## Основные подходы к защите программ от несанкционированного копирования.

###### 3.4.1. Основные функции средств защиты от копирования

При защите программ от несанкционированного копирования применяются методы, которые позволяют привносить в защищаемую программу функции привязки процесса выполнения кода программы только на тех ЭВМ, на которые они были инсталлированы. Инсталлированная программа для защиты от копирования при каждом запуске должна выполнять следующие действия:

* анализ аппаратно-программной среды компьютера, на котором она запущена, формирование на основе этого анализа текущих характеристик своей среды выполнения;
* проверка подлинности среды выполнения путем сравнения ее текущих характеристик с эталонными, хранящимися на винчестере;
* блокирование дальнейшей работы программы при несовпадении текущих характеристик с эталонными.

Этап проверки подлинности среды является одним из самых уязвимых с точки зрения защиты. Можно детально не разбираться с логикой защиты, а немного "подправить" результат сравнения, и защита будет снята.

При выполнении процесса проверки подлинности среды возможны три варианта: с использованием множества операторов сравнения того, что есть, с тем, что должно быть, с использованием механизма генерации исполняемых команд в зависимости от результатов работы защитного механизма и с использованием арифметических операций. При использовании механизма генерации исполняемых команд в первом байте хранится исходная ключевая контрольная сумма BIOS, во второй байт записывается подсчитанная контрольная сумма в процессе выполнения задачи. Затем осуществляется вычитание из значения первого байта значение второго байта, а полученный результат добавляется к каждой ячейки оперативной памяти в области операционной системы. Понятно, что если суммы не совпадут, то операционная система функционировать не будет. При использовании арифметических операций осуществляется преобразование над данными арифметического характера в зависимости от результатов работы защитного механизма.

Для снятия защиты от копирования применяют два основных метода: статический и динамический.

Статические методы предусматривают анализ текстов защищенных программ в естественном или преобразованном виде. Динамические методы предусматривают слежение за выполнением программы с помощью специальных средств снятия защиты от копирования.

###### 3.4.2. Основные методы защиты от копирования

Криптографические методы

Для защиты инсталлируемой программы от копирования при помощи криптографических методов инсталлятор программы должен выполнить следующие функции:

* анализ аппаратно-программной среды компьютера, на котором должна будет выполняться инсталлируемая программа, и формирование на основе этого анализа эталонных характеристик среды выполнения программы;
* запись криптографически преобразованных эталонных характеристик аппаратно-программной среды компьютер на винчестер.

Преобразованные эталонные характеристики аппаратно-программной среды могут быть занесены в следующие области жесткого диска:

* в любые места области данных (в созданный для этого отдельный файл, в отдельные кластеры, которые должны помечаться затем в FAT как зарезервированные под операционную систему или дефектные);
* в зарезервированные сектора системной области винчестера;
* непосредственно в файлы размещения защищаемой программной системы, например, в файл настройки ее параметров функционирования.

Можно выделить два основных метода защиты от копирования с использованием криптографических приемов:

* с использованием односторонней функции;
* с использованием шифрования.

Односторонние функции это функции, для которых при любом x из области определения легко вычислитьf(x), однако почти для всех y из ее области значений, найти y=f(x) вычислительно трудно.

Если эталонные характеристики программно-аппаратной среды представить в виде аргумента односторонней функции x, то y - есть "образ" этих характеристик, который хранится на винчестере и по значению которого вычислительно невозможно получить сами характеристики. Примером такой односторонней функции может служить функция дискретного возведения в степень, описанная в разделах 2.1 и 3.3 с размерностью операндов не менее 512 битов.

При шифровании эталонные характеристики шифруются по ключу, совпадающему с этими текущими характеристиками, а текущие характеристики среды выполнения программы для сравнения с эталонными также зашифровываются, но по ключу, совпадающему с этими текущими характеристиками. Таким образом, при сравнении эталонные и текущие характеристики находятся в зашифрованном виде и будут совпадать только в том случае, если исходные эталонные характеристики совпадают с исходными текущими.

Метод привязки к идентификатору

В случае если характеристики аппаратно-программной среды отсутствуют в явном виде или их определение значительно замедляет запуск программ или снижает удобство их использования, то для защиты программ от несанкционированного копирования можно использовать методов привязки к идентификатору, формируемому инсталлятором. Суть данного метода заключается в том, что на винчестере при инсталляции защищаемой от копирования программы формируется уникальный идентификатор, наличие которого затем проверяется инсталлированной программой при каждом ее запуске. При отсутствии или несовпадении этого идентификатора программа блокирует свое дальнейшее выполнение.

Основным требованием к записанному на винчестер уникальному идентификатору является требование, согласно которому данный идентификатор не должен копироваться стандартным способом. Для этого идентификатор целесообразно записывать в следующие области жесткого диска:

* в отдельные кластеры области данных, которые должны помечаться затем в FAT как зарезервированные под операционную систему или как дефектные;
* в зарезервированные сектора системной области винчестера.

