Лекция 10. Защита программного обеспечения в кибербезопасности

Защита программ от несанкционированного копирования

пользователи могут создавать неограниченное число резервных копий защищенной программы, но использовать могут только одну, работающую с ключом, установленным в один из портов компьютера;

данные пользователей защищаются от несанкционированного использования, поскольку без ключа расшифровать их практически невозможно;

защищенная программа при запуске может проверяться на наличие вирусов или подвергаться контролю на целостность.

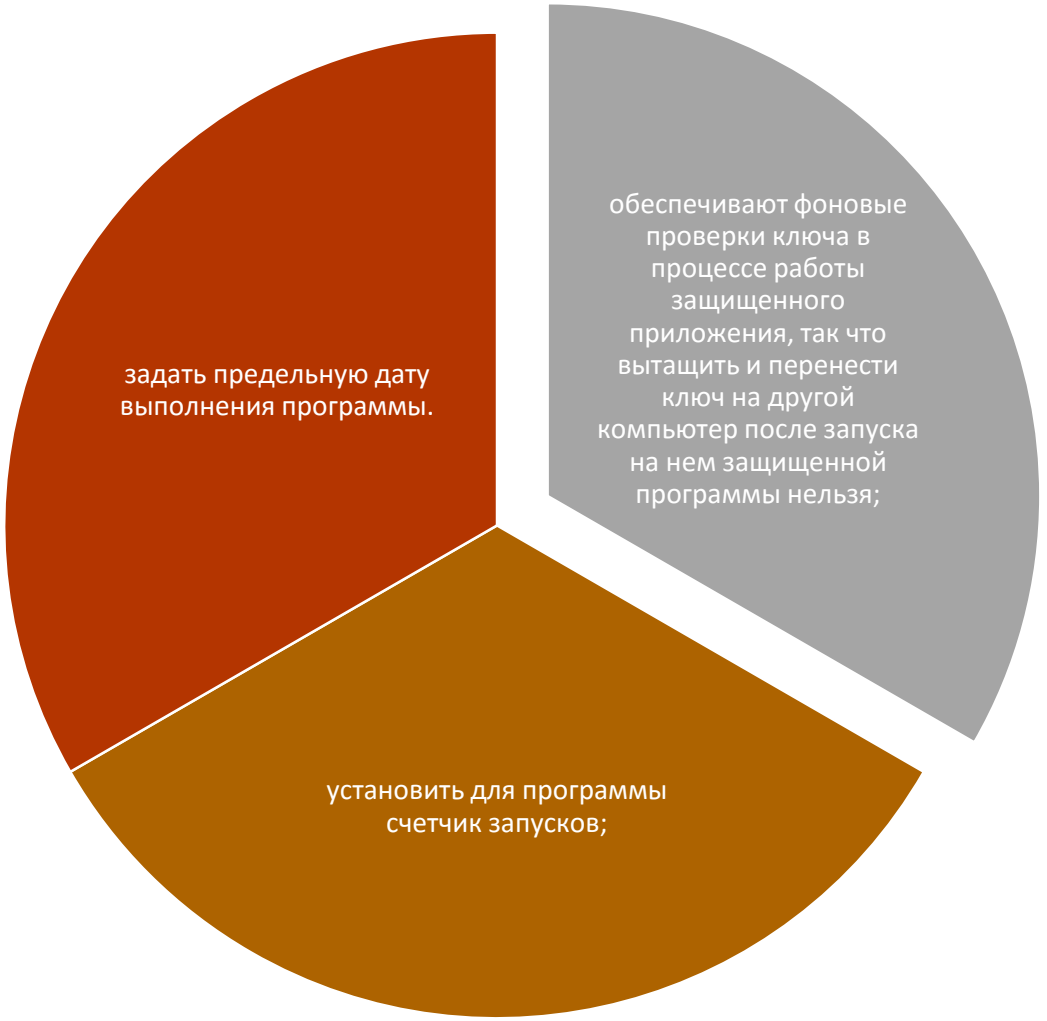
Защитный конверт

проверяется наличие электронного ключа и считывание из него требуемых параметров;

проверка "ключевых" условий и выработка решения;

в случае успешных проверок производится загрузка, расшифровка и передача управления защищенной программе;

в случае неудачных проверок загрузка и расшифровка тела программы в память не производится, выдается сообщение об ошибке и защищенное приложение заканчивает свое выполнение.

