

Лекція 5. Загальний огляд систем захисту програмного забезпечення

Навчальні питання

1. Класифікація систем захисту програмного забезпечення 1
2. Показники застосовності систем захисту програмного забезпечення 7
3. Підходи до захисту програм від несанкціонованого копіювання..... 9

Час – 2 год.

Література.

1. Середя С.А. Оценка эффективности систем защиты программного обеспечения. Журнал "КомпьюЛог" №2, 2000 г.
2. <https://ru.wikipedia.org/wiki/SaaS>.
3. Казарин О.В. Теория и практика защиты программ. – 2004. – 450 с.

Вступ

1. Класифікація систем захисту програмного забезпечення

Напомним основные угрозы программному обеспечению (ПО):

- незаконное использование алгоритмов, являющихся интеллектуальной собственностью автора, при написании аналогов продукта (промышленный шпионаж);
- несанкционированное использование ПО (кража и копирование);
- несанкционированная модификация ПО с целью внедрения программных злоупотреблений;
- незаконное распространение и сбыт ПО (пиратство).

Системы защиты программного обеспечения (СЗПО) можно классифицировать по ряду признаков, среди которых можно выделить

- *метод установки,*
- *используемые механизмы защиты и*
- *принцип функционирования.*

СЗПО по методу установки можно подразделить на:

- системы, устанавливаемые на скомпилированные модули ПО;
- системы, встраиваемые в исходный код ПО до компиляции;
- и комбинированные.

Системы первого типа наиболее удобны для производителя ПО, так как легко можно защитить уже полностью готовое и оттестированное ПО (обычно процесс установки защиты максимально автоматизирован и сводится к указанию имени защищаемого файла и нажатию "Enter"), а потому и наиболее популярны. В то же время стойкость этих систем достаточно низка (в зависимости от принципа действия СЗПО), так как для обхода защиты достаточно определить точку завершения работы "конверта" защиты и передачи управления защищенной программе, а затем принудительно ее сохранить в незащищенном виде.

Системы второго типа неудобны для производителя ПО, так как возникает необходимость обучать персонал работе с программным интерфейсом (API) системы защиты с вытекающими отсюда денежными и временными затратами. Кроме того, усложняется процесс тестирования ПО и снижается его надежность, так как кроме самого ПО ошибки может содержать API системы защиты или процедуры, его использующие. Но такие системы являются более стойкими к атакам, потому что здесь исчезает четкая граница между системой защиты и как таковым ПО.

Наиболее живучими являются комбинированные системы защиты. Сохраняя достоинства и недостатки систем второго типа, они максимально затрудняют анализ и дезактивацию своих алгоритмов.

По используемым механизмам защиты СЗПО можно классифицировать на:

- системы, использующие сложные логические механизмы;

- системы, использующие шифрование защищаемого ПО;
- и комбинированные системы.

Системы первого типа используют различные методы и приёмы, ориентированные на затруднение дизассемблирования, отладки и анализа алгоритма СЗПО и защищаемого ПО. Этот тип СЗПО наименее стоек к атакам, так как для преодоления защиты достаточно проанализировать логику процедур проверки и должным образом их модифицировать.

Более стойкими являются системы второго типа. Для дезактивации таких защит необходимо определение ключа дешифрации ПО.

Самыми стойкими к атакам являются комбинированные системы.

Для защиты ПО используется ряд нижеследующих *механизмов*.

