ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ
«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ КОЛЛЕДЖ
ИМ. А. А. НИКОЛАЕВА»

Конспект лекций
по дисциплине

МДК 04.02 ОБЕСПЕЧЕНИЕ КАЧЕСТВА ФУНКЦИОНИРОВАНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Специальность 09.02.07 Информационные системы и программирование

Москва
2020

Составитель: Антошков А.А. – преподаватель ГБПОУ МАДК им. А.А. Николаева высшей квалификационной категории

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Дисциплина **МДК 04.02.** Обеспечение качества функционирования компьютерных систем является частью программы подготовки специалистов среднего звена, в соответствии с ФГОС СПО по специальности 09.02.07 Информационные системы и программирование.

Цель и планируемые результаты освоения дисциплины:

Код ПК, ОК	Умения	Знания
ОК 1, ОК 2, ОК 4, ОК 5, ОК 9, ОК 10; ПК 4.1 ПК 4.2 ПК 4.3 ПК 4.4	– Определять направления модификации программного продукта. – Разрабатывать и настраивать программные модули программного продукта. – Настраивать конфигурацию программного обеспечения компьютерных систем. – Использовать методы защиты программного обеспечения компьютерных систем. – Анализировать риски и характеристики качества программного обеспечения. – Выбирать и использовать методы и средства защиты компьютерных систем программными и аппаратными средствами. –	– Основные методы и средства эффективного анализа функционирования программного обеспечения. – Основные средства и методы защиты компьютерных систем программными и аппаратными средствами. – Основные принципы контроля конфигурации и поддержки целостности конфигурации ПО.

Информационное обеспечение обучения

Для реализации программы библиотечный фонд образовательной организации имеет печатные и электронные образовательные и информационные ресурсы, рекомендуемые для использования в образовательном процессе

Перечень рекомендуемых учебных изданий, Интернет-ресурсов, дополнительной литературы

Основная литература (печатные издания)

1. Федорова Г.И. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности. Учебное пособие. – М.: Издательство Инфра-М. Среднее профессиональное образование. 2016 г. - 336 с.

2. Рудаков А. Технология разработки программных продуктов: учебник. . –М.: Издательство Academia. Среднее профессиональное образование. 2019 г. - 208 с.

3. Федорова Г., Рудаков А. Технология разработки программных продуктов. Практикум: учебное пособие. . – М.: Издательство Academia. Среднее профессиональное образование. 2017 г. - 192 с.

Дополнительная литература (печатные издания)

4. Орлов С.А., Цилькер Б.Я. Технологии разработки программного обеспечения: учебник.– Издательство Инфра-М.: 2016, - 609 с.

5. Проектирование и реализация прикладного программного обеспечения: учебное пособие. Влацкая И. В., Заельская Н. А., Надточий Н. С. – М.: ОГУ. 2015 г. - 119 с.

Электронные издания (электронные ресурсы)

1. От модели объектов - к модели классов. - http://real.tepkom.ru/Real_OM-CM_A.asp

2. Технология разработки программного обеспечения: - http://window.edu.ru/catalog/pdf2txt/195/19195/1551

ПЕРРЕЧЕНЬ ЛЕКЦИЙ

№ лекции	Темы лекций
	Тема 1 Основные методы обеспечения качества функционирования
1	Многоуровневая модель качества программного обеспечения
2	Объекты уязвимости
3	Дестабилизирующие факторы и угрозы надежности
4	Методы тестирования ПО
5	Методы предотвращения угроз надежности
6	Математические модели описания статистических характеристик ошибок в программах
7	Первичные ошибки, вторичные ошибки и их проявления
8	Целесообразность разработки модулей адаптации
	Тема 2 методы и средства защиты компьютерных систем
9	Вредоносные программы: классификация, методы обнаружения
10	Антивирусные программы: классификация, сравнительный анализ
11	Файрвол: задачи, сравнительный анализ, настройка
12	Архивация системных данных
13	Средства и протоколы шифрования сообщений
14	Средства диагностики оборудования
15	Конфигурирование ИС. Оперативное управление и регламентные работы.
	Тема 3. Администрирование ИС
16	Конфигурирование ИС. Оперативное управление и регламентные работы
17	Решение проблем конфигурации с помощью групповых политик

ЛЕКЦИЯ 1

Многоуровневая модель качества программного обеспечения

План.

1. Общая структура качества
2. **Методы и средства разработки программных продуктов**
3. **Критерии качества**
4. **Стандарт ISO 9126**

Стандарты в области информационных систем

Многоуровневая модель качества ПО в стандарте ISO 9126.

Что такое технология программирования? Методы и средства разработки программных продуктов?

Технология программирования – совокупность принципов разработки, обеспечивающих массовое производство ПО требуемого качества в установленные сроки.

Методами технологии программирования называются способы и приемы организации производственных процессов при разработке программных средств.

Методы ТП определяюторганизационную структуру коллектива разработчиков, способы разбиения процесса разработки на отдельные этапы, последовательность этих этапов и т.д.

Средствами технологии программирования называются утилиты, обеспечивающие автоматизированную или автоматическую поддержку методов.

Совместно используемые утилиты объединяются в системы автоматизированной разработки ПО.

Такие системы принято называть **CASE-средствами** (Computer Aided Software Engineering)

Понятие качества программных продуктов. Критерии качества.

Свойство программы, характеризующееся отсутствием в ней ошибок по отношению к целям разработки, называется **правильностью программы**.

Даже для «малых» программ обеспечение их правильности является чрезвычайно сложной задачей, а для «больших» программ оно становится практически бессмысленным.

Качество ПО – это вся совокупность его характеристик, относящихся к возможности удовлетворять высказанные или подразумеваемые потребности всех заинтересованных лиц (стандарт ISO 9126).

Основными критериями качества ПО (criteria of software quality) являются:

-функциональность (Способность ПО выполнять набор функций (действий), удовлетворяющих заданным или подразумеваемым потребностям пользователей.Набор указанных функций определяется во внешнем описании ПО)

-надежность (это его способность с достаточно большой вероятностью безотказно выполнять определенные функции при заданных условиях и в течение заданного периода времени)

-эффективность (Соотношение уровня услуг, предоставляемых ПО пользователю при заданных условиях, и объема используемых для этого ресурсов. К числу таких ресурсов могут относиться требуемые аппаратные средства, время выполнения программ, затраты на подготовку данных и интерпретацию результатов)

-эргономичность (Характеристики ПО, которые позволяют минимизировать усилия пользователя по подготовке исходных данных, применению ПО и оценке полученных результатов, а также вызывать положительные эмоции определенного или подразумеваемого пользователя)

-модифицируемость (Характеристики ПО, которые позволяют минимизировать усилия по внесению изменений для устранения ошибок и по его модификации в

соответствии с изменяющимися потребностями пользователей. Модифицируемость ПО существенно зависит от степени и качества его документированности)

-мобильность (Способность ПО быть перенесенным из одной среды (окружения) в другую, в частности, с одной аппаратной платформы на другую)

Определение качества ПО в стандарте ISO 9126. Аспекты качества, их взаимное влияние.

Стандарт ISO 9126- Международный стандарт, определяющий оценочные характеристики качества программного обеспечения.

Разделяется на 4 части, описывающие следующие вопросы:

-модель качества;

-внешние метрики качества;

-внутренние метрики качества;

-метрики качества в использовании.

Качество определяется в стандарте ISO 9126как вся совокупность его характеристик, относящихся к возможности удовлетворять высказанные или подразумеваемые потребности всех заинтересованных лиц.

Различаются понятия:

-внутреннего качества,

-внешнего качества,

-качества ПО при использовании

Три аспекта качества ПО

-Внутреннее качество связано с характеристиками ПО самого по себе, без учета его поведения;

-Внешнее качество характеризующего ПО с точки зрения его поведения;

-Качества ПО при использовании – это то качество, которое ощущается пользователями при конкретных сценариях работы ПО.

Многоуровневая модель качества ПО в стандарте ISO 9126.

Модель качества

Качество определяется в стандарте ISO 9126как вся совокупность его характеристик, относящихся к возможности удовлетворять высказанные или подразумеваемые потребности всех заинтересованных лиц.

Стандарт ISO 9126 предлагает использовать для описания внутреннего и внешнего качества ПО многоуровневую модель.

На верхнем уровне выделено 6 основных характеристик качества ПО. Каждая характеристика описывается при помощи нескольких входящих в нее атрибутов. Для каждого атрибута определяется набор метрик, позволяющих его оценить.

Вопросы для самопроверки

1. По какому принципу можно сгруппировать стандарты на разработку информационных систем.
2. Что такое стандарт ISO 9126?
3. Что такое технология программирования ?
4. то называется средствами технологии программирования ?
5. Что является основными критериями качества ПО ?

ЛЕКЦИЯ 2

Объекты уязвимости

План.

1. Контрольный опрос
2. Классификация цязвимостей.
3. уязвимости, вызванные дефектами конфигурирования и управления системой.
4. уязвимости, вызванные дефектами проектирования.

Уязвимости программного обеспечения

Известно, что даже небольшая, казалось бы, уязвимость ПО может привести к большому ущербу. Отсюда и пристальное внимание разработчиков и пользователей ПО к вопросам систематизации уязвимостей. Следует отметить, что проблема составления универсальной классификационной схемы уязвимостей ПО остается актуальной: в основном исследователями приводятся классификационные схемы, охватывающие лишь малую часть предметной области, на которую они ориентируются.

Далее приведем классификацию уязвимостей по критерию «причина возникновения» в общем виде. Данная классификация включает два типа уязвимостей.

Первый тип — уязвимости, вызванные дефектами (ошибки, проблемы) проектирования и программирования системы, такими как [2, 3, 4]:

- ошибки обработки и представления данных;
- неправильная обработка входных и выходных данных;
- отсутствие проверки и представления ввода;
- некорректное кодирование и экранирование вывода;
- некорректная обработка входных данных;
- ошибочная внутренняя трансформация данных;
- ошибки, связанные с использованием строк;
- ошибки типов данных;
- ошибки представления данных;
- числовые ошибки;
- ошибки определения структур данных;
- ошибки доступа к данным;
- ошибки управления информацией;
- неверный доступ к индексируемому ресурсу;
- ошибки модификации постоянных данных;
- нарушение внутренней структуры и зависимости;
- некорректное использование API;
- ошибки, связанные с инкапсуляцией;
- ошибки обработки событий и состояний;
- ошибки временных меток и внутреннего состояния;
- нарушение логики функционирования;
- некорректно написанные обработчики;
- некорректная обработка ошибок и внештатных ситуаций;
- неправильное использование ресурсов и внутренних механизмов системы;
- ошибки при использовании механизмов безопасности;
- ошибки инициализации и очистки областей памяти;
- некорректное использование ссылок и псевдонимов;
- некорректное использование указателей;
- ошибки, свойственные определенному типу функционала;
- ошибки при реализации пользовательского интерфейса;
- некорректное использование сетевых протоколов;

• присутствие в коде намеренно и ненамеренно внедренных объектов (закладок);
• отклонения от стандартов качества проектирования, реализации, документирования;
• несоблюдение качества кода;
• нарушения принципов проектирования безопасного ПО;
• выпуск неполной или некорректной документации.

Второй тип — уязвимости, вызванные дефектами конфигурирования и управления системой и ее окружением, а именно дефекты:
• конфигурации;
• настройки механизмов безопасности;
• настройки структуры и функционала;
• в виде закладок в настройках;
• совместимости версий;
• качества настроек;
• окружения;
• среды компиляции и выполнения программного кода;
• прикладного программного обеспечения;
• системного программного обеспечения (гипервизора, операционной системы, драйверов);
• аппаратного обеспечения.

Представленная классификация основана на дефектах кода и ошибок эксплуатации систем и пересекается с международными таксономиями CWE и Fortify. К достоинству такого подхода следует отнести учет реальных причин уязвимостей, соответствие международной практике классификации и возможность исключения многократного дублирования отдельных позиций, что свойственно академическим общетехническим классификациям.

Контрольные вопросы:
1. В чем заключается поиск уязвимостей;
2. Какие типы уязвимостей бывают;
3. Охарактеризуйте уязвимости, вызванные дефектами (ошибки, проблемы) проектирования и программирования;
4. Охарактеризуйте уязвимости, вызванные дефектами конфигурирования и управления системой.

ЛЕКЦИЯ 3

Дестабилизирующие факторы и угрозы надежности

План.
1. Контрольный опрос
2. Объекты уязвимости
3. Внешние дестабилизирующие факторы
4. Методы повышения надежности

Дестабилизирующие факторы и угрозы надежности.

Анализ надежности ПС базируется на модели взаимодействия следующих компонент:
— объектов уязвимости;
— дестабилизирующих факторов и угроз надежности;
— методов предотвращения угроз надежности;
— методов повышения надежности.

Объектами уязвимости, влияющими на надежность ПС являются:
— вычислительный процесс;

— объектный код программ;
— информация БД;
— информация выдаваемая потребителям.
На эти объекты воздействуют различные дестабилизирующие факторы, которые делятся на внутренние и внешние.
Внутренние источники угроз надежности функционирования сложных ПС присущи самим объектам уязвимости:
— ошибки проектирования при постановке задачи;
— алгоритмические ошибки разработки при спецификации функций ПС, при определении структуры и взаимодействия компонент комплексов программы, а также при использовании информации БД.
— ошибки программирования в текстах программ и ошибки в документации на ПС.
— недостаточное качество средств защиты.
Внешние дестабилизирующие факторы обусловлены средой, в которой функционируют объекты уязвимости. Ими являются:
— ошибки персонала при эксплуатации;
— искажение информации в каналах связи;
— сбои и отказы аппаратуры;
— изменение конфигурации аппаратуры информационной системы.
Полностью исключить все эти факторы невозможно. Поэтому необходимо разрабатывать средства и методы уменьшения их влияния на надежность ПС. Степень влияния всех внутренних дестабилизирующих факторов и некоторых внешних на надежность ПС в наибольшей степени определяется качеством технологий проектирования, разработки, сопровождения и документирования ПС.
Методы предотвращения угроз надежности:
— предотвращение ошибок проектирования;
— систематическое тестирование;
— обязательная сертификация.
Методы повышения надежности:
— временная избыточность;
— информационная избыточность;
— программная избыточность.
Последствия нарушения надежности:
— разрушение вычислительного процесса;
— разрушение информации БД;
— разрушение текста программы;
— разрушение информации для потребителей.
Методы обеспечения НПО: в современных автоматических технологиях создания ПО есть методы, позволяющие:
— создавать программные модули и функциональные компоненты высокого качества;
— предотвращать дефекты проектирования за счет эффективных технологий;
— обнаруживать и устранять различные ошибки и дефекты проектирования, разработки и сопровождения программы путем систем тестирования на всех этапах ЖЦ ПС.
— удостоверять достигнутого качества и надежности ПС в процессе их испытаний и сертификации;
— оперативно выявлять последствия дефектов программ и данных и восстанавливать надежное функционирование программ.

Комплексное применение этих методов позволяет значительно уменьшить влияние угроз. Т.е. уровень достигаемой надежности зависит от ресурсов, выделяемых на его достижение, и от качества технологии, используемой на всех этапах ЖЦ ПС. Предотвращение ошибок и улучшение технико-экономических показателей ПС обеспечивается применением современных технологий и САПР, которые объединяются понятием CASE-технологии и языки IV поколения. CASE-технологии – это высокопроизводительные ресурсосберегающие технологии создания комплексов программ. Они позволяют значительно снизить уровень системных, алгоритмических и программных ошибок. Для обнаружения и устранения ошибок проектирования все этапы разработки и сопровождения ПС д.б. поддержаны методами и средствами системного автоматизированного тестирования.Тестирование – это основной метод измерения качества, определения корректности и реальной надежности функционирования программ на любых этапах разработки. Результаты тестирования должны сравниваться с требованием технического задания или спецификации. Кроме вышеперечисленных методов предотвращения угроз надежности существуют оперативные методы повышения надежности: временная, информационная и программная избыточности.

Контрольные вопросы:

1. Назовите объекты уязвимости
2. Охарактеризуйте внешние дестабилизирующие факторы
3. Охарактеризуйте методы повышения надежности

ЛЕКЦИЯ 4

Методы тестирования ПО

План.

1. Контрольный опрос
2. Тестирование: ручное и автоматизированное
3. Различные типы тестов
4. Как автоматизировать тесты

Есть множество разных типов тестов, которые вы можете применить, чтобы убедиться, что изменения в вашем коде работают по сценарию. Не все типы тестирования идентичны, хотя здесь мы рассмотрим, насколько основные практики тестирования отличаются друг от друга.

