровать либо эмулировать. Чип программируется только с использованием специальной платы Crypto Programmer Card, после чего позволяет шифровать данные блоками длиной 64 бит с ключом, длиной 48 бит, причем для каждого нового блока ASIC-чип генерирует новый сеансовый ключ. Количество комбинаций при кодировании - 248. Напряжение питание ASIC-чипа - всего 1.5V, поэтому он устойчиво работает на всех портах в любых режимах. А малый ток потребления позволяет объединять в каскад практически любое количество ключей. Особенностью ключей USB-HASP является наличие совершенного микроконтроллера, обеспечивающего очень высокий уровень защиты.

- **Улучшенные алгоритмы защиты и антиотладочная технология**
  HASP осуществляет защиту 16- и 32-разрядных приложений и связанных с ними файлов данных в "прозрачном" режиме. При чтении данные автоматически расшифровываются, при записи - зашифровываются с использованием заданного аппаратно реализованного алгоритма. В программном обеспечении HASP использованы новейшие патентованные алгоритмы защиты кода, а также наиболее мощная на сегодняшний день антиотладочная технология.

- **Шифрование связи с ключом**
  Чтобы обезопасить канал обмена между программой и ключом HASP, все передаваемые данные кодируются по случайному закону. Таким образом исключается программная эмуляция аппаратного ключа – один из наиболее часто применяемых способов взлома.

Целью данного эссе является изучить предоставляемые ключами этой фирмы возможности защиты программного обеспечения и рассмотреть некоторые уязвимости таких защит. Также описываются методы, которые могут помочь конечному программисту защитить своё приложение от попыток взлома.

Устройство HASP
Обзор методов защиты

Ключи HASP предоставляют два различных метода защиты программ: автоматически — с использованием утилиты HASP Envelope, вручную — с помощью HASP API. Рассмотрим применение каждой из них.

Утилита HASP Envelope предназначена для защиты уже готовых программ без вмешательства в исходный код программы. При защите данной утилитой тело программы шифруется, в нее добавляется специальный модуль, который при запуске защищенной программы перехватывает управление и содержит специальные антиотладочные и антитрассировочные средства. Таким образом, реализовать ранее описанную уязвимость ассемблерного кода уже намного сложнее. Среди преимуществ такого метода – простота его применения и малый объём затрат. Однако он практически никогда не позволяет настроить функциональность работы связки «программа-ключ» настолько гибко, насколько бы это хотелось конечному пользователю.

В свою очередь, специальный API представляет собой мощнейший механизм защиты, посредством которого программа осуществляет вызовы функций обращения к ключу, маскируя их в теле программы или DLL. Функция API позволяет в любой момент проверять наличие ключа и предпринимать определённые шаги при его отсутствии.
С применением API весь аппарат для контроля поведения программы уже находится в руках у программиста. Главное назначение API — предоставление всего комплекса возможностей по защите программ, которыми обладает ключ.

Обзор уязвимостей

Итак, приступим к рассмотрению уязвимостей данного метода защиты. Главная задача – это «заставить» программу, защищенную HASP API или HASP Envelope, работать в условиях отсутствия легального ключа, подсоединенного к компьютеру. Совершенно очевидно, что любая защищенная программа состоит непосредственно из самой программы и механизмов проверки ключа. Причем эти механизмы, как правило, не являются составными частями алгоритма самой программы, поэтому и их удаление не повлияет на работоспособность программы. Под удалением защитных механизмов здесь не всегда понимается удаление в прямом смысле, гораздо чаще требуется изменить их так, чтобы программа не обнаружила изменения, а сами механизмы при этом работали бы без ключа. В любой программе механизмами защиты являются некоторые функции, но, чтобы с ними разобраться, сначала их нужно найти, что не всегда является простой задачей в большом объеме кода. Примем, для примера, что управлением состоянием программы в зависимости от присутствия или отсутствия ключа заведует некоторая секретная абстрактная функция hasp(). Существует несколько способов нахождения вызовов функций hasp():

