Харківський національний університет імені В.Н. Каразіна

Лекция

«Защита программного обеспечения»

1. Основные аспекты безопасности программного обеспечения.
2. Контроль жизненного цикла программного обеспечения.
3. Защита программного обеспечения

Основные аспекты безопасности программного обеспечения

Под "*правильной*" программой понимается программа, которая ведет себя именно так, как описано в ее спецификации.

Спецификацию требований к ПО иногда называют *документом бизнес-требований*, *функциональной спецификацией*, *спецификацией продукта* или просто *документом о требованиях*.

Спецификации требований к ПО должна содержать достаточно подробное описание поведения системы при различных условиях, а также необходимые качества системы, такие как производительность, безопасность и удобство использования.

Спецификация требований служит основой для дальнейшего планирования, дизайна и кодирования, а также базой для тестирования пользовательской документации (при этом она не должна содержать подробности дизайна, проектирования, тестирования и управления проектом за исключением известных ограничений дизайна и реализации).

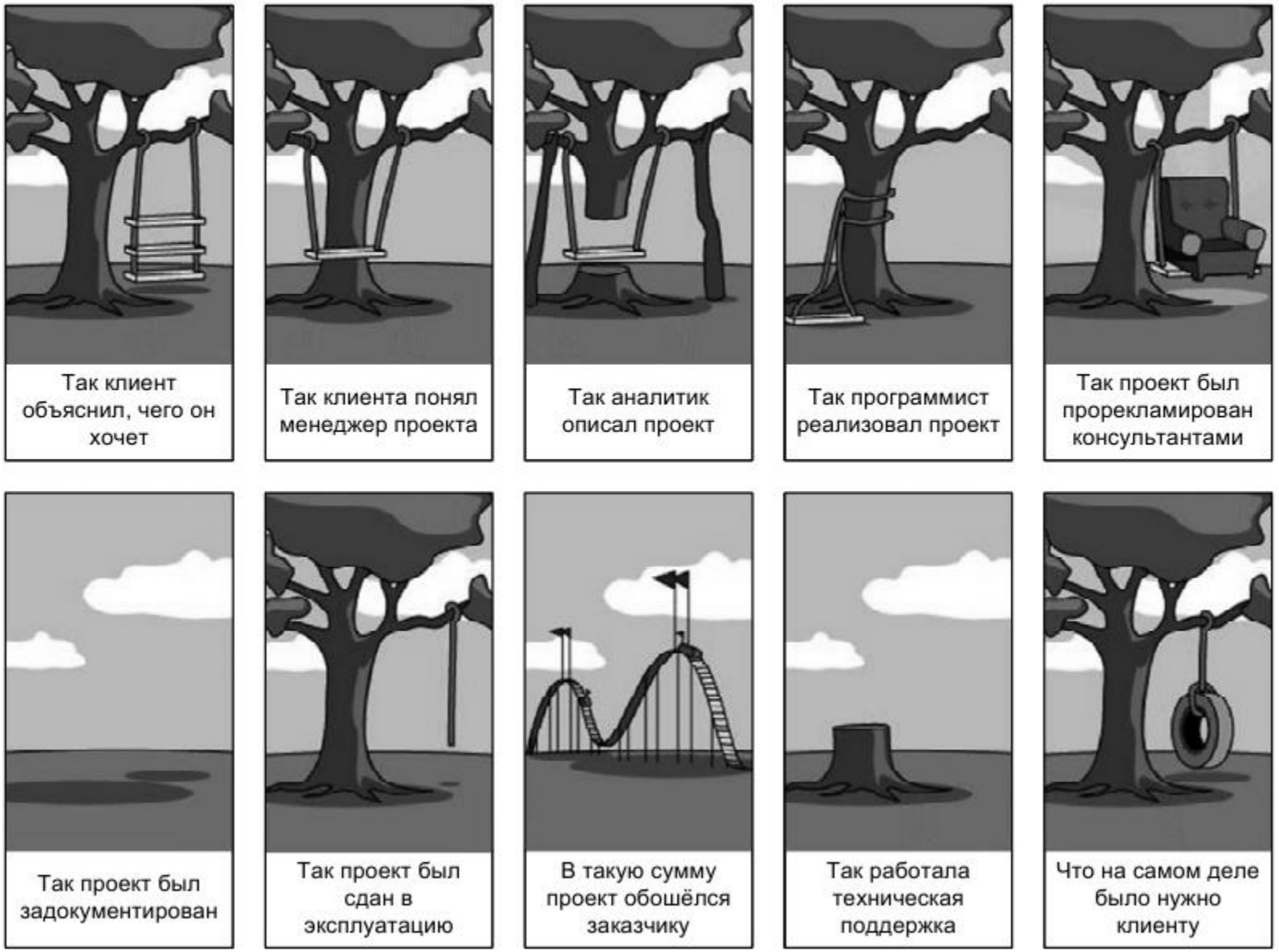
Основные аспекты безопасности программного обеспечения