Сначала надо понять различия между ручными и автоматизированными тестами. Ручное тестирование проводится непосредственно человеком, который нажимает на кнопочки в приложении или взаимодействует с программным обеспечением или API с необходимым инструментарием. Это достаточно затратно, так как это требует от тестировщика установки среды разработки и выполнения тестов вручную. Имеет место вероятность ошибки за счет человеческого фактора, например опечатки или пропуска шагов в тестовом сценарии.

Автоматизированные тесты, с другой стороны, производятся машиной, которая запускает тестовый сценарий, который был написан заранее. Такие тесты могут сильно варьироваться в зависимости от сложности, начиная от проверки одного единственного метода в классе до отработки последовательности сложных действий в UI, чтобы убедиться в правильности работы. Такой способ считается более надежным, однако его работоспособность все еще зависит от того насколько скрипт для тестирования был хорошо написан.

Автоматизированные тесты – это ключевой компонент непрерывной интеграции (Continuous Integration) и непрерывной доставки (continuous delivery), а также хороший способ масштабировать ваш QA процесс во время добавления нового функционала для

вашего приложения. Однако в ручном тестировании все равно есть своя ценность. Поэтому в статье мы обязательно поговорим об исследовательском тестировании (exploratory testing).

Различные типы тестов

Модульные тесты

Модульные тесты считаются низкоуровневыми, близкими к исходному коду вашего приложения. Они нацелены на тестирование отдельных методов и функций внутри классов, тестирование компонентов и модулей, используемых вашей программой. Модульные тесты в целом не требуют особых затрат на автоматизацию и могут отрабатывать крайне быстро, если задействовать сервер непрерывной интеграции (continuous integration server).

Интеграционные тесты

Интеграционные тесты проверяют хорошо ли работают вместе сервисы и модули, используемые вашим приложением. Например, они могут тестировать интеграцию с базой данных или удостоверяться, что микросервисы правильно взаимодействуют друг с другом. Эти тесты запускаются с бОльшими затратами, поскольку им необходимо, чтобы много частей приложения работало одновременно.

Функциональные тесты

Функциональные тесты основываются на требованиях бизнеса к приложению. Они лишь проверяют выходные данные после произведенного действия и не проверяют промежуточные состояния системы во время воспроизведения действия.

Иногда между интеграционными тестами и функциональными тестами возникают противоречия, т.к. они оба запрашивают множество компонентов, взаимодействующих друг с другом. Разница состоит в том, что интеграционные тесты могут просто удостовериться, что доступ к базе данных имеется, тогда как функциональный тест захочет получить из базы данных определенное значение, чтобы проверить одно из требований к конечному продукту.

Сквозные тесты (End-to-end tests)

Сквозное тестирование имитирует поведение пользователя при взаимодействии с программным обеспечением. Он проверяет насколько точно различные пользователи следуют предполагаемому сценарию работы приложения и могут быть достаточно простыми, допустим, выглядеть как загрузка веб-страницы или вход на сайт или в более сложном случае – подтверждение e-mail адреса, онлайн платежи и т.д.

Сквозные тесты крайне полезные, но производить их затратно, а еще их может быть сложно автоматизировать. Рекомендуется проводить несколько сквозных тестов, но все же полагаться больше на низкоуровневое тестирование (модульные и интеграционные тесты), чтобы иметь возможность быстро распознать серьезные изменения.

Приемочное тестирование

Приемочные тесты – это формальные тесты, которые проводятся, чтобы удостовериться, что система отвечает бизнес-запросам. Они требуют, чтобы приложение запускалось и работало, и имитируют действия пользователя. Приемочное тестирование может пойти дальше и измерить производительность системы и отклонить последние изменения, если конечные цели разработки не были достигнуты.

Тесты производительности

Тесты на производительности проверяют поведение системы, когда она находится под существенной нагрузкой. Эти тесты нефункциональные и могут принимать разную форму, чтобы проверить надежность, стабильность и доступность платформы. Например, это может быть наблюдение за временем отклика при выполнении большого количества запросов или наблюдение за тем, как система ведет себя при взаимодействии с большими данными.

Тесты производительности по своей природе проводить достаточно затратно, но они могут помочь вам понять, какие внешние факторы могут уронить вашу систему.

Дымовое тестирование (Smoke testing)

Дымовые тесты – это базовые тесты, которые проверяют базовый функционал приложения. Они отрабатывают достаточно быстро и их цель дать понять, что основные функции системы работают как надо и не более того. Такое тестирование направлено на выявление явных ошибок. Дымовые тесты могут оказаться полезными сразу после сборки нового билда для проверки на то, можете ли вы запустить более дорогостоящие тесты, или сразу после развёртывания, чтобы убедиться, что приложение работает нормально в новой среде.

Как автоматизировать тесты

Тестировщик может проводить все тесты, указанные выше, вручную, но это будет крайне затратно и непродуктивно. Поскольку люди имеют ограниченную возможность производить большое количество действий с повторениями при этом все еще проводя тестирование надежно. Однако машина может с легкостью воспроизводить эти же действия и проверить, допустим, что комбинация логин/пароль будет работать и в сотый раз без каких-либо нареканий. Для автоматизации тестирования, вам для начала придется написать их на каком-то из языков программирования с использованием фреймворка для тестирования, который подойдет для вашего приложения. PHPUnit , Mocha, RSpec – это примеры фреймворков для тестирования, которые вы можете использовать для PHP, Javascript и Ruby, соответственно. В них есть множество возможностей для каждого языка, поэтому вам стоит немного позаниматься исследованием самостоятельно и проконсультироваться с сообществами разработчиков, чтобы понять, какой фреймворк подойдет вам лучше всего. Если ваши тесты могут запускаться с помощью скриптов из терминала, вы можете автоматизировать их, использовав сервер непрерывной интеграции по типу Bamboo или же облачного сервера Bitbucket Pipelines. Эти инструменты будут мониторить ваши репозитории и исполнять наборы тестов, как только новые изменения будут запушены в основной репозиторий. Если вы новичок в вопросах тестирования, обратитесь к нашему руководству по непрерывной интеграции, чтобы создать свой первый набор тестов.

Исследовательское тестирован

Чем больше функций и улучшений добавляется в ваш код, тем больше возрастает потребность в тестировании, поскольку на каждом этапе вам необходимо убеждаться, что система работает корректно. Также это понадобится каждый раз, когда вы исправляете баг, поскольку было бы не лишним убедиться, что он не вернется снова после нескольких релизов. Автоматизация – это ключ к тому, чтобы это стало возможным; написание тестов рано или поздно станет частью вашей практики разработчика.

Вопрос заключается в том, надо ли вообще в таком случае проводить ручное тестирование? Короткий ответ – да, и оно должно быть сфокусировано на том, что называется «исследовательское тестирование» (exploratory testing), которое помогает выявить неочевидные ошибки.

Сессия исследовательского тестирования не должна превышать двух часов и должна иметь четко ограниченную область действия, чтобы помочь тестировщикам сосредоточиться на определенной области программного обеспечения. После информирования всех тестировщиков о границах проведения тестирования, на их усмотрения остаются действия, которые они будут предпринимать, чтобы проверить, как поведет себя система. Такое тестирование является дорогостоящим по своей природе, но очень полезно для выявления проблем с пользовательским интерфейсом или проверки работоспособности сложных рабочих процессов для пользователей. Такое тестирование важно проводить всякий раз, когда в приложение добавляется кардинально новая функция, чтобы понять, как она поведет себя в пограничных условиях.

Контрольные вопросы:

1. Чем отличается тестирование: ручное и автоматизированное?
2. Назовите отличия различных типы тестов?
3. Как автоматизировать тесты?

ЛЕКЦИЯ 5

Методы предотвращения угроз надежности

План.

1. Контрольный опрос
2. Автоматизация программирования
3. Введение избыточности
4. Типовые структуры данных

Большая трудоемкость и стоимость создания ПО систем управления заставляет уделять особое внимание обеспечению его надежной работы. Высокая стоимость ПО во многом обусловлена его низкой надежностью. Следует иметь в виду, что в настоящее время не разработаны методы проектирования программ с гарантированным отсутствием ошибок. Это объясняется рядом причин, и в том числе тем, что программное обеспечение значительно сложнее аппаратуры; характеризуется большей, чем аппаратура, зависимостью от применения; состоит из примитивных составляющих — машинных кодов, из которых синтезируются огромные программные структуры.

Однако накопленный опыт позволяет сформулировать принципы и метода проектирования программ, обеспечивающих их надежную, устойчивую работу, основные из которых показаны на рис. 5.4.1.

По образному выражению *Ф.П. Брукса* «программа — это сообщение, передаваемое человеком машине». Чтобы сделать это сообщение «понятным бессловесной машине», требуется предельная формализация как описания алгоритмов, так и программ.

1. Формализация позволяет в существенной степени исключить, а при наличии ошибок — облегчить их выявление и локализацию.

2. Мощным средством повышения надежности у ПО САПР является *автоматизация программирования, заключающаяся* в использовании ЭВМ для составления машинных программ, т. е. программ, выполненных на языке ЭВМ по исходной программе, составленной на языке высокого уровня.

Системы автоматизированного программирования обеспечивают повышение производительности и облегчение труда программистов. Эти системы применяют для программирования языки высокого уровня, более близкие к естественному языку специалистов, и тем самым освобождают программиста от необходимости составлять программу на языке машинных команд. У нас в стране создан ряд систем автоматизированного программирования для ЭВМ. Эти системы обеспечивают

автоматизацию создания программ и документов с минимальным расходом ручного труда как при основных, так и при вспомогательных работах, в том числе: .трансляцию текстов программ в машинные коды с семантическим и синтаксическим контролем, автономную отладку программ, диалоговое общение программистов со средствами автоматизации, выпуск эксплуатационных и технологических документов, а также их корректировку.

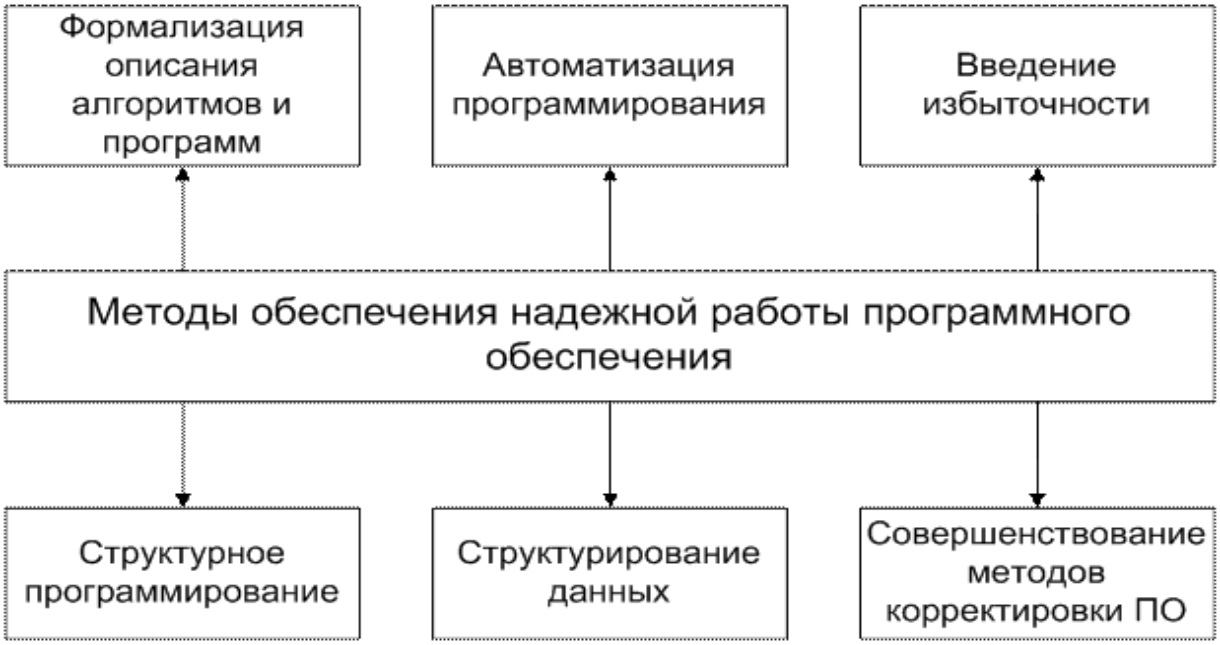


Рис. 5. 4. 1. Классификация методов обеспечения надежной работы программного обеспечения

Эффективным методом повышения надежности ПО является введение избыточности, включающей программную, информационную и временную избыточность.

3. *Программная избыточность* применяет в комплексах ПО несколько вариантов программ, различающихся алгоритмами решения задачи или программной реализации одного и того же алгоритма.

3. 1. Временная избыточность использует часть производительности ЭВМ для контроля исполнения программ и восстановления вычислительного процесса. Как следствие при проектировании ПО необходимо предусматривать резерв производительности ЭВМ, обычно резерв составляет 5-10%.

3. 2. Информационная избыточность заключается в резервировании (дублировании) исходных и промежуточных данных, что обеспечивает как обнаружение искажения данных, так и устранение ошибок.

4. Структурное программирование позволяет облегчить проектирование и повысить надежность сложных программных комплексов. Структурное программирование развилось на основе технологии процедурного и модульного программирования, а также блочно-иерархического подхода; представляет собой технологию программирования, построенную на совокупности определенных принципов и правил, среди которых прежде всего можно выделить модульность структуры, иерархию модулей, нисходящее проектирование.

5. *Структурирование данных* позволяет уменьшить сложность комплекса программ и снизить вероятность появления ошибок из-за их неправильного использования. Совокупность данных можно разделить на два иерархических уровня: простые переменные и массивы. Простые переменные представляют собой минимальный компонент данных, имеющий имя и описание. Массивы образуются из нескольких

простых переменных по определенным правилам объединения и имеют собственное имя, описание и структуру.

С целью упорядочения создан ряд типовых структур данных, применение которых зависит от назначения и метода использования переменных. Структура массивов определяется на основе компромисса между объемом памяти для хранения массива и затратами производительности ЭВМ, необходимыми для выборочного поиска и обращения к данным в массиве. Так, простые структуры массивов экономны по затратам производительности ЭВМ для взаимодействия с данными, но требуют большого объема памяти. Для повышения надежности ПО целесообразно использовать простейшие структуры массивов данных,

Следует иметь в виду, что, как правило, устранение ошибок, обнаруженных в ПО, приводит к внесению новых ошибок, трудно обнаруживаемых, так как их последствия не проявляются на тестах. При проведении корректировок в комплексах программ необходим дополнительный анализ возможных последствий внесенных изменений. Любая корректировка ПО может быть сведена к трем типовым операциям: исключение части или всей подпрограммы; вставка компонентов или новой целой подпрограммы на имеющееся свободное место; замена части или всей подпрограммы в пределах освобождающегося свободного места или с расширением программы и использованием дополнительной памяти.

Контрольные вопросы:

1. *Что тактое автоматизация программирования?*
2. *Что тактое* введение избыточности?
3. *Что тактое* типовые структуры данных?

ЛЕКЦИЯ 6

Математические модели описания статистических характеристик ошибок в программах.

План.

1. Контрольный опрос
2. Предназначение математических моделей;
3. Первая группа допущений;
4. Вторая группа допущений;
5. Третья группа допущений.

Детальный анализ проявления ошибок показывает их глубокую связь с методами структурного построения программ, типом языка программирования, степенью автоматизации технологии проектирования и многими другими факторами. Статистические характеристики различных типов ошибок трудно описать математическими моделями, и более доступны для математического описания обобщенные характеристики ошибок в комплексе программ. Путем анализа и обобщения экспериментальных данных реальных разработок предложено несколько математических моделей, описывающих основные закономерности изменения *суммарного количества вторичных ошибок* в программах. Модели имеют вероятностный характер, и достоверность прогнозов в значительной степени зависит от точности исходных данных и глубины прогнозирования по времени. Эти математические модели предназначены для оценки:

1. надежности функционирования комплекса программ в процессе отладки, испытаний и эксплуатации;
2. числа ошибок, оставшихся невыявленными в анализируемых программах;
3. времени, требующегося для обнаружения следующей ошибки в функционирующей программе;

4. времени, необходимого для выявления всех ошибок с заданной вероятностью.