1. **Алгоритмы запутывания** - используются хаотические переходы в разные части кода, внедрение ложных процедур - "пустышек", холостые циклы, искажение количества реальных параметров процедур ПО, разброс участков кода по разным областям ОЗУ и т.п.
2. **Алгоритмы мутации** - создаются таблицы соответствия операндов-синонимов и замена их друг на друга при каждом запуске программы по определенной схеме или случайным образом, случайные изменения структуры программы.
3. **Алгоритмы компрессии данных** - программа упаковывается, а затем распаковывается по мере выполнения.
4. **Алгоритмы шифрования данных** - программа шифруется, а затем расшифровывается по мере выполнения.
5. **Вычисление сложных математических выражений в процессе отработки механизма защиты** - элементы логики защиты зависят от результата вычисления значения какой-либо формулы или группы формул.
6. **Методы затруднения дизассемблирования** - используются различные приемы, направленные на предотвращение дизассемблирования в пакетном режиме.
7. **Методы затруднения отладки** - используются различные приемы, направленные на усложнение отладки программы.
8. **Эмуляция процессоров и операционных систем** - создается виртуальный процессор и/или операционная система (не обязательно реально существующие) и программа-переводчик из системы команд IBM в систему команд созданного процессора или ОС, после такого перевода ПО может выполняться только при помощи эмулятора, что резко затрудняет исследование алгоритма ПО.
9. **Нестандартные методы работы с аппаратным обеспечением** - модули системы защиты обращаются к аппаратуре ЭВМ, минуя процедуры операционной системы, и используют малоизвестные или недокументированные её возможности.

В свою очередь злоумышленники так же применяют ряд методов и средств для нарушения систем защиты. Ситуация противостояния разработчиков СЗПО и злоумышленников постоянно изменяется за счет комбинирования уже известных методов защиты и нападения, а так же за счет создания и использования новых методов. В целом это взаимодействие может быть описано схемой на рис. 1.

По *принципу функционирования* СЗПО можно подразделить на

- упаковщики/шифраторы;
- СЗПО от несанкционированного копирования и
- СЗПО от несанкционированного доступа (НСД).



Рис. 1

Упаковщики/шифраторы.

Первоначально, основной целью упаковщиков/шифраторов являлось уменьшение объема исполняемого модуля на диске без ущерба для функциональности программы, но позднее на первый план вышла цель защиты ПО от анализа его алгоритмов и несанкционированной модификации. Для достижения этого используются:

- алгоритмы компрессии данных;
- приёмы, связанные с использованием недокументированных особенностей операционных систем (ОС) и процессоров;
- шифрование данных,
- алгоритмы мутации,
- запутывание логики программы,
- приведение ОС в нестабильное состояние на время работы ПО и др.

Достоинства.

1. В рамках периода безопасного использования данные системы обеспечивают высокий уровень защиты ПО от анализа его алгоритмов.

2. Методы упаковки/шифрования многократно увеличивают стойкость систем защиты других типов.

Недостатки.

1. Практически все применяемые методы замедляют выполнение кода ПО.
2. Шифрование/упаковка кода ПО вызывает затруднения при обновлении (update) и исправлении ошибок (bugfix, servicepack).
3. Возможно повышение аппаратно-программных требований ПО.
4. В чистом виде данные системы не применимы для авторизации использования ПО.
5. Эти системы применимы лишь к продуктам небольшого объема (до 1 мегабайта).
6. Данный класс систем уязвим, так как программный код, в конечном итоге, распаковывается или расшифровывается для выполнения.
7. Обладают небольшим сроком безопасного использования, ввиду п.4
8. Упаковка и шифрование исполняемого кода вступает в конфликт с запрещением самомодифицирующегося кода в современных ОС.

СЗПО от несанкционированного копирования.

СЗПО от несанкционированного копирования осуществляют "привязку" ПО к дистрибутивному носителю (жесткий диск, DVD ...). Данный тип защит основывается на глубоком изучении работы контроллеров накопителей, их физических показателей, нестандартных режимов разбивки, чтения/записи и т.п. При этом

на физическом уровне создается дистрибутивный носитель, обладающий (предположительно) неповторимыми свойствами (обычно это достигается при помощи нестандартной разметки носителя информации или/и записи на него дополнительной информации - пароля или метки),

а на программном - создается модуль, настроенный на идентификацию и аутентификацию носителя по его уникальным свойствам. При этом возможно применение приемов, используемых упаковщиками/шифраторами.

Достоинства.

1. Затруднение нелегального копирования и распространения ПО.
2. Защита прав пользователя на приобретенное ПО.

Недостатки.

