Анализ кода и информационная безопасность

Лекция 01

Технологии разработки безопасного ПО

МГУ / ВМК / СП

0101

МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА ПО

Жизненный цикл ПО

- Жизненный цикл ПО — непрерывный процесс, начинающийся с момента принятия решения о создании ПО, до момента полного прекращения его использования.
- Создание сложного ПО — составная задача, для решения декомпозируется на более простые задачи.
- Для описания используются термины: процессы, виды деятельности.
- Большое количество стандартов: ГОСТ Р ISO/МЭК 12207–2010 (43 процесса), ГОСТ Р ISO/МЭК 15288–2005, IEEE 1074-1997…
- Модель жизненного цикла — попытка описать процесс разработки любого ПО.

Этапы жизненного цикла

1. Выработка требований — функциональных и нефункциональных, оформление спецификации.
2. Анализ и проектирование отдельных компонент и взаимодействий между ними.
3. Разработка и реализация компонент и их отладка.
4. Тестирование и проверка на соответствие спецификации.
5. Внедрение: подготовка дистрибутива и развёртывание ПО.
6. Поддержка и сопровождение ПО в процессе и эксплуатации. Выявление и исправление ошибок, расширение возможностей.

Разработка требований

- Функциональные требования — ответ на вопрос «что должна делать программа?».
- Нефункциональные требования — ответ на вопрос «как она должна это делать?»:
  - атрибуты качества (скорость работы, надёжность…);
  - ограничения реализации (ОС, библиотеки…);
  - взаимодействие с пользователем (GUI, CLI…);
  - внешние интерфейсы (входы и выходы).
- Результат оформляется в виде технического задания, тактико-технических требований, спецификации.

Анализ и проектирование

- Выделение функциональных компонент, их интерфейсов и связей между ними.
- Нужно учитывать текущие функциональные и нефункциональные требования:
  - функционал;
  - библиотеки, среда разработки;
  - внешние интерфейсы.
- Возможное расширения требований (минимизация глобальных изменений).
- Количество разработчиков (независимость компонент).

Результат может оформляться, например, в виде UML-диаграмм.

Реализация и отладка

- Написание кода, в том числе с использованием сторонних компонент.
- Компиляция, выявление и устранение ошибок времени компиляции.
- Сборка кода с компоновкой с собственными и сторонними библиотеками.
- Отладка для выявления и устранения ошибок времени выполнения.
- Профилирование для оценки производительности и выявления узких мест.

Используются: среды разработки, системы контроля версий, отладчики, профилировщики, CI.

Тестирование

- Функциональное — проверка корректной обработки на подготовленном наборе входных данных
- Нефункциональное:
  - производительность;
  - пользовательский интерфейс;
  - совместимость;
  - масштабируемость;
  - надёжность;
  - защищённость.

Используются: системы тестирования и инструментирования, CI.

Внедрение

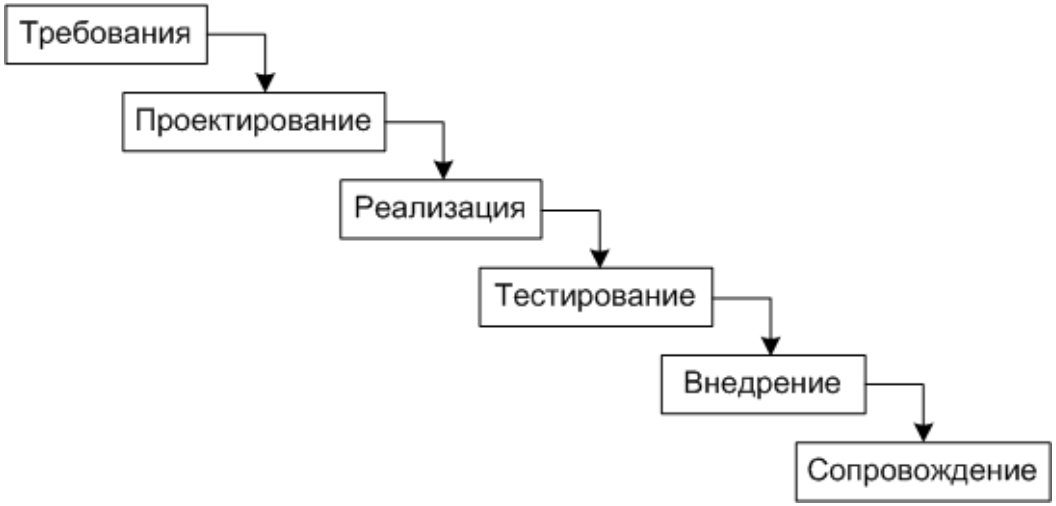
- Подготовка дистрибутива:
  - упаковка программы и данных;
  - разрешение внешних зависимостей — программ, библиотек;
  - продумывание системы распространения обновлений;
  - выбор инсталлятора.
- Развёртывание:
  - в простейшем случае — установка программы;
  - в более сложном — установка и настройка всех программных компонент (библиотек, БД, web-серверов и т.д.), в том числе на нескольких компьютерах, настройка их взаимодействия и т.д.