Точное определение полного числа невыявленных ошибок в комплексе программ *прямыми методами измерения невозможно.* Однако имеются пути для приближенной статистической оценки их полного числа или вероятности ошибки в каждой команде программы. Такие оценки базируются на построении математических Моделей в предположении о жесткой корреляции между общим количеством и проявлениями ошибок в комплексе программ после его отладки в течение времени Т, т. е. между:

- суммарным количеством первичных ошибок в комплексе программ (n_0) или вероятностью ошибки в каждой команде программы (p_0);
- количеством ошибок, выявляемых в единицу времени в процессе тестирования и отладки при постоянных усилиях на ее проведение *{dn/dt);*
- интенсивностью искажений результатов в единицу времени (λ) на выходе комплекса программ вследствие невыявленных первичных ошибок при функционировании программ.

Наиболее доступно для измерения количество вторичных ошибок в программе, выявляемых в единицу времени в процессе отладки. Возможна также непосредственная регистрация отказов и наиболее крупных искажений результатов, выявляемых средствами оперативного контроля в процессе функционирования программ. Все три показателя связаны некоторыми коэффициентами пропорциональности, значения которых зависят, в частности, от интервала времени, на котором производится сопоставление, от быстродействия ЭВМ, от эффективности средств автоматизации отладки и от некоторых других параметров. При фиксированных условиях разработки и функционирования конкретного комплекса программ эти коэффициенты имеют вполне определенное значения. Для другой подобной системы коэффициенты могут несколько измениться, однако оценки, полученные для нескольких конкретных систем, позволяют прогнозировать эти характеристики, а следовательно, и соответствующие показатели надежности ПС в зависимости от длительности отладки и ряда других факторов.

Описаны несколько математических моделей, основой которых являются различные гипотезы о характере проявления вторичных ошибок в программах. Эти гипотезы в той или иной степени апробированы при обработке данных реальных разработок, и их можно разделить на три группы. В **первую группу** входят очевидные допущения, статистическая проверка которых невозможна и нецелесообразна. Вторую группу составляют допущения, определяющие специфические характеристики модели и требующие статистической проверки и обоснования на базе экспериментальных исследований. В третью группу включены второстепенные допущения, расширяющие и уточняющие возможности применения модели и частично доступные экспериментальной проверке.

Первая группа допущений включает предположение о *наблюдаемости искажений данных,* программ или вычислительного процесса, обусловленных первичными ошибками в программах. Первичная ошибка, являющаяся причиной искажения результатов, либо фиксируется и исправляется после завершения этапа тестирования, либо вообще не обнаруживается, так как проявление ее последствий незначительно.

Предполагается, что множество тестов более или менее *равномерно покрывает* все множество реальных исходных данных и отсутствуют априорные сведения для искусственного повышения интенсивности ошибок при некоторых тестах. Тем самым состав тестов представляется случайным относительно области изменения входных данных программы и содержащихся в ней необнаруженных первичных ошибок.

Наличие большого числа разнообразных данных, необходимых для исполнения программ, и практически некоррелированное их изменение приводит к *внешне случайному выбору маршрута,* по которому исполняется программа в каждом конкретном случае. В

результате интенсивность проявления ошибок при реальном функционировании программ зависит от среднего быстродействия ЭВМ и практически не зависит от конкретного распределения типов команд на маршрутах обработки данных между ошибками.

Коллектив специалистов, их квалификация и загруженность предполагаются *постоянными* на интервале проектирования и исследования характеристик ошибок. Также постоянным считается доступное машинное время для проведения проверок программ.

Вторая группа допущений при построении математических моделей ошибок является основной и проверена интегрально по обобщенным характеристикам частости обнаружения ошибок и дифференцирование путем анализа правомерности каждого допущения. Ниже рассмотрены допущения при построении экспоненциальной модели.

Интервалы времени между обнаруживаемыми искажениями результатов предполагаются *статистически независимыми.* Время измеряется по фактической наработке длительностей исполнения программ τ без учета дополнительных затрат календарного времени на локализацию, диагностику и исправление ошибок.

Предполагается, что *интенсивность проявления ошибок остается постоянной,* пока не произведено исправление первичной ошибки или не изменена программа по другой причине. Если каждая обнаруженная ошибка исправляется, то значения интервалов времени между их проявлениями изменяются по экспоненциальному закону. Интегральная проверка распределения интервалов времени между обнаружениями ошибок показала, что оно достаточно хорошо аппроксимируется экспонентой.

Логично предположить, что *интенсивность обнаружения ошибок пропорциональна суммарному числу первичных ошибок,* имеющихся в данный момент в комплексе программ. Это допущение подтверждено расчетом значений суммарного числа ошибок для хорошо отлаженных и переданных в эксплуатацию комплекса программ на ряд предыдущих моментов времени, когда проводилась отладка.

Каждая обнаруженная ошибка подлежит исправлению, поэтому предполагается, что *частота исправления ошибок пропорциональна частоте их обнаружения.* Однако некоторые исправления, в свою очередь, содержат ошибки. Кроме того, некоторые ошибки являются связанными, и при обнаружении проявления одной ошибки следует исправление нескольких первичных ошибок. Из-за этого частота обнаружения ошибок и частота их исправления не равны, а должны быть связаны некоторым коэффициентом пропорциональности. Коэффициенты корреляции для исследованных комплексов программ довольно высокие — от 0,52 до 0,82 при среднем значении около 0,68, т. е. достаточно хорошо подтверждают гипотезу.

Третья группа допущений детализирует использование ресурсов на корректировку программ и повышение **их** качества.

Приведенные предположения позволяют построить экспоненциальную математическую модель распределения моментов обнаружения ошибок в программах и установить связь между интенсивностью обнаружения ошибок при отладке $dn/d\tau$, интенсивностью проявления ошибок при нормальном функционировании программ λ и числом первичных ошибок n. Предположим, что в начале отладки комплекса программ при $\tau=0$ в нем содержалось N_0 первичных ошибок. После отладки в течение времени осталось n_0 первичных ошибок и устранено n ошибок ($n_0 + n = N_0$). Время τ соответствует длительности исполнения программы на ЭВМ для обнаружения ошибок и не учитывает время, необходимое для анализа результатов и проведения корректировок. Календарное время $\tau_к$ отладочных и испытательных работ с реальным комплексом программ значительно больше, так как после тестирования программ, на которое затрачивается машинное время т, необходимо время на анализ результатов, на обнаружение и локализацию ошибок, а также на их устранение.

При постоянных усилиях на отладку интенсивность обнаружения искажений вычислительного процесса, программ или данных вследствие еще не выявленных ошибок

пропорциональна количеству оставшихся первичных ошибок в комплексе программ. Предположение о сильной корреляции между значениями n_o и $dn/d\tau$ проверено анализом реальных характеристик процесса обнаружения ошибок. Тогда

$$\frac{dn}{d\tau} = K'\lambda = Kn_\theta = K(N_0 - n), \qquad (4.15)$$

(11.1)_

где коэффициенты К и К' учитывают масштаб времени, используемый для описания процесса обнаружения ошибок, быстродействие ЭВМ и другие параметры. Значение коэффициента К' можно определить как изменение темпа проявления искажений при переходе от функционирования программ на специальных тестах к функционированию на типовых исходных данных. В начале отладки это различие может быть значительным, однако при завершении отладки и при испытаниях тестовые данные практически совпадают с исходными данными при нормальной эксплуатации. Поэтому ниже К' полагается равным единице (К'=1).

Таким образом, интенсивность обнаружения ошибок в программе и абсолютное число устраненных первичных ошибок связываются уравнением

$$\frac{dn}{d\tau} + Kn = KN_0, \qquad (4.16)$$

(11.2)

Так как выше предполагалось, что в начале отладки при $\tau = 0$ отсутствуют обнаруженные ошибки, то решение уравнения (11.2) имеет вид

$$n = N_0[1 - \exp(-K\tau)]. \qquad (4.17)$$

(11.3)

пропорционально интенсивности их обнаружения $dn/d\tau$ с точностью до коэффициента К.

Наработка между проявлениями ошибок, которые, рассматриваются как обнаруживаемые искажения программ, данных или вычислительного процесса, равны величине, обратной интенсивности обнаружения ошибок:

$$T = 1/\frac{dn}{d\tau} = \frac{1}{KN_0}\exp(K\tau). \qquad (4.19)$$

(11.4)

Если учесть, что до начала отладки в комплексе программ содержалось N_0 первичных ошибок и этому соответствовала наработка T_0 , то функцию наработки между проявлениями ошибок от длительно-

$$T = T_0 \exp[\tau/(N_0 T_0)]. \qquad (4.20)$$

(11.5)

Если известны все моменты обнаружения ошибок t_i и каждый раз в эти моменты обнаруживается и достоверно устраняется одна первичная ошибка, то, используя метод максимального правдоподобия, получим уравнение для определения значения начального количества первичных ошибок N_0

$$\sum_{i=1}^{n} \frac{1}{N_0 - (i-1)} = \frac{n}{N_0 \sum_{i=1}^{n} t_i - \sum_{i=1}^{n} (i-1)t_i}, \qquad (4.21)$$

(11.6)

и выражение для расчета коэффициента пропорциональности

$$K = n/\left[N_0 \sum_{i=1}^{n} t_i - \sum_{i=1}^{n} (i-1)t_i\right]. \qquad (4.22)$$

(11.7)

Необходимые для расчетов К и N_0 экспериментальные значения t_i определяются в процессе отладки данного или аналогичных комплексов программ, созданных тем же коллективом разработчиков и при такой же технологии. В результате можно рассчитать

число оставшихся в программе первичных ошибок и среднюю наработку ***T*** до обнаружения следующей ошибки.

В процессе отладки и испытания программ для повышения наработки между проявлениями ошибок от величины T_1 до значения T_2 необходимо обнаружить и устранить Δn ошибок. Эту величину можно определить, выразив число обнаруживаемых ошибок через длительность наработки на проявление ошибок, для чего подставим (11.4) в {11.2). Тогда

$$\Delta n = N_0 T_0 \left[1/T_1 - 1/T_2 \right]. \qquad (4.23)$$

(11.8)

Аналогичными несложными преобразованиями можно получить затраты времени на проведение отладки $\Delta\tau$, которые позволяют устранить Δn ошибок и соответственно повысить наработку между очередными обнаружениями ошибок от значения T_1 до T_2:

$$\Delta\tau = \frac{N_0 T_0}{K} \ln(T_2/T_1). \qquad (4.24)$$

(11.9)

Следует подчеркнуть статистический характер приведенных соотношений. Неравномерность выбора маршрутов исполнения программы при нормальной эксплуатации, разное влияние конкретных типов ошибок в программах на проявление их при функционировании, а также сравнительно небольшие значения n и Δn, особенно на заключительных этапах отладки, приводят к тому, что флюктуации интервалов времени между обнаружением ошибок $\Delta\tau$ могут быть весьма значительными

Контрольные вопросы:

1. Предназначение математических моделей?
2. Что такое первая группа допущений?
3. Что такое вторая группа допущений?
4. Что такое третья группа допущений?

ЛЕКЦИЯ 7

Первичные ошибки, вторичные ошибки и их проявления

План.

1. Контрольный опрос
2. Алгоритмы выявления ошибок ПО
3. Процесс модернизации ИС
4. Анализ последствий применения обновлений

Рассмотрим классификацию ошибок по месту их возникновения, которая рассмотрена в книге С. Канера «Тестирование программного обеспечения». Фундаментальные концепции менеджмента бизнес-приложений. . Главным критерием программы должно быть ее качество, которое трактуется как отсутствие в ней недостатков, а также сбоев и явных ошибок. Недостатки программы зависят от субъективной оценкой ее качества потенциальным пользователем. При этом авторы скептически относятся к спецификации и утверждают, что даже при ее наличии, выявленные на конечном этапе недостатки говорят о ее низком качестве. При таком подходе преодоление недостатков программы, особенно на заключительном этапе проектирования, может приводить к снижению надежности. Очевидно, что для разработки ответственного и безопасного программного обеспечения (ПО) такой подход не годится, однако проблемы наличия ошибок в спецификациях, субъективного оценивания пользователем качества программы существуют и не могут быть проигнорированы. Должна быть разработана система некоторых ограничений, которая бы учитывала эти факторы при разработке и сертификации такого рода ПО. Для обычных программ все

проблемы, связанные с субъективным оцениванием их качества и наличием ошибок, скорее всего неизбежны.

В краткой классификации выделяются следующие ошибки.

- Ошибки пользовательского интерфейса.
- Ошибки вычислений.
- Ошибки управления потоком.
- Ошибки передачи или интерпретации данных.
- Перегрузки.
- Контроль версий.
- Ошибка выявлена и забыта.
- Ошибки тестирования.

1. Ошибки пользовательского интерфейса.

Многие из них субъективны, т.к. часто они являются скорее неудобствами, чем «чистыми» логическими ошибками. Однако они могут провоцировать ошибки пользователя программы или же замедлять время его работы до неприемлемой величины. В результате чего мы будем иметь ошибки информационной системы (ИС) в целом. Основным источником таких ошибок является сложный компромисс между функциональностью программы и простотой обучения и работы пользователя с этой программой. Проблему надо начинать решать при проектировании системы на уровне ее декомпозиции на отдельные модули, исходя из того, что вряд ли удастся спроектировать простой и удобный пользовательский интерфейс для модуля, перегруженного различными функциями. Кроме того, необходимо учитывать рекомендации по проектированию пользовательских интерфейсов. На этапе тестирования ПО полезно предусмотреть встроенные средства тестирования, которые бы запоминали последовательности действий пользователя, время совершения отдельных операций, расстояния перемещения курсора мыши. Кроме этого возможно применение гораздо более сложных средств психо-физического тестирования на этапе тестирования интерфейса пользователя, которые позволят оценить скорость реакции пользователя, частоту этих реакций, утомляемость и т.п. Необходимо отметить, что такие ошибки очень критичны с точки зрения коммерческого успеха разрабатываемого ПО, т.к. они будут в первую очередь оцениваться потенциальным заказчиком.

2. Ошибки вычислений.

Выделяют следующие причины возникновения таких ошибок:

- неверная логика (может быть следствием, как ошибок проектирования, так и кодирования);
- неправильно выполняются арифметические операции (как правило - это ошибки кодирования);
- неточные вычисления (могут быть следствием, как ошибок проектирования, так и кодирования). Очень сложная тема, надо выработать свое отношение к ней с точки зрения разработки безопасного ПО.

Выделяются подпункты: устаревшие константы; ошибки вычислений; неверно расставленные скобки; неправильный порядок операторов; неверно работает базовая функция; переполнение и потеря значащих разрядов; ошибки отсечения и округления; путаница с представлением данных; неправильное преобразование данных из одного формата в другой; неверная

3. Ошибки управления потоком.

В этот раздел относится все то, что связано с последовательностью и обстоятельствами выполнения операторов программы.

Выделяются подпункты:

- очевидно неверное поведение программы;
- переход по GOTO;
- логика, основанная на определении вызывающей подпрограммы;

- использование таблиц переходов;

- выполнение данных (вместо команд). Ситуация возможна из-за ошибок работы с указателями, отсутствия проверок границ массивов, ошибок перехода, вызванных, например, ошибкой в таблице адресов перехода, ошибок сегментирования памяти.

4. Ошибки обработки или интерпретации данных.

Выделяются подпункты:

- проблемы при передаче данных между подпрограммами (сюда включены несколько видов ошибок: параметры указаны не в том порядке или пропущены, несоответствие типов данных, псевдонимы и различная интерпретация содержимого одной и той же области памяти, неправильная интерпретация данных, неадекватная информация об ошибке, перед аварийным выходом из подпрограммы не восстановлено правильное состояние данных, устаревшие копии данных, связанные переменные не синхронизированы, локальная установка глобальных данных (имеется в виду путаница локальных и глобальных переменных), глобальное использование локальных переменных, неверная маска битового поля, неверное значение из таблицы);

- границы расположения данных (сюда включены несколько видов ошибок: не обозначен конец нуль-терминированной строки, неожиданный конец строки, запись/чтение за границами структуры данных или ее элемента, чтение за пределами буфера сообщения, чтение за пределами буфера сообщения, дополнение переменных до полного слова, переполнение и выход за нижнюю границу стека данных, затирание кода или данных другого процесса);

- проблемы с обменом сообщений (сюда включены несколько видов ошибок: отправка сообщения не тому процессу или не в тот порт, ошибка распознавания полученного сообщения, недостающие или несинхронизированные сообщения, сообщение передано только N процессам из N+1, порча данных, хранящихся на внешнем устройстве, потеря изменений, не сохранены введенные данные, объем данных слишком велик для процесса-получателя, неудачная попытка отмены записи данных).

5. Повышенные нагрузки.

