**Задание на 10.02.2022**

МДК 03.02 Программно-аппаратные средства защиты информации

1. **Сделать конспект лекции**
2. **Сделать практическую работу №20 Настройка файрволла Comodo**
3. **Отчет отправить на почту** [**svebalch@mail.ru**](mailto:svebalch@mail.ru)
4. **СРС:** [**https://stepik.org/course/94299/promo**](https://stepik.org/course/94299/promo) **пройти онлайн курсы на платформе Степик. Тема: Кибербезопасность**

**Лекция 21**

**ПРОГРАММНЫЕ И АППАРАТНЫЕ**

**СРЕДСТВА ЗАЩИТЫ КОМПЬЮТЕРНЫХ**

**ИНФОРМАЦИОННЫХ СИСТЕМ.**

**Защита программ от несанкционированного копирования.**

Все средства защиты информационных систем можно разделить на аппаратные и программные. Программные средства защиты, реализуются чисто софтверным путем, в них не затрагиваются физические характеристики носителей информации, специальное оборудование и т.п.

Аппаратные средства защиты, используют специальное оборудование (например, электронные ключи, подключаемые к портам компьютера) или физические особенности носителей информации (компакт-дисков, дискет), чтобы идентифицировать оригинальную версию программы и защитить продукт от нелегального использования.

Аппаратные средства защиты могут быть реализованы в виде специальных модулей, устанавливаемых в слоты расширения материнской платы компьютера, либо подключаемых к последовательному или параллельному порту. Эти модули могут содержать однокристальные микро-ЭВМ или специальные заказные микросхемы, выполняющие обмен кодовыми последовательностями с программой. Можно также использовать специальные версии BIOS.

**1. Подходы к защите информационных систем**

***Устойчивость к прямому копированию***

Основой любой защиты можно считать ее способность к идентификации носителя, с которым она попала к пользователю. И не просто к идентификации, а к способности отличить данный носитель от нелегальной копии. Причем, уровень защиты на данном этапе должен быть такими, чтобы та условная метка или характеристика, которая была присуща данному носителю, не воспроизводилась любыми средствами битового копирования. Эффективность данного этапа определяет стойкость защиты к элементарному копированию, когда пользователю достаточно запустить CloneCD, и не о чем больше не думать. Следует отметить, что многие системы защиты используют различные физические метки, по которым программа сможет идентифицировать подлинность носителя. Как правило, физически установленные метки плохо копируются, но неплохо эмулируются специальными драйверами, что делает защиту, основанную только на физических метках, очень уязвимой. Еще, как правило, все системы защиты представляют собой «вставную челюсть». То есть модуль, который отвечает за идентификацию диска, в большинстве случаев не может противостоять хакерской атаке, в то время как менее важные участки кода оказываются хорошо защищенными.

Наиболее эффективным можно считать способ, при котором на диск не наносятся специальные метки, то есть, когда диск спокойно можно копировать, и распространять его содержимое, но старт будет производиться только при наличии оригинального диска.

Подобные защиты основываются на том факте, что любой диск (CD/DVD/R/RW) имеет ряд уникальных характеристик, присущих только одному диску, и эти характеристики теряются при копировании на другой диск. Их реализация в конкретных системах (Star-Force, Tages) остается тайной за семью печатями.

***Устойчивость к взлому***

Второй по счету, но не по важности компонент защиты — это устойчивость к взлому. Если защиту не удается обойти копированием и эмулированием, хакеру необходимо при помощи механизмов дизассемблирования и пошаговой отладки просканировать приложение, выделить логику защиты и нейтрализовать ее. Соответственно, если у защиты имеется надежный способ работы с метками (или с характеристиками носителя), но модуль идентификации не защищен должным образом, то подобную систему нет смысла использовать, так как она не выдержит даже минимальной хакерской атаки.

***Аппаратные ключи***

Часто для реализации нераспространения продукта при его эксплуатации используются электронные ключи (HASP) или Sentinel. HASP представляет собой программно-аппаратный комплекс, содержащий код, процедуры или любые другие уникальные данные, по которым защита может идентифицировать легальность запуска.

Электронный ключ — это современное средство защиты от незаконного копирования программ и несанкционированного доступа к ним. Защита состоит в том, что программа не работает в отсутствие электронного ключа, к которому она привязана. Сейчас электронные ключи становятся все более функциональными и предоставляют разработчику все больше разнообразных возможностей защиты программного обеспечения и данных. К преимуществам использования электронных ключей можно отнести возможность создания дополнительного уровня защищенности ПО. Если сравнивать просто программный механизм защиты и аппаратный, можно заметить, что, используя доступные методы программной защиты, можно усилить защищенность ПО, просто добавив механизм защиты электронных ключей, как дополнительный барьер для взломщика. Электронный ключ обычно представляет собой небольшое устройство (размером со спичечный коробок), которое присоединяется к компьютеру через один из. возможных интерфейсов (обычно это: СОМ, LPT или USB порты, хотя бывают варианты и в ISA-card исполнении).

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

Большинство моделей ключей HASP имеют энергонезависимую программно-перезаписываемую память (так называемую EEPROM). В зависимости от реализации HASP память может быть от одного до четырех килобитов).