Способы представления требований:

- документация, в которой используется четко структурированный и аккуратно используемый естественный язык;
- графические модели, иллюстрирующие процессы преобразования, состояния системы и их изменения, отношения данных, а также логические потоки и т. п.;
- формальные спецификации, где требования определены с помощью математически точных, формальных логических языков.

Основные аспекты безопасности программного обеспечения

Типичный проект с плохими требованиями



Основные аспекты безопасности программного обеспечения

Этапы создания спецификации:

1) *требования* (неформальное описание того, для чего предназначена программа; в большинстве случаев требования носят обобщенный характер и не содержат мелких деталей, они сконцентрированы на программе в целом);

2) *функциональная спецификация* (подробное и исчерпывающее описание поведения программы; функциональная спецификация должна быть полной; она должна содержать описание абсолютно всех функциональных элементов программы; все, что не было занесено в функциональную спецификацию, не должно быть реализовано; все элементы, указанные в функциональной спецификации, могут и должны быть протестированы);

3) *план реализации* (описывает внутреннее поведение программы, в нем указаны все те элементы, которые не могут быть протестированы извне; хороший план реализации зачастую разбивает программу на несколько модулей и описывает функциональность каждого из них; описание модуля, в свою очередь, можно рассматривать как составление требований к модулю с последующим созданием функциональной спецификации и плана реализации));

Пример спецификации требований к ПО

Спецификация требований к ПО

1. Введение

1.1. Назначение

Эта спецификация требований к ПО описывает функциональные и нефункциональные требования к выпуску 1.0 XXX. Этот документ предназначен для команды, которая будет реализовывать и проверять корректность работы системы. Кроме специально обозначенных случаев, все указанные здесь требования имеют высокий приоритет и приписаны к выпуску 1.0.

1.2. Соглашения, принятые в документах

В этой спецификации нет никаких типографских условных обозначений.

1.3. Границы проекта

Система XXX позволит реализовать систему цифровой подписи на основе биометрических данных пользователя.

1.4. Ссылки

1. Вигерс К., Битти Д. Разработка требований к программному обеспечению. 3-е изд., дополненное / Пер. с англ. – М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург, 2014. – 736 с.

2. ...

Пример спецификации требований к ПО

2. Общее описание

2.1. Общий взгляд на продукт

XXX – это новый продукт ...

2.2. Классы и характеристики пользователей

2.3. Операционная среда

OE-1 Система XXX работает со следующими браузерами: Windows Internet Explorer, Firefox версии с 12 по 26, Google Chrome.

…

2.4. Ограничения дизайна и реализации

CO-1 Документация системы по дизайну, коду и сопровождению должна соответствовать стандарту ...

2.5. Предположения и зависимости

3. Системные функции

3.1. .....

3.1.1. Описание

3.1.2. Функциональные требования

.....