При повышенных нагрузках или нехватке ресурсов могут возникнуть дополнительные ошибки. Выделяются подпункты: требуемый ресурс недоступен; не освобожден ресурс; нет сигнала об освобождении устройства; старый файл не удален с накопителя; системе не возвращена неиспользуемая память; лишние затраты компьютерного времени; нет свободного блока памяти достаточного размера; недостаточный размер буфера ввода или очереди; не очищен элемент очереди, буфера или стека; потерянные сообщения; снижение производительности; повышение вероятности ситуационных гонок; при повышенной нагрузке объем необязательных данных не сокращается; не распознается сокращенный вывод другого процесса при повышенной загрузке; не приостанавливаются задания с низким приоритетом.

В этом разделе хотелось бы обратить внимание на следующее:

1) Часть ошибок из этого раздела могут проявляться и при не очень высоких нагрузках, но, возможно, они будут проявляться реже и через более длительные интервалы времени;

2) Многие ошибки из 2-х предыдущих разделов уже в своей формулировке носят вероятностный характер, поэтому следует предположить возможность использования вероятностных моделей и методов для их выявления.

6. Контроль версий и идентификаторов.

Выделяются подпункты: таинственным образом появляются старые ошибки; обновление не всех копий данных или программных файлов; отсутствие заголовка; отсутствие номера версии; неверный номер версии в заголовке экрана; отсутствующая или неверная информация об авторских правах; программа, скомпилированная из архивной копии, не соответствует проданному варианту; готовые диски содержат неверный код или данные.

7. Ошибки тестирования.

Являются ошибками сотрудников группы тестирования, а не программы. Выделяются подпункты:

- пропущенные ошибки в программе;

- не замечена проблема (отмечаются следующие причины этого: тестировщик не знает, каким должен быть правильный результат, ошибка затерялась в большом объеме выходных данных, тестировщик не ожидал такого результата теста, тестировщик устал и невнимателен, ему скучно, механизм выполнения теста настолько сложен, что тестировщик уделяет ему больше внимания, чем результатам);

- пропуск ошибок на экране;

- не документирована проблема (отмечаются следующие причины этого: тестировщик неаккуратно ведет записи, тестировщик не уверен в том, что данные действия программы являются ошибочными, ошибка показалась слишком незначительной, тестировщик считает, что ошибку не будет исправлена, тестировщика просили не документировать больше подобные ошибки).

8. Ошибка выявлена и забыта.

Описываются ошибки использования результатов тестирования. По-моему, раздел следует объединить с предыдущим. Выделяются подпункты: не составлен итоговый отчет; серьезная проблема не документирована повторно; не проверено исправление; перед выпуском продукта не проанализирован список нерешенных проблем.

Необходимо заметить, что изложенные в 2-х последних разделах ошибки тестирования требуют для устранения средств автоматизации тестирования и составления отчетов. В идеальном случае, эти средства должны быть проинтегрированы со средствами и технологиями проектирования ПО. Они должны стать важными инструментальными средствами создания высококачественного ПО. При разработке средств автоматизированного тестирования следует избегать ошибок, которые присущи любому ПО, поэтому нужно потребовать, чтобы такие средства обладали более высокими характеристиками надежности, чем проверяемое с их помощью ПО.

Основные пути борьбы с ошибками

Учитывая рассмотренные особенности действий человека при переводе можно указать следующие пути борьбы с ошибками:

· сужение пространства перебора (упрощение создаваемых систем),

· обеспечение требуемого уровня подготовки разработчика (это функции менеджеров коллектива разработчиков),

· обеспечение однозначности интерпретации представления информации,

· контроль правильности перевода (включая и контроль однозначности

Контрольные вопросы:

1. Что такое тестирование ИС;
2. Опишите процесс доработки ИС;
3. Что такое первичные ошибки ?;
4. Что такое вторичные ошибки ?;

ЛЕКЦИЯ 8

Целесообразность и риски разработки адаптации

План.

1. Контрольный опрос
2. Типы рисков
3. Классификация рисков

Риски при разработке программного обеспечения
Риски при разработке программного обеспечения: как с ними справляться?

Любой проект по разработке мобильных приложений связан с определенными рисками той или иной степени. Риски могут различаться в зависимости от характера проекта, но в целом их можно разделить на 5 категорий:

1. Бюджет: риск превышения выделенного на проект бюджета. Пожалуй, это самая распространенная ошибка при разработке программного обеспечения, которая влечет за собой другие ошибки.
2. Кадры: риск потери или нехватки членов проектной команды. Даже если эта проблема длится недолго, она может привести к ошибкам и задержке сроков.

3. Знания: команда может иметь лишь узконаправленные знания, или неверно передавать между собой информацию. В этом случае членам команды приходится переучиваться, что приводит к дополнительным тратам сил, времени и ресурсов.

4. Продуктивность: этот риск чаще всего угрожает долгосрочным проектам. Большой запас времени порой приводит к медленной и непродуктивной работе.

5. Время: при разработке программного обеспечения очень распространены задержки релизов продукции, что является результатом неправильного планирования, крайне сжатых сроков и неспособности разработчиков адаптироваться к меняющимся требованиям по отношению к продукту.

Методы управления рисками при динамичной разработке ПО

Метод динамичной разработки ПО учитывает большинство вышеуказанных рисков. Тем не менее, даже в среде гибкой разработки от них не застрахован никто: риски возникают из-за ошибок проектной команды, неверного планирования, сбоев в рабочем процессе и неожиданных изменений в процессе работы над продуктом. Давайте рассмотрим способы управления этими рисками.

1. Бюджет

Решение: планирование методом «набегающей волны»

Цели и задачи по разработке ПО могут меняться в процессе его создания, и для того, чтобы продукт оставался жизнеспособным, необходимо уметь подстраиваться под возможные изменения. Именно для этого существует планирование методом «набегающей волны». Команды принимают решения по продукту по мере продвижения работы, вместо того, чтобы разрабатывать подробнейший план действий на самом старте проекта. Действенные решения, принятые на основе новых знаний о развитии продукта, снижают вероятность рисков, связанных с бюджетом, так как команде не нужно тратить время и ресурсы на повторное планирование.
Тем не менее, в начале работы очень важно составить бюджетный план, в котором будут учтены все возможные затраты на проект. Многие компании недооценивают стоимость разработки функционального ПО и допускают ряд ошибок в расчете бюджета.

2. Кадры/Знания

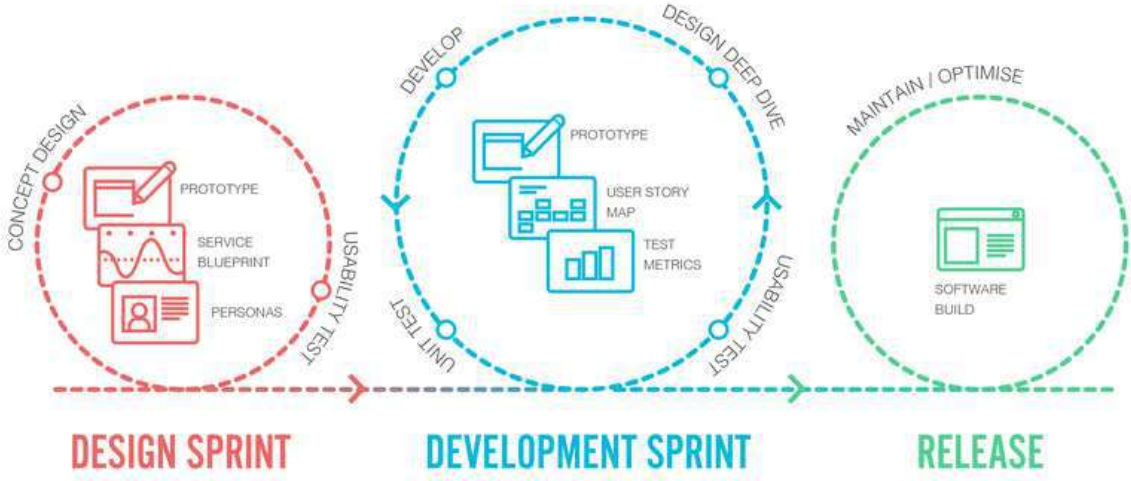
Решение: разбить разработчиков на небольшие группы

Идеальная команда для разработки ПО – несколько групп по 10-12 человек, которые совместно планируют проект, делятся друг с другом опытом, выполняют проверку кода и сообща работают над задачей от начала до конца. Разработчики должны обладать максимальным объемом знаний, что помогает решать проблемы, связанные как с персоналом, так и с риском нехватки нужных знаний. Также у команды должна быть возможность беспрепятственно выполнять работу в том случае, если один из ее членов временно отсутствует или покинул команду.

3. Продуктивность

Решение: разработка на основе спринтов

Спринты – это краткосрочные этапы разработки с целью создания демо-версии продукта в заданные сроки (1-2 недели). Они служат для обозначения правильных целей и задач для проектных команд и позволяют увидеть промежуточные результаты работы. Это придает разработчикам уверенности в правильности создания продукта, и позволяет придерживаться нужной скорости выполнения задачи.



4. Время

Решение: правильная организация процесса

Ошибки, связанные с нехваткой времени, могут возникнуть из-за неправильного планирования рабочего процесса или излишней уверенности в его успешном протекании.

Процесс должен быть гибким, чтобы разработчики могли быстро адаптироваться к меняющимся требованиям, имели возможность быстро предоставлять исправленный продукт заказчику, и могли точно определять количество времени, необходимое для выполнения той или иной задачи.

Все эти методы могут помочь командам разработчиков лучше управлять временем и свести риски при создании ПО к минимуму.

Контрольные вопросы:

1. Назовите типы рисков ?
2. Классификация рисков?

3. Охарактеризуйте риск «бюджет»;
4. Охарактеризуйте риск «время»;
5. Охарактеризуйте риск «бкадры/знания»;

ЛЕКЦИЯ 9

Вредоносные программы: классификация, методы обнаружения

План.

1. Контрольный опрос
2. Действия вредоносных программ.
3. Классификация вредоносного ПО
4. Троянский конь
5. Макровирус;
6. Сетевой червь.

Теоретическая часть

Вредоносная программа - *программа*, предназначенная для осуществления несанкционированного доступа к информации и (или) воздействия на информацию или ресурсы информационной системы[44]. Иными словами вредоносной программой называют некоторый самостоятельный набор инструкций, который способен выполнять следующее:

1. скрывать свое присутствие в компьютере;
2. обладать способностью к самоуничтожению, маскировкой под легальные программы и копирования себя в другие области оперативной или внешней памяти;
3. модифицировать (разрушать, искажать) код других программ;
4. самостоятельно выполнять деструктивные функции - копирование, модификацию, уничтожение, блокирование и т.п.
5. искажать, блокировать или подменять выводимую во внешний канал связи или на внешний носитель информацию.

Основными путями проникновения вредоносных программ в *АС*, в частности, на *компьютер*, являются сетевое взаимодействие и съемные носители информации (флешки, диски и т.п.). При этом внедрение в систему может носить случайный характер.

Основными видами вредоносных программ являются:

- программные закладки;
- программные вирусы;
- сетевые черви;
- другие вредоносные программы, предназначенные для осуществления НСД.

К программным закладкам относятся программы и фрагменты программного кода, предназначенные для формирования недекларированных возможностей легального программного обеспечения.

Недекларированные возможности программного обеспечения - функциональные возможности программного обеспечения, не описанные в документации[45]. Программная закладка часто служит проводником для других вирусов и, как правило, не обнаруживаются стандартными средствами антивирусного контроля.

Закладки иногда делят на программные и аппаратные, но фактически все закладки - программные, так как под аппаратными закладками подразумеваются так называемые прошивки.

Программные закладки различают в зависимости от метода их внедрения в систему:

- программно-аппаратные. Это закладки, интегрированные в программно-аппаратные средства ПК (BIOS, прошивки периферийного оборудования);
- загрузочные. Это закладки, интегрированные в программы начальной загрузки (программы-загрузчики), располагающиеся в загрузочных секторах;

- драйверные. Это закладки, интегрированные в драйверы (файлами, необходимые операционной системе для управления подключенными к компьютеру периферийными устройствами);
- прикладные. Это закладки, интегрированные в прикладное программное обеспечение (текстовые редакторы, графические редакторы, различные утилиты и т.п.);
- исполняемые. Это закладки, интегрированные в исполняемые программные модули. Программные модули чаще всего представляют собой пакетные файлы, которые состоят из команд операционной системы, выполняемых одна за другой, как если бы их набирали на клавиатуре компьютера;
- закладки-имитаторы. Это закладки, которые с помощью похожего интерфейса имитируют программы, в ходе работы которых требуется вводить конфиденциальную информацию;

Для выявления программных закладок часто используется качественный подход, заключающийся в наблюдении за функционированием системы, а именно:

1. снижение быстродействия;
2. изменение состава и длины файлов;
3. частичное или полное блокирование работы системы и ее компонентов;
4. имитация физических (аппаратных) сбоев работы вычислительных средств и периферийных устройств;
5. переадресация сообщений;
6. обход программно-аппаратных средств криптографического преобразования информации;
7. обеспечение доступа в систему с несанкционированных устройств.

Существуют также диагностические методы обнаружения закладок. Так, например, антивирусы успешно находят загрузочные закладки. С инициированием статической ошибки на дисках хорошо справляется *Disk* Doctor, входящий в распространенный комплекс утилит Norton Utilities. К наиболее распространенным программным закладкам относится "*троянский конь*".

Троянским конем называется:

- программа, которая, являясь частью другой программы с известными пользователю функциями, способна втайне от него выполнять некоторые дополнительные действия с целью причинения ему определенного ущерба;
- программа с известными ее пользователю функциями, в которую были внесены изменения, чтобы, помимо этих функций, она могла втайне от него выполнять некоторые другие (разрушительные) действия.

Перечислим основные виды троянских программ и их возможности:

- Trojan-Notifier - Оповещение об успешной атаке. Троянцы данного типа предназначены для сообщения своему "хозяину" о зараженном компьютере. При этом на адрес "хозяина" отправляется информация о компьютере, например, IP-адрес компьютера, номер открытого порта, адрес электронной почты и т. п.
- Trojan-PSW - Воровство паролей. Они похищают конфиденциальные данные с компьютера и передают их хозяину по электронной почте.
- Trojan-Clicker - интернет-кликеры - Семейство троянских программ, основная функция которых - организация несанкционированных обращений к интернет-ресурсам (обычно к веб-страницам). Методы для этого используются разные, например установка злонамеренной страницы в качестве домашней в браузере.
- Trojan-DDoS - Trojan-DDoS превращают зараженный компьютер в так называемый бот, который используется для организации атак отказа в доступе на определенный сайт. Далее от владельца сайта требуют заплатить деньги за прекращение атаки.

- Trojan-Proxy - Троянские прокси-сервера. Семейство троянских программ, скрытно осуществляющих анонимный доступ к различным Интернет-ресурсам. Обычно используются для рассылки спама.
- Trojan-Spy - Шпионские программы. Они способны отслеживать все ваши действия на зараженном компьютере и передавать данные своему хозяину. В число этих данных могут попасть пароли, аудио и видео файлы с микрофона и видеокамеры, подключенных к компьютеру.
- Backdoor - Способны выполнять удаленное управление зараженным компьютером. Его возможности безграничны, весь ваш компьютер будет в распоряжении хозяина программы. Он сможет рассылать от вашего имени сообщения, знакомиться со всей информацией на компьютере, или просто разрушить систему и данные без вашего ведома.
- Trojan-Dropper - Инсталляторы прочих вредоносных программ. Очень похожи на Trojan-Downloader, но они устанавливают злонамеренные программы, которые содержатся в них самих.
- Rootkit - способны прятаться в системе путем подмены собой различных объектов. Такие трояны весьма неприятны, поскольку способны заменить своим программным кодом исходный код операционной системы, что не дает антивирусу возможности выявить наличие вируса.

Абсолютно все программные закладки, независимо от метода их внедрения в компьютерную систему, срока их пребывания в оперативной памяти и назначении, имеют одну общую черту: обязательное выполнение *операции* записи в оперативную или внешнюю *память* системы. При отсутствии данной *операции* никакого негативного влияния программная закладка оказать не может.

Вирус (компьютерный, программный) - исполняемый программный код или интерпретируемый набор инструкций, обладающий свойствами несанкционированного распространения и самовоспроизведения. Созданные дубликаты компьютерного вируса не всегда совпадают с оригиналом, но сохраняют способность к дальнейшему распространению и самовоспроизведению. Таким образом, **обязательным свойством программного вируса является способность создавать свои копии** и внедрять их в вычислительные сети и/или файлы, системные области компьютера и прочие выполняемые объекты. При этом дубликаты сохраняют способность к дальнейшему распространению.

