Тема 1.1 Многоуровневая модель качества программного обеспечения

Стандарты качества программного обеспечения.

Основным стандартом качества в области инженерии программного обеспечения в настоящее время является стандарт ISO/IEK 9126. Он определяет номенклатуру, атрибуты и метрики требований качества программного обеспечения. Относительно недавно этот стандарт стал одним из определяющих факторов при моделировании качества программного обеспечения и остается им до сих пор.

В дополнение к нему выпущен набор стандартов ISO/IEK 14598, регламентирующий способы оценки этих характеристик. В совокупности они образуют модель качества, известную под названием SQuaRE.

Общий подход заключается в том, чтобы сначала идентифицировать небольшой набор атрибутов качества самого высокого уровня абстракции и затем в направлении «сверху вниз» разбить эти атрибуты на наборы подчиненных атрибутов. Стандарт ISO/IEK 9126 является типичным примером такого подхода.

В рамках модели SQuaRE выделяются следующие 6 основных характеристик качества:

1. **Функциональность** (точность, согласованность, интероперабельность, безопасность, пригодность). Функциональные требования традиционно составляют основной предмет спецификации, моделирования, реализации и аттестации программного обеспечения. Они формулируются в виде утверждений в императивной модальности, описывающих поведение системы. Использование формальных методов позволяет довести уровень отклонения фактического поведения системы от требуемого практически до нуля. Это достигается путём выражения функциональных требований в виде предложений подходящих формальных исчислений, так что верификация сводится к строгому доказательству.
2. **Надёжность** (устойчивость, завершенность, восстанавливаемость). Показатели надёжности характеризуют поведение системы при выходе за пределы штатных значений параметров функционирования по причине сбоя в окружении либо в самой системе. При оценке

атрибутов надёжности применяются методы теории вероятностей и математической статистики. Требования к надёжности особенно важны при разработке критических систем обеспечения безопасности жизнедеятельности. Хотя использование формальных методов способствует снижению количества внутренних ошибок (т.е. росту завершенности системы), обеспечение надёжности в целом требует специальных подходов, учитывающих специфику различных типов систем.

3. **Удобство** (эффективность освоения, эргономичность, понимаемость). Соответствие системы требованиям к удобству чрезвычайно трудно поддается оценке. Предлагаемые подходы включают замеры расхода нормативных единиц труда(нормо-часов), затрачиваемого пользователями на овладение системой, а также проведение и анализ экспертных оценок, в том числе с применением методов нечеткой логики(fuzzy logic). В контексте использования формальных методов наилучшим решением представляется изначальная ориентация на формализмы, способные максимально точно отразить структуру исходной предметной области. Например, при создании вычислительных систем критерием адекватности формализма с точки зрения будущего пользователя является поддержка абстрактного математического языка, не зависящего от концептуальных ограничений, накладываемых компьютерными технологиями.
4. **Эффективность** (по ресурсам и по времени). Атрибуты эффективности традиционно относятся к числу важнейших количественных показателей качества программных систем. Их значения подлежат фиксации в эксплуатационной документации программным и аппаратным изделием. Имеется высокоразвитый инструментарий для измерения этих значений. Разработаны также методики, позволяющие прогнозировать интегральные значения показателей эффективностей системы исходя из значений этих показателей для компонентов самой системы и её окружения. Выбору формальных методов обеспечения эффективности следует уделять особое внимание. При этом, следует иметь ввиду, что, хотя имеется тесная взаимосвязь между производительностью и ресурсоёмкостью, подходы к обеспечению каждого из этих требований в общем случае имеют различную природу.
5. **Сопровождаемость (**простота анализа, изменяемость, стабильность**).** Требования сопровождаемости направлены в первую очередь на минимизацию усилий по сопровождению и модернизации

системы, затрачиваемых эксплуатационным персоналом. Для их оценки используется различные методики прогнозирования затрат на выполнение типовых процедур сопровождения. Интегральная стоимость сопровождения долгоживущих систем может существенно превышать стоимость их разработки. Сопровождение существенно упрощается в случае, когда разработка велась с использованием формальных методов, поскольку имеется в определённом смысле, исчерпывающий комплект технологической документации и проверочных тестов.

6. **Переносимость (**адаптируемость, согласованность со стандартами и правилами, гибкость инсталляции, заменяемость**).** Переносимость систем характеризует степень свободы при выборе компонентов системного окружения, необходимых для её функционирования. Оценка переносимости затрудняется принципиальной незавершённостью, динамичностью списка возможных вариантов окружения, обусловлены быстрым прогрессом в сфере IT. Системы, разрабатываемые с использованием формальных методов, как правило, отличаются высоким уровнем переносимости. В частности, если такая система не поддерживает некоторую целевую технологическую платформу, создание «клона» реализация её абстрактной модели с использованием целевых средств программирования требует существенно меньших затрат, чем замена самой системы либо платформы.

Модель качества, создаваемая в рамках данного стандарта, определяется общими характеристиками продукта. Характеристики же, в свою очередь, могут быть уточнены, другими словами, иерархично разбиты на подхарактеристики качества. Так, например, характеристика сопровождаемости может быть представлена такими подхарактеристиками как простота анализа, изменяемость, стабильность, проверяемость.

И, наконец, нижний уровень иерархии представляют непосредственно атрибуты программного обеспечения, поддающиеся точному описанию и измерению. Требования качества в свою очередь могут быть представлены как ограничения на модель качества. Оценка качества продукта в таком случае происходит по следующей схеме. Вначале оцениваются атрибуты программного изделия. Для этого выбирается метрика и градуируется шкала оценки в зависимости от возможных степеней соответствия атрибута накладываемыми ограничениями. Для каждой отдельной оценки атрибута

градация обычно выбирается заново и зависит от требований качества, накладываемых на него. Набор «измеренных» атрибутов представляют собой критерии для оценки подхарактеристики и характеристик, и как результат качества продукта.

Ка́чество програ́ммного обеспечения — характеристика программного обеспечения (ПО) как степени его соответствия требованиям. При этом требования могут трактоваться довольно широко, что порождает целый ряд независимых определений понятия. Чаще всего используется определение ISO 9001, согласно которому качество есть «степень соответствия присущих характеристик требованиям».

Качество исходного кода

Качество кода может определяться различными критериями. Некоторые из них имеют значение только с точки зрения человека. Например, то, как отформатирован текст программы, совершенно не важно для компьютера, но может иметь серьёзное значение для последующего сопровождения. Многие из имеющихся стандартов оформления кода, определяющих специфичные для используемого языка соглашения и задающие ряд правил, улучшающих читаемость кода, имеют своей целью облегчить будущее сопровождение ПО, включающее отладку и обновление. Существуют и другие критерии, определяющие, «хорошо» ли написан код, например, такие, как структурированность — степень логического разбиения кода на ряд управляемых блоков.

- Читаемость кода
- Лёгкость поддержки, тестирования, отладки, исправления ошибок, изменения и портируемости
- Низкая сложность кода
- Низкое использование ресурсов: памяти и процессорного времени
- Корректная обработка исключительных ситуаций
- Малое число предупреждений при компиляции и линковке
- Методы улучшения качества кода: рефакторинг.

Факторы качества

Фактор качества ПО — это нефункциональное требование к программе, которое обычно не описывается в договоре с заказчиком, но, тем не менее, является желательным требованием, повышающим качество программы.

Некоторые из факторов качества:

- понятность

Назначение ПО должно быть понятным, из самой программы и документации.

- полнота

Все необходимые части программы должны быть представлены и полностью реализованы.

- краткость

Отсутствие лишней, дублирующейся информации. Повторяющиеся части кода должны быть преобразованы в вызов общей процедуры. То же касается и документации.

- портируемость

Лёгкость в адаптации программы к другому окружению: другой архитектуре, платформе, операционной системе или её версии.

- согласованность

По всей программе и в документации должны использоваться одни и те же соглашения, форматы и обозначения.

- сопровождаемость

Насколько сложно изменить программу для удовлетворения новых требований. Это требование также указывает, что программа должна быть хорошо документирована, не слишком запутана, и иметь резерв роста по использованию ресурсов (память, процессор).

- тестируемость

Позволяет ли программа выполнить проверку приёмочных характеристик, поддерживается ли возможность измерения производительности.

- удобство использования

Простота и удобство использования программы. Это требование относится прежде всего к интерфейсу пользователя.

- надёжность

отсутствие отказов и сбоев в работе программ, а также простота исправления дефектов и ошибок:

- структурированность
- эффективность

Насколько рационально программа относится к ресурсам (память, процессор) при выполнении своих задач.

- безопасность

ISO 9126 (ГОСТ Р ИСО / МЭК 9126-93) — «Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению». ISO 9126 это международный стандарт, определяющий оценочные характеристики качества программного обеспечения (далее ПО). Российский аналог стандарта ГОСТ 28195. Стандарт разделяется на 4 части, описывающие следующие вопросы: модель качества; внешние метрики качества; внутренние метрики качества; метрики качества в использовании

Модель качества, установленная в первой части стандарта ISO 9126-1, классифицирует качество ПО в 6-ти структурных наборах характеристик, которые в свою очередь детализированы под-характеристиками(субхарактеристиками), такими как:

Функциональность — Набор атрибутов характеризующий, соответствие функциональных возможностей ПО набору требуемой пользователем функциональности. Детализируется следующими подхарактеристиками (субхарактеристиками):

- Пригодностью для применения
- Корректностью (правильностью, точностью)
- Способностью к взаимодействию (в частности сетевому)
- Защищенностью
- Надёжность — Набор атрибутов, относящихся к способности ПО сохранять свой уровень качества функционирования в установленных

условиях за определенный период времени. Детализируется следующими подхарактеристиками (субхарактеристиками):

- Уровнем завершенности (отсутствия ошибок)
- Устойчивостью к дефектам
- Восстанавливаемостью
- Доступностью
- Готовностью
- Практичность (применимость) — Набор атрибутов, относящихся к объему работ, требуемых для исполнения и индивидуальной оценки такого исполнения определенным или предполагаемым кругом пользователей. Детализируется следующими подхарактеристиками (субхарактеристиками):
- Понятностью
- Простотой использования
- Изучаемостью
- Привлекательностью
- Эффективность — Набор атрибутов, относящихся к соотношению между уровнем качества функционирования ПО и объемом используемых ресурсов при установленных условиях. Детализируется следующими подхарактеристиками (субхарактеристиками):
- Временной эффективностью
- Используемостью ресурсов
- Сопровождаемость — Набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений (модификаций). Детализируется следующими подхарактеристиками (субхарактеристиками):
- Удобством для анализа;
- Изменяемостью
- Стабильностью
- Тестируемостью
- Мобильность — Набор атрибутов, относящихся к способности ПО быть перенесенным из одного окружения в другое. Детализируется следующими под характеристиками (субхарактеристиками):
- Адаптируемостью
- Простотой установки (инсталляции)
- Сосуществованием (соответствием)
- Замещаемостью

Подхарактеристика Соответствие не приведена в вышеописанном списке, но она принадлежит всем характеристикам. Эта характеристика должна отражать отсутствие противоречий с иными стандартами или характеристиками. Например соответствие надежности и практичности.

Каждая качественная подхарактеристика (субхарактеристика) (например адаптируемость) в дальнейшем разделяется на атрибуты. Атрибут — это сущность, которая может быть проверена или измерена в программном продукте. Атрибуты не определены в стандарте из-за их разнообразия в различных программных продуктах.

В стандарте выделена модель характеристик качества в использовании. Основными характеристиками качества программных средств (далее ПС) в использовании рекомендуются:

Системная эффективность — 'Применения программного продукта по назначению'

Продуктивность — 'Производительность при решении основных задач ПС, достигаемая при реально ограниченных ресурсах в конкретной внешней среде применения'

Безопасность — 'Надежность функционирования комплекса программ и возможный риск от его применения для людей, бизнеса и внешней среды'

Удовлетворение требований и затрат пользователей в соответствии с целями применения ПС

Вторая и третья части стандарта ISO 9126-2,3 посвящены формализации соответственно внешних и внутренних метрик характеристик качества сложных ПС. В ней изложены содержание и общие рекомендации по использованию соответствующих метрик и взаимосвязей между типами метрик.

Четвертая часть стандарта ISO 9126-4 предназначена для покупателей, поставщиков, разработчиков, сопровождающих, пользователей и менеджеров качества ПС. В ней повторена концепция трех типов метрик, а также аннотированы рекомендуемые виды измерений характеристик ПС.

Стандартизация в управлении качеством. **Роль стандартизации в управлении качеством**

Стандартизация - один из основных принципов управления качеством

По ИСО:

Стандартизация – процесс установления и применения правил с целью упорядочения в данной области на пользу и при участии всех заинтересованных сторон, в частности – для достижения всеобщей максимальной экономии с соблюдением функциональных условий и требований безопасности

Цели стандартизации:

- Повышение уровня безопасности жизни и здоровья, имущества, и содействия соблюдению технических регламентов
- Обеспечение научно-технического прогресса
- Повышение конкурентоспособности продукции, работ и услуг
- Сопоставимость результатов исследований и измерений, технических и экономико-статических данных
- Повышение уровня безопасности объектов с учетом риска возникновения чрезвычайной ситуации
- Рациональное использование ресурсов
- Техническая и информационная совместимость
- Взаимозаменяемость продукции

Принципы международной стандартизации:

Комплексность стандартизации – заключается в систематизации и увязке комплекса факторов, обеспечивающих требуемый уровень качества продукции, в процессе установления и применения НД

Опережающее развитие стандартизации – развитие стандартизации с учетом изменения во времени показателей качества объектов стандартизации

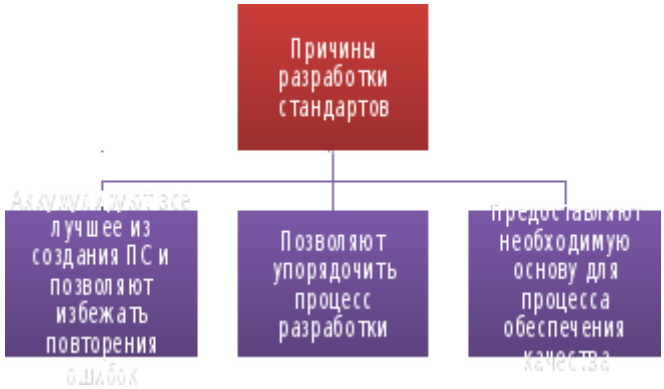
Классификация – выделение у объекта стандартизации классификационных признаков и их ранжирование по значимости для определения объекта

Международная организация по стандартизации (ИСО) – создана решением Комитета по координации стандартов ООН в 1946 г., официальную деятельность начала с февраля 1947 г.