Поддержка и сопровождение

- Реакция на возникающие непредвиденные ситуации, такие как ошибки, аварийные завершения, неожиданное поведение:
  - выявление ситуаций;
  - определение условий их возникновения;
  - распределение работ по их анализу и устранению;
  - устранение ошибки;
  - отслеживание состояния работ;
  - выпуск и распространение обновлений.
- Возникновение новых и уточнение старых требований к программе в процессе эксплуатации.
- Доработка программы под обновлённые требования, расширение возможностей.
- Выпуск новых версий программы.

Каскадная модель

- Простейшая классическая модель жизненного цикла ПО — каскадная (модель водопада).
- Впервые сформулирована в 70х-80х годах.
- Основная идея: каждый вид деятельности выполняется на фиксированной фазе жизненного цикла, всё необходимое должно быть выполнено ранее.



Спиральная модель

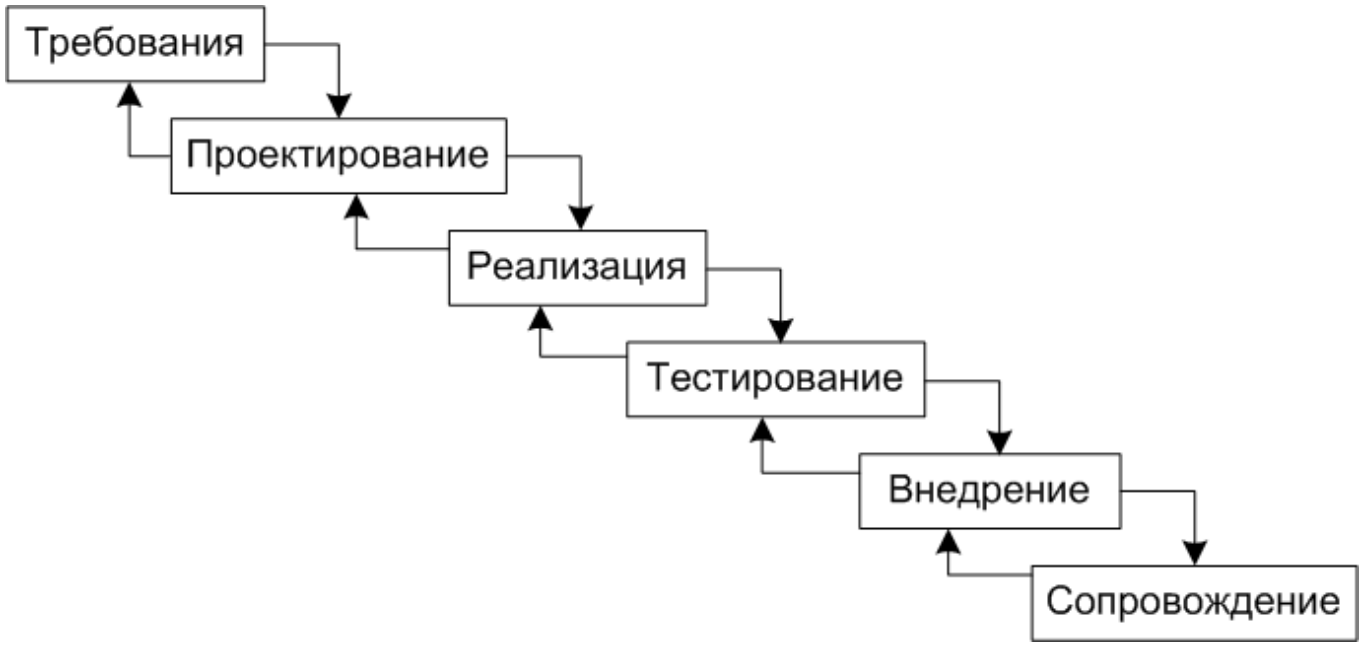
- Спиральная модель (водоворот) жизненного цикла ПО учитывает производство новых версий.
- Полярные координаты: угол — выполняемый этап, удаление от начала координат — версия, затраченные ресурсы, возможности, область применимости.