Жизненный цикл вируса состоит из следующих этапов:

- Проникновение на компьютер
- Активация вируса
- Поиск объектов для заражения
- Подготовка вирусных копий
- Внедрение вирусных копий

Классификация вирусов и сетевых червей представлена на рисунке 5.1.

Вирусный код загрузочного типа позволяет взять управление компьютером на этапе инициализации, еще до запуска самой системы. **Загрузочные вирусы** записывают себя либо в в *boot*-сектор, либо в сектор, содержащий системный *загрузчик* винчестера, либо меняют *указатель* на *активный boot*-сектор. Принцип действия загрузочных вирусов основан на алгоритмах запуска ОС при включении или перезагрузке компьютера: после необходимых тестов установленного оборудования (памяти, дисков и т. д.) *программа* системной загрузки считывает первый физический сектор загрузочного диска и передает управление на A:, C: или *CD-ROM*, в зависимости от параметров, установленных в *BIOS Setup*.

В случае дискеты или CD-диска управление получает *boot*-сектор диска, который анализирует таблицу параметров диска (BPB - *BIOS Parameter Block*), высчитывает адреса системных файлов ОС, считывает их в *память* и запускает на выполнение. Системными

файлами обычно являются MSDOS.SYS и IO.SYS, либо IBMDOS.*COM* и IBMBIO.*COM*, либо другие в зависимости от установленной версии *DOS*, и/или *Windows*, или других ОС. Если же на загрузочном диске отсутствуют файлы операционной системы, *программа*, расположенная в *boot*-секторе диска, выдает *сообщение об ошибке* и предлагает заменить загрузочный *диск*.

В случае винчестера управление получает *программа*, расположенная в *MBR* винчестера. Она анализирует таблицу разбиения диска (*Disk Partition Table*), вычисляет *адрес* активного *boot*-сектора (обычно этим сектором является *boot*-сектор диска C:), загружает его в *память* и передает на него управление. Получив управление, *активный boot*-сектор винчестера проделывает те же действия, что и *boot*-сектор дискеты.

При заражении дисков загрузочные вирусы подставляют свой код вместо какой-либо программы, получающей управление при загрузке системы. Принцип заражения, таким образом, одинаков во всех описанных выше способах: *вирус* "заставляет" систему при ее перезапуске считать в *память* и отдать управление не оригинальному коду загрузчика, а коду вируса.

Пример: Вредоносная *программа Virus.Boot.Snow*.a записывает свой код в *MBR* жесткого диска или в загрузочные сектора дискет. При этом оригинальные загрузочные сектора шифруются вирусом. После получения управления *вирус* остается в памяти компьютера (резидентность) и перехватывает прерывания *INT* 10h, 1Ch и 13h. Иногда *вирус* проявляет себя визуальным эффектом - на экране компьютера начинает падать снег.

Рис. 5.1. Классификация программных вирусов и сетевых червей

Файловые вирусы - вирусы, которые заражают непосредственно файлы. Файловые вирусы можно разделить на три группы в зависимости от среды, в которой распространяется *вирус*:

1. файловые вирусы - работают непосредственно с ресурсами операционной системы. Пример: один из самых известных вирусов получил название "Чернобыль". Благодаря своему небольшому размеру (1 Кб) вирус заражал PE-файлы таким образом, что их размер не менялся. Для достижения этого эффекта вирус ищет в файлах "пустые" участки, возникающие из-за выравнивания начала каждой секции файла под кратные значения байт. После получения управления вирус перехватывает IFS API, отслеживая вызовы функции обращения к файлам и заражая исполняемые файлы. 26 апреля срабатывает деструктивная функция вируса, которая заключается в стирании Flash BIOS и начальных секторов жестких дисков. Результатом является неспособность компьютера

загружаться вообще (в случае успешной попытки стереть Flash BIOS) либо потеря данных на всех жестких дисках компьютера.

2. **Макровирусы** - вирусы, написанные на макроязыках, встроенных в некоторые системы обработки данных (текстовые редакторы, электронные таблицы и т.п.). Самыми распространенными являются вирусы для программ Microsoft Office. Для своего размножения такие вирусы используют возможности макроязыков и при их помощи переносят себя (свои копии) из одного документа в другой.

Для существования макровирсуов в конкретном редакторе встроенный в него макроязык должен обладать следующими возможностями:

1. привязка программы на макроязыке к конкретному файлу;
2. копирование макропрограмм из одного файла в другой;
3. получение управления макропрограммой без вмешательства пользователя (автоматические или стандартные макросы).

Данным условиям удовлетворяют прикладные программы Microsoft Word, Excel и Microsoft Access. Они содержат в себе макроязыки: Word Basic, Visual Basic for Applications. Современные макроязыки обладают вышеперечисленными особенностями с целью предоставления возможности автоматической обработки данных.

Большинство макровирусов активны не только в момент открытия (закрытия) файла, но до тех пор, пока активен сам редактор. Они содержат все свои функции в виде стандартных макросов Word/Excel/Office. Существуют, однако, вирусы, использующие приемы скрытия своего кода и хранящие свой код в виде не макросов. Известно три подобных приема, все они используют возможность макросов создавать, редактировать и исполнять другие макросы. Как правило, подобные вирусы имеют небольшой макрос-загрузчик вируса, который вызывает встроенный редактор макросов, создает новый макрос, заполняет его основным кодом вируса, выполняет и затем, как правило, уничтожает (чтобы скрыть следы присутствия вируса). Основной код таких вирусов присутствует либо в самом макросе вируса в виде текстовых строк (иногда - зашифрованных), либо хранится в области переменных документа[46].

3. **Сетевые вирусы** - вирусы, которые для своего распространения используют протоколы и возможности локальных и глобальных сетей. Основным свойством сетевого вируса является возможность самостоятельно тиражировать себя по сети. При этом существуют сетевые вирусы, способные запустить себя на удаленной станции или сервере.

Основные деструктивные действия, выполняемые вирусами и червями:

1. перезагрузка каналов связи
2. атаки "отказ в обслуживании"
3. потеря данных
4. нарушение работы ПО
5. загрузка ресурсов компьютера
6. хищение информации.

Помимо всего вышеописанного, существуют вирусы комбинированного типа, которые объединяют в себе свойства разных типов вирусов, например, файлового и загрузочного. В виде примера приведем популярный в минувшие годы файловый *загрузочный вирус* под названием "OneHalf". Этот вирусный код, оказавшись в компьютерной среде операционной системы "MS-*DOS*" заражал основную *запись* загрузки. В процессе инициализации компьютера он шифровал секторы основного диска, начиная с конечных. Когда *вирус* оказывается в памяти, он начинает контролировать любые обращения к шифровальным секторам и может расшифровать их таким образом, что все программы будут работать в штатном режиме. Если *вирус* "OneHalf" просто стереть из памяти и сектора загрузки, то *информация*, записанная в шифровальном секторе диска, станет недоступной. Когда *вирус* зашифровывает часть диска, он предупреждает об этом следующей надписью:

"*Dis* is *one half,* Press any key to *continue...*". После этих действий он ждет, когда вы нажмете на любую кнопку и продолжите работать. В вирусе "OneHalf" использованы разные маскировочные *механизмы*. Он считается невидимым вирусом и выполняет полиморфные алгоритмические функции. Обнаружить и удалить вирусный код "OneHalf" весьма проблематично, потому что, его могут увидеть не все антивирусные программы.

На этапе подготовки вирусных копий современные вирусы часто используют методы маскировки копий с целью затруднения их нахождения антивирусными средствами[47]:

- Шифрование - вирус состоит из двух функциональных кусков: собственно вирус и шифратор. Каждая копия вируса состоит из шифратора, случайного ключа и собственно вируса, зашифрованного этим ключом.
- Метаморфизм - создание различных копий вируса путем замены блоков команд на эквивалентные, перестановки местами кусков кода, вставки между значащими кусками кода "мусорных" команд, которые практически ничего не делают. Сочетание этих двух технологий приводит к появлению следующих типов вирусов.
- Шифрованный вирус - вирус, использующий простое шифрование со случайным ключом и неизменный шифратор. Такие вирусы легко обнаруживаются по сигнатуре шифратора.
- Метаморфный вирус - вирус, применяющий метаморфизм ко всему своему телу для создания новых копий.
- Полиморфный вирус - вирус, использующий метаморфный шифратор для шифрования основного тела вируса со случайным ключом. При этом часть информации, используемой для получения новых копий шифратора также может быть зашифрована. Например, вирус может реализовывать несколько алгоритмов шифрования и при создании новой копии менять не только команды шифратора, но и сам алгоритм.

Червь (*сетевой червь*) - тип вредоносных программ, распространяющихся *по* сетевым каналам, способных к автономному преодолению систем защиты автоматизированных и компьютерных сетей, а также к созданию и дальнейшему распространению своих копий, не всегда совпадающих с оригиналом, и осуществлению иного вредоносного воздействия.

Самым знаменитым червем является червь Moriss, *механизмы* работы которого подробно описаны в литературе. Червь появился в 1988 году и в течение короткого промежутка времени парализовал работу многих компьютеров в Интернете. Данный червь является "классикой" вредоносных программ, а *механизмы* нападения, разработанные автором при его написании, до сих пор используются злоумышленниками. Moriss являлся самораспространяющейся программой, которая распространяла свои копии *по* сети, получая привилегированные *права* доступа на хостах сети за счет использования уязвимостей в операционной системе. Одной из уязвимостей, использованных червем, была уязвимая версия программы sendmail (*функция* "*debug*" программы sendmail, которая устанавливала отладочный режим для текущего сеанса связи), а другой - *программа* fingerd (в ней содержалась ошибка переполнения буфера). Для поражения систем червь использовал также *уязвимость* команд rexec и rsh, а также неверно выбранные пользовательские пароли.

На этапе проникновения в систему черви делятся преимущественно *по* типам используемых протоколов:

- Сетевые черви - черви, использующие для распространения протоколы Интернет и локальных сетей. Обычно этот тип червей распространяется с использованием неправильной обработки некоторыми приложениями базовых пакетов стека протоколов tcp/ip
- Почтовые черви - черви, распространяющиеся в формате сообщений электронной почты. Как правило, в письме содержится тело кода или ссылка на зараженный ресурс. Когда вы запускаете прикрепленный файл, червь активизируется;

когда вы щелкаете на ссылке, загружаете, а затем открываете файл, червь также начинает выполнять свое вредоносное действие. После этого он продолжает распространять свои копии, разыскивая другие электронные адреса и отправляя по ним зараженные сообщения. Для отправки сообщений червями используются следующие способы: прямое подключение к SMTP-серверу, используя встроенную в код червя почтовую библиотеку; использование сервисов MS Outlook; использование функций Windows MAPI.

Для поиска адресов жертв чаще всего используется адресная книга MS Outlook, но может использоваться также адресная база WAB. Червь может просканировать файлы, хранящиеся на дисках, и выделить из них строки, относящиеся к адресам электронной почты. Черви могут отсылать свои копии по всем адресам, обнаруженным в почтовом ящике (некоторые обладают способностью отвечать на письма в ящике). Встречаются экземпляры, которые могут комбинировать способы.

- IRC-черви - черви, распространяющиеся по каналам IRC (Internet Relay Chat). Черви этого класса используют два вида распространения: посылание пользователю URL-ссылки на файл-тело; отсылку пользователю файла (при этом пользователь должен подтвердить прием).
- P2P-черви - черви, распространяющиеся при помощи пиринговых (peer-to-peer) файлообменных сетей. Механизм работы большинства подобных червей достаточно прост: для внедрения в P2P-сеть червю достаточно скопировать себя в каталог обмена файлами, который обычно расположен на локальной машине. Всю остальную работу по его распространению P2P-сеть берет на себя - при поиске файлов в сети она сообщит удаленным пользователям о данном файле и предоставит весь необходимый сервис для его скачивания с зараженного компьютера.

Существуют более сложные P2P-черви, которые имитируют сетевой протокол конкретной файлообменной системы и положительно отвечают на поисковые запросы (при этом червь предлагает для скачивания свою копию).

- IM-черви - черви, использующие для распространения системы мгновенного обмена сообщениями (IM, Instant Messenger - ICQ, MSN Messenger, AIM и др.). Известные компьютерные черви данного типа используют единственный способ распространения - рассылку на обнаруженные контакты (из контакт-листа) сообщений, содержащих URL на файл, расположенный на каком-либо веб - сервере. Данный прием практически полностью повторяет аналогичный способ рассылки, использующийся почтовыми червями.

Мы перечислили наиболее распространенные категории сетевых червей, на практике их значительно больше. Например, в настоящее время всё большую "популярность" приобретают мобильные черви и черви, распространяющие свои копии через общие сетевые ресурсы. Последние используют функции операционной системы, в частности, перебирают доступные сетевые папки, подключаются к компьютерам в глобальной сети и пытаются открыть их диски на полный *доступ*. Отличаются от стандартных сетевых червей тем, что пользователю нужно открыть *файл* с копией червя, чтобы активизировать его.

По деструктивным возможностям вирусы и *сетевые черви* можно разделить на:

- безвредные, т. е. никак не влияющие на работу компьютера (кроме уменьшения свободной памяти на диске в результате своего распространения);
- неопасные, влияние которых ограничивается уменьшением свободной памяти на диске и графическими, звуковыми и прочими эффектами;
- опасные вирусы, которые могут привести к серьезным сбоям в работе компьютера;
- очень опасные - в алгоритм их работы заведомо заложены процедуры, которые могут вызвать потерю программ, уничтожить данные, стереть необходимую для работы компьютера информацию, записанную в системных областях памяти, и даже, как гласит одна из непроверенных компьютерных легенд, способствовать быстрому износу

движущихся частей механизмов - вводить в резонанс и разрушать головки некоторых типов винчестеров.

Но даже если в алгоритме вируса не найдено ветвей, наносящих *ущерб* системе, этот *вирус* нельзя с полной уверенностью назвать безвредным, так как проникновение его в *компьютер* может вызвать непредсказуемые и порой катастрофические последствия. Ведь *вирус*, как и всякая *программа*, имеет ошибки, в результате которых могут быть испорчены как файлы, так и сектора дисков (например, вполне безобидный на первый взгляд *вирус* DenZuk довольно корректно работает с 360-килобайтовыми дискетами, но может уничтожить информацию на дискетах большего объема). До сих пор попадаются вирусы, определяющие COM или EXE не *по* внутреннему формату файла, а *по* его расширению. Естественно, что при несовпадении формата и расширения имени *файл* после заражения оказывается неработоспособным. Возможно также "заклинивание" резидентного вируса и системы при использовании новых версий *DOS*, при работе в *Windows* или с другими мощными программными системами. И так далее.

Если проанализировать всё вышесказанное, то можно заметить схожесть сетевых червей и компьютерных вирусов, в частности, полное совпадение жизненного *цикла* и самотиражирование. Основным отличием червей от программных вирусов является способность к распространению *по* сети без участия человека. Иногда сетевых червей относят к подклассу компьютерных вирусов.

В связи с бурным развитием Интернета и информационных технологий количество вредоносных программ и вариантов их внедрения в информационную систему неустанно растет. Наибольшую опасность представляют новые формы вирусов и сетевых червей, сигнатуры которых не известны производителям средств защиты информации. В настоящее время всё большую популярность получают такие методы борьбы, как *анализ* аномального поведения системы и искусственные иммунные системы, позволяющие обнаруживать новые формы вирусов.

Контрольные вопросы:

1. Опишите действия вредоносных программ?
2. Классификация вредоносного ПО?
3. Что такое «Троянский конь»?;
4. Что такое «Макровирус»?;
5. Что такое «Сетевой червь»?.

ЛЕКЦИЯ 9

Методы обнаружения вредоносного ПО

План.

1. Контрольный опрос
2. Сканирование
3. Эвристический анализ
4. Вакцинирование

Методы обнаружения вирусов

В настоящее время существует множество антивирусных программных средств, но их можно разделить на группы по методам поиска вирусов, которые они реализовывают.

Рассмотрим эти методы.

1. **Сканирование.**

Это самый простой метод поиска вируса. Он основан на последовательном просмотре памяти компьютера, загрузочных секторов и проверяемых файлов в поиске так называемых сигнатур (масок) известных вирусов.

Сигнатура вируса - это уникальная последовательность байтов, принадлежащая вирусу и не встречающаяся в других программах.