1. Большая трудоёмкость реализации системы защиты.
2. Замедление продаж из-за необходимости физической передачи дистрибутивного носителя информации.
3. Повышение системных требований из-за защиты (наличие накопителя).
4. Снижение отказоустойчивости ПО.
5. Несовместимость защиты и аппаратуры пользователя (накопитель, контроллер).
6. На время работы ПО занимается накопитель.
7. Угроза кражи защищённого носителя.

СЗПО от НСД.

СЗПО от НСД осуществляют предварительную или периодическую аутентификацию пользователя ПО или его компьютерной системы путём запроса дополнительной информации. К этому типу СЗПО можно отнести:

- системы парольной защиты ПО,
- системы "привязки" ПО к компьютеру пользователя,
- аппаратно-программные системы с электронными ключами.

В первом случае "ключевую" информацию вводит пользователь,

во втором - она содержится в уникальных параметрах компьютерной системы пользователя,

в третьем случае "ключевая" информация считывается с микросхем электронного ключа.

Парольные защиты

Этот класс СЗПО, на сегодняшний день, является самым распространённым. Основной принцип работы данных систем заключается в идентификации и аутентификации пользователя ПО путём запроса дополнительных данных, это могут быть название фирмы и/или имя и фамилия пользователя и его пароль либо только пароль/регистрационный код. Эта информация может запрашиваться в различных ситуациях, например, при старте программы, по истечении срока бесплатного использования ПО, при вызове процедуры регистрации либо в процессе установки на

ПК пользователя. Процедуры парольной защиты просты в реализации и, поэтому, очень часто применяются производителями ПО. Большинство паролевых СЗПО использует логические механизмы, сводящиеся к проверке правильности пароля/кода и запуску или не запуску ПО, в зависимости от результатов проверки.

Существуют так же системы, шифрующие защищаемое ПО и использующие пароль или производную от него величину как ключ дешифрации, большинство таких систем использует слабые или простейшие алгоритмы шифрования, нестойкие к направленным атакам. Это происходит из-за сложности корректной реализации стойких криптоалгоритмов и нецелесообразности их применения для защиты недорогих условно-бесплатных программных продуктов, составляющих большинство ПО, использующего паролевые защиты. Лишь в последнее время разработаны паролевые СЗПО, реализующие стойкие криптоалгоритмы типа DES и RSA, они реализованы в виде защитного модуля и вспомогательных библиотек и устанавливаются на уже скомпилированные модули ПО.

Слабым звеном паролевых защит является блок проверки правильности введенного пароля/кода. Для такой проверки можно сравнивать введенный пароль с записанным в коде ПО правильным либо с правильно сгенерированным из введенных дополнительных данных паролем. Возможно так же сравнение производных величин от введенного и правильного паролей, например их ХЭШ-функций, в таком случае в коде можно сохранять только производную величину, что повышает стойкость защиты. Путём анализа процедур проверки можно найти реальный пароль, записанный в коде ПО, найти правильно сгенерированный пароль из введенных данных либо создать программу для перебора паролей для определения пароля с нужной ХЭШ-суммой. Кроме того, если СЗПО не использует шифрования, достаточно лишь принудительно изменить логику проверки для получения беспрепятственного доступа к ПО.

Шифрующие системы более стойки к атакам, но при использовании простейших или некорректно реализованных криптоалгоритмов есть опасность дешифрации ПО.

Для всех паролевых систем существует угроза перехвата пароля при его вводе авторизованным пользователем. Кроме того, в большинстве СЗПО данного типа процедура проверки используется лишь единожды, обычно при регистрации или установке ПО, затем система защиты просто отключается, что создаёт реальную угрозу для НСД при незаконном копировании ПО.

Достоинства.

1. Надёжная защита от злоумышленника-непрофессионала.
2. Минимальные неудобства для пользователя.
3. Возможность передачи пароля/кода по сети.
4. Отсутствие конфликтов с системным и прикладным ПО и аппаратным обеспечением.
5. Простота реализации и применения.
6. Низкая стоимость.

Недостатки.

1. Низкая стойкость большинства систем защиты данного типа.
2. Пользователю необходимо запоминать пароль/код.