Некопируемый стандартным образом идентификатор может помещаться на дискету, к которой должна будет обращаться при каждом своем запуске программа. Такую дискету называют ключевой. Кроме того, защищаемая от копирования программа может быть привязана и к уникальным характеристикам ключевой дискеты. Следует учитывать, что при использовании ключевой дискеты значительно увеличивается неудобство пользователя, так как он всегда должен вставлять в дисковод эту дискету перед запуском защищаемой от копирования программы.

Методы, основанные на работа с переходами и стеком

Данные методы основаны на включение в тело программы переходов по динамически изменяемым адресам и прерываниям, а также самогенерирующихся команд (например, команд, полученных с помощью сложения и вычитания). Кроме того, вместо команды безусловного перехода (JMP) может использоваться возврат из подпрограммы (RET). Предварительно в стек записывается адрес перехода, который в процессе работы программы модифицируется непосредственно в стеке.

При работе со стеком, стек определяется непосредственно в области исполняемых команд, что приводит к затиранию при работе со стеком. Этот способ применяется, когда не требуется повторное исполнение кода программы. Таким же способом можно генерировать исполняемые команды до начала вычислительного процесса.

Манипуляции с кодом программы

При манипуляциях с кодом программы можно привести два следующих способа:

* включение в тело программы "пустых" модулей;
* изменение защищаемой программы.

Первый способ заключается во включении в тело программы модулей, на которые имитируется передача управления, но реально никогда не осуществляется. Эти модули содержат большое количество команд, не имеющих никакого отношения к логике работы программы. Но "ненужность" этих программ не должна быть очевидна потенциальному злоумышленнику.

Второй способ заключается в изменении начала защищаемой программы таким образом, чтобы стандартный дизассемблер не смог ее правильно дизассемблировать. Например, такие программы, как Nota и Copylock, внедряя защитный механизм в защищаемый файл, полностью модифицируют исходный заголовок EXE-файла.

Все перечисленные методы были, в основном направлены на противодействия статическим способам снятия защиты от копирования. В следующем подразделе рассмотрим методы противодействия динамическим способам снятия защиты.

###### 3.4.2. Методы противодействия динамическим способам снятия защиты программ от копирования

Набор методов противодействия динамическим способам снятия защиты программ от копирования включает следующие методы.

* Периодический подсчет контрольной суммы, занимаемой образом задачи области оперативной памяти, в процессе выполнения. Это позволяет:
  + заметить изменения, внесенные в загрузочный модуль;
  + в случае, если программу пытаются "раздеть", выявить контрольные точки, установленные отладчиком.
* Проверка количества свободной памяти и сравнение и с тем объемом, к которому задача "привыкла" или "приучена". Это действия позволит застраховаться от слишком грубой слежки за программой с помощью резидентных модулей.
* Проверка содержимого незадействованных для решения защищаемой программы областей памяти, которые не попадают под общее распределение оперативной памяти, доступной для программиста, что позволяет добиться "монопольного" режима работы программы.
* Проверка содержимого векторов прерываний (особенно 13h и 21h) на наличие тех значений, к которым задача "приучена". Иногда бывает полезным сравнение первых команд операционной системы, отрабатывающих этим прерывания, с теми командами, которые там должны быть. Вместе с предварительной очисткой оперативной памяти проверка векторов прерываний и их принудительное восстановление позволяет избавиться от большинства присутствующих в памяти резидентных программ.
* Переустановка векторов прерываний. Содержимое некоторых векторов прерываний (например, 13h и 21h) копируется в область свободных векторов. Соответственно изменяются и обращения к прерываниям. При этом слежение за известными векторами не даст желаемого результата. Например, самыми первыми исполняемыми командами программы копируется содержимое вектора 21h (4 байта) в вектор 60h, а вместо команд int 21h в программе везде записывается команда int 60h. В результате в явном виде в тексте программы нет ни одной команды работы с прерыванием 21h.
* Постоянное чередование команд разрешения и запрещения прерывания, что затрудняет установку отладчиком контрольных точек.
* Контроль времени выполнения отдельных частей программы, что позволяет выявить "остановы" в теле исполняемого модуля.

Многие перечисленные защитные средства могут быть реализованы исключительно на языке Ассемблер. Одна из основных отличительных особенностей этого языка заключается в том, что для него не существует ограничений в области работы со стеком, регистрами, памятью, портами ввода/вывода и т.п.

Автокорреляция представляет значительный интерес, поскольку дает некоторую числовую характеристику программы. По всей вероятности автокорреляционные функции различного типа можно использовать и тестировании программ на технологическую безопасность, когда разработанную программу еще не с чем сравнивать на подобие с целью обнаружения программных дефектов. Таким образом, программы имеют целую иерархию структур, которые могут быть выявлены, измерены и использованы в качестве характеристик последовательности данных. При этом в ходе тестирования, измерения не должны зависеть от типа данных, хотя данные, имеющие структуру программы, должны обладать специфическими параметрами, позволяющими указать меру распознавания программы. Поэтому указанные методы позволяют в определенной мере выявить те изменения в программе, которые вносятся нарушителем либо в результате преднамеренной маскировки, либо преобразованием некоторых функций программы, либо включением модуля, характеристики которого отличаются от характеристик программы, а также позволяют оценить степень обеспечения безопасности программ при внесении программных закладок.