Наличие энергонезависимой памяти дает возможность программировать HASP, размещая внутри модуля различные процедуры, либо хранить дополнительные ключи, а также:

• управлять доступом к различным программным модулям и пакетам программ;

• назначать каждому пользователю защищенных программ уникальный номер;

• сдавать программы в аренду и распространять их демо-версии с Ограничением количества запусков;

• хранить в ключе пароли, фрагменты кода программы, значения переменных и другую важную информацию.

У каждого ключа HASP с памятью имеется уникальный Опознавательный номер, или идентификатор (ID-number), доступный для считывания защищенными программами. Идентификаторы позволяют различать пользователей программы. Проверяя в программе идентификатор HASP, пользователь имеет возможность предпринимать те или иные действия в зависимости от наличия конкретного ключа. Идентификатор присваивается электронному ключу в процессе изготовления, что делает Невозможным его замену, но гарантирует надежную защиту от повтора. С использованием идентификатора можно шифровать содержимое памяти и использовать возможность ее дистанционного перепрограммирования

Система HASP позволяет защищать программное обеспечение двумя различными способами: автоматически (стандартно) и вручную (через специальный АРI).

**2. Структура системы защиты от несанкционированного копирования**

В общем случае система защиты от несанкционированного копирования представляет собой комплекс средств, предназначенный для затруднения (предотвращения) нелегального копирования защищаемого программного модуля, с которым она ассоциирована.

Структура системы защиты от несанкционированного копирования включает: подсистему внедрения управляющих механизмов, подсистему реализации защитных функций, блок установки, блок сравнения, блок ответной реакции.

Подсистема внедрения управляющих механизмов представляет собой комплекс программных средств, предназначенный для подключения внедряемого защитного кода к защищаемому программному модулю. Внедряемый защитный код — это программный модуль, задача которого состоит в противодействии попыткам запуска (исполнения) нелегальной копии защищаемой программы.

Подсистема реализации защитных функций представляет собой программную секцию, решающую задачу распознавания легальности запуска защищаемой программы.

Подсистема противодействия нейтрализации защитных механизмов предназначена для борьбы с возможными попытками нейтрализации системы защиты от несанкционированного копирования и/или ее дискредитации.

Блок установки характеристик среды отвечает за получение характеристик, идентифицирующих вычислительную среду.

Блок сравнения характеристик среды устанавливает факт легальности запуска защищаемой программы.

Блок ответной реакции реализует ответные действия системы защиты на попытки несанкционированного исполнения защищаемой программы.

Поскольку стойкость системы защиты определяется стойкостью каждого ее элемента, то в качестве объекта атаки может использоваться любая из описанных подсистем. Здесь необходимо отметить неоднородный уровень как самих идей, лежащих в основе той или иной подсистемы, так и их реализаций, что, в первую очередь, связано с развитием приемов, методов и средств для нейтрализации систем защиты. Учитывая современное состояние вопроса, наиболее актуальной задачей является разработка подсистемы внедрения управляющих механизмов системы защиты и подсистемы установки характеристик среды, хотя остальные подсистемы должны быть разработаны не менее тщательно. Показательным примером является блок ответной реакции, который может, как просто выводить сообщение о незаконности копии, так и предпринимать более сложные действия, позволяющие на определенное время замаскировать наличие защиты, увеличивая тем самым время атаки. Но если функционирование блока ответной реакции может влиять на надежность системы лишь косвенным образом, то зачастую самым слабым местом всей системы является блок сравнения характеристик среды, и именно против него в первую очередь направлены атаки злоумышленников.

Системы защиты от несанкционированного копирования можно классифицировать по способу внедрения защитного механизма:

— встроенная (внедряется при создании программного продукта);

— пристыковочная (подключается к уже готовому программному продукту).

Наибольшую популярность в последнее время приобрели системы второго типа.

Это обусловлено рядом преимуществ, которые дает их использование:

— простота тиражирования программных систем защиты на объекты заказчика и разработчика;

— простота технологии применения; обеспечение достаточного уровня защищенности данных в силу специализации разработчиков;

— более оптимальное соотношение «надежность функционирования/затраты на разработку» по сравнению со встроенными системами, подготовленными непрофессионалами. Рассмотрим способы установки защитных механизмов в защищаемые программные модули. Одним из вариантов встраивания пристыковываемого модуля в исполняемый модуль является дописывание его по вирусному принципу.

При этом код защиты дописывается в некоторую область защищаемого файла и защищаемый файл модифицируется таким образом, чтобы управление передавалось на пристыкованный модуль защиты, который проверяет летальность копии, и в случае положительного ответа передает управление на исполняемый модуль.

Такой подход к внедрению в исполняемые файлы имеет ряд существенных недостатков. Во-первых, исходный код защищаемой программы остается практически в неизменном виде, что значительно упрощает нейтрализацию защиты. Во-вторых, если предполагается защищать файлы большого размера и, как следствие, со сложной (может быть, и с оверлейной) структурой, то необходимо учитывать то, что код, находящийся в конце, загружен не будет, а значит, программный модуль не будет отработан. Некоторые недостатки можно устранить, если писать в начало программы не одну команду перехода, а весь код защиты. При этом необходимо модифицировать таблицу перемещения (которая находится в конце заголовка ЕХЕ - файла), в случае если защищается файл типа ЕХЕ. Основным недостатком защит такого типа является то, что они могут быть нейтрализованы динамически, путем определения момента, когда защитная часть уже отработала и начал выполняться сам защищаемый код. Примером такого подхода к нейтрализации защит являются программы SNAPSHOT и INTRUDER. Основная идея метода, реализованного в подобных программах, заключается в двукратной загрузке исследуемого модуля по разным адресам памяти с последующим вычислением всей необходимой информации на основе анализа дампов памяти (под необходимой информацией здесь понимается заголовок ЕХЕ - файла, содержащий всю информацию, которая требуется операционной системе для загрузки программы). Другим важным недостатком описанных методов внедрения является то, что существующие защитные механизмы данного типа не могут обеспечить корректную защиту самомодифицирующихся программ, которые в процессе выполнения изменяют свой образ, хранящийся на диске. Исходя из указанных недостатков, можно сформулировать следующие требования к пристыковываемым модулям:

- пристыковываемый модуль должен подключаться к файлам любого размера;

— результирующий файл, полученный после подключения пристыковываемого модуля, должен быть устроен таким образом, чтобы максимально затруднить выделение исходной защищаемой программы;

— пристыковываемый модуль не должен накладывать ограничений на функционирование защищенной программы, в частности, он должен позволять модифицировать в процессе работы свой собственный дисковый файл.

***Блок установки характеристик среды***

Если структура системы защиты от несанкционированного копирования практически не зависит от применяемых способов защиты, то конкретная реализация блока установки характеристик среды зависит от многих параметров, среди которых можно выделить способ предполагаемой организации распространения программного обеспечения.

В общем случае программное обеспечение может распространяться:

— бесплатно (из альтруизма и соображений саморекламы);

— условно бесплатно (по принципу «попробуй и купи» (try and buy), когда оплата производится добровольно и только тогда, когда пользователь соглашается с реальной пользой для себя, данного продукта);

— на коммерческой основе.

Последний случай (распространение программного продукта на коммерческой основе) предусматривает наличие защиты, которая может включать или не включать в себя технические меры. При наличии технических мер защиты производитель может распространять свой программный продукт тремя основными способами, которые определяют конкретную реализацию блока установки характеристик среды:

с помощью специальной службы распространения;

— через торговые организации;

через свободное распространение дистрибутивных (демонстрационных) пакетов с последующей регистрацией.

При наличии специальной службы распространения контроль за дистрибутивными носителями вместо технических мер осуществляется организационными мерами. Сотрудники службы распространения выезжают с дистрибутивными комплектами к заказчикам, где производят установку программного обеспечения на жесткий диск. Поскольку наличие у пользователя резервных копий не предусматривается, то в случае сбоя требуется повторный выезд сотрудника службы распространения, что обычно рассматривается как определенное неудобство. При данном способе блок установки характеристик среды должен уметь идентифицировать только параметры компьютера, благодаря чему разработка блока несколько упрощается. В случае распространения программных продуктов через торговые организации возможны следующие варианты:

программа ассоциирует (связывает) себя с дистрибутивным носителем без «привязки» к конкретному компьютеру;

— программа ассоциирует себя со специальным аппаратным устройством, подключаемым к компьютеру и входящим в дистрибутивный комплект;

программа ассоциирует себя как с дистрибутивным носителем (при инсталляции), так и с параметрами компьютера (в рабочем режиме).

Первые два варианта удобны тем, что позволяют переносить защищенные программы с компьютера на компьютер, но как плату за это требуют постоянного присутствия либо ключевой дискеты, либо специального аппаратного устройства.

Свободный от этих недостатков третий вариант требует, помимо обеспечения надежности ключевой дискеты, решения далеко не простой проблемы счетчика инсталляций. достаточно интересен третий способ распространения программного обеспечения — посредством свободного распространения дистрибутивных (демонстрационных) пакетов. Суть этого способа лучше всего описать следующим сценарием. Пользователь, все равно каким способом, получает программу и запускает ее на своем компьютере. При запуске ему сообщается некоторый код, который вместе с квитанцией об оплате он должен передать разработчику. В ответ ему сообщается другой код, являющийся паролем для регистрации копии программы и ассоциации ее с компьютером. По такому способу, например, защищен некогда популярный текстовый процессор «Слово и дело». Таким образом, в зависимости от выбранного способа распространения программного продукта, блок установки характеристик среды должен уметь идентифицировать параметры компьютера, дистрибутивного носителя, либо специального аппаратного устройства.

***3. Защита дискет от копирования***

Дискета, предназначенная для установки защищенного от копирования программного обеспечения, должна быть сама защищена от копирования.

Копирование дискет можно выполнить как по файлам (с помощью команд операционной системы COPY или XCOPY), так и по секторам (командой DIKCOPY, программами PCTOOLS, PCSHELL и аналогичными).

Кроме того, существуют программы, специально предназначенные для копирования дискет, защищенных от копирования, например COPY2PC, TeleDisk. Специальные программы могут копировать дискеты, содержащие только определенные защищенные программные пакеты, или они могут повторять структуру дорожек диска с точностью до бита.