Первый способ позволяет обнаруживать вызов функции hasp() по используемым этой функцией несекретным вспомогательным функциям (которые запросто можно «подцеплять» из библиотек динамической компоновки и исследовать уже при помощи специально написанных программ). Пусть одной из таких функций является Vulneurable_Function(), поэтому для обнаружения вызова функции hasp() ставим точку останова на обращение к функции Vulneurable_Function(). После того, как Vulneurable_Function() будет найдена, следует проверить, откуда она была вызвана, и решить, является ли функция, вызвавшая ее, функцией hasp() или это

некоторая другая функция. Таким образом, данный метод позволяет сильно сократить число функций, которые претендуют на то, чтобы являться функцией hasp().

Как этому противостоять:

Очевидно, что чем больше будет вызовов hasp() из самых разных функций, тем сложнее определить, hasp() это была или нет. Поэтому следует использовать как можно больше вызовов процедуры hasp() для противодействия атакам на Ваше защищенное приложение. Сложные многочисленные вызовы hasp() создадут серьезные затруднения для взломщика в понимании Вашей схемы защиты и атаках на нее.

Второй способ работает непосредственно с защитным механизмом. Не изменяя логики работы программы, он заставляет защитный механизм, не обращаясь к ключу, возвращать ожидаемые программой значения. Этот метод является универсальным как для защит с использованием HASP API, так и с использованием HASP Envelope. Он не требует нейтрализации множества ловушек для отладчика, присутствующих в теле программы. Это стало возможным благодаря тому, что точка входа в HASP API оставлена открытой. Отыскать ее просто: достаточно найти в тексте программы строку “HASPDOSDRV” (или нечто схожее, в общем, заинтересованный – найдёт); следующая за ней осмысленная команда, после некоторого, ничего не значащего набора байт, будет прыжком на то место, где происходит вызов функции. Сразу возникает целесообразная мысль: Написать свою функцию, которая будет возвращать «правильные» результаты и подменить её. Все, что необходимо далее сделать, это – заменить найденный вызов функции на вызов функции, эмулирующей работу ключа.

Третий метод был найден в Internet. Данный способ сопровождался подробным описанием того, как можно отыскать функцию hasp() немного нестандартным, но действительно 100% методом, которым не стоит пренебрегать:

Как бы функции hasp() ни прятались, в конечном счете - вызов ключа это электрический сигнал, подаваемый на LPT-порт. Поскольку защита использует аппаратные средства, то и взлом может выполняться с применением тех же средств. Все, что необходимо для этого – несколько светодиодов, подключенных к LPT-порту, и любой отладчик, позволяющий выполнять пошаговую трассировку программы. Метод состоит в пошаговом проходе программы отладчиком без вхождения в процедуры до тех пор, пока не произойдет интересующее нас событие (вспышка светодиодов) во время выполнения одной из процедур, затем необходимо углубиться в эту процедуру и т.д. Этот метод здесь наиболее уместен ввиду того, что защитный механизм не может, как в случае чисто программной защиты, разделить во времени проверку условия и реакцию на нее. Эти два способа позволяют обнаружить вызовы ключа, после чего работа ведется с самой программой, а именно с изменением ее реакции на отсутствие ключа.

Приведенный анализ позволяет сделать следующий вывод: защиту программ с помощью ключей HASP в существующем варианте вряд ли можно признать достаточно надёжной. Но, тем не менее, следует признать, что ключи HASP, пожалуй, одно действительно эффективное средство защиты программ.

Список литературы:

[1] Руководство программиста HASP4 / Aladdin Knowledge Systems, 2002.
http://www.aladdin.ru/solutions/hasp/hasptechdetails/deveperhowto.pdf

[2] FAQ Сайта: http://www.alladin.ru
http://www.aladdin.ru/solutions/hasp/hasptechdetails/deveperhowto.pdf

[3] Законы о защите интеллектуальной собственности
http://www.sibai.ru/law.php?ID=56

[4] Новое поколение электронных ключей HASP
http://www.aladdin.ru/solutions/hasp/HASP4.pdf