Министерство образования и науки Российской Федерации
Казанский национальный исследовательский технический университет им. А.Н. Туполева
Кафедра систем информационной безопасности

ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТ НЕСАНКЦИОНИРОВАННОГО ИСПОЛЬЗОВАНИЯ.

Подготовил
студент группы 4106
Ишанкулов А.Д

Казань-2016

Проблема защиты программного обеспечения от несанкционированного использования.

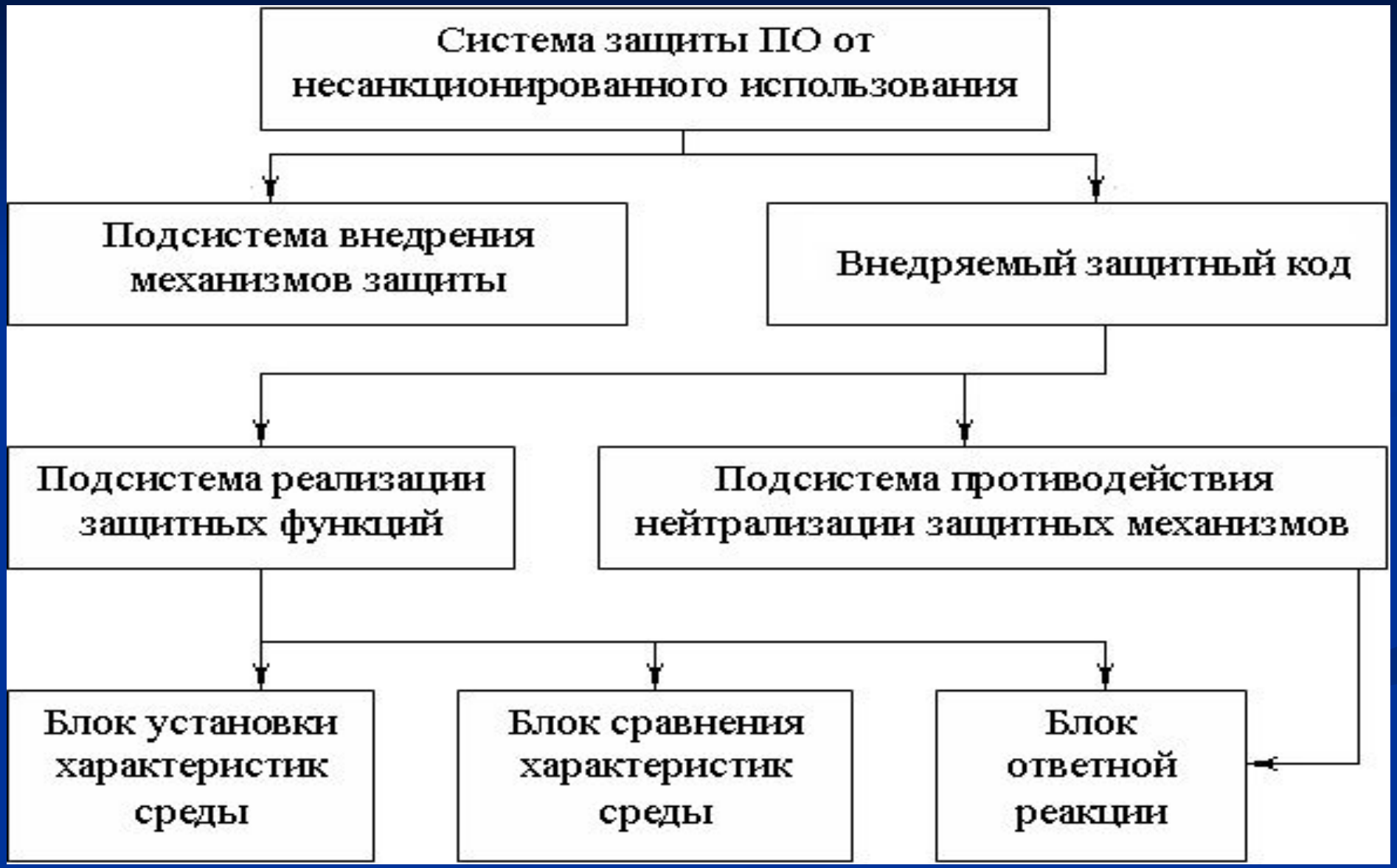
Наиболее типичными ситуациями, которые возникают при получении злоумышленником нелегальных копий, являются следующие:

1. Изготовление официальным распространителем нелегальных копий программ.
2. Изготовление и коммерческое распространение злоумышленником поддельных нелегальных копий программ без разрешения правообладателей.
3. Несанкционированное правообладателями коммерческое распространение программ.
4. Получение нелегальной копии программы у законного пользователя .
5. Превышение числа разрешенных правообладателями инсталляций законно приобретенной программы.