Наиболее просто обеспечить защиту от программ копирования дискет по секторам. Можно предложить следующие достаточно простые способы, использующие нестандартное форматирование отдельных дорожек дискеты:

- форматирование отдельных дорожек с размером секторов, отличным от стандартного для MS-DOS, например, 128 или 1024 байт;

- создание дорожек за пределами рабочей зоны диска, например, создание 41 дорожки для дискеты емкостью 360 Кбайт или 81 дорожки для дискеты емкостью 1,44 Мбайт;

- создание большего, чем стандартное, количества секторов на дорожке;

- форматирование отдельных дорожек с использованием фактора чередования секторов с последующим анализом времени доступа к секторам для обычных стандартных дорожек и для нестандартных дорожек;

- использование нестандартного символа заполнения при форматировании.

Очевидно, что все эти способы непригодны для защиты от таких программ копирования, которые способны копировать битовую структуру дорожек диска.

В этом случае можно использовать специальную аппаратуру при записи установочных дискет, которая позволяет записывать отдельные дорожки или секторы как бы с промежуточным уровнем записи. Эти участки дорожки будут читаться нестабильно.

Если скопировать такую дискету на обычной аппаратуре (с использованием обычных НГМД и программ битового копирования), то все дорожки будут читаться стабильно. Если при многократном контрольном чтении указанных секторов или дорожек каждый раз будут получены разные данные — мы имеем дело с оригиналом, в противном случае — с незаконной копией.

Однако дискеты с промежуточным уровнем записи все-таки могут быть скопированы с использованием специальной аппаратуры, копирующей содержимое дорожек «аналоговым» способом (как в бытовом магнитофоне).

для защиты от аналогового копирования можно использовать дискеты, где — в некоторых местах искусственно созданы дефекты магнитного покрытия — выжженные лазером небольшие точки или просто царапины. Проверка основывается на том, что в дефектные места невозможно ничего записать. Если мы имеем дело с копией, то на месте дефектных секторов окажутся хорошие — копируется только информация, но не дефекты дискеты!

Можно использовать комбинации различных методов защиты от копирования. При этом легко распознаваемые методы (нестандартный размер сектора и т. п.) можно использовать для маскировки какого-либо другого, более тонкого метода защиты.

Самое простое, что можно сделать для того чтобы защитить установочную дискету от копирования — изменить размер секторов на дорожке.

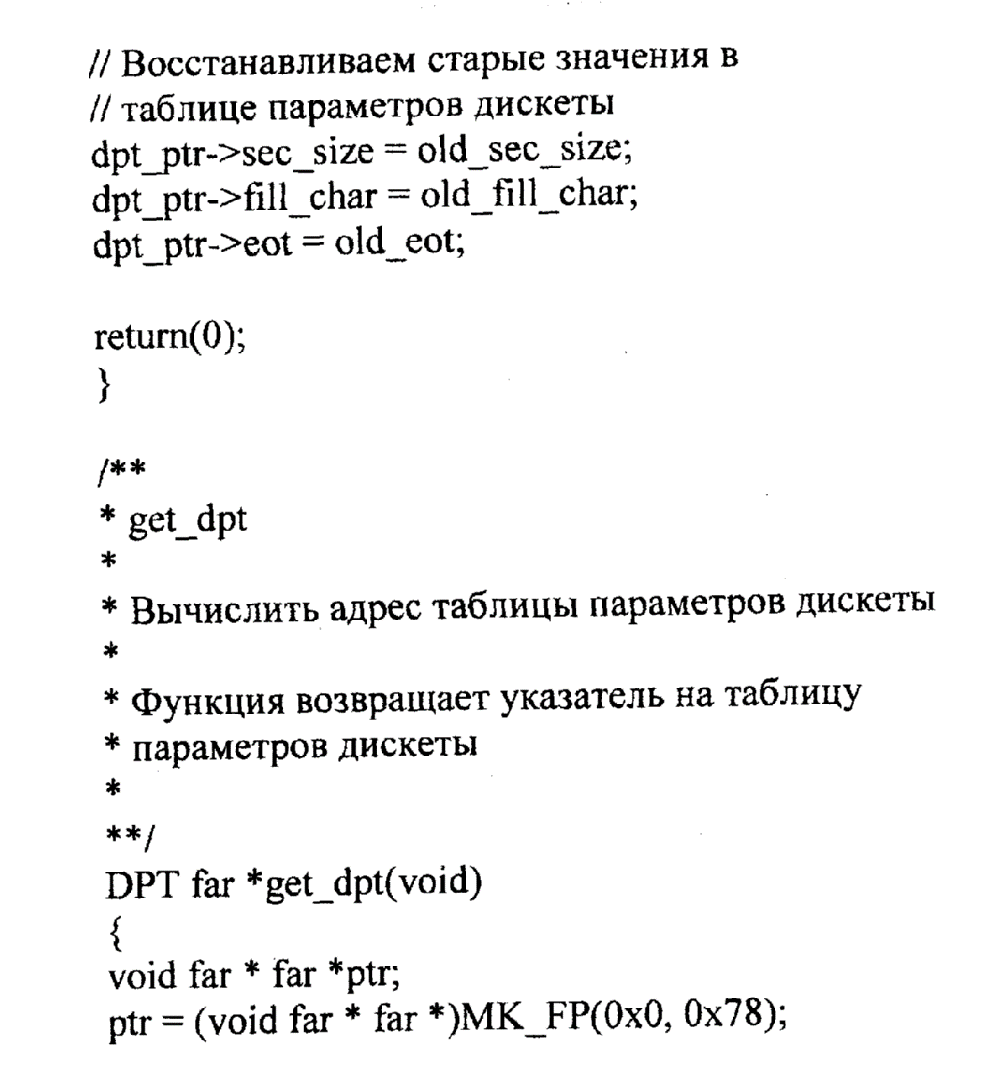
Приведем простую программу FMT256, которая форматирует двадцатую дорожку диска емкостью 1,44 Мбайт в устройстве А:, создавая на ней секторы размером 256 байт.