Основная цель – содействие стандартизации в мировом масштабе

Стандарт – нормативный документ, разработанный на основе консенсуса, утвержденный признанным органом, направленный на достижение оптимальной степени упорядочения в определенной области

В стандарте устанавливаются для всеобщего использования общие принципы, правила, характеристики рекомендательного характера, касающиеся различных видов деятельности или их результатов.



Вид стандарта – характеристика, определяющая его содержанием в зависимости от объекта стандартизации

Стандарт на продукцию – устанавливающий требования, которым должна удовлетворять продукция или группа продукции, чтобы обеспечить ее соответствие своему назначению.

Стандарт на процессы – требования к выполнению работ на этапах жизненного цикла продукции или услуги (разработка, изготовление, хранение, транспортирование, эксплуатация, утилизация) для обеспечения их технического единства и оптимальности.

Стандарт на методы контроля должны обеспечивать всестороннюю проверку всех обязательных требований к качеству продукции (услуги).

Для каждого метода устанавливаются:

- Средства испытаний и вспомогательные устройства
- Порядок подготовки к проведению испытаний
- Порядок проведения испытаний
- Правила обработки результатов испытаний
- Правила оформления результатов испытаний
- Допустимая погрешность испытаний

Стандарт на услугу – требования к группе однородных услуг (услуги транспортные) или конкретной услуге (грузовые перевозки), чтобы обеспечить соответствие услуги ее назначению

Стандарт на термины и определения – устанавливающий термины, к которым даны определения, содержащие необходимые и достаточные признаки понятия.

Выполняют одну из главных задач стандартизации – обеспечение взаимопонимания между всеми сторонами, заинтересованными в объекте стандартизации.

Сертификация - процедура подтверждения соответствия, посредством которой независимая от изготовителя и потребителя организация удостоверяет в письменной форме, что продукция соответствует установленным требованиям.

Цели:

-содействие в компетентном выборе ПО

-функциональная стандартизация ПО

-улучшение качества ПО

-защита от недобросовестности производителей ПО

-подтверждение показателей качества ПО, заявленных его изготовителями

Объекты сертификации

Система качества- проверяются технологии создания ПО и АС по требованиям качества и ИБ в рамках реализованных на предприятиях – разработчиках систем качества.

Разрабатываемое и созданное ПО - устанавливаются факты:

-отсутствия закладных элементов

-соответствия реальных и декларированных функциональных возможностей

-соответствия требованиям стандартов для обеспечения взаимодействия, совершенствования и развития АС

Разрабатываемые и созданные АС:

-проверяется адекватность функционирования АС

-оцениваются возможность к взаимодействию, совершенствованию и развитию АС

-проверяется степень обеспечения надежного и своевременного представления полной, достоверной и конфиденциальной информации

Правовое обеспечение сертификации

Федеральный закон РФ от 27 декабря 2002 г. №184-ФЗ «О техническом регулировании»

- Госстандарт РФ
- Министерство экономического развития и торговли РФ
- Соглашение о технических барьерах в торговле ВТО
- Директива ЕС «О процедуре представления информации в области технических регламентов и стандартов»

Техническое регулирование:

- техническое законодательство
- стандартизация
- оценка соответствия-сертификация ->

->Федеральное агенство по техническому регулированию и метрологии-Росстандарт

- Гос. Органы
- Испытательные лаборатории
- Изготовители и потребители
- Организации, создающие систему сертификации
- Центральные органы систем сертификации

Содержание процедуры сертификации

В руководстве ИСО определены *8 схем сертификации* третьей стороной:

1. Испытания образца продукции
2. Испытания образца продукции с последующим контролем на основе надзора за заводскими образцами, закупаемыми на открытом рынке
3. Испытания образца продукции с последующим контролем на основе надзора за заводскими образцами
4. Испытания образца продукции с последующим контролем на основе надзора за образцами, приобретенными на открытом рынке и полученными с завода
5. Испытания образца продукции и оценка заводского управления качеством с последующим контролем на основе надзора за заводским управлением качества и испытаний образцов, полученных с завода и открытого рынка
6. Только оценка заводского управления качеством
7. Проверка партий изделий
8. 100%-ый контроль

Подача заявки на сертификацию -> Решение (1 месяц): условия, схема, перечень документов, исп. Лаборатории

Отбор, идентификации образцов и их испытания-> Испытания проводятся только на серийных образцах, идентичных продукции

Оценка производства->Метод оценки производства указывается в сертификате

Выдача сертификатов соответствия->По результатам оценки документации составляется заключение эксперта

Применение знака соответствия->Право маркировки сертифицированной продукции

Инспекционный контроль за сертифицированной продукцией->Результаты оформляются актом (хранится в органе по сертификации)

Корректирующие мероприятия->назначаются в случаях нарушения соответствия продукции требованиям и правил применения знакам соответствия

Сертификация ПО:

- Только рабочие версии (без ограничений функциональности)
- Сертификат-только на конкретную версию (с указанием номера и даты выпуска версии)

- *Методика оценки показателей:*

- показатели адекватности функционирования

- показатели надежности и своевременности представления информации

- показатели полноты, безошибочности, актуальности, защищенности от несанкционированного доступа и компьютерной вирусной инфекции, конфиденциальности информации:

Показатели адекватности функционирования:

- способность системы качества предприятия обеспечить условия для достижения требуемого качества АС

- соответствие реальных функциональных возможностей ПО, декларируемым в программной документации

- отсутствие опасных закладных элементов в ПО

Модель бесприоритетного обслуживания

При бесприоритетном обслуживании нет приоритетов, при поступлении нового запроса не происходит прерывания обработки предыдущего запроса. В этом случае запросы обслуживания в порядке их поступления в информационную систему.

Факторы, влияющие на качество пс

Характеристики качества ПС ⎧Функциональная пригодность

Конструктивные характеристики качества ПС

- Корректность
- Способность к взаимодействию
- Защищенность
- Надежность
- Эффективность
- Практичность
- Сопровождаемость
- Мобильность

Метрики характеристик качества ПС

- Внутренние метрики
- Внешние метрики
- Метрики в использовании

Негативные факторы, влияющие на характеристики качества ПС

- Внутренние дефекты:
  - Проектирования
  - Алгоритмизации
  - Программирования
  - Защиты
- Внешние воздействия:
  - Ошибки персонала
  - Искажения в каналах
  - Отказы аппаратуры

Ресурсы, ограничивающие характеристики качества ПС

- Экономические
- Временные
- Кадры специалистов
- Вычислительные

Функциональные характеристики (функциональность) – определяющее значение, свойства и задачи, решаемые комплексом программ для основных пользователей.

Конструктивные характеристики – номенклатура которых может быть использована для определения характеристик качества, поддерживающих реализацию функциональных требований к качеству объектов жизненного цикла ПС.

Сравнение функционального качества программ -> В пределах, ограниченных классов ПС, выполняющих подобные функции:

- Административные
- Банковские
- Медицинские
- Авиационные

Функциональная пригодность непосредственно определяет основное назначение и функции ПС для пользователя (ISO 9126)

Функциональная пригодность обозначается как основная цель и главная характеристика для всего множества типов программных средств.

Конструктивные характеристики (играют подчиненную роль) - должны обеспечивать и поддерживать высокое качество реализации функций ПС и его применения по основному назначению:

- Корректность
- Защищенность
- Мобильность
- Сопровождаемость
- Ресурсная эффективность
- Практичность
- Надежность

Исходная номенклатура этой группы характеристик, субхарактеристик и их атрибутов практически инвариантна к функциям ПС.

Метрики характеристик качества

Процессы формирования качества ПС

Внутреннее качество – проявляется в процессе разработки и других промежуточных этапах жизнедеятельности цикла ПС

Внешнее качество – задаётся требованиями заказчика в спецификациях и отражается в характеристиках конечного продукта

Качество в использовании – определяется результативностью достижения потребностей пользователей с учетом затрат

Измерение качества:

- Внутренне – статическим анализом мер программного кода
- Внешне – наблюдение и измерение показателей кода при его исполнении

Подходящие внутренние атрибуты качества ПС -> Требуемое внешнее поведение -> Достижение качества в использовании

Любое ПС может быть частью большой информационной системы ПС.

Интерфейсы:

- Аппаратных средств
- Персонала операторов
- Рабочих потоков данных

ПС оценивается уровнями отобранных внешних метрик.

Внешнее качество – степень, в которой продукт удовлетворяет установленные и зафиксированные потребности в среде эксплуатации определенными пользователями, **для достижения заданных целей с необходимой результативностью, производительностью и качеством.**

Разработка ПС -> Промежуточные продукты -> Внутренние метрики

Цель применения внутренних метрик – обеспечивать получение требуемого внешнего качества.

Внутренние метрики:

- Отражают определенные **функциональные и конструктивные свойства программ** и могут быть выведены из моделируемого поведения
- Позволяют измерять **внутренние атрибуты или формировать признаки внешних атрибутов** путем анализа статических свойств промежуточных или поставляемых программных компонентов
- Используют **свойства, категории, числа или характеристики элементов из состава ПС**, которые, например, имеются в процедурах исходного программного текста, в графе потока управления, потоке данных

1. Общие требования к функциям ПС
2. Выделение характеристик и субхарактеристик

(Полный набор показателей качества конкретного комплекса программ)

3. Определение внешних метрик, их мер и диапазонов значений (ПС удовлетворяет потребностям заказчика и пользователей)
4. Определение и спецификация внутренних метрик, атрибуты качества (удовлетворение требуемых внешних характеристик качества в конечном продукте и обеспечение их промежуточных продуктах в процессе разработки)

Метрики качества в использовании отражают, в какой степени продукт удовлетворяет потребности пользователей **в достижении заданных целей с результативностью, продуктивностью и удовлетворенностью** в заданном контексте применения.

- **Результативность** подразумевает **точность и полноту достижения определенных целей** пользователями при применении ПС
- **Продуктивность** соответствует **соотношению израсходованных ресурсов и результатов** при эксплуатации ПС
- **Удовлетворенность – технологическое и психологическое отношению к качеству процессов и результатов** использования программного продукта

Качество в использовании – **объединенных эффект функциональных и конструктивных характеристик качества ПС** для пользователя

Зависит от типа задач их пользователя:

- Конечный оперативный пользователь ПС по основному назначению -> Характеристики функциональных возможностей, надежности, практичности и эффективности
- Персонал сопровождения ПС -> Преимущественно сопровождаемость
- Персонал, выполняющий перенос ПС на иные платформы, а также инсталляцию и адаптацию -> Мобильность

Особенности измерения и оценивания характеристик качества

Оценка качества:

- Эконометрика
- Квалиметрия
- Программометрика

Оценка качества включает в себя: статистический анализ данных, экспертное оценивание.

Показатели качества – выделение измеримых свойств продукции, важных для потребителя.

Использование определенных мер и шкал:

- Выбор характеристик качества
- Сравнение с требованиями
- Сопоставление значения

Меры, используемые для сравнения:

1. Должны быть утверждены
2. Должны иметь точность, достаточную для выполнения надежных сравнений
3. Измерения должны быть объективны и воспроизводимы
4. Наличие системы измерений и методов оценки

Система измерений характеристик программного обеспечения – совокупность измеряемых характеристик, единиц измерения, измеряемых шкал и связей, установленных между ними

- Иерархическая
- Одноранговая

Измерительная шкала устанавливает границы (диапазон) и точность, которая необходима для измерений характеристик свойств в условленных единицах.

1. Результаты измерений
2. Сходство и различие в свойствах ПО
3. Оценка и классификация

Для обеспечивания объективности измерений необходимо:

Документированная и согласованная процедура для присвоения числового значения, свойства или категории каждому атрибуту программного продукта

Использование **наблюдений** или одобренных **вопросников** с применением **номинальной, интервальной** или **порядковой** шкалы.

Характеристики, субхарактеристики и атрибуты качества ПС

- **Категорийныет** Описательные (набор свойств и общие характеристики) – его функции, категории ответственности, защищенности и важности, которые могут быть представлены **номинальной шкалой**
- **Количественные** Представляемые множеством упорядоченных числовых точек, отражающих непрерывные закономерности и описываемые **интервальной** или **относительной шкалой**.
- **Качественные** Характеризуются **порядковой** или **точечной шкалой** набора категорий (есть – нет, хорошо – плохо), устанавливаются, выбираются и оцениваются в значительной степени субъективно и экспертно.

Типы шкал

Использование модулей	0 = Нет 1 = Да
Структура программы	1 = Линейная программа 2 = Программа с указателями 3 = Программа с модулями
Использование указателей	1 = Не используются

		2 = Используются умеренно 3 = Используются интенсивно 4 = Очень интенсивно	
Объем программы		1 = до 500кБ 2 = 501 – 1000 кБ 3 = более 1000 кБ	
	ПС 1	ПС 2	ПС 3
Количество операторов	2000	4000	6000
Объем (кБ)	300	900	1800

Если значимы интервалы – интервальная шкала

Если интересует «во сколько раз» - шкала отношений.

Статистическая шкала	Эмпирическая значимость
Номинальная	Нет
Порядковая	Порядок чисел
Интервальная	Разность чисел
Шкала отношений	Отношения чисел

Группа категорийных показателей:

Характеризуются наибольшим разнообразием значений – свойств программ и наборов данных и охватывают весь спектр классов, значений, функций современных ПС.

Свойства можно сравнивать только в пределах однотипных ПС и трудно упорядочивать по принципу предпочтительности.

Функциональная пригодность – самая важная и доминирующая характеристика любых ПС.

Выбор функциональной пригодности ПС, **подробное и максимально корректное описание её свойств** являются **основными исходными**

данными для установления требуемых значений всех остальных стандартизированных показателей качества.

Группа количественных показателей:

Достаточно достоверно и объективно измеряемые численные характеристики ПС

Их субхарактеристики могут быть описаны упорядоченными шкалами объективно измеряемых значений, требуемые численные величины которых могут быть установлены и выбраны заказчиками или пользователями ПС.

Надежность – может отражаться **временем наработки на отказ, средним временем восстановления, а также коэффициентом готовности** – вероятностью застать ПС **в работоспособном состоянии при нормальной эксплуатации.**

Атрибуты временной эффективности тесно связаны между собой и также значительно влияют на функциональную пригодность ПС.

Группа качественных показателей

Трудно полностью описать измеряемыми количественными значениями, их некоторые субхарактеристики имеют описательный, качественный вид.

В зависимости от функционального назначения ПС по согласованию с заказчиком можно **определять экспертно степень необходимости этих свойств и бальные значения** уровня реализации их атрибутов в жизненном цикле конкретного ПС.