Определение сигнатуры вируса очень сложная задача. Необходимо тщательно изучить принцип работы вируса и сравнить программы, зараженные данным вирусом, и незараженные. Кроме того, сигнатура не должна содержаться в других программах, иначе возможны ложные срабатывания.

Надежность принципа поиска по маске ограниченной длины не очень высока. Вирус легко модифицировать, чем и занимаются многие авторы вирусов. Достаточно изменить строковую константу, текст выдаваемого сообщения или одну из первых команд, чтобы вирус стал совсем новым.

Надежность увеличивается при приведении вируса к каноническому виду: то есть обнулении всех байтов, приходящихся на переменные и константы.

Хранение сигнатур канонических форм всех известных вирусов (вспомните, что на сегодняшний день их число огромно) требует неоправданно много памяти. Достаточно хранить только контрольную сумму сигнатур вирусов.

При подозрении на вирус необходимо привести подозреваемый код к каноническому виду, подсчитать контрольную сумму и сравнить с эталоном.

Часто в качестве сигнатуры берется характерный для этого вируса фрагмент кода, например, фрагмент обработчика прерывания. Не все вирусы имеют сигнатуры в виде строк байт, для некоторых удается в качестве сигнатур использовать регулярные выражения, а некоторые вирусы могут вообще не иметь сигнатур, например, полиморфные.

2. **Обнаружение изменений, или контроль целостности.**

Контроль целостности основан на выполнении двух процедур:

- постановка на учет;
- контроль поставленного на учет.

При внедрении вируса в компьютерную систему обязательно происходят изменения в системе (которые некоторые вирусы успешно маскируют). Это и изменение объема доступной оперативной памяти, и изменение загрузочных секторов дисков, и изменения самих файлов.

Достаточно запомнить характеристики, которые подвергаются изменениям в результате внедрения вируса, а затем периодически сравнивать эти эталонные характеристики с действующими.

3. **Эвристический анализ.**

Метод, который стал использоваться для обнаружения вирусов сравнительно недавно. Он предназначен для обнаружения новых неизвестных вирусов. Программы, реализующие этот метод, тоже проверяют загрузочные секторы дисков и файлы, только уже пытаются обнаружить в них код, *характерный* для вирусов. Например, таким может быть код, устанавливающий резидентный модуль в памяти и т.п.

4. **Метод резидентного сторожа.**

Этот метод направлен на выявление «подозрительных» действий пользовательских программ, например, таких, как запись на диск по абсолютному адресу, форматирование диска, изменение загрузочного сектора, изменение или переименование выполняемых программ, появление новых резидентных программ, изменение системных областей DOS и других. При обнаружении «подозрительного» действия необходимо «спросить разрешение» у пользователя на выполнение такого действия.

5. **Вакцинирование программ.**

Этот метод заключается в дописывании к исполняемому файлу дополнительной подпрограммы, которая первой получает управление при запуске файла, выполняет проверку целостности программы. Проверяться могут любые изменения, например, контрольная сумма файла или другие характеристики.

Что такое ложные тревоги?

Задача любого антивирусного средства - обнаружить вирус в системе с максимальной степенью надежности, то есть не вызывая при этом ложные тревоги.

Ложные тревоги делят на два типа:

- ложная позитивная (положительная);
- ложная негативная (отрицательная).

Ложная позитивная тревога - это когда антивирусное средство сообщает о наличии вируса, а на самом деле его нет.

Ложная негативная тревога - это когда антивирусное средство говорит, что вируса нет, а на самом деле он есть.

Некоторые специалисты ложные тревоги называют ошибками антивирусов, причем ложные негативные тревоги (несрабатывания) называются ошибками первого рода, а ложные позитивные тревоги - ошибками второго рода.

В идеале антивирус должен обнаруживать все возможные вирусы и не должен вызывать ложных тревог.

Контрольные вопросы:

1. Как можно обнаружить вирус?
2. Что такое Сканирование?
3. Что такое Эвристический анализ?
4. Что такое Вакцинирование?

ЛЕКЦИЯ 10

Антивирусные программы: классификация, сравнительный анализ

План.

1. Контрольный опрос;
2. Функции антивирусной программы;
3. Характеристики современного антивирусного ПО;
4. Настройка и обновление.

Антивирусная программа – программа, предназначенная для борьбы с компьютерными вирусами.

В своей работе эти программы используют различные принципы для поиска и лечения зараженных файлов.

Для нормальной работы на ПК каждый пользователь должен следить за обновлением антивирусов.

Если антивирусная программа обнаруживает вирус в файле, то она удаляет из него программный код вируса. Если лечение невозможно, то зараженный файл удаляется целиком.

Имеются различные типы антивирусных программ – полифаги, ревизоры, блокировщики, сторожа, вакцины и пр.

Типы антивирусных программ:

- Антивирусные сканеры – после запуска проверяют файлы и оперативную память и обеспечивают нейтрализацию найденного вируса
- Антивирусные сторожа (мониторы) – постоянно находятся в ОП и обеспечивают проверку файлов в процессе их загрузки в ОП
- Полифаги – самые универсальные и эффективные антивирусные программы. Проверяют файлы, загрузочные сектора дисков и ОП на поиск новых и неизвестных вирусов. Занимают много места, работают не быстро

- Ревизоры – проверяют изменение длины файла. Не могут обнаружить вирус в новых файлах (на дискетах, при распаковке), т.к. в базе данных нет сведений о этих файлах
- Блокировщики – способны обнаружить и остановить вирус на самой ранней стадии его развития (при записи в загрузочные сектора дисков). Антивирусные блокировщики могут входить в BIOS Setup

Антивирус Касперского

Антивирус Касперского давно завоевал популярность у миллионов пользователей благодаря своей эффективности и быстроте реакции на различные угрозы. Несмотря на все свои преимущества, до сих пор многие не знающие товарищи критикуют его за то, что он якобы очень сильно нагружает систему. Так было в недалёких 2003-2004 годах, тогда многие антивирусные продукты не особо выделялись «лёгкостью». Теперь совсем другие времена и одним из главных требований к современным антивирусам является небольшое потребление ресурсов компьютера, хотя стоит заметить, что если дать команду на полное и глубокое сканирование, то для этого будет задействованы все средства, зато в других случаях система остаётся почти незатронутой. Последними версиями являются антивирус Касперского 2010, комплексная защита Kaspersky Internet Security 2010 и Kaspersky CRYSTAL.

Антивирус Касперского 2010 обладает всеми основными функциями защиты, вот некоторые из них: классический антивирус, антишпионский модуль, онлайн сканер и веб-антивирус, защищающий в режиме реального времени при просмотре Интернет страниц. Имеется новая функция – виртуальная клавиатура, позволяющая безопасно вводить данные и не бояться программ, крадущих логины и пароли.

В Kaspersky Internet Security 2010 присутствуют те же функции, плюс есть наличие сетевого экрана, защищающего от внешних вторжений из Интернета. Kaspersky CRYSTAL в отличии от Kaspersky Internet Security 2010 оснащён ещё и функцией резервного копирования и восстановления данных.

Также в продуктах лаборатории Касперского присутствует функция Родительского контроля, ограждающая детей от посещения нежелательных Интернет сайтов. Учитывая наличие мощной защиты в данном продукте, не стоит удивляться его цене. К примеру, лицензия Kaspersky CRYSTAL для 2-х компьютеров стоит около 88у.е., а остальные версии, ввиду отсутствия некоторых функций, стоят немного дешевле. Прежде чем приобретать какую-либо версию, лучше скачать установочный дистрибутив с официального сайта и установить пробную версию продукта, которая будет доступна 30 дней.

Dr.Web

Теперь рассмотрим **Dr.Web** – ещё один антивирус российского производства, кстати, сертифицированный Министерством обороны Российской Федерации. Если говорить о функциях защиты, то они во многом схожи с антивирусом Касперского, да и с функциями многих других популярных продуктов, таких как Nod32, Norton, Panda, но только реализованные под другой графической оболочкой.

Помимо модулей антивируса, антишпиона, антируткита, антиспама, веб-антивируса и брандмауэра, имеющихся во всех продуктах Dr.Web, Dr.Web Бастион Pro предлагает функцию Криптограф, предназначенную для шифрования информации в специальных файловых контейнерах. Также как и в «Касперском», есть возможность активировать пробную версию на 30 дней, но с той лишь разницей, что эту процедуру можно повторять один раз в четыре месяца.

Бесплатная для домашнего использования утилита Dr.Web CureIt! не защищает ПК в реальном времени, зато позволяет без установки просканировать систему на наличие вредоносного программного обеспечения. Если компьютер заражен настолько, что даже запуск Windows или Unix проблематичен, то на помощь придет Dr.Web LiveCD – диск аварийного восстановления системы. С

помощью него Вы легко восстановите работоспособность пораженной системы совершенно бесплатно! Dr.Web LiveCD поможет не только очистить компьютер от инфицированных и подозрительных файлов, но и скопировать важную информацию на сменные носители или другой компьютер, а также попытается вылечить зараженные объекты.

ESET NOD32

Немного слов об антивирусе **ESET NOD32**, имеющем, также как и остальные продукты, антивирус, антишпион, веб-антивирус, а в ESET NOD32 Smart Security добавлен антиспам и сетевой экран. NOD32 является одним из самых быстрых антивирусов, также хочется отметить, что и более ранние продукты этой серии отличались быстротой реакции и незаметностью для системных ресурсов компьютера. Аналогично «Касперскому» имеется возможность бесплатного тестирования в течение 30 дней.

ESET NOD32 Business Edition – централизованная защита серверов и рабочих станций от троянов, червей, рекламных вирусов, фишинг-атак и других угроз в организациях и офисах. Продукт включает в себя приложение ESET Remote Administrator для централизованного администрирования корпоративных сетей.

ESET NOD32 Business Edition Smart Security – комплексная защита класса Internet Security для серверов и рабочих станций в офисах и больших предприятий, включающая в себя антивирус, антишпион, антиспам и персональный файервол, а также приложение ESET Remote Administrator для администрирования в корпоративных сетевых средах предприятия.

Panda Security

Заслуживает также внимания такой бренд в мире антивирусных программ, как **Panda Security.**

Panda Security – один из мировых лидеров в разработке продуктов для защиты IT-ресурсов от вирусов и других компьютерных угроз. Огромный выбор готовых продуктов для дома и офисов. Для домашних пользователей Panda Security предлагает продукты: Panda Antivirus Pro (минимальный но достаточный уровень защиты), Panda Internet Security for Netbooks, Panda Internet Security, Panda Global Protection (максимальная защита вашего компьютера). Для офиса линейка антивирусных продуктов гораздо разнообразен. В любом варианте продукта есть возможность бесплатной полноценной работы.

Есть у Panda Security интересные решения, как Panda Cloud Protection.

Panda Cloud Protection – это решение для малых и средних предприятий, которое избавляет от лишних расходов на аппаратное обеспечение, персонал и другие ресурсы, выделяемые на антивирусную защиту ПК, ноутбуков, рабочих станций, серверов и почтового трафика. Этот продукт, как бы всегда установлен на серверах Panda Security, а для проверки компьютеров всегда доступен из любого места в мире с помощью собственной веб-консоли.

Panda Cloud Protection включает следующие сервисы:

- Panda Cloud Office Protection
- Panda Cloud Email Protection
- Panda Cloud Internet Protection

Основным недостатком перечисленных выше коммерческих продуктов является их немаленькая стоимость. Согласитесь, не каждый может себе позволить потратить сумму в 80-90у.е., пусть даже на очень качественный программный продукт, поэтому приходится прибегать к различным «крякам» или искать бесплатную альтернативу. Благо сегодня имеется большое количество бесплатных антивирусов, защищающих ваш компьютер не хуже платных аналогов. Но следует помнить, что их бесплатность ограничивается только домашним использованием.

avast!5 Free Antivirus

Одним из таких антивирусных продуктов является **avast!5 Free Antivirus**, отличающийся от своих платных вариантов отсутствием брандмауэра и некоторыми другими непринципиальными функциями. В avast!5 Free Antivirus имеется возможность активации функции, позволяющей сканировать жёсткие диски ещё до загрузки операционной системы, что является несомненным преимуществом даже перед очень популярными антивирусными пакетами.

Также у компании AVAST Software есть и платные продукты:

- avast! Pro Antivirus – продукт для домашнего пользования, содержащий avast! Sandbox для проверки подозрительных веб-сайтов и приложений и с ускоренным процессом обновления.
- avast! Internet Security – максимальная защита для дома включающая защиту почтовых сообщений и блокировку доступа к личной информации.
- avast! Standard Suite – это офисный комплект продуктов в одном решении, который обеспечивают защиту от вирусов для серверов и рабочих станций на базе Windows.

.

Контрольные вопросы:

1. Назовите функции антивирусной программы.
2. Каик характеристики имеет современное антивирусное ПО?
3. Как осуществляется обновление?

ЛЕКЦИЯ 11

Файрвол: задачи, сравнительный анализ, настройка

План.

1. Контрольный опрос
2. Назначение межсетевых экранов;
3. Как происходит фильтрация трафика;
4. Типы МСЭ, достоинства и недостатки.

Сеть нуждается в защите от внешних угроз. Хищение данных, несанкционированный доступ и повреждения могут сказаться на работе сети и принести серьезные убытки. Используйте специальные программы и устройства, чтобы обезопасить себя от разрушительных воздействий. В этом обзоре мы расскажем о межсетевом экране и рассмотрим его основные типы.

Назначение межсетевых экранов

Межсетевые экраны (МСЭ) или файрволы — это аппаратные и программные меры для предотвращения негативных воздействий извне. Файрвол работает как фильтр: из всего потока трафика просеивается только разрешенный. Это первая линия защитных укреплений между внутренними сетями и внешними, такими как интернет. Технология применяется уже на протяжении 25 лет.

Необходимость в межсетевых экранах возникла, когда стало понятно, что принцип полной связности сетей больше не работает. Компьютеры начали появляться не только в университетах и лабораториях. С распространением ПК и интернета возникла необходимость отделять внутренние сети от небезопасных внешних, чтобы уберечься от злоумышленников и защитить компьютер от взлома.

Для защиты корпоративной сети устанавливают аппаратный межсетевой экран — это может быть отдельное устройство или часть маршрутизатора. Однако такая практика применяется не всегда. Альтернативный способ — установить на компьютер, который нуждается в защите, программный межсетевой экран. В качестве примера можно привести файрвол, встроенный в Windows.

Имеет смысл использовать программный межсетевой экран на корпоративном ноутбуке, которым вы пользуетесь в защищенной сети компании. За стенами организации вы попадаете в незащищенную среду — установленный файрвол обезопасит вас в командировках, при работе в кафе и ресторанах.

Как работает межсетевой экран

Фильтрация трафика происходит на основе заранее установленных правил безопасности. Для этого создается специальная таблица, куда заносится описание допустимых и недопустимым к передаче данных. Межсетевой экран не пропускает трафик, если одно из запрещающих правил из таблицы срабатывает.

Файрволы могут запрещать или разрешать доступ, основываясь на разных параметрах: IP-адресах, доменных именах, протоколах и номерах портов, а также комбинировать их.

- IP-адреса. Каждое устройство, использующее протокол IP, обладает уникальным адресом. Вы можете задать определенный адрес или диапазон, чтобы пресечь попытки получения пакетов. Или наоборот — дать доступ только определенному кругу IP-адресов.
- Порты. Это точки, которые дают приложениям доступ к инфраструктуре сети. К примеру, протокол ftp пользуется портом 21, а порт 80 предназначен для приложений, используемых для просмотра сайтов. Таким образом, мы получаем возможность воспрепятствовать доступу к определенным приложениям и сервисам.
- Доменное имя. Адрес ресурса в интернете также является параметром для фильтрации. Можно запретить пропускать трафик с одного или нескольких сайтов. Пользователь будет огражден от неприемлемого контента, а сеть от пагубного воздействия.
- Протокол. Файрвол настраивается так, чтобы пропускать трафик одного протокола или блокировать доступ к одному из них. Тип протокола указывает на набор параметров защиты и задачу, которую выполняет используемое им приложение.

Типы МСЭ

1. Прокси-сервер

Один из родоначальников МСЭ, который выполняет роль шлюза для приложений между внутренними и внешними сетями. Прокси-серверы имеют и другие функции, среди которых защита данных и кэширование. Кроме того, они не допускают прямые подключения из-за границ сети. Использование дополнительных функций может чрезмерно нагрузить производительность и уменьшить пропускную способность.