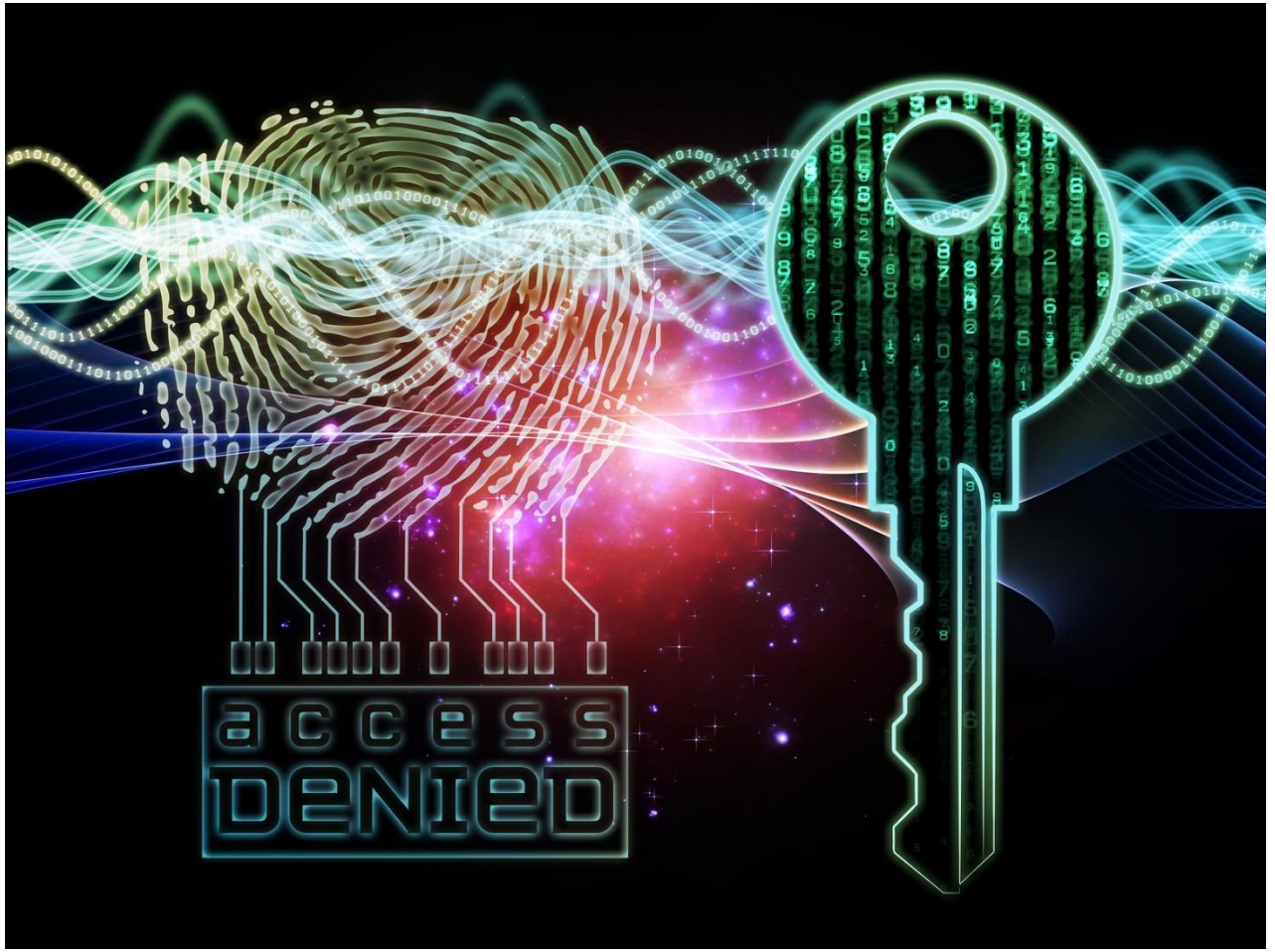


Ключи защиты программ

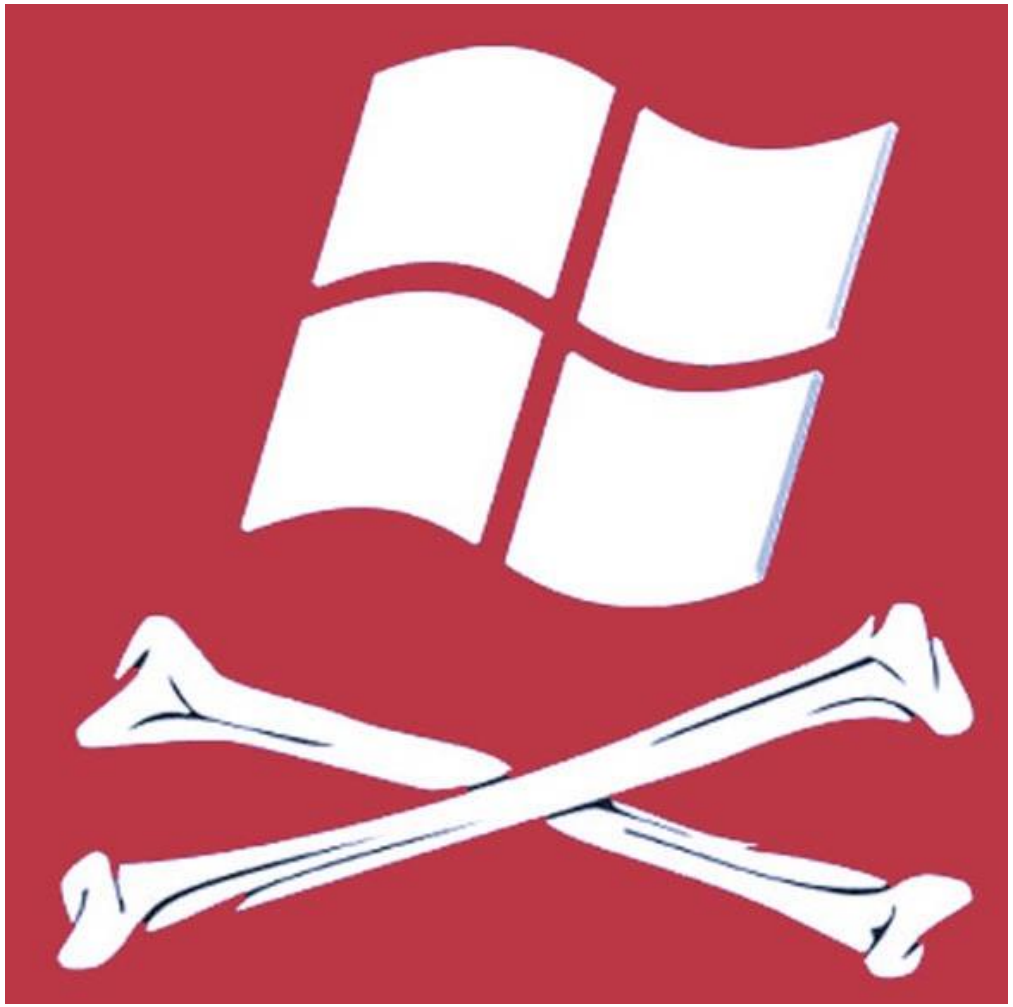
микросхем энергонезависимой электрически перепрограммируемой памяти;

заказных микросхем с памятью или без памяти;