Системы "привязки" ПО

Системы этого типа при установке ПО на ПК пользователя осуществляют поиск уникальных признаков компьютерной системы либо они устанавливаются самой системой защиты. После этого модуль защиты в самом ПО настраивается на поиск и идентификацию данных признаков, по которым в дальнейшем определяется авторизованное или неавторизованное использование ПО. При этом возможно применение методик оценки скоростных и иных показателей процессора, материнской платы, дополнительных устройств, ОС, чтение/запись в микросхемы энергонезависимой памяти, запись скрытых файлов, настройка на наиболее часто встречаемую карту использования ОЗУ и т.п.

Слабым звеном таких защит является тот факт, что на ПК пользователя ПО всегда запускается на выполнение, что приводит к возможности принудительного сохранения ПО после отработки системы защиты, исследование самой защиты и выявление данных, используемым СЗПО для аутентификации ПК пользователя.

Достоинства.

1. Не требуется добавочных аппаратных средств для работы защиты.

2. Затруднение несанкционированного доступа к скопированному ПО.
3. Простота применения.
4. "Невидимость" СЗПО для пользователя.

Недостатки.

1. Ложные срабатывания СЗПО при любых изменениях в параметрах ПК.
2. Низкая стойкость при доступе злоумышленника к ПК пользователя.
3. Возможность конфликтов с системным ПО.

Программно-аппаратные средства защиты ПО с электронными ключами

Этот класс СЗПО в последнее время приобретает все большую популярность среди производителей программного обеспечения (ПО). Под программно-аппаратными средствами защиты, в данном случае, понимаются средства, основанные на использовании так называемых "аппаратных (электронных) ключей". *Электронный ключ* - это аппаратная часть системы защиты, представляющая собой плату с микросхемами памяти и, в некоторых случаях, микропроцессором, помещенную в корпус и предназначенную для установки в один из стандартных портов ПК (COMM, LPT, PCMCIA, USB ...) или слот расширения материнской платы. Так же в качестве такого устройства могут использоваться smart-карты. Программно-аппаратные средства защиты в настоящий момент являются одними из самых стойких систем защиты ПО от НСД.

Электронные ключи по архитектуре можно подразделить на

- **ключи с памятью** (без микропроцессора) и
- **ключи с микропроцессором** (и памятью).

Наименее стойкими (в зависимости от типа программной части) являются системы с аппаратной частью первого типа. В таких системах критическая информация (ключ дешифрации, таблица переходов) хранится в памяти электронного ключа. Для дезактивации таких защит в большинстве случаев необходимо наличие у злоумышленника аппаратной части системы защиты (основная методика: перехват диалога между программной и аппаратной частями для доступа к критической информации).

Самыми стойкими являются системы с аппаратной частью второго типа. Такие комплексы содержат в аппаратной части не только ключ дешифрации, но и блоки шифрации/дешифрации данных, таким образом при работе защиты в электронный ключ передаются блоки зашифрованной информации, а принимаются оттуда расшифрованные данные. В системах этого типа достаточно сложно перехватить ключ дешифрации так как все процедуры выполняются аппаратной частью, но остается возможность принудительного сохранения защищенной программы в открытом виде после отработки системы защиты. Кроме того, к ним применимы методы криптоанализа.

Достоинства.

1. Значительное затруднение нелегального распространения и использования ПО.
2. Избавление производителя ПО от разработки собственной системы защиты.
3. Высокая автоматизация процесса защиты ПО.
4. Наличие API системы для более глубокой защиты.
5. Возможность легкого создания демо-версий.
6. Достаточно большой выбор таких систем на рынке.

Недостатки.

1. Затруднение разработки и отладки ПО из-за ограничений со стороны СЗПО.
2. Дополнительные затраты на приобретение системы защиты и обучение персонала.
3. Замедление продаж из-за необходимости физической передачи аппаратной части.
4. Повышение системных требований из-за защиты (совместимость, драйверы).
5. Снижение отказоустойчивости ПО.
6. Несовместимость систем защиты и системного или прикладного ПО пользователя.
7. Несовместимость защиты и аппаратуры пользователя.
8. Ограничения из-за несовместимости электронных ключей различных фирм.
9. Снижение расширяемости компьютерной системы.
10. Затруднения или невозможность использования защищенного ПО в переносных и блокнотных ПК.
11. Наличие у аппаратной части размеров и веса.
12. Угроза кражи аппаратного ключа.