Мобильность программ - возможность перехода на иные операционные и аппаратные платформы.

Сопровождаемость – возможность полной замены программ на вновь разработанные версии или осуществляться, как непрерывная поддержка.

Практичность – можно отразить трудоёмкостью и длительностью, которые необходимы для изучения и полного освоения функций и технологий применения соответствующего ПС.

Тема 1.2 Объекты уязвимости, дестабилизирующие факторы и угрозы надежности

Классификация уязвимостей

В 1996 году компания ISS (Internet Security Systems) разработала следующую классификацию уязвимостей:

- Уязвимости, реализованные или созданные продавцом (разработчиком) программного или аппаратного обеспечения. Включают: ошибки, не установленные обновления (SP, patch и hotfix) операционной системы, уязвимые сервисы и незащищенные конфигурации по умолчанию.
- Уязвимости, добавленные администратором в процессе управления компонентами системы. Представляют собой доступные, но неправильно используемые настройки и параметры информационной системы, не отвечающие политике безопасности (например, требования к минимальной длине пароля и несанкционированные изменения в конфигурации системы).
- уязвимости, привнесенные пользователем в процессе эксплуатации системы. Включают уклонения от предписаний принятой политики безопасности, например, отказ запускать ПО для сканирования вирусов или использование модемов для выхода в сеть Internet в обход межсетевых экранов и другие, более враждебные действия.

В более общем виде уязвимости могут быть классифицированы по этапам жизненного цикла ИС:

- Уязвимости проектирования (проектирование)
- Уязвимости реализации (реализация)
- Уязвимости конфигурации (эксплуатация)

Уязвимости проектирования наиболее серьезны — они обнаруживаются и устраняются с большим трудом. В этом случае уязвимость свойственна проекту или алгоритму и, следовательно, даже совершенная его реализация (что в принципе невозможно) не избавит от заложенной в нем слабости. Например, уязвимость стека протоколов TCP/IP. Недооценка требований по безопасности при создании этого стека протоколов привела к тому, что не проходит месяца, чтобы не было объявлено о новой уязвимости в протоколах стека TCP/IP. И раз и навсегда устранить эти недостатки уже невозможно — существуют только временные или неполные меры. Однако бывают и ис-

ключения. Например, внесение в проект корпоративной сети множества модемов, облегчающих работу персонала, но существенно усложняющих работу службы безопасности. Это приводит к появлению потенциальных путей обходов межсетевого экрана, обеспечивающего защиту внутренних ресурсов от несанкционированного использования. И обнаружить, и устранить эту уязвимость достаточно легко.

Уязвимости реализации состоят в появлении ошибки на этапе реализации в программном или аппаратном обеспечении корректного с точки зрения безопасности проекта или алгоритма. Яркий пример такой уязвимости — *"переполнение буфера"* во многих реализациях программ, например, sendmail или Internet Explorer. Обнаруживаются и устраняются подобного рода уязвимости относительно легко. Если нет исходного кода программного обеспечения, в котором обнаружена уязвимость, то ее устранение заключается или в обновлении версии уязвимого ПО или в полной его замене или отказе от него.

Уязвимости конфигурации состоят в ошибках при конфигурации программного или аппаратного обеспечения. Этот вид наряду с уязвимостями реализации является самой распространенной категорией уязвимостей. Существует множество примеров таких уязвимостей. К их числу можно отнести, например, доступный, но не используемый на узле сервис Telnet, разрешение "слабых" паролей или паролей длиной менее 6 символов, учетные записи и пароли, остановленные по умолчанию (например, SYSADM или DBSNMP в СУБД Oracle), и т. д. Локализовать и исправить такие уязвимости проще всего. Основная проблема — определить, является ли конфигурация уязвимой.

Наиболее распространенные уязвимости

По статистике, опубликованной в 1998 году институтом SANS (System Administrator and Network Security), пятерка наиболее распространенных уязвимостей выглядела следующим образом:

1. Выслеживание информации, особенно паролей и иной конфиденциальной информации.
2. Переполнение буфера, приводящее к удаленному выполнению произвольных команд.
3. Уязвимости системы защиты узлов, например, уязвимости сценариев CGI или ошибки в sendmail.

4. Подверженность атакам типа *"отказ в обслуживании"*.
5. Допустимость загрузки враждебного кода, к которому можно отнести программы типа *"троянский конь",* вирусы, апплеты Java, элементы управления ActiveX.

Можно заметить, что в первую пятерку вошли все три категории уязвимостей. Выслеживание паролей возможно благодаря отсутствию механизмов шифрования в стандартных протоколах Internet. Переполнение буфера, уязвимости защиты узлов и подверженность атакам типа "отказ в обслуживании" могут быть отнесены к разряду уязвимостей реализации и конфигурации. Ну и, наконец, возможность загрузки враждебного кода может быть причислена к разряду уязвимостей конфигурации.

В 2001 году пятерка наиболее распространенных уязвимостей по данным SANS обновилась:

1. Слабости BIND (службы доменных имен в Internet).
2. Уязвимые CGI-сценарии и расширения приложений, установленные на Web-сервере.
3. Уязвимости RPC.
4. Уязвимости Remote Data Services (RDS) в MS IIS.
5. Переполнение буфера в почтовой программе sendmail.

Эта пятерка частично совпадает с исследованиями компании ISS:

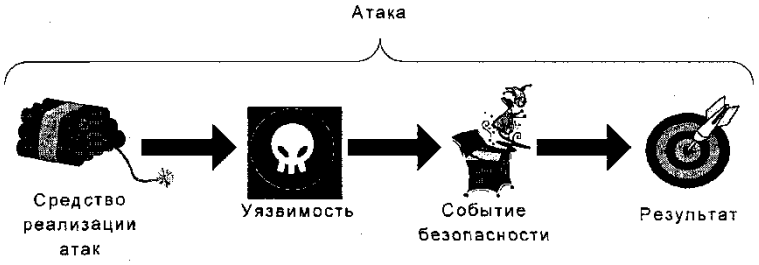
1. Подверженность атакам типа "отказ в обслуживании" (в том числе и распределенным атакам этого типа).
2. "Слабые" учетные записи (для серверов, маршрутизаторов и т. д.).
3. Уязвимости ПО MS IIS.
4. Уязвимости СУБД (неправильные права доступа к расширенным хранимым процедурам, пароли, заданные по умолчанию и т. д.).
5. Приложения eCommerce (Netscape FastTrack, MS FrontPage и др.).

Атаки

До сих пор у профессионалов в области информационной безопасности нет точного определения термина "атака". *Атаку* на информационную систему можно понимать как действие или последовательность связанных между собой действий нарушителя, которые приводят к реализации угрозы путем использования уязвимостей этой информационной системы. Таким образом, атака отличается от события безопасности тем, что в случае атаки

злоумышленник пытается достичь некоторого результата, противоречащего политике безопасности. Например, доступ пользователя к файлу или вход в систему — это событие безопасности. Однако, если этот доступ или вход осуществляется в нарушение прав доступа, то это уже атака.

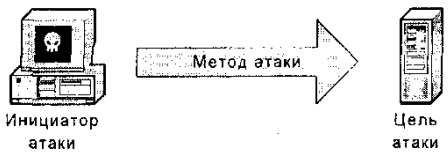
Если построить неформальную модель атаки, которая расширяет описанную выше для события безопасности, то получится модель, состоящая из 4-х элементов.



Для того чтобы реализовать атаку, *злоумышленник* моделирует некоторое событие безопасности, которое приводит к искомому результату при помощи некоего средства, использующего уязвимости системы. Первые два элемента данной модели применяются для реализации события безопасности, т. е. некоторого действия по отношению к адресату для достижения результата, приводящего к нарушению политики безопасности.

Неформальная модель атаки

Атаку можно представить в виде 3-х элементов: инициатор, метод, цель.



В частном случае инициатор атаки (злоумышленник) и *цель атаки* совпадают. В этом случае злоумышленник уже получил или имеет в рамках своих полномочий доступ к узлу (или группе узлов), к ресурсам которого (которых) он намерен несанкционированно обращаться. Целью атаки, также как и инициатором атаки, может выступать одиночный узел или группа узлов (например, подсеть).

Логично предположить, что устранение одного из этих элементов позволит полностью защититься от атаки. Удалить цель атаки на практике зачастую

невозможно из-за особенностей технологии обработки информации. Хотя это решение было бы идеальным. Раз нет цели для атаки, то неосуществима и сама атака. Одним из механизмов попытки удаления объекта атаки является сетевая трансляция адресов, блокирование доступа к определенным узлам корпоративной сети извне при помощи межсетевого экрана, физическое исключение доступа к защищаемой сети.

Поскольку нельзя удалить цель атаки, то необходимо пытаться устранить *инициатора* или *метод атаки.*

Современные средства защиты, как правило, сосредотачивают свое внимание на объекте атаки и немного на методе атаки.

Метод атаки зависит от нескольких параметров.

- *Тип инициатора атаки и цели атаки*. От этого зависит, какой метод атаки следует ожидать. Например, если объектом атаки является почтовый сервер MS Exchange, то вряд ли для нападения на него будет использоваться метод, применяемый для атаки на sendmail.

- *Результат воздействия*. От того, какого результата нарушитель ждет от атаки (отказ в обслуживании, компрометация и т. п.), зависит, какой метод атаки он применит. Например, если злоумышленник планирует получить несанкционированный доступ к файлу паролей вашего Web-сервера, то он будет скрывать свои несанкционированные действия и искать уязвимости в открытых сервисах Web-сервера (HTTP, FTP, IMAP и т. д.).

- *Механизм воздействия*.

- *Средство воздействия*.

Модель "традиционной" атаки

Традиционная модель атаки строится по принципу "один-к-одному" или "один-ко-многим", т. е. атака исходит из одной точки.

Рис. 2.4. Отношение "один-к-одному"

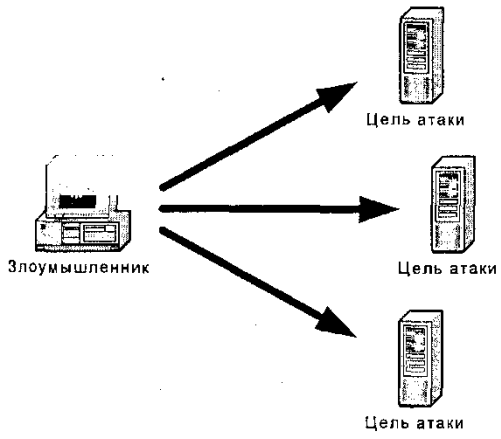


Рис. 2.5. Отношение "один-ко-многим"

Очень часто для сокрытия источника атаки или затруднения его нахождения используется метод промежуточных хостов. Злоумышленник реализует атаку не напрямую на выбранную цель, а через цепь узлов. Нередко эти узлы находятся в разных странах. В результате объекту атаки "кажется", что угроза исходит с промежуточного узла 2.

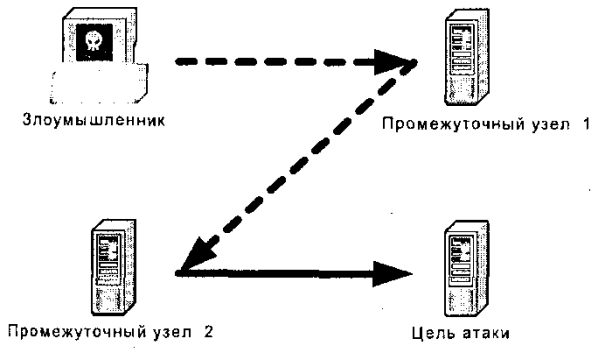


Рис. 2.6. Реализация атаки через промежуточные узлы

Модель распределенной атаки

В ноябре 1999 года впервые была организована конференция в области информационной безопасности на тему *распределенных атак*. Эти атаки позволяют одному или нескольким злоумышленникам проводить сотни и тысячи нападений, осуществляемых в один момент времени, на один или несколько узлов.

Традиционная модель атаки обычно оперирует одним узлом в качестве источника атаки. Именно этот принцип и заложен как основополагающий во многие средства защиты сетей.

Модель *распределенной* или *скоординированной атаки* опирается на иные принципы. В отличие от традиционной модели, распределенная модель основана на отношениях "много-к-одному" и "много-ко-многим".

Все распределенные атаки основаны на "классических" атаках типа "отказ в обслуживании", точнее — на их подмножестве – лавинных атаках. Смысл данных атак заключается в посылке большого количества пакетов на заданный узел сети (цель атаки), что может привести к выведению этого узла из строя, поскольку он "захлебнется" в потоке посылаемых пакетов и не сможет обрабатывать запросы авторизованных пользователей. Однако в том случае, когда полоса пропускания канала до цели атаки превышает пропускную способность атакующего, к "успеху" обычная атака "отказ в обслуживании" не приведет. В случае же распределенной атаки ситуация коренным образом меняется. Атака происходит уже не из одной точки Internet, а сразу из нескольких, что обуславливает резкое возрастание трафика и выход атакуемого узла из строя.

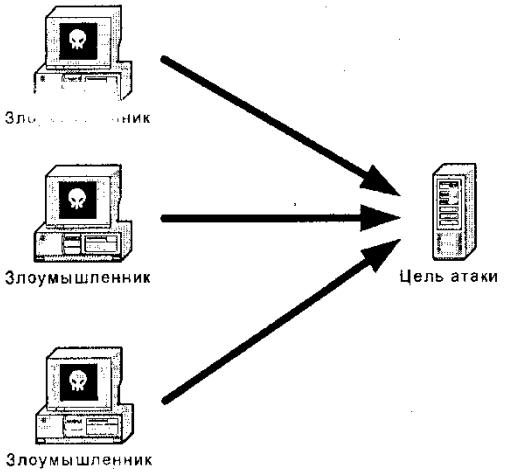


Рис. 2.7. Отношение "многие-к-одному"

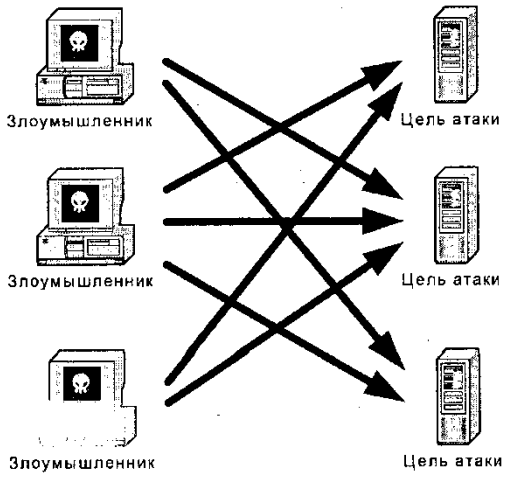


Рис. 2.8. Отношение "многие-ко-многим"

Реализуется распределенная атака в два этапа. Первый этап заключается в поиске в Internet узлов, которые можно было бы задействовать для реализации распределенной атаки. Чем больше будет найдено таких узлов, тем эффективнее будут последствия, "Изюминка" в том, что в Internet таких узлов миллионы. Проводимые регулярно исследования показывают, что многие компании не следят за безопасностью своих узлов, имеющих выход в Internet. После нахождения уязвимых узлов злоумышленник осуществляет установку на них компонентов, реализующих атаку.