После форматирования программа записывает в первый сектор нестандартной дорожки строку, введенную с клавиатуры. Затем для контроля содержимое этого сектора считывается и отображается на экране. Также обратим внимание на изменения в таблице параметров дискеты они необходимы для использования нестандартного размера сектора.

Какую информацию можно записать в нестандартный сектор?

Если вы делаете установочную (инсталляционную) дискету, которая рассчитана на ограниченное количество установок, нестандартный сектор — самое подходящее место для хранения счетчика установок. Даже такие программы, как Norton Disk Editiotor не помогут прочитать или изменить значение этого счетчика. В этот же сектор можно записать и другую информацию, необходимую для правильной установки защищенного программного обеспечения.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Безымянный | | |
|  | |  |



**Лекция 22**

## Электронные ключи HASP

Недостатком таких характеристик среды, как серийный номер, ключевой файл, ключевой носитель, конфигурация аппаратуры, информация в секретном секторе диска и т.д. является возможность сравнительно легкого раскрытия их злоумышленником, взлом посредством эмуляции характеристик среды. Для устранения этого недостатка данные характеристики должны быть вынесены во внешнее, максимально защищенное от несанкционированного доступа аппаратное устройство, затрудняющее свою эмуляцию и дублирование. Данную возможность предоставляют специализированные аппаратные средства, называемые электронными ключами. Одним из наиболее распространенных типов электронных ключей являются электронные ключи HASP.

Электронные ключи HASP являются разработкой фирмы Aladdin и представляют собой современное аппаратное средство защиты ПО от несанкционированного использования, позволяющее предотвратить несанкционированный доступ к защищаемым программам и их исполнение. Внешний вид электронных ключей HASP представлен на рис. 8.2.



Рис. 8.2. Внешний вид электронных ключей HASP

Базовой основой ключей HASP является специализированная заказная микросхема (ASIC – Application Specific Integrated Circuit), имеющая уникальный для каждого ключа алгоритм работы. В процессе своего исполнения защищенная программа опрашивает подключенный к ПК HASP. Если HASP возвращает правильные ответы, работает по требуемому алгоритму и обладает требуемыми эталонными характеристиками, то программа выполняется нормально. В противном случае реализуется определенная реакция на несанкционированное использование: запуск в демонстрационном режиме, блокировка отдельных функций, отказ в запуске и т.д.

Семейство электронных ключей HASP включает в себя следующие модели:

1. HASP4.
2. MemoHASP.
3. TimeHASP.
4. NetHASP.

HASP4 Standard

Данный тип ключей является простейшей модификацией электронных ключей HASP. Основным элементом их защиты является аппаратно реализованная в них на ASIC микросхеме функция шифрования и связанная с ней функция отклика *f*(*x*), принимающая на вход 32-битный аргумент и формирующая на выходе четыре 32-битных значения.

Для предотвращения несанкционированного использования ПО система защиты может осуществлять:

1. проверку наличия HASP Standard;

2. проверку соответствия выходов, формируемых функцией отклика *f*(*x*) для различных *x*, эталонным;

3. использовать функцию шифрования электронного ключа для шифрования и дешифрования своего исполняемого кода, используемых данных и т.д.

С любым электронным ключом HASP связана его серия, идентифицирующая защищаемое программное обеспечение (например, IXGGR, RAOMG). Каждая из серий обладает своей уникальной функцией шифрования данных, что вносит уникальность в алгоритм защиты каждого защищаемого продукта.

Данный тип ключей HASP является наиболее хорошим решением для защиты недорогих программ. Стоимость данных ключей составляет порядка 14$.

MemoHASP

Ключи MemoHASP имеют в своем составе все компоненты HASP Standard. Базовым отличием данного типа ключей от HASP4 Standard является наличие встроенной в них энергонезависимой памяти (EEPROM), доступной для чтения и записи во время выполнения защищенной программы. Каждому из данных типов ключей также присваивается свой уникальный 32-битовый идентификационный номер ID, который позволяет идентифицировать конкретного пользователя продукта.

Модификации данных ключей связаны с объемом доступной энергонезависимой памяти.

HASP4 M1 – 112 байт EEPROM, возможность одновременной защиты до 16 программ, ориентировочная стоимость 20$.

HASP4 M4 – 496 байт EEPROM, возможность одновременной защиты до 112 программ, ориентировочная стоимость 29$.

Кроме подходов к защите ПО, свойственных HASP Standard, с помощью MemoHASP могут быть реализованы, например, следующие подходы к защите:

4. Хранение в энергонезависимой памяти MemoHASP конфиденциальной информации – ключей шифрования, части исполняемого кода и т.д.

5. Хранение в энергонезависимой памяти информации о модулях защищенного ПО, к которым пользователь имеет доступ и о тех, к которым не имеет (в зависимости от заплаченной суммы за приобретение ПО).

6. Хранение в энергонезависимой памяти информации о количестве запусков ПО, либо об оставшемся количестве запусков. Данный подход актуален при создании демонстрационных версий ПО, работа с которыми ограничена количеством запусков.