Необходимо отметить, что пользователем явно ощущаются лишь отрицательные стороны систем защит. А производители ПО рассматривают только относящиеся к ним "плюсы" и "минусы" систем защиты и практически не рассматривают факторы, относящиеся к конечному потребителю.

Отдельно можно упомянуть о **программном обеспечении как услуге SaaS** (англ. software as a service — программное обеспечение как услуга; также англ. software on demand — программное обеспечение по требованию) — бизнес-модель продажи и использования программного обеспечения, при которой поставщик разрабатывает веб-приложение и самостоятельно управляет им, предоставляя заказчику доступ к программному обеспечению через Интернет. Основное преимущество модели SaaS для потребителя услуги состоит в отсутствии затрат, связанных с установкой, обновлением и поддержкой работоспособности оборудования и работающего на нём программного обеспечения.

В модели SaaS:

- приложение приспособлено для удаленного использования;
- одним приложением пользуется несколько клиентов (приложение коммунально);
- оплата взимается либо в виде ежемесячной абонентской платы, либо на основе объёма операций;
- техническая поддержка приложения включена в оплату;
- модернизация и обновление приложения происходит оперативно и прозрачно для клиентов.

В рамках модели SaaS заказчики платят не за владение программным обеспечением как таковым, а за его аренду (то есть за его использование через веб-интерфейс). Таким образом, в отличие от классической схемы лицензирования ПО, заказчик несет сравнительно небольшие периодические затраты, и ему не требуется инвестировать значительные средства в приобретение ПО и аппаратной платформы для его развертывания, а затем поддерживать его работоспособность. Схема периодической оплаты предполагает, что если необходимость в программном обеспечении временно отсутствует, то заказчик может приостановить его использование и заморозить выплаты разработчику.

С точки зрения разработчика некоторого проприетарного программного обеспечения модель SaaS позволяет эффективно бороться с нелегальным использованием программного обеспечения, поскольку ПО как таковое не попадает к конечным заказчикам. Кроме того, концепция SaaS часто позволяет уменьшить затраты на развертывание и внедрение систем технической и консультационной поддержки продукта, хотя и не исключает их полностью.

2. Показники застосовності систем захисту програмного забезпечення

Технические

Соответствие СЗПО функциональным требованиям производителя ПО и требованиям по стойкости, системные требования ПО и системные требования СЗПО, объём ПО и объём СЗПО, функциональная направленность ПО, наличие и тип СЗ у аналогов ПО - конкурентов.

Экономические

Соотношение потерь от пиратства и общего объёма прибыли, соотношение потерь от пиратства и стоимости СЗПО и её внедрения, соотношение стоимости ПО и стоимости СЗПО, соответствие стоимости СЗПО и её внедрения поставленным целям.

Организационные

Распространённость и популярность ПО, условия распространения и использования ПО, уникальность ПО, наличие угроз, вероятность превращения пользователя в злоумышленника, роль документации и поддержки при использовании ПО.

Критерии оценки СЗПО

Защита как таковая

Затруднение нелегального копирования, затруднение нелегального доступа, защита от мониторинга, отсутствие логических брешей и ошибок в реализации системы.

Стойкость к исследованию/взлому

Применение стандартных механизмов, новые/нестандартные механизмы.

Отказоустойчивость (надёжность)

Вероятность отказа защиты (НСД), время наработки на отказ, вероятность отказа программы защиты (крах), время наработки на отказ, частота ложных срабатываний.

Независимость от конкретных реализаций ОС

Использование недокументированных возможностей, "вирусных" технологий и "дыр" ОС

Совместимость

Отсутствие конфликтов с системным и прикладным ПО, максимальная совместимость с будущим ПО.

Неудобства для конечного пользователя ПО

Необходимость и сложность дополнительной настройки системы защиты, доступность документации, доступность информации об обновлении модулей системы защиты из-за ошибок/несовместимости/нестойкости, доступность сервисных пакетов, безопасность сетевой передачи пароля/ключа, задержка из-за физической передачи пароля/ключа, нарушения прав потребителя.