Второй этап представляет собой посылку большого количества пакетов на атакуемый узел. Особенность этого этапа в том, что отправка пакетов осуществляется не с узла, за которым "сидит" злоумышленник, а с скомпрометированных им систем-посредников, на которых установлены специальные агенты, реализующие распределенную атаку. Существуют два типа таких агентов: "мастера" (master) и "демоны" (daemon). Злоумышленник управляет небольшим числом "мастеров", которые, в свою очередь, командуют "демонами".

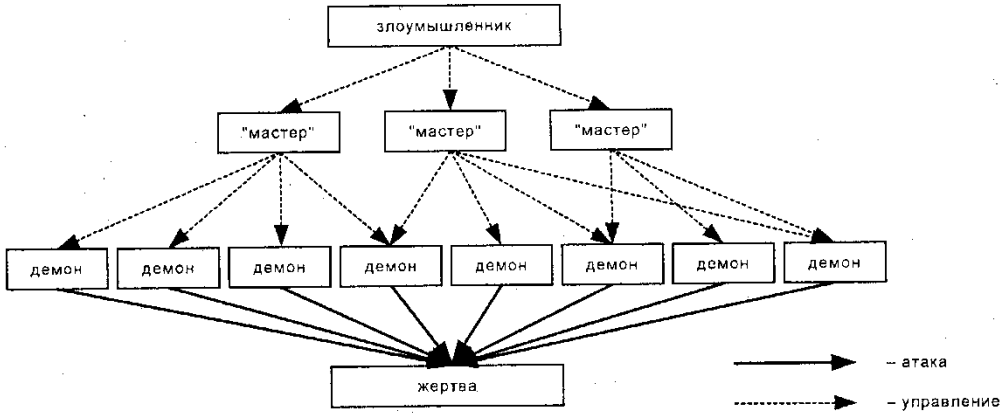


Рис. 2.9. Распределенная атака

Трудность блокирования подобных атак состоит в том, что обнаружение и блокирование одного или нескольких "мастеров" или "демонов" не приводит к окончанию атаки, поскольку каждый "демон" действует независимо от других и, получив соответствующие команды от "мастера", уже не нуждается в дальнейшем поддержании связи с ним. То есть "демон" работает автономно, что существенно затрудняет обнаружение и блокирование всех демонов, участвующих в распределенной атаке. Кроме того, при атаке возможна подмена адреса отправителя враждебных пакетов, что также отрицательно сказывается на эффективности контрмер.

В зафиксированных в 1998—1999 годах случаях распределенные атаки использовали сотни и даже тысячи демонов. Эти демоны устанавливаются путем использования на скомпрометированных узлах различных уязвимостей, в т. ч. и позволяющих получить права администратора на узле с установленным демоном. Как только демон установлен, он уведомляет об этом "мастера" (обычно трех или четырех). После получения определенных команд от злоумышленника "мастер" программирует "демона" на выполнение соответствующих действий против жертвы. Эти команды содержат адрес жертвы, тип атаки, время и продолжительность атаки.

Результат атаки

Результаты атаки можно классифицировать следующим образом:

- *Расширение прав доступа* — любое несанкционированное действие, приводящее к расширению прав доступа в сети или на конкретном узле.
- *Искажение информации* — любое несанкционированное изменение информации, хранящейся на узлах сети или при ее передаче по сети.
- *Раскрытие информации* — распространение информации среди лиц без соответствующих полномочий доступа.
- *Кража сервисов* — несанкционированное использование компьютера или сетевых сервисов без ухудшения качества обслуживания других пользователей.
- *Отказ в обслуживании* — умышленное снижение производительности или блокировка доступа к сети или компьютеру и его ресурсам.

Этапы реализации атак

Можно выделить следующие этапы реализации атаки: "сбор информации", "реализация атаки" и завершение атаки. Обычно, когда говорят об атаке, то подразумевают именно второй этап, забывая о первом и последнем. Сбор информации и завершение атаки, в свою очередь, также могут являться атакой и соответственно разбиваться на три этапа.

Сбор информации

Основной этап — это сбор информации об атакуемой системе или узле. Именно эффективность работы злоумышленника на данном этапе является залогом успешной атаки. В первую очередь выбирается цель нападения и собирается информация о ней (ОС, сервисы, конфигурация). Затем

идентифицируются наиболее уязвимые места атакуемой системы, воздействие на которые приводит к нужному результату. На первом этапе злоумышленник пытается выявить все каналы взаимодействия объекта атаки с другими узлами. Это позволит не только выбрать тип реализуемой атаки, но и источник ее реализации.

На этапе сбора информации могут применяться следующие методы.

Изучение окружения

Выполняя эту задачу, нападающий исследует области вокруг предполагаемой цели атаки. К таким областям относятся, например, узлы internet-провайдера "жертвы". На этом шаге злоумышленник может пытаться определить адреса "доверенных" систем (например, сети партнера), узлов, которые напрямую соединены с целью атаки.

Идентификация топологии сети

Можно назвать два метода определения топологии сети, используемых злоумышленниками: "изменение TTL" и "запись маршрута". Программы traceroute для Unix и tracert для Windows используют первый способ определения топологии сети. Протокол RIP может быть использован для получения информацию о таблице маршрутизации в сети и т. д.

Идентификация узлов

Идентификация узла, как правило, осуществляется путем посылки при помощи утилиты ping команды ECHO_REQUEST протокола ICMP. Ответное сообщение ECHO_REPLY говорит о том, что узел доступен. Существуют программы, которые автоматизируют и ускоряют процесс параллельной идентификации большого числа узлов, например, fping или nmap. Другим способом идентификации узлов сети является так называемая разведка DNS (Domain Name Service).

Идентификация сервисов или сканирование портов

Идентификация сервисов, как правило, выполняется путем обнаружения открытых портов — сканированием. Такие порты очень часто связаны с сервисами, основанными на протоколах TCP или UDP. Например, открытый 80-й порт подразумевает наличие Web-сервера, 25-й порт — почтового SMTP-сервера, 31337-й — сервера троянского коня BackOrifice и т. д.

Идентификация операционной системы

Основной механизм удаленного определения ОС — анализ TCP/IP-стека. В каждой ОС стек протоколов TCP/IP реализован по-своему, что позволяет при помощи специальных запросов и ответов на них определить, какая ОС установлена на удаленном узле.

Определение роли узла

Этот шаг выполняется на основе уже собранной информации об активных сервисах, именах узлов, топологии сети и т.п. Например, открытый 80-й порт может указывать на наличие Web-сервера, блокировка ICMP-пакета свидетельствует о потенциальном наличии межсетевого экрана, а имя узла proxy.domain.ru или fw.domain.ru говорит само за себя.

Определение уязвимостей узла

Последняя стадия на этапе сбора информации — поиск уязвимостей. На этом шаге злоумышленник при помощи различных автоматизированных средств или вручную определяет уязвимости, которые могут быть использованы для реализации атаки. В качестве таких автоматизированных средств могут быть использованы ShadowSecurityScanner, nmap, Retina и т. д.

Реализация атаки

Реализация атаки заключается в попытке доступа на атакуемый узел. При этом доступ может быть как непосредственный, т. е. проникновение на узел, так и опосредованный, как при реализации атаки типа "отказ в обслуживании". Реализация атак в случае непосредственного доступа также может быть разделена на два этапа: проникновение, установление контроля.

Проникновение

Проникновение подразумевает под собой преодоление средств защиты периметра (например, межсетевого экрана). Реализовываться это может различными путями. Например, использованием уязвимости сервиса компьютера, "смотрящего" наружу, или путем передачи враждебного содержания по электронной почте (макровирусы) или через апплеты Java. Такое содержание может задействовать так называемые "туннели" в межсетевом экране, через которые затем и проникает злоумышленник. К этому же шагу можно отнести

подбор пароля администратора или иного пользователя при помощи специализированной утилиты (например, LOphtCrack или Crack).

Установление контроля

После проникновения злоумышленник устанавливает контроль над атакуемым узлом. Это может быть осуществлено путем внедрения программы типа "троянский конь" (например, NetBus или BackOrifice). После установки контроля над нужным узлом и "заметания" следов злоумышленник может производить все необходимые несанкционированные действия дистанционно без ведома владельца атакованного компьютера. При этом установление контроля над узлом корпоративной сети должно сохраняться и после перезагрузки операционной системы. Это может быть реализовано путем замены одного из загрузочных файлов или вставкой ссылки на враждебный код в файлы автозагрузки или системный реестр. Известен случай, когда злоумышленник смог перепрограммировать EEPROM сетевой карты и даже после переустановки ОС он смог повторно реализовать несанкционированные действия.

Методы реализации атак

Если нарушитель имеет физический доступ к компьютеру, он сможет проникнуть в него или провести на него атаку. Методы могут быть различными — от использования специальных привилегий, которые имеет консоль или терминал, до процедуры снятия жесткого диска и его чтения/записи на другом компьютере. Это — *физическое вторжение.*

Системное вторжение — тип несанкционированной деятельности, предполагающий, что нарушитель уже имеет учетную запись в атакуемой системе как пользователь с некоторыми (обычно невысокими) привилегиями. Если в системе не установлены самые последние "заплаты", то у нарушителя есть хороший шанс попытаться реализовать известную атаку для получения дополнительных административных полномочий.

Несанкционированная деятельность *удаленного вторжения* подразумевает, что нарушитель пытается дистанционно проникнуть в систему через сеть. При этом нарушитель действует без каких-либо специальных привилегий. Существует несколько типов такой деятельности:

- локальное сетевое вторжение: злоумышленник атакует компьютер или группу компьютеров, находящихся в одном с ним сегменте.

- вторжение через сети открытого доступа - злоумышленник атакует компьютер или группу компьютеров, находящиеся в другом сегменте. При этом атака осуществляется через сети открытого доступа, как правило, через Internet.

- вторжение через Dial-up - злоумышленник атакует компьютер или группу компьютеров через модем.

Завершение атаки

Этапом завершения атаки является "заметание следов" со стороны злоумышленника. Обычно это реализуется путем удаления соответствующих записей из журналов регистрации узла и выполнением других действий, возвращающих атакованную систему в исходное состояние. При этом могут применяться следующие способы.

Подмена адреса источника атаки

Большинство злоумышленников организуют свои атаки с промежуточных серверов, которые они уже взломали, или с proxy-серверов. Таким образом, найти того, кто атакует ваш узел, будет очень трудно. Активное блокирование атак с помощью межсетевых экранов, фильтров на маршрутизаторах и других устройствах может привести к отрицательному результату, т. е. к тому, что вы заблокируете не злоумышленника, а вполне реальный адрес, которому возможно необходим доступ к вашим информационным ресурсам.

Создание фальшивых пакетов

Сканер nmap имеет возможность обманного сканирования, когда вместо реальных IP-адресов источника проставляются другие IP-адреса. Тем самым перед администратором системы обнаружения атак ставится непростая задача: обнаружить среди множества зафиксированных в журналах регистрации IP-адресов только один реальный, с которого действительно производилось сканирование.

Фрагментация атаки

Фрагментация — механизм разбиения IP-пакета на множество более мелких. При получении таких пакетов TCP/IP-устройство собирает эти пакеты и передает конечному приложению или повторно фрагментирует их и передает

дальше. Большинство современных систем обнаружения атак не имеет механизма дефрагментации IP-пакетов. Эти системы пропускают такого рода пакеты (возможно, выдавая на консоль администратора соответствующее сообщение об обнаружении фрагментированных пакетов).

Отказ от значений по умолчанию

Очень часто механизмы обнаружения атак исходят из предположения, что порт однозначно идентифицирует протокол или сервис. Например, порт 80 относится к протоколу HTTP, порт 25 — к протоколу SMTP, порт 23 — к протоколу Telnet, порт 31337 — к "троянцу" BackOrifice, и т. д. Злоумышленники пользуются этим и могут задействовать стандартные протоколы на нестандартных портах. Например, злоумышленник может заменить значение по умолчанию для BackOrifice (31337) на 31338.

Изменение стандартного сценария атаки

Многие механизмы обнаружения атак работают по принципу сопоставления с образцом (шаблоном). Использование баз данных известных атак позволяет обнаруживать атаки с высокой степенью достоверности. Однако от таких систем можно очень легко уклониться, немного изменяя шаблон. Пример — замена символа пробела в действиях, реализующих команду, на символ табуляции.

Замедление атаки

Из-за большого объема регистрируемых данных механизмы обнаружения атак неэффективно отслеживают атаки, растянутые во времени. Таким образом, трудно обнаруживать "распределенное по времени сканирование", при котором нарушители проверяют один порт/адрес каждые 5 минут или даже каждый час.

Чистка журналов регистрации

Достаточно распространенный способ, заключающийся в удалении всех записей журнала регистрации, фиксирующих произведенные несанкционированные действия. Это позволяет скрыть от администратора атакованной системы все следы подозрительной деятельности.

Скрытые файлы и данные

Скрытие файлов и данных очень часто используется для того, чтобы замаскировать несанкционированную деятельность злоумышленника. При этом могут быть применены совершенно различные методики, различающиеся по сложности реализации. Например, установка атрибута Hidden на файл, внедрение вредоносного кода в ядро операционной системы (для Unix-подобных систем) или присоединение такого кода к какому-либо исполняемому файлу или библиотеке.

Скрытие процессов

Аналогично предыдущему примеру, данный метод используется для скрытия деятельности злоумышленника на атакуемом узле. Для этого может быть также проведено изменение ядра операционной системы или специальных утилит, отвечающих за работу с процессами (например, утилиты ps в Unix). Самым простым методом сокрытия несанкционированной деятельности процесса является изменение его имени на "стандартное" или похожее на стандартное. Например, враждебный процесс может иметь имя internetd, iexplorer.exe.

Средства реализации атак

Средства (механизмы) реализации атак могут быть разделены в соответствии с нижеприведенным списком.