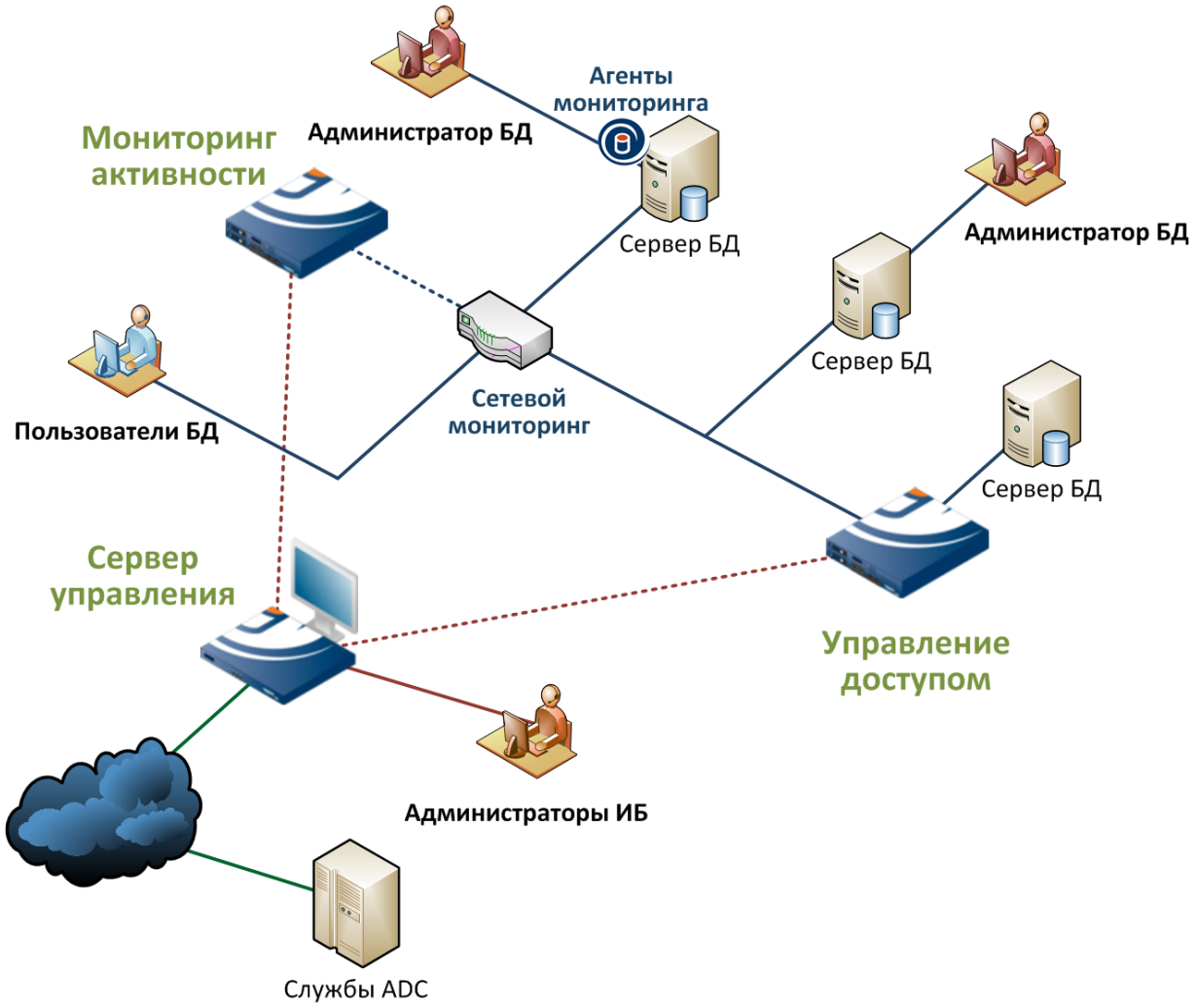
на базе микроконтроллеров.