Побочные эффекты

Перегрузка трафика, отказ в обслуживании, замедление работы защищаемого ПО, замедление работы ОС, захват системных ресурсов, перегрузка ОЗУ, нарушение стабильности ОС.

Стоимость

Стоимость/эффективность, стоимость/цена защищаемого ПО, стоимость/ликвидированные убытки.

Доброкачественность

Правдивая реклама, доступность результатов независимой экспертизы, доступность информации о побочных эффектах, полная информация о СЗ для конечного пользователя.

В целом, общая картина взаимодействия агентов рынка программного обеспечения может быть представлена на схеме на рис. 2.



Рис. 2

Из четырёх указанных выше видов среды взаимодействия защищающей стороне подконтрольны (или частично подконтрольны) три вида - организационная, техническая и

экономическая среда. Важнейшей средой взаимодействия является экономическая среда, так как экономическое взаимодействие, в данном случае, является первопричиной и целью всего взаимодействия. При разработке и анализе защиты программного обеспечения необходимо учитывать существующую законодательную базу, при этом нужно проводить подробный экономический анализ ситуации, применяя различные критерии оценки, а затем создавать стратегию защиты, включающую применение технических и организационных мер защиты программного обеспечения.

3. Підходи до захисту програм від несанкціонованого копіювання

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 КС, 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.

Криптографические методы защиты от копирования

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 компьютера 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 (которое также может использовать односторонние функции).

Односторонние функции это функции, для которых при любом x из области определения легко вычислить $f(x)$, 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 экспоненцирования 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.

Методы противодействия статическим способам снятия защиты от копирования

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 метод 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 внешний носитель данных, к которому должна будет обращаться при каждом своем запуске программа. Такой носитель называют ключевым. Кроме того, защищаемая от копирования программа может быть привязана и к уникальным характеристикам ключевого внешнего носителя. Следует учитывать, что при использовании ключевого носителя значительно увеличивается неудобство пользователя, так как он всегда должен его подключать вставлять перед запуском защищаемой от копирования программы.

Методы, основанные на работе с переходами и стеком

Данные методы основаны на включение в тело программы переходов по динамически изменяемым адресам и прерываниям¹, 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 стекком, стек определяется непосредственно в области исполняемых команд, что приводит к затиранию при работе со стекком. 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.

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, обрабатывающих 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

¹ **Прерывание** (англ. interrupt) — сигнал, сообщаящий процессору о наступлении какого-либо события. При этом выполнение текущей последовательности команд приостанавливается, и управление передается обработчику прерывания, который реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код

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.

Висновки

Контрольні питання

1. Як можна класифікувати системи захисту програмного забезпечення?
2. Які використовуються механізми для захисту програмного забезпечення?
3. Як захищають програмне забезпечення пакувальники/шифратори, якими є переваги та недоліки такого захисту?
4. Як функціонують системи захисту програмного забезпечення від несанкціонованого копіювання, якими є переваги та недоліки такого захисту?
5. Як захищає програмне забезпечення паролльний захист, якими є переваги та недоліки такого захисту?
6. Як функціонують системи "прив'язки" програмного забезпечення, якими є переваги та недоліки такого захисту?
7. Як функціонують програмно-апаратні засоби захисту програмного забезпечення, якими є переваги та недоліки такого захисту?
8. Яким чином програмне забезпечення SaaS вирішує проблеми захисту
9. Якими є показники застосовності систем захисту програмного забезпечення?
10. Якими є критерії оцінки систем захисту програмного забезпечення?
11. Як можна зобразити взаємодію агентів ринку програмного забезпечення у контексті захисту?
12. Які є основні функції засобів захисту від несанкціонованого копіювання програмного забезпечення?
13. Які є криптографічні методи захисту від несанкціонованого копіювання програмного забезпечення?
14. Якими є методи протидії статичним способам зняття захисту від несанкціонованого копіювання?
15. Якими є методи протидії динамічним способам зняття захисту програм від несанкціонованого копіювання?