- Информационный обмен — средство получения определенной информации из других источников (например, FIDO) или от атакуемых людей (социальный инжиниринг).
- Команды пользователей — механизм использования уязвимостей путем ввода определенных команд в интерфейсе командной строки или процесса. В качестве примера можно назвать ввод команд ОС через соединение Telnet или FTP.
- Сценарий или программа — программы или последовательности команд, описанные в сценарии, вызываемые нарушителем для использования уязвимости. Иными словами, сценарий или программа — это оболочка для одной или нескольких команд пользователей. Например, к таковым относятся программы Crack или LOphtCrack для подбора паролей или "троянский конь" NetBus.
- Автономный агент — аналог предыдущего "оружия", но в отличие от него, цель атаки выбирается по определенному алгоритму независимо от пользователя. Для программ или сценариев объект атаки

нарушитель определяет "вручную". Примером автономного агента является "червь Морриса" (Morris worm) или компьютерный вирус (например, макровирус).

- Комплект утилит — пакет программ, сценариев или автономных агентов, используемых для осуществления атаки. Например, rootkit.
- Распределенные средства — программы, сценарии или автономные агенты, распределенные по нескольким узлам сети. Атака осуществляется в определенное время сразу из нескольких точек сети. Это наиболее сложный (как для осуществления атак, так и для защиты от него) тип средств реализации атаки. Характерные примеры: TFN2K, Stacheldraht.

Автоматизированные средства для реализации атак

В Internet и других сетях (например, FIDONET) доступны многочисленные источники, облегчающие злоумышленникам несанкционированный доступ к корпоративным ресурсам. Информация об уязвимостях постоянно публикуется в различных списках новостей, бюллетенях и т. п. Тысячи программ, реализующих атаки, использующих эти уязвимости, "лежат" в свободном доступе на множестве серверов в Internet, провоцируя рост "круши-ломай" настроений. Многие компьютерные атаки теперь не являются уделом лишь избранных. Любой начинающий пользователь Internet может загрузить к себе на компьютер готовый исполняемый файл и "напустить" его на своего "соседа". Несколько лет назад для запуска таких программ необходимо было знать Unix и уметь скомпилировать исходный текст программы-атаки. Сегодня ситуация изменилась коренным образом. Многие инструментальные средства для реализации атак обладают графическим интерфейсом и выполняются под Windows, что намного облегчило задачу начинающих "хакеров".

Тема 1.3 Оперативные методы повышения надежности: временная, информационная, программная избыточность

Влияние избыточности на повышение надежности программ

. можно преобразовать отказы в сбои и тем самым улучшить показатели надежности функционирования системы, Автоматизируя процесс и сокращая время восстановления. Главной задачей становится восстановление за время, не превышающее порогового значения между сбоем и отказом. то можно

добиваться высокой автоматизации программного восстановления,Так как в программах нет необходимости “ремонта” компонент с участием человека

Комплексы программ в процессе функционирования находятся под воздействием *шумов-возмущений* Для снижения их влияния на результаты следует применять.различных типов *фильтры*. каждый из которых способен селектировать некоторые виды искажений, Разнообразие видов искажений приводит к необходимости построения систем фильтров. устранять или уменьшать их вредные последствия, позволяющие обнаруживать искажения,

используется, обеспечивающих повышение надежности функционирования программ и защиту вычислительного процесса и информации программно-алгоритмическими методами,Для реализации фильтров программная, информационная и временная *избыточность*. . Любые аномалии при исполнении программ необходимо сводить до уровня сбоя путем быстрого восстановления. соответствующих отказу системы,Основная задача ввода избыточности состоит в исключении возможности аварийных последствий от возмущений

Временная избыточность . либо за счет сокращения времени решения функциональных задач, который выделяется либо за счет резерва, На это требуется в общем случае небольшой интервал времени. их диагностику и на реализацию операций восстановления, Временная избыточность используется на обнаружение искажений. Величина временной избыточности зависит от требований к надежности функционирования системы и находится в пределах от 5-10% производительности однопроцессорной ЭВМ до трех-четырех кратного дублирования производительности машины.состоит в использовании некоторой части производительности ЭВМ для контроля исполнения программ и восстановления вычислительного процесса

Информационная избыточность . позволяющих только обнаружить искажение, Для менее важных данных информационная избыточность используется в виде помехозащитных кодов. которые в наибольшей степени влияют на нормальное функционирование программ или требуют значительного времени восстановления, Избыточность используется для сохранения достоверности данных. обрабатываемых комплексом программ,состоит в дублировании накопленных исходных и промежуточных данных

Программная избыточность . использующих временную избыточность, Программная избыточность необходима также для реализации программ контроля и восстановления данных с использованием информационной избыточности и для функционирования всех средств защит. различающихся методами решения задачи или программной реализацией одного и того же метода, Она заключается в применении нескольких вариантов программ.используется для контроля и обеспечения достоверности наиболее важных решений по управлению и обработке информации

Эффективность применения избыточности для повышения надежности комплексов программ

. которая эффективно используется для оперативной защиты при отказовых ситуациях и непосредственно влияет на надежность функционирования программ, Наибольшее внимание уделяется временной избыточности. позволяющие оценить эффективность использования избыточности, Здесь представлены некоторые рекомендации. информационной и программной избыточности для обеспечения надежности программ,Выше рассмотрены принципы использования временной

Для реализации стратегий резервирования программ необходима *временная избыточность*. При этом временная избыточность используется в основном для оперативного контроля состояния данных и вычислительного процесса, а также для автоматического восстановления при возникновении отказовых ситуаций. Резерв времени для выполнения этих операций можно считать достаточным независимо от числа предыдущих отказов, времени на их устранение и наработки на отказ. Кроме того, суммарное время восстановления работоспособности обычно не ограничено.

Эффективность оперативного использования временной избыточности для повышения надежности функционирования программ определяется затратами на контрольно-восстановительные операции, изменением показателей надежности в зависимости от затрат и связью этих изменений с отлаженностью программ. В результате для оценки эффективности введения временной избыточности в программе необходимо:

- определить совокупные затраты на контроль, на работу при необнаруженном искажении и на восстановление, обеспечивающие заданную вероятность обнаружения отказовой ситуации при исполнении программ ;

- определить основные показатели надежности функционирования программ в зависимости от совокупных затрат на оперативный контроль и восстановление;
- оптимизировать суммарные затраты на отладку программ и оперативную защиту от искажений для обеспечения заданной надежности функционирования комплекса программ.

При решении последней задачи источниками искажений предполагаются только невыявленные ошибки в программах.

Тема 2.1. Антивирусные программы: классификация, сравнительный анализ

Компьютерные вирусы, их виды и антивирусные программы.

Широкое распространение компьютеров, а так же бурное развитие сети Интернет способствовало к появлению и распространению маленьких вредоносных программ вирусов. Вирусы порой делают работу на компьютере невозможной. Они, в зависимости от ситуации могут нанести значительный ущерб как информации, так и даже самому компьютеру. **Что такое вирус? Компьютерный вирус** это специальная программа, как правило, очень маленького размера, способная внедряться в тело другой программы, перехватывать управление этой программой, саморазмножаться (распространяться) и внедряться в другие программы с задачей дестабилизации работы компьютера и порче информации. Неспециалисты к компьютерным вирусам иногда причисляют и другие виды вредоносных программ, такие как троянские кони и программы-шпионы. Вирусы распространялись, внедряя себя в исполняемый код других программ или же заменяя собой другие программы. Какое-то время даже считалось, что, являясь программой, вирус может заразить только программу какое угодно изменение не программы является не заражением, а просто повреждением данных. Подразумевалось, что такие копии вируса не получат управления, будучи информацией, не используемой процессором в качестве инструкций. Так, например неформатированный текст не мог бы быть переносчиком вируса. Однако, позднее хакеры показали, что вирусным поведением может обладать не только исполняемый код, содержащий машинный код процессора. Были написаны вирусы на языке пакетных файлов. Потом появились макровирусы, внедряющиеся через макросы в документы таких программ, как Microsoft Word и Excel. Некоторое время спустя появились вирусы, использующие уязвимости в популярном программном обеспечении (например, Adobe Photoshop, Internet Explorer, Outlook), в общем случае

обрабатывающем обычные данные. Вирусы стали распространяться посредством внедрения в последовательности данных (например, картинки, тексты, и т. д.) специального кода. использующего уязвимости программного обеспечения. **История вредоносных программ** Время рождения первого вируса спорно. Известно лишь, что на машине Чарльза Бэббиджа, считающегося изобретателем первого компьютера, вирусов не было, а на UNIVAC 1108 и IBM 360/370 в середине 1970-х годов они уже были. Но, конечно, идея вирусов появилась намного раньше. Труды Джона фон Неймана по изучению самовоспроизводящихся математических автоматов можно считать началом. Труды стали известны общественности в 40-х годах. А в 50-х ученый предложил методы, демонстрирующие эти автоматы. В 60-х годах журнал “Scientific American” опубликовал статью посвященную, самовоспроизводящимся механическим структурам. Позже они были реализованы на IBM 650. В 1962 г. инженеры из американской компании Bell Telephone Laboratories - В.А. Высотский, Г.Д. Макилрой и Роберт Моррис - создали игру “Дарвин”. Игра предполагала присутствие в памяти вычислительной машины так называемого супервизора, определявшего правила и порядок борьбы между собой программ-соперников, создававшихся игроками. Смысл игры заключался в удалении всех копий программы противника и захвате поля битвы. На этом теоретические исследования ученых и безобидные упражнения инженеров ушли в тень, и совсем скоро мир узнал, что теория саморазмножающихся структур с не меньшим успехом может быть применена и в несколько иных целях. Сегодня история компьютерной вирусологии представляется постоянной «гонкой за лидером», причем, несмотря на век» мощь современных антивирусных программ, лидерами являются именно вирусы. Среди тысяч вирусов лишь несколько десятков являются оригинальными разработками. использующими действительно принципиально новые идеи. Все остальные — <вариации на тему». Но каждая оригинальная разработка заставляет создателей антивирусов приспосабливаться к новым условиям, догонять вирусную технологию. Последнее можно оспорить. Например, в 1989 году американский студент сумел создать вирус, который вывел из строя около 6000 компьютеров Министерства обороны США. Или эпидемия известного вируса Dir–II, разразившаяся в 1991 году. Вирус использовал оригинальную, принципиально новую технологию и на первых порах сумел широко распространиться за счет несовершенства традиционных антивирусных средств. Или всплеск компьютерных вирусов в Великобритании (Кристоферу Пайну удалось создать вирусы Pathogen и Queeq, а также вирус Smeg). Вирус Smeg был самым опасным, его можно было накладывать на первые два

вируса, и из-за этого после каждого прогона программы они меняли конфигурацию. Поэтому их было невозможно уничтожить. Чтобы распространить вирусы, Пайн скопировал компьютерные игры и программы, заразил их, а затем отправил обратно в сеть. Пользователи загружали в свои компьютеры, зараженные программы и инфицировали диски. Ситуация усугубилась тем, что Пайн умудрился занести вирусы и в программу, которая с ними борется. Запустив ее, пользователи вместо уничтожения вирусов получали еще один. В результате этого были уничтожены файлы множества фирм, убытки составили миллионы фунтов стерлингов. Широкую известность получил американский программист Моррис. Его знают как создателя вируса, который в ноябре 1988 года заразил порядка 7 тысяч персональных компьютеров, подключенных к Internet. Причины появления и распространения компьютерных вирусов, с одной стороны, скрываются в психологии человеческой личности и ее теневых сторонах (зависти, мести, тщеславии, невозможности конструктивно применить свои способности), с другой стороны, обусловлены отсутствием аппаратных средств защиты и противодействия со стороны операционной системы персонального компьютера. **Пути проникновения вирусов в компьютер** Основными путями проникновения вирусов в компьютер являются съемные диски (гибкие и лазерные), а также компьютерные сети. Заражение жесткого диска вирусами может произойти при загрузке программы с дискеты, содержащей вирус. Такое заражение может быть и случайным, например, если дискету не вынули из дисковода и перезагрузили компьютер, при этом дискета может быть к не системной. Заразить дискету гораздо проще. На нее вирус может попасть, даже если дискету просто вставили в дисковод зараженного компьютера и, например, прочитали ее оглавление. Вирус, как правило, внедряется в рабочую программу таким образом, чтобы при ее запуске управление сначала передалось ему и только после выполнения всех команд снова вернулось к рабочей программе. Получив доступ к управлению, вирус, прежде всего, переписывает сам себя в другую рабочую программу и заражает ее. После запуска программы, содержащей вирус, становится возможным заражение других файлов. Наиболее часто вирусом заражаются загрузочный сектор диска и исполняемые файлы, имеющие расширения EXE, COM, BAT. Крайне редко заражаются текстовые файлы. После заражения программы вирус может выполнить какую-нибудь диверсию (не слишком серьезную, чтобы не привлечь вниманий). И не забывает возвратить управление той программе, из которой был запущен. Каждое выполнение зараженной программы переносит вирус в следующую программу. Таким образом, будет заражено все программное обеспечение. **Признаки**

появления вирусов. При заражении компьютера вирусом важно его обнаружить. Для этого следует знать об основных признаках проявления вирусов. К ним можно отнести следующие: • прекращение работы или неправильная работа ранее успешно функционировавших программ; • медленная работа компьютера; • невозможность загрузки операционной системы; • исчезновение файлов и каталогов или искажение их содержимого; • изменение даты и времени модификации файлов; • изменение размеров файлов; • неожиданное значительное увеличение количества файлов на диске; • существенное уменьшение размера свободной оперативной памяти; • вывод на экран непредусмотренных сообщений или изображений; • подача непредусмотренных звуковых сигналов;

• частые зависания и сбои в работе компьютера. Следует отметить, что вышеперечисленные явления необязательно вызываются присутствием вируса, а могут быть следствием других причин, Поэтому всегда затруднена правильная диагностика состояния компьютера. **Классификация вирусов.** Сегодня число известных вирусов уже перевалило пятитысячную отметку. Вирусы принято классифицировать: **1. По среде обитания:** - сетевые вирусы (они распространяются по разным компьютерным сетям); - файловые вирусы (заражают исполняемые файлы программ — com, exe, bat); - загрузочные вирусы (заражают загрузочный сектор диска (boot-sector) или сектор, содержащий программу загрузки системного диска Master Boot Records); - файлово-загрузочные (действуют так же как и загрузочные вирусы). 2. **По способу заражения:** - резидентные (такой вирус при инфицировании ПК оставляет в оперативке свою резидентную часть), которая потом перехватывает обращение ОС к объектам заражения и поражает их. Резидентные вирусы живут до первой перезагрузки ПК); – нерезидентные (не заражают оперативную память и могут быть активными ограниченное время). **3. По результату воздействия:** - неопасные (как правило эти вирусы забивают память компьютера путем своего размножения и могут организовывать мелкие пакости — проигрывать заложенную в них мелодию или показывать картинку); - опасные (эти вирусы способны создать некоторые нарушения в функционировании ПК – сбои, перезагрузки, глюки, медленная работа компьютера и т.д.); - очень опасные (опасные вирусы могут уничтожить программы, стереть важные данные, убить загрузочные и системные области жесткого диска, который потом можно выбросить) **4. По алгоритму работы:** - паразитические (меняют содержимое файлов и секторов диска). Такие вирусы легко вычисляются и удаляются; - мутанты (их очень тяжело обнаружить из-за применения в них алгоритмов