3.2. .....

3.2.1. Описание

3.2.2. Функциональные требования

.....

Пример спецификации требований к ПО

4. Требования к данным
4.1. Логическая модель данных

4.2. Словарь данных

4.3. Отчеты

4.4. Целостность, сохранение и утилизация данных

5. Требования к внешним интерфейсам
5.1. Пользовательские интерфейсы
UI-1 Интерфейсы XXX должны соответствовать ….
……..

6. Атрибуты качества
6.1. Требования по удобству использования

6.2.Требования к производительности

6.3. Требования к безопасности
……..

Основные аспекты безопасности программного обеспечения

Причины, по которым возникают сложности при написании правильных программ:

- **отсутствие *спецификации*;**
- **несовершенством метода разработки "*протестировать и исправить*»**
- **недобросовестное (халатное) отношение к создаваемому программному обеспечению разработчиков.**

Несколько простых правил отслеживания ошибок:

a) если вы нашли ошибку, реализуйте тест, который будет обнаруживать эту ошибку; убедитесь, что он действительно ее обнаруживает, затем исправьте ошибку и убедитесь, что тест больше не обнаруживает ее; впоследствии применяйте этот тест ко всем будущим версиям программы, чтобы проверить, не появится ли указанная ошибка снова;

b) найдя ошибку, подумайте, чем она вызвана. Нет ли в программе других мест, в которых может находиться аналогичная ошибка? Проверьте все эти места;

c) ведите учет всех найденных ошибок. Простой статистический анализ обнаруженных ошибок может показать, какая часть программы является самой проблемной, ошибки какого типа встречаются наиболее часто и т. п. Это необходимо для системы контроля качества.

Основные аспекты безопасности программного обеспечения

В чем же различие между правильным и безопасным программным обеспечением?

Правильное программное обеспечение обладает заявленной функциональностью: если вы щелкнете на кнопке *А*, случится событие *Б*.

К безопасному программному обеспечению выдвигается еще одно требование – *недостаточности функциональности:* что бы ни делал злоумышленник, он не должен выполнить операцию *X*.

Основное правило написания систем безопасности:
уничтожайте всю информацию, как только она вам больше не понадобится.
Чем дольше она будет храниться в памяти компьютера, тем вероятнее, что до нее доберется посторонний. Более того, не забывайте уничтожать данные прежде, чем вы потеряете контроль над носителем, используемым для их хранения).

Сложность – главный враг безопасности.
Каждый разработчик системы безопасности должен стремиться к простоте. Уберите из системы все параметры и настройки, какие только возможно, и которые очень часто никому не нужны.

Следите за переполнением буфера.
Данная проблема основывается на возможности при нарушении границ памяти аварийно завершить приложение или выполнить произвольный бинарный код от имени пользователя, под которым работала уязвимая программа.

Основные аспекты безопасности программного обеспечения

Разработка программного обеспечения с точки зрения информационной безопасности охватывает обычно три аспекта:

1. Проектирование механизмов, предназначенных для поддержания модели безопасности, заложенной в системе.
2. Контроль программного обеспечения на протяжении всего его жизненного цикла на предмет внесения несанкционированных изменений или наличия недокументированных возможностей.
3. Защита самого программного обеспечения как интеллектуальной собственности или объекта распространения авторского права.

Контроль жизненного цикла программного обеспечения

Контроль жизненного цикла программного обеспечения необходим для того, чтобы обеспечить уверенность в том, что информационная система на основе данного ПО функционирует точно так, как предполагалось в техническом задании, и что она не имеет недокументированных возможностей в виде так называемых программных закладок (*backdoor, logical bomb*).

Каким способом организация может обезопасить себя?

В качестве решения, как правило, используются известные принципы безопасности:

- ✓ *минимальных привилегий* (*least privileges*);
- ✓ *разделения обязанностей* (*separation of duties*) применительно к разработке и использованию ПО как к полноценному информационному процессу.