TimeHASP

Кроме функций MemoHASP, данные ключи обладают встроенными часами реального времени с автономным питанием от литиевой батарейки (отражающие время и дату). Используя часы реального времени производитель может защищать свое ПО по времени использования и на основании этого строить гибкую маркетинговую политику - сдачу программ в аренду, лизинг ПО и периодический сбор платы за его использование и т.д. Стоимость данных ключей составляет порядка 33$.

NetHASP

Данные ключи имеют в своем составе все компоненты MemoHASP и предназначены для защиты ПО в сетевых средах. Один ключ, установленный на любом компьютере сети, способен защитить ПО от тиражирования, а также ограничить количество рабочих мест (лицензий), на которых ПО используется одновременно. Ключ может работать на выделенном либо невыделенном сервере, либо любой станции. Он поддерживает различные протоколы – IPX/SPX, NetBIOS, NetBEUI, TCP/IP.

Существует несколько моделей ключей HASP4 Net, которые позволяют лицензировать программы для 5, 10, 20, 50, 100 и для неограниченного количества пользователей. Все они могут защищать до 112 различных программ (или модулей программы). Номер модели определяет максимальное число рабочих мест для любой из этих программ. Например, HASP4 Net-5 защищает до 112 программ, и каждая программа может лицензироваться на количество рабочих мест от нуля до пяти (например, три).

Работа с NetHASP осуществляется через менеджер лицензий.

*Менеджер лицензий* HASP4 Net - это программа-посредник, обеспечивающая связь защищённых приложений с сетевым ключом. При этом сам ключ может стоять на любом компьютере в сети - выделенном или невыделенном файл-сервере или на любой станции. Чтобы использовать в сети сервис NetHASP, нужно выбрать рабочую станцию, присоединить к ней ключ NetHASP и загрузить на ней менеджер лицензий NetHASP.

Менеджер лицензий NetHASP способен обслуживать до 250 защищенных программ, работающих в сети, одновременно обслуживая их связь с несколькими ключами NetHASP.

Когда защищённое приложение стартует на компьютере, подключенном к сети, оно обращается к менеджеру лицензий и запрашивает разрешение выполняться дальше (NetHASP LOGIN), для отключения от сервера используется процедура (NetHASP LOGOUT). При этом менеджер лицензий проверяет соблюдение ряда условий:

* наличие соответствующего ключа HASP4 Net на машине, где он загружен;
* наличие лицензии на выполнение данной программы;
* лимит рабочих мест для этой программы на данный момент не исчерпан.

Если результаты всех проверок положительны, менеджер лицензий даёт запросившей его программе разрешение на выполнение и заносит данные о ней в журнал доступа.

Менеджер лицензий ведёт журнал доступа, в котором отмечаются все подключившиеся приложения, выполнившие LOGIN. В журнале содержатся сведения о том, какая программа и на какой рабочей станции была запущена. Эти данные сохраняются в журнале до тех пор, пока программа не выполнит отключение (LOGOUT). При помощи журнала доступа менеджер лицензий отслеживает количество машин, на которых одновременно выполняется защищённая программа, и не допускает превышения максимального их числа, заданного разработчиком программы.

Цена на ключи NetHASP в зависимости от модели 37,5 – 260$.

Общая схема взаимодействия с электронным ключом HASP

Общую схему функционирования HASP можно представить в следующем виде:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Серия | ID | F(x) – функция шифрования | MEMO | TIME MEMO | Таймер |

|  |
| --- |
| Pass1, Pass2 |

|  |
| --- |
| Program |

На данной схеме представлены основные элементы электронных ключей HASP и особенности взаимодействия с ними. Направление стрелок указывает направление потоков информации от защищенной программы к элементам HASP.

Доступ к функциям электронного ключа HASP возможен только при указании кодов доступа. Коды доступа представляют собой два целых 16-битовых числа. Они уникальны для каждой из серий HASP. Внутри серии данные коды определены однозначно. Не указав код, пользователь не сможет обеспечить себе доступ к функциям HASP. Таким образом выполняется противодействие эмуляции и копированию HASP.

В таблице 8.1. представлены основные элементы электронных ключей HASP и типы HASP, в которых данные элементы присутствуют.

Существует два способа внедрения защитных механизмов в ПО с помощью электронных ключей HASP.

1. HASP API – с помощью API функций.
2. Пакетный режим (HASP Envelope).

Первый способ защиты используется, для встраивания защитных механизмов в исходные тексты. Фирма Aladdin предлагает набор функций для взаимодействия с HASP практически для всех платформ. Сам код API функций защищен и зашифрован. С помощью данных функций можно обратиться к HASP из любой точки программы и на основании проведенных проверок предпринять необходимые шаги. Использование API функций позволяет программировать разработчику любую реакцию на несанкционированный запуск.