шифрования; каждая следующая копия размножающегося вируса не будет похожа на предыдущую); - репликаторы (вирусы-репликаторы, они же сетевые черви, проникают через компьютерные сети, они находят адреса компьютеров в сети и заражают их); - троянский конь (один из самых опасных вирусов, так как Трояны не размножаются, а воруют ценную (порой очень дорогую) информацию пароли, банковские счета, электронные деньги и т.д.); - невидимки (это трудно обнаружимые вирусы, которые перехватывают обращения ОС к зараженным файлам и секторам дисков и подставляют вместо своего незараженные участки. **Как защититься от вирусов.** Значительно снизить вероятность заражения своего компьютера вирусами можно следуя ниже написанным правилам: - установите на свой ПК современную (желательно последней версии) лицензионную антивирусную программу (NOD32, DrWeb, антивирус Касперского); - перед просмотром информации принесенной на флэш-карте (дискете) с другого компьютера – проверьте носитель антивирусником; - после разархивирования архивных файлов сразу проверьте их на вирусы (не все антивирусные программы могут искать вредоносный код в архивах или могут делать это не корректно); - периодически проверяйте компьютер на вирусы (если активно пользуетесь Интернетом – запускайте раз в неделю, а то и чаще); - как можно чаще делайте резервные копии важной информации (backup); - используйте совместно с антивирусной программой файервол (firewall) если компьютер подключен к Интернет;

- настройте браузер (программа просмотра Интернет страниц IE, Opera и т.д.) для запрета запуска активного содержимого html-страниц. **Антивирусная программа** Антивирусная программа (антивирус) изначально программа для обнаружения и лечения программ, заражённых компьютерным вирусом, а также для предотвращения заражения файла вирусом (например, с помощью вакцинации). Многие современные антивирусы позволяют обнаруживать и удалять также троянские программы и прочие вредоносные программы. И напротив программы, создававшиеся как файерволы, также обретают функции, роднящие их с антивирусами (например, Outpost Firewall), что со временем может привести к ещё более очевидному распространению смысла термина на средства защиты вообще. Можно также вспомнить сходство назначений ADinf (позиционировавшийся как антивирус) и tripwire (IDS). Первые наиболее простые антивирусные программы появились почти сразу после появления вирусов. Сейчас разработкой антивирусов занимаются крупные компании. Как и у создателей вирусов, в этой сфере также сформировались оригинальные приемы, но уже для поиска и борьбы с

вирусами. Современные антивирусные программы могут обнаруживать десятки тысяч вирусов. Антивирусное программное обеспечение состоит из компьютерных программ, которые пытаются обнаружить, предотвратить размножение и удалить компьютерные вирусы и другие вредоносные программы. **Методы обнаружения вирусов** Антивирусное программное обеспечение обычно использует два отличных друг от друга метода для выполнения своих задач: • Сканирование файлов для поиска известных вирусов, соответствующих определению в антивирусных базах • Обнаружение подозрительного поведения любой из программ, похожего на поведение заражённой программы. 1 .Метод соответствия определению вирусов в словаре **Обнаружение, основанное на сигнатурах.** Это метод, когда антивирусная программа, просматривая файл, обращается к антивирусным базам, которые составлены производителем программы-антивируса. В случае соответствия, какого либо участка кода просматриваемой программы известному коду (сигнатуре) вируса в базах, программа антивирус может по запросу выполнить одно из следующих действий: 1. Удалить инфицированный файл. 2. Заблокировать доступ к инфицированному файлу. 3. Отправить файл в карантин (то есть сделать его недоступным для выполнения, с целью недопущения дальнейшего распространения вируса). 4. Попытаться восстановить файл, удалив сам вирус из тела файла. 5. В случае невозможности лечения/удаления. выполнить эту процедуру при перезагрузке. Для того, чтобы такая антивирусная программа успешно работала на протяжении долгого времени, в словарь вирусов нужно периодически загружать (обычно, через Интернет) обновленные данные. Если бдительные и имеющие склонность к технике пользователи определят вирус по горячим следам, они могут послать зараженные файлы разработчикам антивирусной программы, а они затем добавят информацию о новых вирусах в свой словарь. Для многих антивирусных программ со словарем характерна проверка файлов в тот момент, когда операционная система создает, открывает, закрывает или посылает их по почте. Таким образом, программа может обнаружить известный вирус сразу после его получения. Заметьте, также, что системный администратор может установить в антивирусной программе расписание для регулярной проверки (сканирования) всех файлов на жестком диске компьютера. Хотя антивирусные программы, созданные на основе поиска соответствия определению вируса в словаре, при обычных обстоятельствах, могут достаточно эффективно препятствовать вспышкам заражения компьютеров, авторы вирусов стараются держаться на полшага впереди таких программ-антивирусов, создавая «олигоморфические»,

«полиморфические» и, самые новые, «метаморфические» вирусы, в которых некоторые части шифруются или искажаются так, чтобы невозможно было обнаружить совпадение с определением в словаре вирусов. 2. Метод обнаружения странного поведения программ **Обнаружение аномалий** Антивирусы, использующие метод обнаружения подозрительного поведения программ не пытаются идентифицировать известные вирусы, вместо этого они прослеживают поведение всех программ. Если программа пытается записать какие-то данные в исполняемый файл (exe-файл), программа-антивирус может пометить этот файл, предупредить пользователя и спросить что следует сделать. В настоящее время, подобные превентивные методы обнаружения вредоносного кода, в том или ином виде, широко применяются в качестве модуля антивирусной программы, а не отдельного продукта. Другие названия: Проактивная защита, Поведенческий блокиратор, Host Intrusion Prevention System (HIPS). В отличие от метода поиска соответствия определению вируса в антивирусных базах, метод обнаружения подозрительного поведения даёт защиту от новых вирусов, которых ещё нет в антивирусных базах. Однако следует учитывать, что программы или модули, построенные на этом методе, выдают также большое количество предупреждений (в некоторых режимах работы), что делает пользователя мало восприимчивым ко всем предупреждениям. В последнее время эта проблема ещё более ухудшилась, так как стало появляться всё больше не вредоносных программ, модифицирующих другие exe - файлы, несмотря на существующую проблему ошибочных предупреждений. Несмотря на наличие большого количества предупреждающих диалогов, в современном антивирусном программном обеспечении этот метод используется всё больше и больше. Так, в 2006 году вышло несколько продуктов, впервые реализовавших этот метод: Kaspersky Internet Security, Kaspersky Antivirus, Safe’n’See, F-Secure Internet Security, Outpost Firewall Pro, DefenceWall. Многие программы класса файерволл издавна имели в своем составе модуль обнаружения странного поведения программ. 3. Метод обнаружения при помощи эмуляции **Обнаружение, основанное на эмуляции** Некоторые программы-антивирусы пытаются имитировать начало выполнения кода каждой новой вызываемой на исполнение программы перед тем как передать ей управление. Если программа использует самоизменяющийся код или проявляет себя как вирус (то есть немедленно начинает искать другие exe-файлы например), такая программа будет считаться вредоносной, способной заразить другие файлы. Однако этот метод тоже изобилует большим количеством ошибочных предупреждений. 4. Метод «Белого списка» Общая технология по борьбе с вредоносными программами — это «белый список».

Вместо того, чтобы искать только известные вредоносные программы, это технология предотвращает выполнение всех компьютерных кодов за исключением тех, которые были ранее обозначены системным администратором как безопасные. Выбрав этот параметр отказа по умолчанию, можно избежать ограничений, характерных для обновления сигнатур вирусов. К тому же, те приложения на компьютере, которые системный администратор не хочет устанавливать, не выполняются, так как их нет в «белом списке». Так как у современных предприятий есть множество надежных приложений, ответственность за ограничения в использовании этой технологии возлагается на системных администраторов и соответствующим образом составленные ими «белые списки» надежных приложений. Работа антивирусных программ с такой технологией включает инструменты для автоматизации перечня и эксплуатации действий с «белым списком».

5. Другие методы обнаружения вирусов Ряд других методов предлагается в исследованиях и используется в антивирусных программах . **Эвристическое сканирование –** метод работы антивирусной программы, основанный на сигнатурах и эвристике, призван улучшить способность сканеров применять сигнатуры и распознавать модифицированные версии вирусов в тех случаях, когда сигнатура совпадает с телом неизвестной программы не на 100 %, но в подозрительной программе налицо более общие признаки вируса. Данная технология, однако, применяется в современных программах очень осторожно, так как может повысить количество ложных срабатываний. Практически все современные антивирусные средства применяют технологию эвристического анализа программного кода. Эвристический анализ нередко используется совместно с сигнатурным сканированием для поиска сложных шифрующихся и полиморфных вирусов. Методика эвристического анализа позволяет обнаруживать ранее неизвестные инфекции, однако, лечение в таких случаях практически всегда оказывается невозможным. В таком случае, как правило, требуется дополнительное обновление антивирусных баз для получения последних сигнатур и алгоритмов лечения, которые, возможно, содержат информацию о ранее неизвестном вирусе. В противном случае, файл передается для исследования антивирусным аналитикам или авторам антивирусных программ. **Классификация антивирусов** Касперский использовал следующую классификацию антивирусов в зависимости от их принципа действия (определяющего функциональность): **Сканеры** (устаревший вариант «полифаги»). Определяют наличие вируса по БД, хранящей

сигнатуры (или их контрольные суммы) вирусов. Их эффективность определяется актуальностью вирусной базы и наличием эвристического анализатора **Ревизоры.** Запоминают состояние файловой системы, что делает в дальнейшем возможным анализ изменений. (Класс близкий к Юз). Ревизоры запоминают исходное состояние программ, каталогов и системных областей диска тогда, когда компьютер не заражен вирусом, а затем периодически или по желанию пользователя сравнивают текущее состояние с исходным. Обнаруженные изменения выводятся на экран монитора. Как правило, сравнение состояний производят сразу после загрузки операционной системы. При сравнении проверяются длина файла, код циклического контроля (контрольная сумма файла), дата и время модификации, и другие параметры. Программы-ревизоры имеют достаточно развитые алгоритмы, обнаруживают стелсвирусы и могут даже очистить изменения версии проверяемой программы от изменений, внесенных вирусом. К числу программ-ревизоров относится широко распространенная и России программа Kaspersky Monitor. **Сторожа (мониторы или фильтры).** Отслеживают потенциально опасные операции, выдавая пользователю соответствующий запрос на разрешение/запрещение операции. При попытке какой- либо программы произвести указанные действия «сторож» посылает пользователю сообщение и предлагает запретить или разрешить соответствующее действие. Программы-фильтры весьма полезны, так как способны обнаружить вирус на самой ранней стадии его существования до размножения. Однако они не «лечат» файлы и диски. Для уничтожения вирусов требуется применить другие программы, например фаги. К недостаткам программ-сторожей можно отнести их «назойливостью (например, они постоянно выдают предупреждение о любой попытке копирования исполняемого файла), а также возможные конфликты с другим программным обеспечением. **Вакцины.** Изменяют прививаемый файл таким образом, чтобы вирус, против которого делается прививка, уже считал файл заражённым. В современных условиях, когда количество возможных вирусов измеряется десятками тысяч, этот подход неприменим. **Наиболее распространенные современные вирусы и вредоносные программы Internet-черви** Самым распространённым типом вирусов в последние два года являются Интернет – черви. Именно они представляют главную угрозу для всех пользователей глобальной сети. Почти все Интернет черви - это почтовые черви, и лишь малая доля - это не почтовые черви, применяющие уязвимости программного обеспечения (как правило, серверного). Примеры не почтовых Интернет – червей: IIS-Worm, CodeRed, , CodeBlue и др. Почтовые черви можно делить на подклассы по-разному, но

для конечного пользователя они делятся на два основных класса: 1. Черви, которые запускаются сами (без ведома пользователя); 2. Черви, которые активизируются, только если пользователь сохранит присоединённый к письму файл и запустит его. К первому типу относятся черви, которые используют уязвимости (ошибки) почтовых клиентов. Чаще всего такие ошибки находятся в почтовом клиенте Outlook, а вернее даже не в нём, а в Интернет браузере Internet Explorer. Дело в том, что Outlook создаёт письмо в виде HTML страницы и при отображении этих страниц он использует функции браузера Internet Explorer. Наиболее распространённая уязвимость, применяемая червями, **ошибка IFRAME.** Применяя соответствующий код, вирус, имеет возможность при просмотре письма автоматически сохранить присоединённый к письму файл на диск и запустить его. Самое обидное то, что данная уязвимость обнаружена более двух лет назад. Тогда же компанией Microsoft были выпущены заплатки для всех версий браузера Internet Explorer, исправляющие эту ошибку. И, тем не менее, черви, применяющие данную уязвимость, по-прежнему являются наиболее распространёнными.