2. МСЭ с контролем состояния сеансов

Экраны с возможностью контролировать состояние сеансов — уже укоренившаяся технология. На решение принять или блокировать данные влияет состояние, порт и протокол. Такие версии следят за всей активностью сразу после открытия соединения и вплоть до самого закрытия. Блокировать трафик или не блокировать система решает, опираясь на установленные администратором правила и контекст. Во втором случае учитываются данные, которые МСЭ дали прошлые соединения.

3. МСЭ Unified threat management (UTM)

Комплексное устройство. Как правило, такой межсетевой экран решает 3 задачи:

- контролирует состояние сеанса;
- предотвращает вторжения;
- занимается антивирусным сканированием.

Порой фаерволы, усовершенствованные до версии UTM, включают и другой функционал, например: управление облаком.

4. Межсетевой экран Next-Generation Firewall (NGFW)

Ответ современным угрозам. Злоумышленники постоянно развивают технологии нападения, находят новые уязвимости, совершенствуют вредоносные программы и усложняют для отражения атаки на уровне приложений. Такой файрвол не только

фильтрует пакеты и контролирует состояние сеансов. Он полезен в поддержании информационной безопасности благодаря следующим функциям:

- учет особенностей приложений, который дает возможность идентифицировать и нейтрализовать вредоносную программу;
- оборона от непрекращающихся атак из инфицированных систем;
- обновляемая база данных, которая содержит описание приложений и угроз;
- мониторинг трафика, который шифруется с помощью протокола SSL.

5. МСЭ нового поколения с активной защитой от угроз

Данный тип межсетевого экрана — усовершенствованная версия NGFW. Это устройство помогает защититься от угроз повышенной сложности. Дополнительный функционал умеет:

- учитывать контекст и находить ресурсы, которые находятся под наибольшим риском;
- оперативно отражать атаки за счет автоматизации безопасности, которая самостоятельно управляет защитой и устанавливает политики;
- выявлять отвлекающую или подозрительную активность, благодаря применению корреляции событий в сети и на компьютерах;

В этой версии межсетевого экрана NGFW введены унифицированные политики, которые значительно упрощают администрирование.

Недостатки МСЭ

Межсетевые экраны обороняют сеть от злоумышленников. Однако необходимо серьезно отнестись к их настройке. Будьте внимательны: ошибившись при настройке параметров доступа, вы нанесете вред и файрвол будет останавливать нужный и ненужный трафик, а сеть станет неработоспособной.

Применение межсетевого экрана может стать причиной падения производительности сети. Помните, что они перехватывают весь входящий трафик для проверки. При крупных размерах сети чрезмерное стремление обеспечить безопасность и введение большего числа правил приведет к тому, что сеть станет работать медленно.

.

Контрольные вопросы:

1. Каково назначение межсетевых экранов?
2. Как происходит фильтрация трафика?
3. Назовите типы МСЭ, достоинства и недостатки?

ЛЕКЦИЯ 12

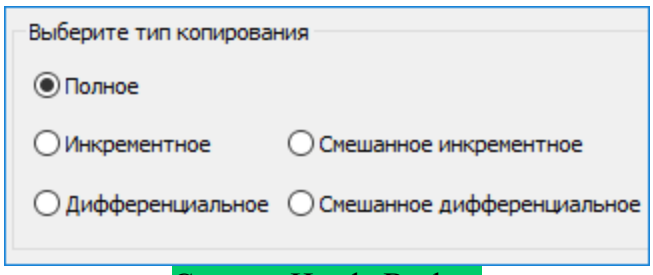
Архивация системных данных и программ

План.

1. Контрольный опрос
2. Необходимость создания информационной избыточности
3. Типы архивов
4. Регламенты архивирования

Тип резервного копирования данных выбирается в зависимости от задачи копирования, от объема данных, который вам необходимо копировать, и от необходимости обеспечить такие параметры, как скорость работы и полнота копии.

Handy Backup использует разные виды резервного копирования (полный, дифференциальный, инкрементальный и смешанный).



Скачать Handy Backup

Версия 8.1.2 от 21 февраля 2020

106 MB

Классификация типов резервного копирования

Полное резервное копирование

Каждый раз при выполнении задачи бэкапа из источника копируются все данные без изъятия. Этот тип резервного копирования наиболее медленный, но обеспечивает наибольшую полноту и точность сохранения данных.

Данный тип подходит для копирования небольшого объёма данных или для сохранности всех копируемых данных, например бэкап жесткого диска.

Инкрементальное резервное копирование

При таком типе бэкапа первый раз выполняется полное копирование, а каждый последующий раз копируются только новые или изменившиеся файлы с момента последней операции бэкапа.

Данный тип копирования наиболее эффективен там, где нужно следить за историей версий: копирование рабочих документов, проектов, отчётов и т.д.

Инкрементальное копирование с временными метками — разновидность инкрементального бэкапа, при котором каждая новая копия (инкремент) снабжается меткой, содержащей информацию о времени создания копии, для простоты автоматической обработки.

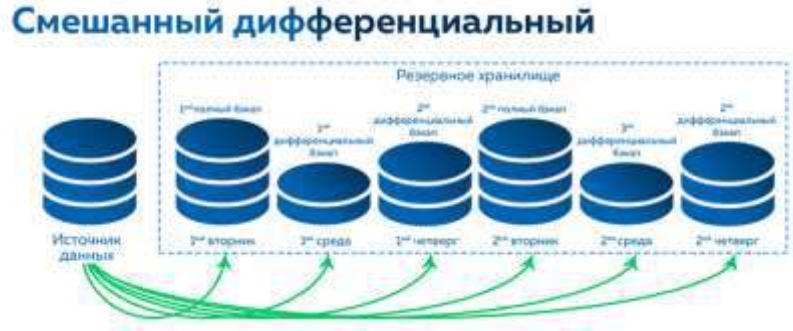
Дифференциальное резервное копирование

При этом методе в первый раз выполняется полный бэкап, а при последующих операциях задача копирует только обновлённую информацию, включая данные, изменившиеся по сравнению с полным бэкапом (а не с предыдущим бэкапом, как при инкрементальном копировании).

Такой вид копирования позволяет сэкономить место и время, поэтому пригодно для сохранения и восстановления часто меняющейся информации: бэкапа веб-сайтов, баз данных, массивов виртуальных машин и т.п.

Смешанное инкрементальное копирование

Создаёт полную копию данных, а затем — указанное количество инкрементальных копий в течение указанного промежутка времени. По истечении этого промежутка весь цикл повторяется, начиная с создания полной копии данных.



Смешанное дифференциальное копирование

Создаёт полную копию данных, за которой следует указанное количество дифференциальных копий, создаваемое в течение определённого промежутка времени. По достижении указанного количества копий цикл повторяется сначала.

Зеркальное резервное копирование

Этот тип копирования помогает сохранять одинаковое содержимое в двух папках. Все новые или изменённые данные копируются из одной синхронизируемой папки в другую. При зеркальном типе копирования (двухсторонней синхронизации папок) происходит взаимное обновление содержимого папок.

Контрольные вопросы:

1. Что архивация данных?
2. Опишите тип резервного копирования «полный»;
3. Опишите тип резервного копирования «инкрементальный»;
4. Опишите тип резервного копирования «дифференциальный»;
5. Опишите тип резервного копирования «зеркальный»;

ЛЕКЦИЯ 13

Средства и протоколы шифрования сообщений.

План.

1. Контрольный опрос
2. Безопасная передача сообщений;
3. Принципы шифрования;
4. Протоколы шифрования.

Единственный способ защиты от физического чтения данных это шифрование файлов. Простейший случай такого шифрования — архивирование файла с паролем. Однако здесь есть ряд серьезных недостатков. Во-первых, пользователю требуется каждый раз вручную шифровать и дешифровать (то есть, в нашем случае архивировать и разархивировать) данные перед началом и после окончания работы, что уже само по себе уменьшает защищенность данных. Пользователь может забыть зашифровать (заархивировать) файл после окончания работы, или (еще более банально) просто оставить на диске копию файла. Во-вторых, пароли, придуманные пользователем, как правило, легко угадываются. В любом случае, существует достаточное количество утилит, позволяющих распаковывать архивы, защищенные паролем. Как правило, такие утилиты осуществляют подбор пароля путем перебора слов, записанных в словаре.

Система EFS была разработана с целью преодоления этих недостатков. Ниже мы рассмотрим более подробно детали технологии шифрования, взаимодействие EFS с пользователем и способы восстановления данных, познакомимся с теорией и реализацией EFS в Windows 2000, а также рассмотрим пример шифрования каталога при помощи EFS.

Технология шифрования

EFS использует архитектуру Windows CryptoAPI. В ее основе лежит технология шифрования с открытым ключом. Для шифрования каждого файла случайным образом генерируется ключ шифрования файла. При этом для шифрования файла может применяться любой симметричный алгоритм шифрования. В настоящее же время в EFS используется один алгоритм, это DESX, являющийся специальной модификацией широко распространенного стандарта DES.

Ключи шифрования EFS хранятся в резидентном пуле памяти (сама EFS расположена в ядре Windows), что исключает несанкционированный доступ к ним через файл подкачки.

Взаимодействие с пользователем

По умолчанию EFS сконфигурирована таким образом, что пользователь может сразу начать использовать шифрование файлов. Операция шифрования и обратная поддерживаются для файлов и каталогов. В том случае, если шифруется каталог, автоматически шифруются все файлы и подкаталоги этого каталога. Необходимо отметить, что если зашифрованный файл перемещается или переименовывается из зашифрованного каталога в незашифрованный, то он все равно остается зашифрованным. Операции шифрования/дешифрования можно выполнить двумя различными способами — используя Windows Explorer или консольную утилиту Cipher.

Для того чтобы зашифровать каталог из Windows Explorer, пользователю нужно просто выбрать один или несколько каталогов и установить флажок шифрования в окне расширенных свойств каталога. Все создаваемые позже файлы и подкаталоги в этом

каталоге будут также зашифрованы. Таким образом, зашифровать файл можно, просто скопировав (или перенеся) его в «зашифрованный» каталог.

Зашифрованные файлы хранятся на диске в зашифрованном виде. При чтении файла данные автоматически расшифровываются, а при записи — автоматически шифруются. Пользователь может работать с зашифрованными файлами так же, как и с обычными файлами, то есть открывать и редактировать в текстовом редакторе Microsoft Word документы, редактировать рисунки в Adobe Photoshop или графическом редакторе Paint, и так далее.

Необходимо отметить, что ни в коем случае нельзя шифровать файлы, которые используются при запуске системы — в это время личный ключ пользователя, при помощи которого производится дешифровка, еще недоступен. Это может привести к невозможности запуска системы! В EFS предусмотрена простая защита от таких ситуаций: файлы с атрибутом «системный» не шифруются. Однако будьте внимательны: это может создать «дыру» в системе безопасности! Проверяйте, не установлен ли атрибут файла «системный» для того, чтобы убедиться, что файл действительно будет зашифрован.

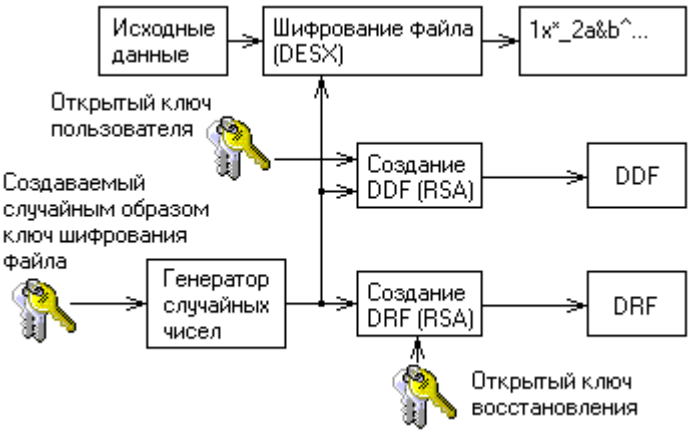
Важно также помнить о том, что зашифрованные файлы не могут быть сжаты средствами Windows и наоборот. Иными словами, если каталог сжат, его содержимое не может быть зашифровано, а если содержимое каталога зашифровано, то он не может быть сжат.

В том случае, если потребуется дешифровка данных, необходимо просто снять флажки шифрования у выбранных каталогов в Windows Explorer, и файлы и подкаталоги автоматически будут дешифрованы. Следует отметить, что эта операция обычно не требуется, так как EFS обеспечивает «прозрачную» работу с зашифрованными данными для пользователя.

Восстановление данных

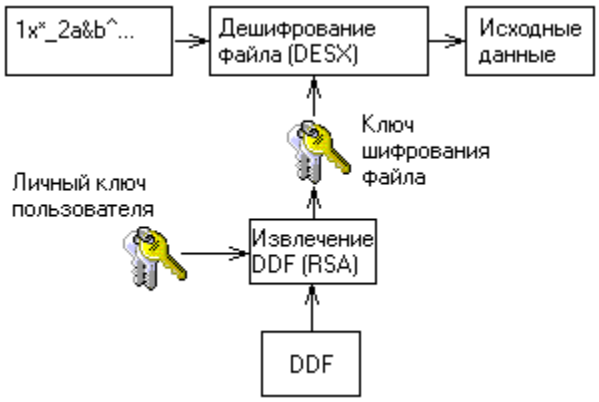
EFS обеспечивает встроенную поддержку восстановления данных на тот случай, если потребуется их расшифровать, но, по каким-либо причинам, это не может быть выполнено обычным. По умолчанию, EFS автоматически сгенерирует ключ восстановления, установит сертификат доступа в учетной записи администратора и сохранит его при первом входе в систему. Таким образом, администратор становится так называемым агентом восстановления, и сможет расшифровать любой файл в системе. Разумеется, политику восстановления данных можно изменить, и назначить в качестве агента восстановления специального человека, ответственного за безопасность данных, или даже несколько таких лиц.

Процесс шифрования



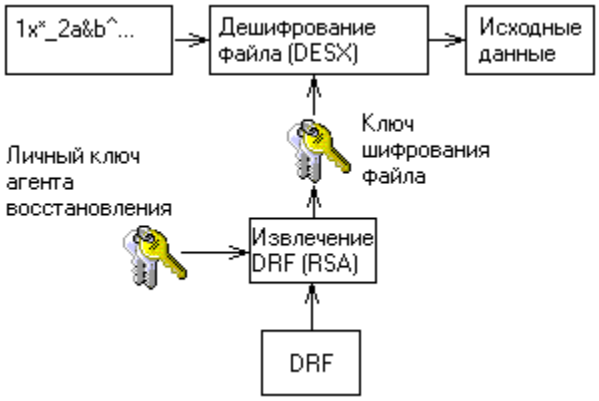
Незашифрованный файл пользователя шифруется при помощи случайно сгенерированного ключа FEK. Этот ключ записывается вместе с файлом, файл дешифруется при помощи общего ключа пользователя (записанного в DDF), а также при помощи общего ключа агента восстановления (записанного в DRF).

Процесс дешифрования



Сначала используется личный ключ пользователя для дешифрации FEK — для этого используется зашифрованная версия FEK, которая хранится в DDF. Расшифрованный FEK используется для поблочного дешифрования файла. Если в большом файле блоки считываются не последовательно, то дешифруются только считываемые блоки. Файл при этом остается зашифрованным.

Процесс восстановления



Этот процесс аналогичен дешифрованию с той разницей, что для дешифрования FEK используется личный ключ агента восстановления, а зашифрованная версия FEK берется из DRF.

Контрольные вопросы

1. Что такое сертификат и для чего он необходим?
2. В чем суть механизма защиты шифрованием?
3. В чем идея прозрачного шифрования?
4. Назначение системы EFS.
5. Шифрование системных файлов, доступно для всех пользователей?
6. При перемещении незашифрованных файлов в зашифрованную папку они автоматически шифруются в новой папке. Приведет ли к расшифровке файлов обратная операция?

ЛЕКЦИЯ 14

Средства диагностики оборудования.

План.

1. Контрольный опрос
2. Порядок инсталляции
3.Базовые настройки
4. Настройка оборудования

Порядок инсталляции программного обеспечения

Каждый программный продукт – это, прежде всего, исполняемый модуль с расширением .exe (например, arj.exe) или .com (например, win.com), и этот модуль может работать либо автономно (например, arj.exe), либо в сопровождении многочисленной «свиты» из множества служебных файлов и других программ (например, win.com).

До появления современных программных продуктов проблемы установки не существовало: исполняемый модуль (с сопутствующими файлами) просто копировали с дискеты или компакт-диска в любой каталог жесткого диска, а затем запускали на выполнение. Если данный программный продукт был совместим с аппаратными средствами, он функционировал в соответствии с соглашениями, предусмотренными программистом-разработчиком.