Табл. 8.1. Элементы электронных ключей HASP

|  |  |  |
| --- | --- | --- |
| Элемент | Описание | Типы ключей |
| Серия | Серия, присваиваемая каждому из производителей защищаемого продукта, либо каждому из продуктов. Различные серии ключей обладают различными функциями шифрования и различными кодами доступа. Серия не может быть программным путем прочитана, либо записана в HASP. | HASP4 Standard,  MemoHASP,  TimeHASP  NetHASP |
| Идентификационный номер ID | Номер, уникально идентифицирующий каждый из выпущенных ключей HASP. Прошивается единожды в заводских условиях. Может быть прочитан программным путем. | MemoHASP,  TimeHASP  NetHASP |
| Функция шифрования | Аппаратно реализованная функция в HASP, позволяющая шифровать и дешифровывать информацию. | HASP4 Standard,  MemoHASP,  TimeHASP  NetHASP |
| Энергонезависимая память MEMO | Защищенная по доступу память для долговременного хранения конфиденциальной информации. В нее могут быть программным путем записана информация, либо прочитана из нее. | MemoHASP,  TimeHASP  NetHASP |
| Энергонезависимая память TIME MEMO | Защищенная по доступу память для долговременного хранения информации, используемой TimeHASP при защите ПО по времени работы. | TimeHASP |
| Таймер | Таймер, используемый для защиты ПО по времени своей работы. Ограничение по доступу к таймеру с помощью кодов доступа HASP не позволяет злоумышленнику несанкционированно использовать ПО путем корректировки таймера. | TimeHASP |

Второй способ служит непосредственно для защиты исполняемых файлов. Исполняемый файл заключается в защитную программную оболочку, кодирующую файл, и обладающую такими свойствами, как распознавание ключа и антиотладка.

При защите ПО более предпочтительно одновременное использование как первого, так и второго способа.

Система полного управления доступом в HASP (FAS)

Система полного управления доступом (Full Authorization System) в HASP позволяет производителю защитить несколько программ одним и тем же ключом HASP, определив условия, при которых может работать конкретная программа. Возможно ограничение использования программ посредством:

1. Задания количества запусков программы (MemoHASP, NetHASP). Эта возможность полезна при разработке демо-версий защищенных программ.
2. Задания допустимого срока работы программы (HASP4 Time). Эта возможность полезна при лизинге и аренде программ.
3. Задания числа станций, на которых программа может работать одновременно (NetHASP).

В случае защиты с помощью подсистемы FAS, защищенная программа осуществляет несколько проверок.

1. В первую очередь, проверяется, присоединен ли к компьютеру соответствующий ключ.

2. Если ответ положителен, память HASP проверяется на предмет того, занесена ли программа в список разрешенных к работе.

3. При положительном ответе выполняется серия проверок в зависимости от используемой модели HASP.

В случае MemoHASP, память ключа опрашивается на предмет превышения допустимого количества запусков программ. С каждым новым запуском число допустимых запусков уменьшается на 1. Как только это число станет=0, работа программ прекращается, и выдается сообщение об ошибке.

В случае HASP4 Time, в памяти ключа опрашивается список допустимых сроков, и результат сравнивается с реальным временем на таймере ключа.

HASP API

Все функции API HASP (кроме NetHASP) вызываются через единую функцию hasp(), которая имеет следующий синтаксис:

*Hasp(Service,SeedCode,LptNum,Pass1,Pass2,Par1,Par2,Par3,Par4)*

Для NetHASP та же функция имеет следующий формат

*Hasp(Service,SeedCode,ProgNum,Pass1,Pass2,Par1,Par2,Par3,Par4)*

Здесь, *Service* – номер вызываемой функции (и соответствующей ей операции). *LptNum* – номер параллельного порта, к которому подключен HASP (если 0, то драйвер его ищет автоматически, 1-LPT1, 2-LPT2, 3-LPT3,201-255-определенный ключ HASP для порта USB). *SeedCode* – значение, посылаемое в функцию отклика *f*(*x*). *Pass,Pass2* – пароль для HASP. Он должен быть указан при доступе ко всем функциям, кроме функции проверки наличия HASP. *Par1,Par2,Par3,Par4* – параметры, через которые передаются значения в HASP и через которые возвращаются.

Система удаленного обновления (RUS)

Система удаленного обновления (RUS) представляет собой утилиту, позволяющую безопасным образом удаленно обновить содержимое ключей HASP у пользователя без необходимости раскрытия паролей. Эта возможность позволяет отказаться от необходимости отправки разработчиком нового ключа HASP, когда покупатель желает обновить его содержимое.

Использование RUS позволяет, например, принять разработчиком ПО оплату от покупателя, после чего удаленно разблокировать запрет использования определенного модуля. Это позволяет использовать так называемую концепцию многоуровневого лицензирования.

Технология применения утилиты RUS включает в себя 2 этапа:

1. Создание утилиты RUS.
2. Обновление памяти ключей у пользователя.

С помощью утилиты RUS продавец создает 2 утилиты – *утилиту продавца* и *утилиту пользователя*.

Утилита продавца остается у разработчика, а утилита пользователя передается клиенту. Для обновления содержимого ключей HASP разработчик и пользователь каждый раз используют соответствующие утилиты.

Чтобы обновить память ключа HASP

1. Покупатель использует утилиту Пользователя для нахождения идентификатора своего ключа, нуждающегося в обновлении, а затем эту информацию передает разработчику.
2. Разработчик вводит идентификатор и обновляемые данные в утилиту Продавца.
3. Разработчик генерирует пароли RUS в утилите Продавца и передает их пользователю.
4. Пользователь вводит эти пароли в утилиту Пользователя и обновляет память своего ключа HASP.