Почтовые черви второго типа рассчитаны на то, что пользователь, по каким то соображениям сам запустит программу, присоединённую к письму. Для того чтобы подтолкнуть пользователя к запуску инфицированного файла авторами червей применяются различные психологические ходы. Самый распространённый приём - выдать зараженный файл, за какой то важный документ, картинку или полезную программку создаёт ответы на письма, содержащиеся в почтовой базе. Практически всегда червями применяются “двойные расширения”. Данный принцип рассчитан на то, что почтовые клиенты не отображают полное имя файла (если оно слишком длинное), и пользователь не увидит второго расширения, которое и является “реальным”. То есть пользователь думает, что файл является документом или картинкой, а тот на самом деле является исполняемым файлом с расширением вроде: EXE, COM, PIF, SCR, BAT, CMD и т.п. Если такой файл “открыть”, то тело червя активизируется. Кроме основной функции, размножения, черви почти всегда несут в себе и боевую нагрузку. Действительно, зачем писать червя и выпускать его “в свет”, предварительно не заложив бомбу. Вложенные функции чрезвычайно разнообразны. Так, например, очень часто почтовые черви призваны для того, чтобы установить на зараженный компьютер троянскую программу или утилиту скрытого администрирования и сообщить адрес компьютера творцу червя. Не редко просто уничтожают информацию или просто делают невозможной дальнейшую работу на компьютере. Так червь I–Worm.Magistr выполнял те же действия, что и печально-известный

WinCIH - стирал содержимое FLASH BIOS и затирал мусорными данными информацию на жёстком диске. В любом случае, независимо от наличия или отсутствия вредоносных функций и их “опасности” почтовые черви вредны уже только потому, что они существуют. Это связано с тем, что при размножении они загружают каналы связи и нередко настолько, что полностью парализуют работу человека или целой организации. **Макро-Вирусы** Вторыми по распространённости в диком виде являются макро-вирусы. Данные вирусы являются макросами, хранящимися во внешних файлах программного обеспечения (документах Microsoft Office, AutoCAD, CorelDRAW и пр.) и при открытии документа исполняются внутренними интерпретаторами данных программ. Широкое распространение они получили благодаря огромным возможностям интерпретатора языка Visual Basic, интегрированного в Microsoft Office. Излюбленным местом обитания этих вирусов являются офисы с большим документооборотом. В таких организациях людям, работающим за компьютерами (секретари, бухгалтеры, операторы ЭВМ) некогда заниматься такими мелочами как компьютерные вирусы. Документы лихо переносятся с компьютера на компьютер, без какого либо контроля (особенно при наличии локальной сети).

К сожалению, людям свойственно не воспринимать всерьёз макровирусы, а напрасно. На самом деле макрос, написанный на языке VBA и интегрированный в документ того же Word или Excel, обладает всеми теми же возможностями, что и обычное приложение. Он может отформатировать Ваш винчестер или просто удалить информацию, украсть какие то файлы или пароли и отправить их по электронной почте. Фактически вирусы этого класса способны парализовать работу целого офиса, а то даже и не одного. Опасность макровирусов заключается ещё и в том, что распространяется вирус целиком в исходном тексте. Если человек, к которому попал вирус, более-менее умеет писать на Visual Basic, то он без труда сможет модифицировать вирус, вложить в него свои функции и сделать его невидимым для антивирусов. Не забывайте, что авторы вирусов пользуются теми же антивирусными программами и модифицируют свои вирусы до тех пор, пока те не перестают детектироваться антивирусами. Фактически, таким образом, рождаются новые модификации уже известных вирусов, но для того, чтобы данный вирус обнаруживался антивирусом, он сначала должен попасть в антивирусную лабораторию и только после этого будут добавлены функции детектирования и обезвреживания новой модификации. Так специалистам Украинского Антивирусного Центра известно более 100 модификаций вируса Macro.Word97.Thus, более 200 модификаций

Macro.Word97.Marker и более 50 модификаций Macro.Word97.Ethan (здесь речь идёт о модификациях, значительно отличающихся друг от друга, что требует добавления дополнительных модулей детектирования и лечения данных модификаций вирусов). **Троинские программы и утилиты скрытого администрирования.** Следующими по распространённости являются **Trojan и Backdoor программы.** Отличие этих двух типов программ заключается в том, что троянская программа выполняет активные действия (уничтожение данных, сбор данных и отправка через Internet, выполнение каких либо действий в определённое время), в то время как **Backdoor программы** открывают удалённый доступ к компьютеру и ожидают команды злоумышленника. Для простоты будем называть оба этих класса троянскими программами. Главное отличие "троянов" от всех перечисленных выше творений человеческого разума является то, что троянские программы не размножаются сами. Они единоразово устанавливаются на компьютер и долгое время (как правило, либо до момента обнаружения, либо до переустановки операционной системы, по какой либо причине) выполняет свои функции. При этом троянский конь не может самостоятельно переместиться с одного компьютера в локальной сети на другой. Так почему же Трояны так распространены? Причина таится именно в том, что они максимально «полезны» и незаметны. Часто они являются спутниками сетевых или почтовых червей. Так, почтовый червь I-Worm.LovGate при попадании на компьютер устанавливает в систему backdoor модуль, открывающий доступ к компьютеру по TCP/IP и отправляет разработчику червя письмо, в котором указывается имя пользователя, имя компьютера и сетевой адрес зараженного компьютера. Все троянекие программы можно разделить на три основных класса по выполняемым действиям:

1. Логические (временные) бомбы - программы, различными методами удаляющие/модифицирующие информацию в определённое время, либо по какому то условию.

2. **Шпионы** - собирающие информацию (имена, пароли, нажатия на клавиши) и складирующие её определённым образом, а нередко и отправляющие собранные данные по электронной почте или другим методом.

3. Собственно **backdoor программы** - удалённое управление компьютером или получение команд от злоумышленника (через локальную/глобальную сеть, по электронной почте, в файлах, от других приложений, например тех

же червей или вирусов). Одинаково опасны все три типа программ. Каждый из них способен либо уничтожить данные, либо украсть ценную информацию (хотя бы те же имена и пароли доступа к различным ресурсам). Стоит отметить, что многие троянские программы постоянно обновляются, выходят всё новые и новые модификации. Учитывая то, что троянская программа не может попасть к вам случайно, злоумышленник старательно выбирает: какой бы Троян вам установить. Очень велика вероятность того, что он пойдёт в Интернет и выкачает что-то свеженькое. Именно поэтому необходимо регулярно обновлять базы антивирусного продукта. Так обновления для системы антивирусной защиты “Украинский Национальный Антивирус” (УНА) выходят каждый день, и если вы активно пользуетесь Интернетом, рекомендуем обновлять антивирус минимум раз в неделю (хотя конечно идеально было бы обновляться каждый день). В завершение хотелось бы сказать: процесс развития вирусов и антивирусов - это постоянная война технологий. Регулярно в вирусах реализовываются оригинальные идеи, что требует адекватных действий от разработчиков антивирусного ПО. Поэтому рядовому пользователю рекомендуется следить за новостями на сайтах антивирусных компаний и прислушиваться к советам специалистов по информационной безопасности о необходимости обновления программного обеспечения (не только антивирусного) или выполнении специфических действий по улучшению защищённости ПК.

Термин «хакинг» обычно ассоциируется с незаконной активностью и нарушениями. Однако, в контексте информационной безопасности, хакинг может применяться для добросовестной проверки качества ПО бизнеса. В этом случае речь идёт про тестирование на проникновение или пентестинг, который представляет собой контролируемую попытку проникновения в систему для определения её уязвимостей и обеспечения защиты от реальных кибератак.

Тема 2.2. Тестирование защиты программного обеспечения

Что такое тестирование на проникновение?
Тестирование на проникновение (penetration testing) — это контролируемый процесс, при котором специалисты пытаются проникнуть в систему ПО, используя способы, которые могли бы использовать злоумышленники.

Цель тестирования на проникновение: поиск уязвимостей в ИТ-продукте,

чтобы компания могла устранить их до того, как их используют злоумышленники.

Пример проведения тестирования на проникновение: тестировщик получает разрешение от компании на пентестинг и начинает сканирование сети на предмет уязвимостей. Затем он использует определённые методы, которые мы опишем ниже, чтобы попытаться войти в систему. После завершения тестирования специалист предоставляет отчёт, содержащий обнаруженные уязвимости, и составляет рекомендации по их устранению.

Задачи у этого вида тестирования могут быть следующие:

1. Определение того, насколько эффективны текущие меры безопасности.
2. Поиск слабых мест в ПО, которые могут использовать злоумышленники.
3. Обучение сотрудников компании выявлять угрозы безопасности и правильно реагировать на них.
4. Соблюдение регулятивных требований: многие отраслевые стандарты и законодательство требуют проведения тестирования на проникновение для обеспечения безопасности данных.

Этапы и методики проведения тестирования на проникновение

Этапы тестирования на проникновение:

1. Подготовка: определение целей и объёма тестирования, получение разрешения от компании, определение сроков и бюджета.
2. Сбор информации: специалисты собирают данные о целевой системе, в том числе её архитектуру, используемые технологии и версии ПО, сетевую инфраструктуру, список пользователей и групп, доступные порты и сервисы.
3. Сканирование: проверка сети на предмет уязвимостей. Это может быть сканирование портов, веб-приложений и т.д.
4. Анализ результатов: определение возможных путей входа в систему и оценка уровня риска. На основании этого специалисты составляют список рекомендаций по устранению найденных уязвимостей.

Методики тестирования на проникновение:

1. Ручное тестирование, при котором тестировщик вручную ищет уязвимости в ПО. Это можно делать через перебор паролей, анализ защиты периметра сети и социальная инженерия.
2. При автоматизированном тестировании применяются специальные инструменты для автоматической проверки системы на наличие уязвимостей.
3. Сочетание ручного и автоматизированного тестирования, при котором применяют как ручные, так и автоматизированные методы. Так можно получить более полную картину об уровне безопасности.

В зависимости от своих целей компания может выбрать оптимальный подход к проведению тестирования.

Типы испытаний на проникновение

Испытания на проникновение классифицируются в зависимости от используемых объектов, а также на основе подходов к проверке качества.

Типы в зависимости от используемых объектов

- Социальная инженерия: этот тип заключается в манипуляции людьми для получения нужной злоумышленникам информации. К ней относится фишинг, поддельные звонки, обман и остальные способы воздействия на человеческий фактор.
- Веб-приложения: испытания оценивают уязвимости ПО, такие как межсайтовый скриптинг (XSS), инъекции SQL, уязвимости аутентификации и авторизации и недостатки конфигурации.
- Сетевая служба: эти испытания оценивают безопасность сетевых сервисов, таких как серверы файлов, почтовые серверы, DNS-серверы и другие. Они включают анализ протоколов, сканирование портов и обнаружение уязвимостей.
- Клиентская часть: анализ безопасности мобильных приложений, тестирование web-приложений на проникновение, поиск уязвимостей в локальном ПО и технологии клиентской стороны.
- Удалённое подключение: эти испытания оценивают безопасность удалённого доступа к системам, включая VPN, удалённый рабочий стол и SSH.

- Тестирование на проникновение для беспроводной сети: в этом случае специалисты оценивают безопасность Wi-Fi-сетей, Bluetooth соединений и подобных технологий.

Типы в зависимости от используемых объектов на основе подходов к тестированию

1. Белый ящик (White Box): при этом подходе тестировщикам доступен исходный код, документация и другие данные о системе.

2. Чёрный ящик (Black Box): тестировщики проводят тестирование без доступа к внутреннему устройству системы или её компонентам.

3. Серый ящик (Gray Box): этот подход сочетает элементы белого и чёрного ящика, где у QA-специалистов есть ограниченная информация о системе, что помогает имитировать атаку.

Инструменты тестирования на проникновение

1. Nmap — сканер сети, который способен обнаруживать устройства в сети, анализировать открытые порты и определять операционные системы.
2. Nessus представляет собой инструмент для автосканирования. Он помогает находить уязвимости сети и серверов, анализировать безопасность сети и создавать отчёт тестирования на проникновение.
3. Metasploit — фреймворк для разработки и выполнения эксплойтов. Позволяет тестировать уязвимости и проводить атаки на системы.
4. Wireshark — анализатор сетевого трафика, который помогает мониторить и собирать информацию, которая передаётся по сети.
5. OpenSSL представляет собой библиотеку криптографических функций, которая используется для шифрования данных и обеспечения безопасности коммуникаций.

Раскрытие уязвимостей и оценка рисков в тестировании на проникновение

При раскрытии уязвимостей и оценке рисков можно получить следующие результаты:

- Выявление конкретных уязвимостей: проведение тестирования на проникновение помогает обнаружить уязвимости в системе, такие как недостатки в конфигурации или ошибки в коде.
- Оценка вероятности эксплуатации уязвимостей: при помощи тестирования можно определить, насколько легко или сложно для злоумышленников использовать обнаруженные уязвимости для атаки на ПО.
- Оценка потенциальных последствий эксплуатации уязвимостей: специалисты анализируют, какие последствия могут возникнуть, если будет совершена атака на уязвимости ИТ-продукта. В этом случае может произойти утечка данных или нарушение целостности системы.

Методы, которые используются для раскрытия уязвимостей:

- Активное сканирование сети и портов: здесь помогут инструменты для обнаружения открытых портов и сервисов, такие как Nmap.

- Эксплойтинг уязвимостей: попытки использования известных эксплойтов для проверки уязвимостей.

- Социальная инженерия: проверка наличия слабых мест в человеческом факторе.

Одного раскрытия уязвимостей не хватит для обеспечения безопасности системы. Для полноценной оценки рисков и разработки соответствующих мер по обеспечению безопасности специалисты могут применять следующие методы:

1. Матрица рисков: оценка вероятности и воздействия уязвимостей на систему с последующей приоритизацией мер по обеспечению безопасности.
2. Анализ вероятности и последствий: оценка вероятности эксплуатации уязвимостей и анализ потенциальных последствий для ПО и бизнеса в целом.

По итогу раскрытия уязвимостей и оценки рисков можно получить:

- Список конкретных уязвимостей, выявленных в ИТ-продукте.
- Оценку вероятности эксплуатации каждой уязвимости.
- Оценку потенциальных последствий эксплуатации каждой уязвимости.

- Рекомендации по устранению или смягчению уязвимостей.
- Список мер по обеспечению безопасности системы.

Преимущества и рекомендации по улучшению безопасности на основе результатов тестирования на проникновение

Тестирование на проникновение важно для бизнеса по нескольким причинам:

1. Поиск уязвимостей: тестирование на проникновение помогает обнаружить уязвимости в ИТ-продукте, которые злоумышленники могут использовать для доступа к системе или данным. Благодаря этому можно предотвратить потенциальные атаки и защитить данные компании.
2. Оценка рисков: результаты тестирования помогают оценить потенциальные риски и угрозы для ПО компании. На их основании можно разработать и реализовать меры по обеспечению безопасности.
3. Повышение доверия пользователей: если компания устранит найденные уязвимости в ПО, то клиенты будут больше доверять ИТ-продукту, поскольку они могут быть уверены в его надёжности и защите их данных.

Ниже мы привели советы о том, как компания может улучшить безопасность своего ИТ-продукта:

- Обращение к профессиональным специалистам, которые смогут устранить уязвимости через внесение изменений в код продукта, обновления программного обеспечения или изменения конфигурации системы.
- Усиление авторизации через внедрение более надёжных способов аутентификации, таких как двухфакторная аутентификация, а также установление строгих правил авторизации для ограничения доступа к защищённой информации.
- Обеспечение шифрования конфиденциальных данных в покое и в движении для защиты от правонарушителей.
- Внедрение обучения сотрудников по вопросам информационной безопасности, в том числе осведомлённость о социальной инженерии и фишинговых атаках.
- Установка систем мониторинга, которые позволят фиксировать подозрительную активность и предотвращать потенциальные атаки.