Пиратские версии программ



Структура системы технической защиты



Источник: https://elvis.ru/services/application/database/

- система защиты должна выявлять факт несанкционированного запуска программы;

- система защиты должна реагировать на факт несанкционированного запуска программы;

- система защиты должна противостоять возможным атакам злоумышленников.

1. Разработчик программы внедряет защитные механизмы в защищаемую программу.

2. В защитные механизмы закладываются эталонные характеристики объекта, которые идентифицируют конкретную копию программы и относительно которых будет проверяться легальность запуска.

3. При каждом запуске программы:

- снимаются текущие характеристики объекта;

- текущие характеристики сравниваются с эталонными;

- если сравнение характеристик дало положительный результат — запускается (либо продолжает работать) защищаемая программа;

- если сравнение характеристик дало отрицательный результат — запускается блок ответной реакции.

Подсистема внедрения управляющих механизмов

1) встроенные (внедряются при создании программного продукта);

2) пристыковочные (подключаются к уже готовому программному продукту).

Подсистема внедрения управляющих механизмов

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

- простота технологии применения — защита поставляется в виде законченного продукта, которым нужно обработать защищаемую программу;

- обеспечение в большинстве случаев достаточного уровня защищенности данных;

- построение собственной встроенной системы

Подсистема внедрения управляющих механизмов

- первая загрузка исследуемой программы;
- определение границ занятой программой памяти;
- сброс первого дампа в файл;
- вторая загрузка исследуемой программы;
- определение границ занятой программой памяти;

Подсистема противодействия нейтрализации защитных механизмов

Блок сравнения характеристик среды

1. Установка значений характеристик среды функционирования защищаемой программы и, следовательно, сравнение значения характеристик с эталонными, должны производиться много раз.

2. Сравнение текущих значений характеристик с эталонными должно производиться периодически в течение всего сеанса работы программы.

3. Значения, полученные от блока установки характеристик среды, можно использовать как ключ для расшифровки кода, по которому (перед передачей ему управления) подсчитывается контрольная сумма.

4. Получение результатов сравнения должно быть принудительно распределено в коде.

5. Если код выработки результатов сравнения не может быть распределен, для подтверждения правильности передаваемых результатов должна также передаваться аутентифицирующая информация ("пароль" обращения к блоку сравнения характеристик среды).

Блок установки характеристик среды

- компьютер;

- биометрические характеристики пользователя;

- специальные аппаратные устройства;

- машинные носители.

• возможностью создания резервных копий программ;

• независимостью от аппаратных особенностей компьютера;

• возможностью свободного переноса защищенных программ с компьютера на компьютер.

Методы нейтрализации защиты

1. Определение модели программно – аппаратной среды функционирования защищенной программы, которая описывает перечень идентифицирующих элементов среды, а также их параметров с целью воспроизведения копии среды. Примером этого направления является воспроизведение копии ключевой дискеты.

2. Определение модели алгоритма сравнения установленных значений характеристик программно – аппаратной среды с эталонными с целью искажения кода сравнения таким образом, чтобы результат сравнения всегда был истинным, т.е. чтобы система защиты всегда опознавала текущую копию как легальную.

Business-to-Business (B2B) Marketing on the Web

09D8:0140
CMP BYTE PTR [A76C],00

0908:0145
JNZ 014B

0908:0148
CALLOC50

09D8:014B
MOV AH,18

Системы защиты персональных данных Система защиты данных методом прозрачного кодирования Secret Disk

ключ для параллельного порта HASP;

микропроцессорная карта ASE (рекомендуется использовать, когда одним компьютером пользуются несколько человек);

плату PCMCIA (для пользователей портативных компьютеров);

устройство e-Token-ключ для порта USB

1. Априорно неизвестно наличие в каком-либо множестве программ фрагментов РПВ. Ставится задача определения факта их наличия или отсутствия; при этом программы не выполняются (статическая задача).

2. При условиях, рассматриваемых в п.1, прикладные программы используются по своему прямому назначению. Также ставится задача выявления закладки, но в данном случае динамическая (по результатам работы).