Это означает, что
процесс разработки программного обеспечения должен быть отделен от распространения, установки, эксплуатации и сопровождения системы!

Контроль жизненного цикла программного обеспечения

Типовой сценарием для обеспечения безопасной работы

- В организации создаются отдельные службы: разработки, хранения эталонных копий, внедрения и поддержки.
- Первая рабочая копия передается службой разработки службе хранения в виде исходных текстов, где проходит тщательный анализ на отсутствие недокументированных возможностей.
- После успешной верификации отсутствия недокументированных возможностей служба хранения производит компиляцию исходных текстов в исполняемые модули, возможно, подписывает их с помощью ЭЦП, рассчитывает и фиксирует контрольную сумму каждого модуля как файла данных либо применяет другие механизмы контроля целостности программного обеспечения.
- Скомпилированные модули и средства контроля целостности перелаются службе внедрения и поддержки. Исходные коды и эталоны программных модулей сохраняются в специальном хранилище, куда имеют доступ только уполномоченные сотрудники службы, ни в коем случае ни разработчики, ни служба поддержки.

Контроль жизненного цикла программного обеспечения

Типовой сценарием для обеспечения безопасной работы

(продолжение)

- Группа внедрения и поддержки обеспечивает регулярную проверку рабочих копий системы на целостность.
- В следующем цикле разработки, когда группа программирования представляет новую версию с обновлениями/изменениями и соответствующей документацией, служба хранения производит проверку только измененных составляющих, после чего процесс компиляции, фиксации и передачи в работу повторяется. Контроль изменений проще всего осуществлять, если разработка ведется объектно-ориентированным способом или с применением специализированных средств поддержки версий (например, *CVS - Concurrent Versions System* - система конкурирующих (одновременных) версий или более поздним средством – *Subversion* - программа разработана специально для замены широко распространённой, но явно устаревшей системы *CVS*).

Теоретически, в рамках данной схемы злоупотребления могут произойти в службе хранения, если несанкционированные изменения будут внесены в программный код после проверки на недокументированные возможности, но до компиляции.

Контроль жизненного цикла программного обеспечения

Усилить работу схемы возможно, если служба разработки будет передавать исходные коды службе хранения, а уже скомпилированные модули, службе поддержки (рис. 1).