- Проведение периодических проверок безопасности ИТ-продукта для поиска новых уязвимостей и поддержания высокого уровня защиты данных компании.

Тестирование на проникновение — важная часть проверки качества. Оно позволяет обнаружить уязвимости в программном обеспечении, оценить риски и предложить рекомендации по улучшению безопасности данных. Тестирование поможет защитить ИТ-продукт и снизить риск возможных атак и утечек данных.

На бесплатной консультации наши QA-специалисты ответят на ваши вопросы о тестировании программного обеспечения.

Тема 2.3. Шифрование информации (средства и протоколы шифрования сообщений)

Алгоритмы традиционного шифрования
Шифр Цезаря

В шифре Цезаря каждая буква алфавита заменяется буквой, которая находится на три позиции дальше в этом же алфавите. Проще всего увидеть это на примере.

Открытый текст: meet me after the toga party

Шифрованный текст: PHHW PH DIWHU WKH WRJD SDUMB

Если каждой букве назначить числовой эквивалент (a = 1, b = 2 и т.д.), то алгоритм можно выразить следующими формулами. Каждая буква открытого текста p заменяется буквой шифрованного текста C:

C = E(p) = (p + 3) mod(26).

В общем случае сдвиг может быть любым, поэтому обобщенный алгоритм Цезаря записывается формулой

C = E(p) = (p+k)mod(26),

где k принимает значения в диапазоне от 1 до 25. Алгоритм дешифрования также прост:

p = D(C) = (C - k)mod(26) .

Если известно, что определенный текст был шифрован с помощью шифра Цезаря, то с помощью простого перебора всех вариантов раскрыть шифр очень просто — для этого достаточно проверить 25 возможных вариантов ключей.

Шифр Плейфейера

Шифр Плейфейера (Playfair) базируется на методе многобуквенного шифрования. Биграммы открытого текста рассматриваются как самостоятельные единицы, преобразуемые в заданные биграммы шифрованного текста.

Алгоритм Плейфейера основан на использовании матрицы букв размерности 5x5, созданной на основе некоторого ключевого слова.

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

В данном примере ключевым словом является *monarchy*. Матрица создается путем размещения букв, использованных в ключевом слове, слева направо и сверху вниз (повторяющиеся буквы отбрасываются). Затем оставшиеся буквы алфавита размещаются в естественном порядке в оставшихся строках и столбцах матрицы. Буквы I и J считаются одной и той же буквой. Открытый текст шифруется порциями по две буквы в соответствии со следующими правилами.

1. Если оказывается, что повторяющиеся буквы открытого текста образуют одну пару для шифрования, то между этими буквами вставляется специальная буква-заполнитель, например x. В частности, такое слово как balloon будет преобразовано к виду ba lx lo on.

2. Если буквы открытого текста попадают в одну и ту же строку матрицы, каждая из них заменяется буквой, следующей за ней в той же строке справа —

с тем условием, что для замены последнего элемента строки матрицы служит первый элемент той же строки. Например, ar шифруется как RM.

3. Если буквы открытого текста попадают в один и тот же столбец матрицы, каждая из них заменяется буквой, стоящей в том же столбце сразу под ней, с тем условием, что для замены самого нижнего элемента столбца матрицы берется самый верхний элемент того же столбца. Например, mu шифруется как CM.

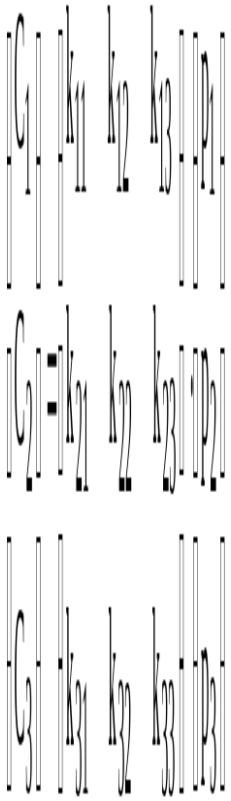
4. Если не выполняется ни одно из приведенных выше условий, каждая буква из пары букв открытого текста заменяется буквой, находящейся на пересечении содержащей эту букву строки матрицы и столбца, в котором находится вторая буква открытого текста. Например, hs шифруется как BP, а ea — как IM (или JM, по желанию шифровальщика),

Шифр Хилла

Алгоритм, разработанный математиком Лестером Хиллом (Lester Hill) в 1929 г., заменяет каждые *m* последовательных букв открытого текста *m* буквами шифрованного текста. Подстановка определяется *m* линейными уравнениями, в которых каждому символу присваивается числовое значение (*a = 0, b = 1, ... z = 25*). Например, при *m = 3* , получаем следующую систему уравнений:

$$C_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_{13}) \bmod 26,$$
$$C_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_{13}) \bmod 26,$$
$$C_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_{13}) \bmod 26.$$

Эту систему уравнений можно записать в виде произведения вектора и матрицы в следующем виде:



$$C = KP$$

где **C** и **P** — векторы длины 3, представляющие соответственно шифрованный и открытый текст, а K — это матрица размерности 3x3, представляющая ключ шифрования. Операции выполняются по модулю 26.

Рассмотрим, например, как будет шифрован текст "paymoremoney" при использовании ключа

$$K = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Первые три буквы открытого текста представлены вектором (15 0 24). Таким образом, **K**(15 0 24) = (275 819 486)mod26 = (11 13 18) = LNS. Продолжая вы-

численния, получим для данного примера шифрованный текст вида LNSHDLEWMTRW.

Для расшифровки нужно воспользоваться матрицей, обратной **К**. При этом для обратной ключевой матрицы должно выполняться соотношение:

$\mathbf{K}\mathbf{K}^{-1} \bmod 26 = \mathbf{K}^{-1}\mathbf{K} \bmod 26 = \mathbf{I}$,

где **I** — это единичная матрица. Для примера

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}.$$

Полиалфавитные шифры

Полиалфавитные шифры основаны на использовании нескольких подстановок, применяемых в ходе шифрования открытого текста в зависимости от определенных условий.

Самым широко известным и одновременно самым простым алгоритмом такого рода является шифр Виженера (Vigennre). Этот шифр базируется на наборе правил моноалфавитной подстановки, представленных 26 шифрами Цезаря со сдвигом от 0 до 25. Каждый из таких шифров можно обозначить ключевой буквой, являющейся буквой шифрованного текста, соответствующей букве а открытого текста. Например, шифр Цезаря, для которого смещение равно 3, обозначается ключевой буквой *d.*

Для облегчения понимания и применения этой схемы была предложена матрица, названная "табло Виженера". Все 26 шифров располагаются по горизонтали, и каждому из шифров соответствует своя ключевая буква, представленная в крайнем столбце слева. Алфавит, соответствующий буквам открытого текста, находится в первой сверху строке таблицы. Процесс шифрования прост — необходимо по ключевой букве *x* и букве открытого текста *y* найти букву шифрованного текста, которая находится на

пересечении строки *x* и столбца *y*. В данном случае такой буквой является буква V.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	SU
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	SU	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	SU	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Чтобы зашифровать сообщение, нужен ключ, имеющий ту же длину, что и само сообщение. Обычно ключ представляет собой повторяющееся нужное число раз ключевое слово, чтобы получить строку подходящей длины.

Расшифровать текст также просто — буква ключа определяет строку, буква шифрованного текста, находящаяся в этой строке, определяет столбец, и в этом столбце в первой строке таблицы будет находиться соответствующая буква открытого текста.

Перестановочные шифры

Все рассмотренные выше методы основывались на замещении символов открытого текста различными символами шифрованного текста. Принципиально иной класс преобразований строится на использовании

перестановок букв открытого текста. Шифры, созданные с помощью перестановок, называют перестановочными шифрами.

Простейший из таких шифров использует преобразование "лесенки", заключающееся в том, что открытый текст записывается вдоль наклонных строк определенной длины ("ступенек"), а затем считывается построчно по горизонтали. Например, чтобы шифровать сообщение "meet me after the toga party" no методу лесенки со ступеньками длиной 2, запишем это сообщение в виде

m e m a t r h t g p r y

e t e f e t e o a a t

Шифрованное сообщение будет иметь следующий вид.

MEMATRHTGPRYETEFETEOAAT

Перестановочный шифр можно сделать существенно более защищенным, выполнив шифрование с использованием перестановок несколько раз. Оказывается, что в этом случае примененную для шифрования перестановку воссоздать уже не так просто. Например, если предыдущее сообщение шифровать еще раз с помощью того же самого алгоритма, то результат будет следующим.

Ключ: 4 3 1 2 5 6 1

Открытый текст: t t n a a p t

m t s u o a o

d w c o i x k

n l y p e t z

Шифрованный текст: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

Защита открытой информационной системы

Современная информационная система представляет собой совокупность неоднородных элементов и реализует широкий диапазон сетевых функций и услуг. Наиболее полной моделью ИС является модель сетевого

взаимодействия открытых систем. Такая ИС представляет универсальную распределенную среду с широкими вычислительными возможностями, ориентированную на большой круг пользователей. Однако при этом она становится мишенью для возможных угроз, попыток несанкционированного доступа, что делает актуальной проблему защиты таких ИС.

Потребность обеспечения надежности вычислительных услуг, целостности, конфиденциальности и доступности информации приводит к целесообразности включения функции защиты в число обязательных функций ИС.

Характеристики безопасной ИС

1. *Целостность*ресурсов ИС предполагает выполнение следующих условий:

а) все ресурсы ИС всегда доступны пользователям независимо от надежности технического или качества программного видов обеспечения, а также несанкционированных действий (защита от потери данных);

б) наиболее важные ресурсы всегда доступны независимо от попыток их разрушения (защита от разрушения данных).

2. *Защита (безопасность)*ресурсов ИС означает, что все операции с этими ресурсами выполняются по строго определенным правилам и инструкциям.

3. *Право владения*ресурсами ИС есть право отдельных субъектов ИС распоряжаться принадлежащей им информацией (имеется в виду ее сбор, хранение, использование и распространение).

Эти три основные характеристики функционирования сети (целостность, безопасность и право владения ресурсами) составляют основу надежности среды, поддерживаемой ИС.

Проблемы защиты ис

Большое число различных компонентов, операций, ресурсов и объектов ИС создает весьма привлекательную среду для различного рода вторжений и несанкционированных операций. Это означает, что защита становится важным аспектом функционирования любой сети, что обусловлено следующими причинами:

- Сетевая структура приводит к **росту сложности вычислительной системы**. Становится невозможным контроль всех объектов и операций в ИС в каждый момент времени. Сетевая структура ИС в большей степени подвержена различного рода вторжениям и ошибочным действиям. Возрастающая сложность определяется следующими факторами: ИС представляет географически распределенную систему, возможно выходящую за пределы границ страны. Сеть может объединять различные устройства и средства управления, использовать различные типы компьютеров и операционных систем, включать в состав большое число объектов.
- **ИС являются неотъемлемой частью жизни общества**. Неправильное функционирование ИС может вызвать гибельные последствия для правительств, общества, бизнеса и отдельного гражданина.
- При включении в сеть бытовых персональных компьютеров существенно возрастает нагрузка из-за **огромного числа индивидуальных пользователей**. Ситуация, когда большое число пользователей одновременно обращается к одним и тем же данным, может создать угрозу праву владения сетью, поскольку достаточно легко выявить, какие новости интересуют ту или иную личность, с кем он общается, какую информацию разыскивает и т. п.
- **Угроза искажения информации существует в любой точке ИС**, начиная от места ввода сообщения в сеть до места назначения. В частности, информация наиболее подвержена угрозе при передаче по линиям связи. Защита информации на всех стадиях передачи по сети должна прежде всего касаться содержания сообщения.

Классификация вторжений в ис

По методу атаки вторжения могут быть:

- Пассивные. Нарушитель наблюдает за прохождением информации по линии связи, не изменяя ни информационный поток, ни содержание передаваемой информации. Как правило, злоумышленник выполняет анализ потока сообщений (трафика), фиксируя пункты назначений и идентификаторы, или только факт прохождения сообщения, его длину и частоту обмена, если содержимое сообщения нераспознаваемо.
- Активные. Нарушитель стремится подменить информацию, передаваемую в сообщении. Он может выборочно изменить или добавить правильное или неправильное сообщение, удалить, задержать или изменить порядок следования сообщений. Злоумышленник может

также аннулировать или задержать все сообщения, передаваемые по сети: такое вторжение равносильно отказу в передаче сообщений.

По принципу несанкционированного доступа:

- физический несанкционированный доступ;
- логический несанкционированный доступ;

По положению источника несанкционированного доступа:

- несанкционированный доступ, источник которого расположен в локальной сети;
- несанкционированный доступ, источник которого расположен вне локальной сети.

По режиму выполнения несанкционированного доступа:

- атаки, выполняемые при постоянном участии человека;
- атаки, выполняемые специально разработанными программами без непосредственного участия человека.

По типу используемых слабостей системы информационно-компьютерной безопасности:

- атаки, основанные на недостатках установленной политики безопасности;
- атаки, основанные на ошибках административного управления компьютерной сетью;
- атаки, основанные на недостатках алгоритмов защиты, реализованных в средствах информационно-компьютерной безопасности;
- атаки, основанные на ошибках реализации проекта системы защиты.

По пути несанкционированного доступа:

- атаки, ориентированные на использование прямого стандартного пути доступа к компьютерным ресурсам;
- атаки, ориентированные на использование скрытого нестандартного пути доступа к компьютерным ресурсам.

По текущему месту расположения конечного объекта атаки:

- атаки на информацию, хранящуюся на внешних запоминающих устройствах;
- атаки на информацию, передаваемую по линиям связи;
- атаки на информацию, обрабатываемую в основной памяти компьютера.

По непосредственному объекту атаки:

- атаки на политику безопасности и процесс административного управления;
- атаки на постоянные компоненты системы защиты;
- атаки на сменные элементы системы безопасности;
- нападения на протоколы взаимодействия;
- нападения на функциональные элементы компьютерной системы.

Конечным объектом нападения всегда является защищаемая информация. Под непосредственным же объектом атаки понимается объект, анализ или использование которого позволяет успешно реализовать несанкционированный доступ к защищаемой информации. Например, непосредственным объектом нападения может быть криптосистема, позволяющая злоумышленнику спрогнозировать значение генерируемого секретного ключа.