Процесс обновления памяти ключей пользователя полностью защищен. Все данные обмена шифруются случайным образом.

Модель защиты структурным кодом (PCS)

Защита структурным кодом (Pattern Code Security – PCS) является средством, значительно повышающим защищенность приложения, защищаемого с помощью электронных ключей HASP.

PCS реализуется в процессе защиты с помощью HASP API. Использование PCS возможно лишь при наличии доступа к исходным текстам защищаемого приложения.

PCS осуществляет последовательность скрытых вызовов процедуры hasp(), не включая эти вызовы в Ваш исходный код явно. После каждого вызова процедуры hasp() происходит переключение на следующий скрытый вызов. Если вызов hasp() вдруг удален из защищенной программы, скрытые вызовы не выполняются, а это означает, что кто-то вмешался в работу программы. Тем самым, PCS не дает удалить либо «заклеить» обращения к процедуре hasp().

Возможно определить до 25 шаблонов в исходном коде. Каждый раз, когда вызывается процедура hasp(), переключается столько скрытых вызовов, сколько определено шаблонов.

Шаблон – специальная статическая структура данных, определенная в приложении. Шаблон включает в себя сигнатуру, номер функции hasp() и параметры, необходимые для работы данной функции.

Каждый вызов процедуры hasp() автоматически обновляет все шаблоны в коде программы, их после выхода из функции можно проверить. Если динамически изменять в шаблонах значения входных переменных, то автоматически будут меняться и выходные. Аналогично, можно изменять и значения сервисов.

Преимуществами использования PCS являются:

1. скрывание обращения к HASP;
2. трассировка вызовов hasp() для шаблонов практически невозможна, так как их нет в исходном коде;
3. легче обнаруживается вмешательство извне (если процедуру отключили); если вызов hasp() будет удален, то шаблоны не обновятся, а это значит – кто-то вмешался в работу приложения;
4. PCS препятствует эмуляции процедуре hasp().

Рекомендации по наиболее надежной защите с помощью HASP

Электронные ключи HASP являются достаточно надежным аппаратным средством защиты ПО. Тем не менее, система защиты HASP надежна лишь настолько, насколько разработчик сделает ее таковой.

Так как аппаратную часть HASP подделать в принципе невозможно, то все атаки будут осуществляться на программную часть – на трассировку кода и отключение защитных механизмов. Существует два способа атаки на защищенное приложение:

* заклеивание вызовов к защищающей приложение процедуре;
* заклеивание программы производителя ключа.

При попытке заклеить вызовы к защищающей приложение процедуре, взломщику придется так изменить защищенное приложение, чтобы оно не посылало вызовов ключу, не проверяло бы результата вызовов, не реагировало бы указанными в приложении способами на результаты проверки. Этот способ применим, если взломщик увидит, что защита реализована не очень хитрым способом.

При попытке заклеить программы производителя ключа, взломщику придется изменить процедуры, отвечающие за связь с ключом (сами API). А когда эти процедуры будут изменены, они будут возвращать ожидаемый результат вызова, даже если надлежащий ключ и не подсоединен.

Для более надежной защиты ПО от несанкционированного использования с помощью HASP рекомендуется использовать следующие приемы.

1. Использовать одновременно методы защиты с помощью оболочки и с помощью API. Они дополняют и усиливают друг друга.
2. Использовать больше вызовов hasp() и шаблонов PCS. Это создаст большие проблемы для взломщика в понимании схемы защиты и атаках на нее. Необходимо как можно больше рассеивать данные вызовы по всему приложению, чтобы затруднить анализ.
3. Шифровать внутренние и внешние данные защищаемого приложения. Дешифровку проводить на ключе HASP. В данном случае взломщику нужно будет не только взломать приложение, но и дешифровать данные. Нет необходимости шифровать все используемые приложением данные, но некоторые ключевые данные можно зашифровать. Объектом шифрования может быть все то, что оказывает влияние на основные функции приложения.
4. Избегать повторяющихся схем. Схему, которая повторяется в защищаемом коде легко обнаружить и трассировать. Как только взломщик поймет схему защиты, для него станет ясно, на что обратить внимание, что облегчит ему работу по снятию защиты.
5. Разделять в коде программы шаги вызова процедуры hasp(), анализа ответных значений, возвращенных данной процедурой, и реакцию программы на результат анализа. В данном случае их хуже трассировать, нежели последовательные шаги.

Например, можно проверять наличие ключа, когда пользователь щелкает мышью по определенной опции меню, после этого при сбое дать ему немного поработать, а затем выдать сообщение об ошибке. Это позволит скрыть действительное место проверки HASP.

6. Использование функционирования программы в качестве ответа на отсутствие HASP. Можно использовать целый ряд реакций на неприсоединение нужного ключа. Наиболее простое – вывод сообщения «HASP not FOUND», однако это подсказывает, что делалась проверка ключа. Лучше запрограммировать другую реакцию, – например, отключить клавиатуру. После подсоединения ключа клавиатура включается.

7. Использовать HASP-зависимые данные. Если осуществляется проверка значений, внесенных в HASP, путем их сравнения с эталонными значениями, то это дает дополнительную информацию для атаки. Необходимо считывать и использовать эти данные в приложении, не проверяя их корректность напрямую. Например, можно хранить в энергонезависимой памяти HASP метку перехода, считывать ее и переходить по этой метке.