Основные требования к системе защиты ПО от несанкционированного использования:

- система защиты должна **выявлять** факт несанкционированного запуска программы;
- система защиты должна **реагировать** на факт несанкционированного запуска программы;
- система защиты должна **противостоять** возможным атакам злоумышленников, направленным на нейтрализацию системы защиты.

Модульная архитектура системы защиты ПО от несанкционированного использования.



Обобщённый алгоритм функционирования системы защиты

1. Разработчик программы внедряет защитные механизмы в защищаемую программу.
2. В защитные механизмы закладываются эталонные характеристики среды, которые идентифицируют конкретную копию программы, и относительно которых будет проверяться легальность запуска.
3. При каждом запуске программы выполняются следующие действия:

- снимаются текущие характеристики среды;
- текущие характеристики сравниваются с эталонными характеристиками;
- если сравнение характеристик дало положительный результат, то программа запускается (либо продолжает работать);
- если сравнение характеристик дало отрицательный результат, то запускается блок ответной реакции.

Функционирование подсистем и модулей системы защиты программного обеспечения от несанкционированного использования.

- **Подсистема внедрения управляющих механизмов:**
- 1) встроенные системы (внедряются при создании ПО);
- 2) пристыковочные системы (подключаются к уже готовому ПО).
- **Подсистема противодействия нейтрализации защитных механизмов;**
- **Блок ответной реакции;**
- **Блок сравнения характеристик среды.**
- **Блок установки характеристик среды.**

Подсистема внедрения механизмов защиты

1. **встроенные системы** (внедряются при создании ПО);

2. **пристыковочные системы** (подключаются к уже готовому ПО).

Преимущества встроенных систем защиты

- более просто реализовать любую реакцию системы защиты ПО на несанкционированный запуск;
- возможно опрашивать идентифицирующий элемент где угодно, когда угодно и столько угодно раз.

Преимущества защит пристыковочного типа

- простота тиражирования программных систем защиты;
- простота технологии применения;
- возможность включения в пристыковочные защиты элементов защиты ПО от изучения с помощью отладчиков и дизассемблеров;
- использование пристыковочных защит не требует наличия исходных текстов программы.

Основные требования к реализации защитных механизмов

- Установка значений характеристик среды и сравнение с эталонными должны производиться многократно и в течение всего сеанса работы программы.
- Не сравнивать характеристики среды в открытом виде (использовать функции хэширования).
- Защищать софт от исследования статическими и динамическими средствами.
- Получение результатов сравнения должно быть принудительно распределено в коде программы.

Основные требования к реализации защитных механизмов

- Удачным приёмом против потенциального злоумышленника считается преобразование значения на выходе блока установки характеристик среды в некоторую переменную, которая затем может быть использована, например, как аргумент для обращения к управляющей таблице при вычислении значения адреса перехода или вызова подпрограммы.
- Блок ответной реакции следует реализовывать в неявном виде. Возможны ситуации, когда данный блок вообще отсутствует, либо выражен неявным образом. Например, информация, полученная от блока установки характеристик среды, может быть использована в качестве ключа для расшифрования кода программы. Если значения характеристик среды отличаются от эталонных, то при расшифровании получится «мусор», который при исполнении, как правило, «подвешивает» систему.

Защита программного обеспечения с помощью электронных ключей HASP

- **Электронные ключи серии HASP 4.**
- HASP4 Standard.
- MemoHASP.
- TimeHASP.
- NetHASP.

- **Электронные ключи серии HASP HL.**

Электронные ключи HASP

Разработка фирмы Aladdin представляют собой современное аппаратное средство защиты ПО от несанкционированного использования.

Базовой основой ключей HASP является специализированная заказная микросхема (ASIC – Application Specific Integrated Circuit), имеющая уникальный для каждого ключа алгоритм работы - функцию шифрования и связанную с ней функцию отклика $f(x)$, принимающую на вход 16-битный аргумент и формирующую на выходе четыре 16-битных значения.

Модели семейства ключей HASP

- HASP4 Standard;
- MemoHASP;
- TimeHASP;
- NetHASP.

- Каждый из ключей обладает определенными кодами доступа (два 16-битных значения), соответствующие его серии.
- Внутри одной серии совпадают коды доступа и алгоритм работы.