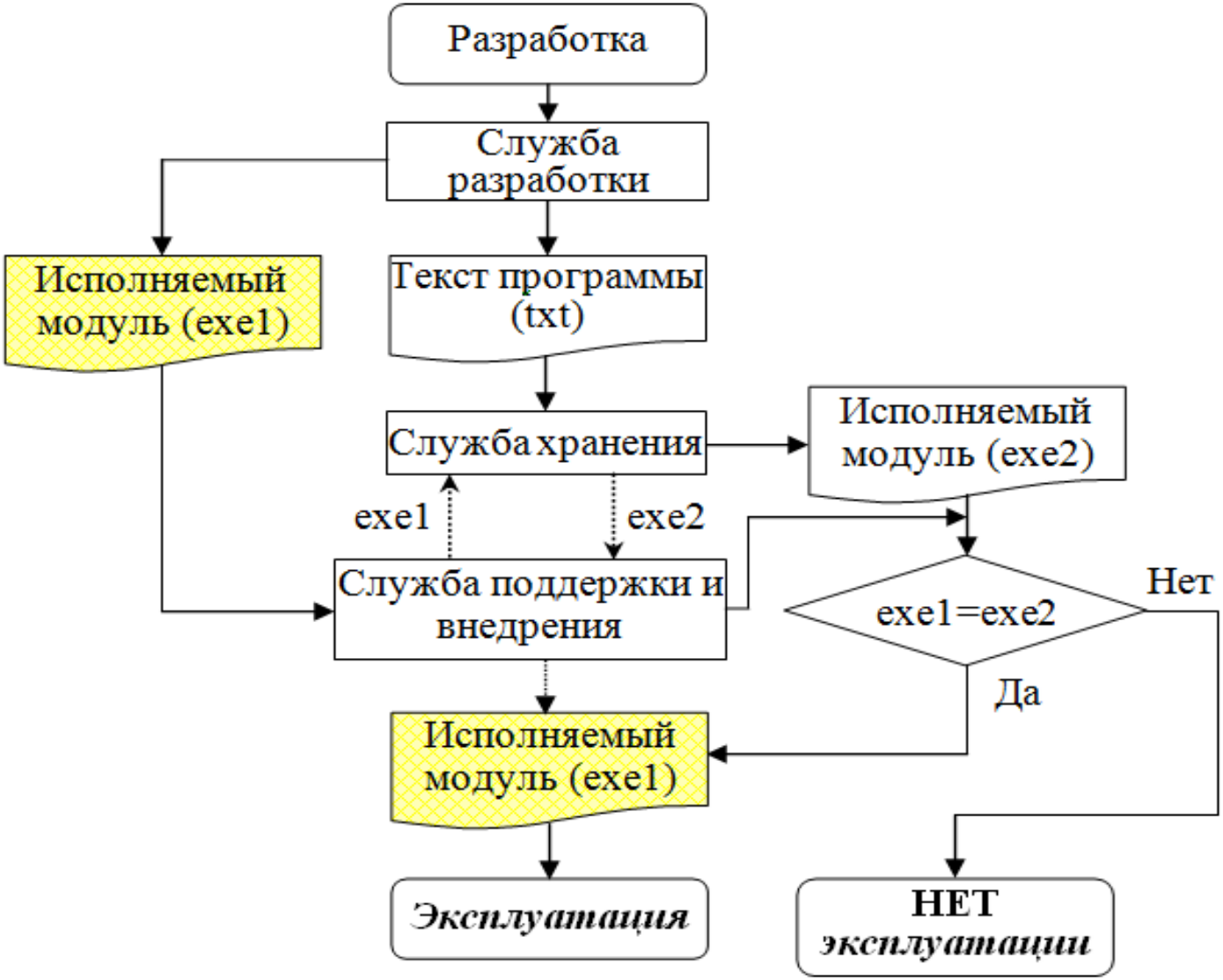


Рис. 1 – Схема процесса разработки, хранения, внедрения

Защита программного обеспечения

Основные подходы к защите программ от несанкционированного использования и копирования

Под системой защиты от несанкционированного использования и копирования (защиты авторских прав, или просто защиты от копирования) понимается комплекс программных или программно-аппаратных средств, предназначенных для усложнения или запрещения нелегального распространения, использования и (или) изменения программных продуктов и иных информационных ресурсов. Термин «нелегальное» здесь понимается как производимое без согласия правообладателя. Нелегальное изменение информационного ресурса может потребоваться нарушителю для того, чтобы измененный им продукт не подпадал под действие законодательства о защите авторских прав.

Под надежностью системы защиты от несанкционированного использования и копирования понимается ее способность противостоять попыткам изучения алгоритма ее работы и обхода реализованных в нем методов защиты. Очевидно, что любая программная или программно-аппаратная система защиты от копирования может быть преодолена за конечное время, так как процессорные команды системы защиты в момент своего исполнения присутствуют в оперативной памяти компьютера в открытом виде. Также очевидно, что надежность системы защиты равна надежности наименее защищенного из ее модулей.

Защита программного обеспечения

Основные подходы к защите программ от несанкционированного использования и копирования

При создании и использовании систем защиты от несанкционированного использования и копирования необходимо учитывать следующее:

- условия распространения программных продуктов;
- возможности пользователей программного продукта по снятию с него системы защиты (наличие достаточных материальных ресурсов, уровень знаний и квалификация лиц, снимающих систему защиты, возможность привлечения необходимых специалистов и т. п.);
- свойства распространяемого программного продукта (предполагаемого тиража, оптовой и розничной цены, частоты обновления, сложности продукта, уровня послепродажного сервиса для легальных пользователей, возможности применения правовых санкций к нарушителю и др.);
- возможные потери при снятии защиты и нелегальном использовании;
- период обновление использованных в системе защиты средств.