продуктом (либо в пространстве – передача по каналу связи или пересылка на магнитном носителе, либо во времени – хранение), априорно

опасных действий.

4. При условиях п.3 решается динамическая задача — защита от воздействия закладки (закладок) в ходе работы программ.

5. При условии потенциальной возможности воздействия закладок решается задача борьбы с их итоговым влиянием, т.е. закладки присутствуют в системе, но либо не активны при выполнении критических действий прикладных программ, либо результат их воздействия не конструктивен.

1.Общие методы защиты программного обеспечения, решающие задачи борьбы со случайными сбоями оборудования и несанкционированным доступом.

а) Контроль целостности системных областей, запускаемых прикладных программ и используемых данных (решение задачи 3).

б) Контроль критических для безопасности системы событий (решение задачи 2).

Данные методы действенны лишь тогда, когда контрольные элементы не подвержены воздействию закладок и разрушающее воздействие либо инициирующее его событие входят в контролируемый класс. Так, система контроля за вызовом прерываний не будет отслеживать обращение к устройствам на уровне портов. С другой стороны, контроль событий может быть обойден путем:

навязывания конечного результата проверок;

влияния на процесс считывания информации;

изменения контрольных элементов (хеш-функций), хранящихся в общедоступных файлах или в оперативной памяти.

в) Создание безопасной и изолированной операционной среды (решение задачи 4).

г) Предотвращение результирующего воздействия вируса или закладки (например, запись на диск только в зашифрованном виде на уровне контроллера – тем самым локальное сохранение информации закладкой не имеет смысла – или запрет записи на диск на аппаратном уровне) (решение задачи 5).

2. Специальные методы выявления программ с потенциально опасными последствиями.

а) Поиск фрагментов кода по характерным последовательностям (сигнатурам), свойственным закладкам, либо, наоборот, разрешение на выполнение или внедрение в цепочку прерываний только программам с известными сигнатурами (решение задач 1,2).

б) Поиск критических участков кода (с точки зрения безопасности компьютерной системы) методом семантического анализа. При этом анализ фрагментов кода на выполняемые ими функции, например выполнение НСЗ, часто сопряжен с дизассемблированием или эмуляцией выполнения (решение задач 1,2).

1. Проверенные программы будут использованы на другом компьютере с другой BIOS, которая может содержать закладки.

2. Проверенные программы будут использованы в аналогичной, но не проверенной операционной среде, в которой могут содержаться закладки.

3. Проверенные программы используются на проверенном компьютере и в проверенной операционной среде, но запускаются еще и не проверенные программы, потенциально несущие в себе закладки.

1) на компьютере с проверенной BIOS установлена проверенная операционная среда;

2) достоверно установлена неизменность операционной среды и BIOS для данного сеанса работы;

3) кроме проверенных программ в данной программно-аппаратной среде не запускалось и не запускается никаких иных программ;

4) исключен запуск проверенных программ в какой-либо иной ситуации, т.е. вне проверенной среды.

Текстовый редактор

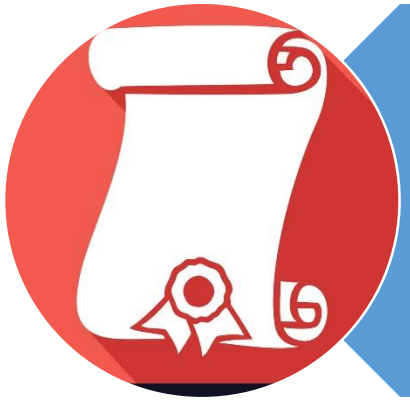
1) невозможность запуска программ в обход контролируемых ИПС событий;

2) отсутствие возможностей влиять на среду функционирования уже запущенных программ

Программный модуль

Алгоритм A

по известному числу M=A(F) очень трудоемким должно быть нахождение другого файла O, не равного F, такого, что M=A(G);



число M должно быть недоступно для изменения.