Система защиты HASP Standard позволяет осуществлять

- проверку наличия HASP Standard;
- проверку соответствия выходов, формируемых функцией отклика $f(x)$ для различных значений x, эталонным значениям;
- использовать функцию шифрования электронного ключа для шифрования и расшифрования своего исполняемого кода, используемых данных и т.д.

MemoHASP

Добавлена встроенная в них энергонезависимой памяти (EEPROM), доступной для чтения и записи во время выполнения защищенной программы.

Модификации данных ключей

- HASP4 M1 – 112 байт EEPROM, возможность одновременной защиты до 16 программ.
- HASP4 M4 – 496 байт EEPROM, возможность одновременной защиты до 112 программ.

С помощью MemoHASP могут быть реализованы

- Хранение в энергонезависимой памяти MemoHASP конфиденциальной информации – ключей шифрования, части исполняемого кода и т.д.
- Хранение в энергонезависимой памяти информации о модулях защищённого программного обеспечения, к которым пользователь имеет доступ и о тех, к которым не имеет (в зависимости от заплаченной суммы за приобретение программы).
- Хранение в энергонезависимой памяти информации о количестве запусков программы, либо об оставшемся количестве запусков. Данный подход актуален при создании демонстрационных версий программ, 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 программное обеспечение по времени использования и на основании этого строить гибкую маркетинговую политику, например, сдачу программ в аренду и периодический сбор платы за его использование и т.д.

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 любом узле компьютерной сети.

Способы внедрения защитных механизмов в ПО с помощью электронных ключей HASP

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

Возможности HASP

- Подсистема полного управления доступом (FAS)
- Защита структурным кодом (PCS)
- Система удаленного обновления (RUS)

Рекомендации по более надежной защите

- Использовать одновременно методы защиты с помощью оболочки и с помощью API. Они дополняют и усиливают друг друга.
- 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.
- Шифровать внутренние и внешние данные защищаемого приложения. Расшифрование проводить на ключе HASP. В данном случае взломщику нужно будет не только взломать приложение, но и расшифровать 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.

Рекомендации по более надежной защите

- 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.
- Разделять в коде программы шаги вызова процедуры hasp(), анализа ответных значений, возвращенных данной процедурой, и реакцию программы на результат анализа.
- Использование функционирования программы в качестве ответа на отсутствие HASP. Например, отключить клавиатуру. После подсоединения ключа клавиатура включается.

HASP HL - решение для защиты ПО

- HASP HL – новое поколение аппаратно-программных средств класса Software Digital Rights Management (система управления электронными правами на ПО).

Возможности HASP HL

- Повысить уровень продаж и увеличить доходы от реализации ПО
- Защитить свою интеллектуальную собственность
- Защитить разработанное и распространяемое программное обеспечение
- Управлять лицензированием ПО
- Реализовать различные модели продаж защищенного ПО

Защита и лицензирование с помощью HASP HL

Построение защиты с помощью HASP HL

Методы защиты HASP HL

- HASP HL предоставляет разработчикам два метода защиты:
- **HASP HL Envelope** – быстрый и простой метод автоматической защиты уже готовых приложений.
- **HASP HL API** – набор функций, встраиваемых в приложение, для создания мощной и гибкой системы защиты, разработки собственной системы лицензирования и использования дополнительных методов и схем защиты.

Модель	Фото	Алгоритм	Уникальный ID	Память	Кол-во лицензий	Комментарии
Basic		AES	—	—	1	Наиболее простое и эффективное решение для защиты недорогого ПО, не требующее управления лицензированием и сохранения в памяти ключа параметров и настроек.
Pro		AES	✔	112 Байт	16	Идеальное решение для защиты ПО с лицензированием по используемым компонентам.
Max		AES	✔	4 КБ	112	Оптимальное решение для сложного ПО с лицензированием по функциональности и/или количественным показателям.
Net		AES	✔	4 КБ	112	Идеальное решение для защиты корпоративного ПО, предназначенного для использования многими пользователями в сети. Могут работать и как локальные. Позволяет управлять и ограничивать количество пользователей, одновременно использующих защищенное ПО, лицензировать компоненты, функциональность и другие количественные показатели.
Time		AES	✔	4 КБ	112	Идеальное решение для организации аренды, проката, лизинга, подписки на ПО, распространения пробных версий (trial). Аналогичен модели Max, но дополнительно имеет встроенные часы реального времени, используемые при лицензировании ПО (ограничения по дате и времени).
Master Key		Ключ содержит уникальные коды и идентификаторы, используемые системой HASP HL, которые присваиваются разработчику компанией Aladdin. Используется при построении защиты с помощью HASP HL.				