Защита программного обеспечения

Основные требования, предъявляемые к системе защиты от несанкционированного использования и копирования:

- обеспечение некопируемости дистрибутивных дисков стандартными средствами (для такого копирования нарушителю потребуется тщательное изучение структуры диска с помощью специализированных программных или программно-аппаратных средств);
- обеспечение невозможности применения стандартных отладчиков без дополнительных действий над машинным кодом программы или без применения специализированных программно-аппаратных средств (злоумышленник должен быть специалистом высокой квалификации);
- обеспечение некорректного дисассемблирования машинного кода программы стандартными средствами (злоумышленнику потребуется использование или разработка специализированных дисассемблеров);
- обеспечение сложности изучения алгоритма распознавания индивидуальных параметров компьютера, на котором установлен программный продукт, и его пользователя или анализа применяемых аппаратных средств защиты (злоумышленнику будет сложно эмулировать легальную среду запуска защищаемой программы).

Защита программного обеспечения

Основные компоненты системы защиты программных продуктов от несанкционированного использования и копирования:

- модуль проверки ключевой информации (некопируемой метки на дистрибутивном диске, уникального набора характеристик компьютера, идентифицирующей информации для легального пользователя) – может быть добавлен к исполнимому коду защищаемой программы, в виде отдельного программного модуля или в виде отдельной функции проверки внутри защищаемой программы;
- модуль защиты от изучения алгоритма работы системы защиты;
- модуль согласования с работой функций защищаемой программы в случае ее санкционированного использования;
- модуль ответной реакции в случае попытки несанкционированного использования (как правило, включение такого модуля в состав системы защиты нецелесообразно по морально-этическим соображениям).

Защита программного обеспечения

Для снятия защиты от несанкционированного использования и копирования применяют два основных метода:

- ✓ *статический*;
- ✓ *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.

Угроза несанкционированного копирования информации блокируется методами, которые могут быть распределены по двум группам:

- ➢ методы, затрудняющие считывание скопированной информации;
- ➢ методы, препятствующие использованию информации.

Методы из первой группы основываются на придании особенностей процессу записи информации, которые не позволяют считывать полученную копию на других накопителях, не входящих в защищаемую ИС.

Методы, препятствующие использованию скопированной информации, имеют целью затруднить использование полученных путем несанкционированного копирования данных при их несанкционированном исполнении, тиражировании и исследовании. Скопированная информация может быть программой или данными.

Защита программного обеспечения

Методы, затрудняющие считывание скопированной информации

Самым простым решением является нестандартная разметка (форматирование) носителя информации. Изменение длины секторов, межсекторных расстояний, порядка нумерации секторов и некоторые другие способы нестандартного форматирования затрудняют их использование стандартными средствами операционных систем.

Перепрограммирование контроллеров внешних запоминающих устройств, аппаратные регулировки и настройки вызывают сбой оборудования при использовании носителей на стандартных внешних запоминающих устройствах (ВЗУ), если форматирование и запись информации производились на нестандартном ВЗУ.

Выбор конкретного метода изменения алгоритма работы ВЗУ (или их композиции) осуществляется с учетом удобства практической реализации и сложности повторения алгоритма злоумышленником.

(При разработке ВЗУ необходимо учитывать потребность использования устройств в двух режимах: в стандартном режиме и в режиме совместимости на уровне ИС. Выбор одного из режимов, а также выбор конкретного алгоритма нестандартного использования должен осуществляться, например, записью в ПЗУ двоичного кода. Число нестандартных режимов должно быть таким, чтобы исключался подбор режима методом перебора. Процесс смены режима должен исключать возможность автоматизированного подбора кода.)