Несложные программы и сейчас «устанавливаются» подобным же образом. Однако для большинства современных программных продуктов разработчики предусматривают специальную процедуру установки ("инсталляции"), при которой используется специальная дистрибутивная копия продукта. Эта копия поставляется на дискете (нескольких дискетах) или компакт-диске и имеет программу установки setup.exe или install.exe. Для установки запускается эта программа, и выполняются ее указания. Подобную программу часто называют «Мастером установки программ».

Типичные шаги, которые выполняются во время установки продукта:

§ **ввод имени пользователя**, обладающего лицензией на использование данной программы, и название организации;

§ **проверка аппаратных элементов системы**;

§ **конфигурирование продукта** в соответствии с требованиями пользователя и запись на жесткий диск всех служебных файлов, необходимых для работы продукта в заказанной конфигурации (при этом часть функций продукта может быть отключена);

§ **создание и (или) модификация файлов настроек** – как системных (например, config.sys), так и специализированных файлов Windows (с расширением .ini).

При установке программного продукта в Windows чаще всего создается *программный элемент* (специальная папка), снабженный соответствующей пиктограммой. По желанию пользователя этот элемент можно включить либо в существующую, либо во вновь созданную программную группу.

В Панели управления Windows имеется специальное окно «Установка и удаление программ», которое содержит список установленных программ.

Контрольные вопросы:

1. Назовите основные этапы инсталляции ПО?
2. Что входит в базовые настройки ОС?
3. Как производится настройка оборудования?

ЛЕКЦИЯ 15

Конфигурирование ИС. Оперативное управление и регламентные работы.

План.

1. Контрольный опрос
2. Функциональные характеристики серверного ПО
3. Виды серверов
4. Виды серверных ОС

Особенности эксплуатации различных видов серверного программного обеспечения.

Сервером сети Интернет называется компьютер, на котором установлена специальная программа (она тоже называется сервером, web-сервером или http-сервером), которая отображает web-страницы по запросу клиентской машины, атак же выполняет множество других полезных функций, которых мы коснемся чуть позже. Когда

ваш домашний компьютер связывается с сервером и получает от него все необходимые данные, например код web-страницы, он выступает в роли клиента, а всю систему в этом случае принято называть связкой клиент-сервер. На этот термин следует обратить особое внимание, поскольку в последствии мы часто будем сталкиваться с ним.

Системой клиент-сервер называют механизм передачи информации между удаленным компьютером, предоставляющим свои ресурсы в распоряжение пользователей, и пользовательским компьютером, эксплуатирующим эти ресурсы. В данном случае компьютер, открывающий доступ к собственным ресурсам, носит название сервера, а получающий такой доступ клиента.

Серверы могут быть разными, причем отличия заключаются, прежде всего, в операционной системе, под управлением которой они работают.В настоящее время на большинстве интернетовских узлов используют два типа серверных программ: либо Internet Information Server, рассчитанный на работу под Windows NT, либо Apache, предназначенный для платформ, совместимых со стандартом UNIX. Как правило, серверы работают на линиях с большой пропускной способностью, например, в сетях с оптоволоконными каналами связи, что по финансовым соображениям доступно лишь крупным предприятиям.

Помимо соответствующей программы настоящий сервер должен иметь собственный домен, то есть адрес DNS, отвечающий стандартам Domain Name System.

Таким образом, сервер это компьютер с установленным на нем специальным программным обеспечением, имеющий собственное доменное имя. Владелец и администратор сервера могут гибко менять необходимые настройки, разрешать или запрещать доступ к его ресурсам, подключать, настраивать и запускать ряд дополнительных программ и функций, таких как скрипты CGI или приложения SSI, то есть полностью конфигурировать его работу по мере необходимости.

Тематическое содержание серверов может варьироваться в широком диапазоне в зависимости от целей, ради которых они были созданы, возможностей или фантазии владельца и многих других условий. Объединяет их все, пожалуй, только одно: полноценный сервер должен представлять собой то, что среди пользователей Интернета принято называть термином информационный портал , то есть в идеальном случае он является достаточно большим виртуальным пространством, состоящим из множества различных тематических разделов меньшего размера, либо некоторого количества самостоятельных проектов.

Файл-серверы и принт - серверы управляют доступом соответственно к файлам и принтерам, на серверах приложений выполняются прикладные части клиент - серверных приложений, а так же находятся данные доступные клиентам. Например, чтобы упростить извлечение данных, серверы хранят большие объемы информации в структурированном виде. Эти серверы отличаются от файл - серверов и принт - серверов.

В принт - серверах, файл или данные целиком копируются на запрашиваемый компьютер. А в сервере приложений на запрашиваемый компьютер посылаются только результаты запроса. Приложение-клиент на удаленном компьютере получает доступ к данным, хранимым на сервере приложений. Однако вместо всей базы данных на ваш компьютер с сервера загружаются только результаты запроса. В расширенной сети использование серверов различных типов становится наиболее актуальным. Необходимо поэтому учитывать всевозможные нюансы, которые могут проявиться при разрастании сети, с тем чтобы изменение роли определенного сервера в дальнейшем не отразилось на работе всей сети. Основным аргументом при работе в сети на основе выделенного сервера является, как правило, защита данных.

В таких сетях, например как Windows NT Server, проблемами безопасности может заниматься один администратор. Поскольку жизненно важная информация расположена централизованно, то есть, сосредоточена на одном или нескольких серверах, нетрудно обеспечить ее регулярное резервное копирование.

Благодаря избыточным системам данные на любом сервере могут дублироваться в реальном времени, поэтому в случае повреждения основной области хранения данных информация не будет потеряна легко воспользоваться резервной копией.

Контрольные вопросы:

1. Какие основные отличия серверного ПО от локального?
2. Какие типы серверного ПО вы знаете?
3. Назовите программные средства для серверов;
4. Назовите программные средства диагностики.

ЛЕКЦИЯ 16

Решение проблем конфигурации с помощью групповых политик.

План.

1. Контрольный опрос
2. Понятие пользователей и групп
3. Преимущества группового администрирования
4. Типы политик использования

Что такое групповые политики и зачем они нужны?

Групповая политика - это инструмент, доступный для администраторов, работающих с архитектурой Active Directory. Он позволяет централизованно управлять настройками на клиентских компьютерах и серверах, подключенных к домену, а также обеспечивает простой способ распространения программного обеспечения.

Групповые политики позволяют настраивать параметры для определенного набора пользователей или компьютеров внутри домена Active Directory. Также позволяют указать политики в одном месте для группы и применить к целевому набору пользователей.

Например, можно обеспечить применение стандартного набора настроек и конфигураций для групп пользователей или компьютеров в домене или по запросу. Во всех компаниях как правило есть различные отделы, например отдел системных администраторов, разработчиков, дизайнеров, каждому из отдела необходим свой стандартный набор программного обеспечения, их рабочие компьютеры должны быть сконфигурированы под специальные задачи и нужды. С помощью групповых политик можно создать наборы настроек для конкретных групп пользователей в домене. С помощью Active Directory GPO можно установить и управлять отдельными унифицированными наборами настроек, конкретно для дизайнеров или разработчиков.

Конфигурации для компьютеров или пользователей проще и эффективнее, т.к. расположены в одном месте и не требуют повтора на каждом компьютере.

Компоненты GPO

Существует два компонента групповых политик - серверный компонент и клиентский, т.е. данная структура относится к архитектуре “клиент-сервер”.

Серверный компонент - оснастка Microsoft Management Console (MMC), которая используется для указания настроек групповой политики. MMC может быть использована для создания политик для контроля и управления административными шаблонами и настройками безопасности (скрипты, установка ПО и прочее). Каждый из них называется расширением и в свою очередь каждый из них имеет дочернее расширение, которое разрешает добавление новых компонентов или обновление существующих без возможности затронуть или подвергнуть риску всю политику.

Клиентский компонент интерпретирует и применяет настройки групповой политики для компьютеров пользователей или целевым пользователям. Клиентские расширения - это компоненты, которые запущены на пользовательской системе и несут ответственность за интерпретацию обработки и применения в объекты групповой политики.

Для администрирования GPO используют Group Policy Management Console (GPMC) и Group Policy Management Editor.

Сценарии использования Active Directory GPO:

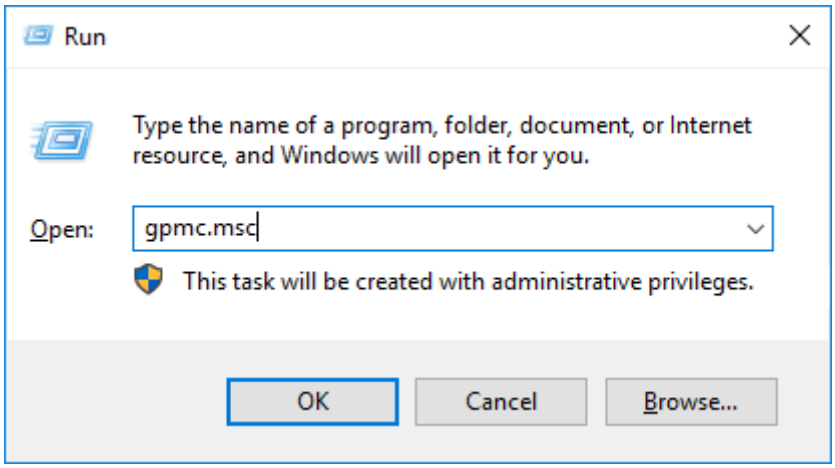
- Централизованная настройка пакета программ Microsoft Office.
- Централизованная настройка управлением питанием компьютеров.
- Настройка веб-браузеров и принтеров.
- Установка и обновление ПО.
- Применение определенных правил в зависимости от местоположения пользователя.
- Централизованные настройки безопасности.
- Перенаправление каталогов в пределах домена.
- Настройка прав доступа к приложениям и системным программам.

Оснастка Управление групповыми политиками

После установки роли Active Directory Domain Service (AD DS) на контроллер домена на сервере появится оснастка **Group Policy Management**. Для того, чтобы ее открыть нажмите комбинацию клавиш *Win+R* и в открывшемся окне введите:

gpmc.msc

Нажмите OK.



Если оснастку не удается открыть, то возможно по определенным причинам она не установлена. Установить ее можно через стандартное меню **Add roles and features** в диспетчере сервера, выбрав компонент **Group Policy Management**.

Контрольные вопросы:

1. Что такое групповые политики и зачем они нужны?
2. Какие преимущества политики дают для администраторов
3. Какие типы групповых политик вы знаете?
4. Как осуществляется управление групповыми политиками?
5. Назовите Компоненты GPO.

ЛЕКЦИЯ 17

Управление и обслуживание ИС.

План

1. Контрольный опрос
2. Виды испытаний ИС
3. Функциональное тестирование
4. Не функциональное тестирование

Основные теоретические сведения

Виды испытаний (тестирования) информационной системы

Испытание информационной системы и тестирование программного продукта на первый взгляд одно и то же, но на практике это не совсем так. Если учесть, что информационная система – это не только используемые в ее составе программные компоненты, но и аппаратное и организационное обеспечение, то и в результатах ее испытаний должны быть отражены показатели выбранных серверов, рабочих станций, сетевого оборудования (их надежность и производительность), а также эффективность разработанного регламента эксплуатации системы. Все виды испытаний информационной системы можно разделить на функциональные и нефункциональные тесты.

Функциональное тестирование призвано показать (доказать), что автоматизированные рабочие места информационной системы предоставляют пользователям ровно ту функциональность, которую они от нее ожидают. Система выполняет свои функции корректно.

Нефункциональное тестирование подтверждает или опровергает соответствие таких свойств информационной системы, как производительность, надежность, эргономичность и т.д. заданным на этапе ее проектирования параметрам. Система выполняет свои функции в срок, в должном объеме и с приемлемым качеством, и пользоваться ею удобно.

Виды функционального тестирования

Компонентное тестирование – испытание отдельных программных компонентов информационной системы, в ходе которых подтверждается корректность проводимых этими компонентами вычислений.

Интеграционное тестирование – испытания, направленные на выявление проблем взаимодействия отдельных компонентов системы. Если программная архитектура информационной системы довольно сложная, то в ней выделяются подсистемы, для каждой из которых проводят последовательно компонентное и интеграционное тестирование. В завершении проводят интеграционное тестирование всех выделенных подсистем, как компонентов единой системы.

Тестирование прототипа – испытания информационной системы на первых этапах ее разработки, когда готовы не все ее функциональные блоки. Отсутствующие компоненты заменяются функциональными заглушками, имитирующими их будущую работу. Информационная система на данном этапе представляет собой прототип целевого программного продукта.

Виды нефункционального тестирования

Нагрузочное тестирование (load testing) – испытание информационной системы в условиях прогнозируемой нормальной нагрузки. Под величиной нагрузки понимается количество запросов к системе, которое она должна успевать обрабатывать, не превышая определенное исходными требованиями время отклика.

Стрессовое тестирование (stress testing) – испытание информационной системы в условиях минимальных аппаратных ресурсах и максимально допустимой нагрузки. Цель стрессового тестирования, как понятно из названия, - проверить работоспособность системы в стрессовых ситуациях.

Объемное тестирование (volume testing) – испытания информационной системы в условиях максимальных (предельно допустимых) объемов информации в базе данных. Основным объектом тестирования в данном случае является зависимость времени отклика и прочих аспектов производительности системы от объемов контролируемых данных.

Тестирование стабильности (stability testing) – проверка, может ли испытываемая информационная система длительное время нормально функционировать в условиях, близких к нормальным условиям (средняя нагрузка, средние объемы данных, рекомендуемые аппаратное ресурсы и т.д.).

Тестирование надежности (reliability testing) – гибрид всех перечисленных ранее видов тестирования, направленный на то, чтобы проверить способность системы

возвращаться к нормальному режиму работы после коротких периодов максимальной нагрузки, стрессов, предельных объемов данных и т.д.

Тестирование эргономики решений – испытания пользовательского интерфейса на предмет удобства и безопасности эксплуатации информационной системы.

Испытание информационной системы на этапах подготовки к эксплуатации

После завершения этапа реализации информационной системы Разработчик, совместно с Заказчиком, может проводить следующие виды испытаний.

Тестирование процесса установки (installation testing) – проверка корректности развертывания программных компонентов системы в различных ее конфигурациях, предусмотренных исходными требованиями.

Тестирование на различных конфигурациях (configuration testing) - проверка работоспособности системы при развертывании отдельных ее компонентов (серверной части, клиентских рабочих мест) в условиях всех возможных (предусмотренных исходными требованиями) вариантах операционных систем и конфигурациях аппаратных и программных ресурсов.

Приемочное тестирование (acceptance testing) – комплексное испытание информационной системы, выполняемое представителями Заказчика по специально разработанной Исполнителем программе и методике испытаний (ПМИ). Цель приемочного испытания – показать, что разработанная и развернутая на территории Заказчика информационная система делает ровно то, что от нее требуется и делает это с заданными параметрами производительности. В программу приемочных испытаний, помимо функциональных тестов, могут входить и тестирование процесса установки системы и тестирование ее работы на различных конфигурациях, а также все виды нефункционального тестирования.

Особенность приемочных испытаний, в сравнении с прочими этапами функционального и нефункционального тестирования как раз в том, что тестируемое решение развернуто на целевых аппаратных и системных программных ресурсах Заказчика (или арендованных Заказчиком), проводится представителями Заказчика (будущими пользователями) по программе, согласованной с Заказчиком. Решение об успешности приемочного тестирования также принимает Заказчик, переводя систему в эксплуатацию или отправляя ее на доработку.

Испытание информационной системы на этапах ее сопровождения

Регрессионное тестирование (regression testing) – тестирование, проводимое по результатам исправления обнаруженных дефектов и ошибок в работе системы и направленное на то, чтобы показать - исправленный дефект или ошибка в настоящий момент не проявляются, а целевая функциональность системы не нарушена.

Предварительное или дымовое тестирование (smoke testing) - вид испытаний, проводимый после выхода новой версии программных компонентов, входящих в состав информационной системы, целью которого является быстро показать общую работоспособность или неработоспособность системы. Если после установки новых версий программных продуктов от системы “не пошел дым”, то это означает, что на первый взгляд все работает, и можно приступать к более детальным видам тестирования. Дымовое тестирование позволяет экономить время, поскольку длится намного меньше, чем весь остальной комплекс испытаний, а его отрицательный результат говорит о том, что дальше можно и не продолжать, поскольку дефекты сборки уже обнаружены.

Контрольные вопросы:

1. Какие виды испытаний ИС вы знаете?
2. Что такое функциональное тестирование?
3. Что такое не функциональное тестирование?