Защита программного обеспечения

Методы, препятствующие использованию скопированной информации

Эта группа методов имеет целью затруднить использование полученных путем несанкционированного копирования данных при их несанкционированном исполнении, тиражировании и исследовании. Скопированная информация может быть программой или данными.

Криптографические методы

Для защиты инсталлируемой программы от дальнейшей возможности ее использования после 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 (хэш-функция);
- ✓ с использованием шифрования.

Защита программного обеспечения

Общий алгоритм механизма защиты от несанкционированного использования программ в «чужой» среде размещения сводится к выполнению следующих шагов.

Шаг 1. Запоминание множества индивидуальных контрольных характеристик ЭВМ и (или) съемного носителя информации на этапе инсталляции защищаемой программы.

Шаг 2. При запуске защищенной программы управление передается на блок контроля среды размещения. Блок осуществляет сбор и сравнение характеристик среды размещения с контрольными характеристиками.

Шаг 3. Если сравнение прошло успешно, то программа выполняется, иначе – отказ в выполнении. Отказ в выполнении может быть дополнен выполнением деструктивных действий в отношении этой программы, приводящих к невозможности выполнения этой программы, если такую самоликвидацию позволяет выполнить ОС.

Привязка программ к среде размещения требует повторной их инсталляции после проведения модернизации, изменения структуры или ремонта ИС с заменой устройств.

Для защиты от несанкционированного использования программ могут применяться электронные ключи, которые распространяются с защищаемой программой. Программа в начале и в ходе выполнения считывает контрольную информацию из ключа. При отсутствии ключа выполнение программы блокируется.

Защита программного обеспечения

Методы противодействия динамическим способам снятия защиты программ от копирования

- 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 ее 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 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.

Методы противодействия динамическим способам снятия защиты программ от копирования (продолжение)

- 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.

Указанные методы позволяют в определенной мере выявить те изменения в программе, которые вносятся злоумышленником 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.

Защита программного обеспечения

Защита программных средств от исследования

Все средства исследования программного обеспечения можно разбить на два класса: *статические и динамические*.
Первые оперируют исходным кодом программы как данными и строят ее алгоритм без исполнения, вторые же изучают программу, интерпретируя ее в реальной или виртуальной вычислительной среде.

Два наиболее известных типа программ, предназначенных для исследования ПО: *отладчик* (динамическое средство) и *дизассемблер* (средство статистического исследования).

Если первый широко применяется пользователем для отладки собственных программ и задачи построения алгоритма для него вторичны и реализуются самим пользователем, то второй предназначен исключительно для их решения и формирует на выходе ассемблерный текст алгоритма.

Методы противодействия дизассемблированию:
✓шифрование;
✓архивация;
✓использование самогенерирующих кодов;
✓«обман» дизассемблера.

Для противодействия трассировке программы **в ее состав вводятся следующие механизмы**:
✓изменение среды функционирования;
✓модификация кодов программы;
✓«случайные» переходы.

Литература

1. Есин В. И., Кузнецов А. А., Сорока Л. С. Безопасность информационных систем и технологий – Х.:ООО «ЭДЭНА», 2010. – 656с.
2. Єсін В. І. Безпека інформаційних систем і технологій / В. І. Єсін, О. О. Кузнецов, Л. С. Сорока. – Х. : ХНУ імені В. Н. Каразіна, 2013. – 632 с.
1. Конев И.Р., Беляев А.В. Информационная безопасность предприятия.- СПб.:БХВ-Петербург, 2003, 752с.:ил.
2. Фергюсон Нильс, Шнайер Брюс Практическая криптография. : Пер.с англ. – М.: Издательский дом "Вильямс", 2005. — 424 с. : ил.
3. Вигерс К., Битти Д. Разработка требований к программному обеспечению. 3-е изд., дополненное / Пер. с англ. – М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург, 2014. – 736 с.