Ошибки на разных этапах

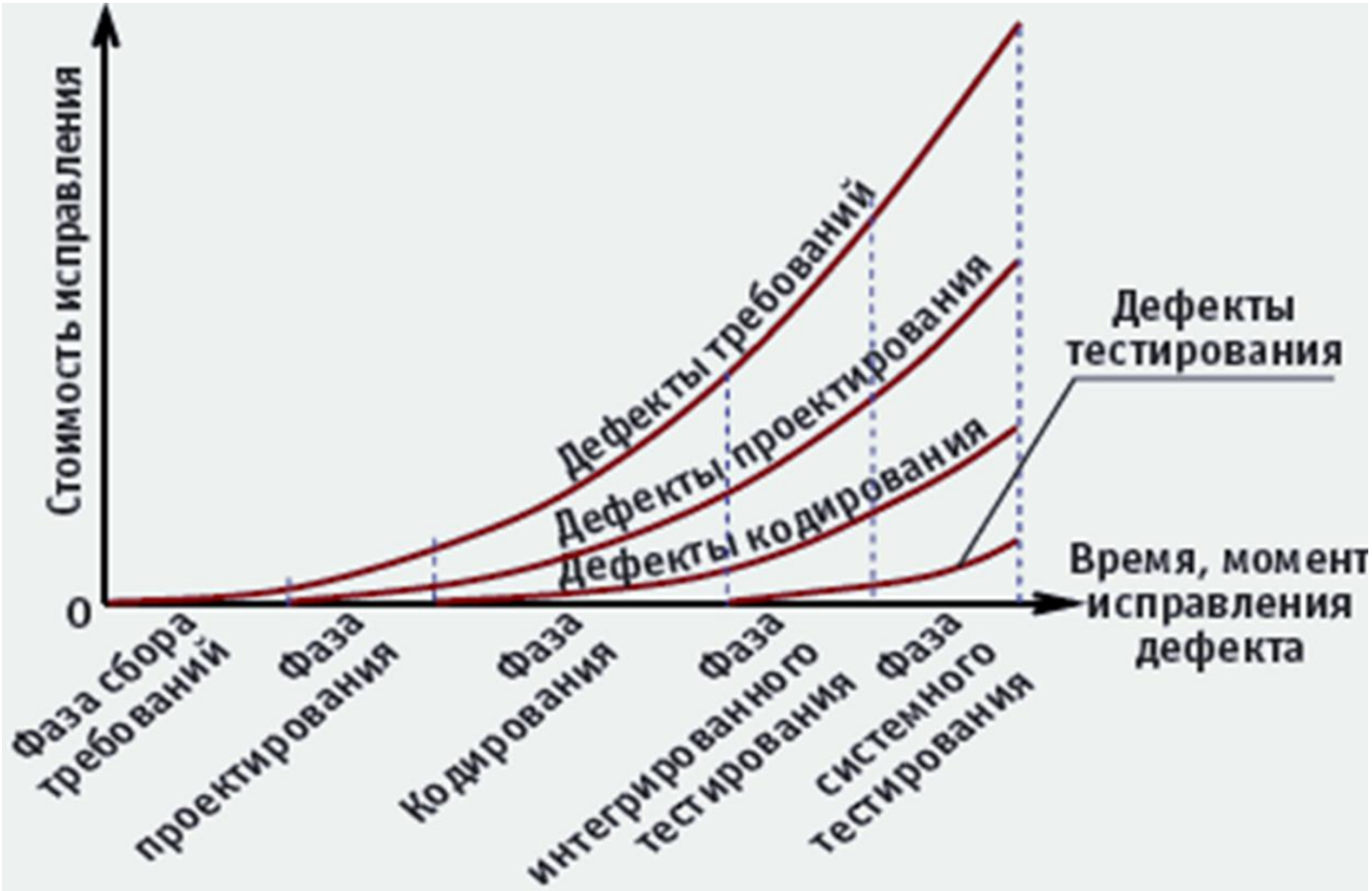
1. Выработка требований — не сформулирована часть функциональных или нефункциональных требований — несоответствие продукта потребностям заказчика.
2. Анализ и проектирование — неверное распределение функций по компонентам, неудачные интерфейсы, неподходящий выбор библиотек — затруднённая отладка и расширение, проблемы с производительностью.
3. Разработка и реализация — ошибки при реализации — некорректное поведение, аварийные завершения, уязвимости.
4. Тестирование и проверка — неполное покрытие тестами — обнаружение проблем на этапе эксплуатации.
5. Внедрение — ошибки при установке и настройке — неудобство/проблемы при эксплуатации.
6. Поддержка и сопровождение — плохая организация обратной связи, контроля за процессом исправления, доставки обновлений — задержка/невозможность получения своевременных обновлений.

Итеративная модель

1. Определение стадии возникновения ошибки.
2. Возвращение на эту стадию, исправление ошибки.
3. Переход на следующую стадию каскадной модели с учётом исправленной ошибки.



Цена ошибок



0102

ТЕХНОЛОГИИ БЕЗОПАСНОЙ РАЗРАБОТКИ ПО

Жизненный цикл, направленный на обеспечение безопасности

- Дополнение к классическим моделям жизненного цикла.
- Параллельно разработке ПО осуществляются мероприятия для проверки и повышения безопасности ПО.
- Для каждой стадии разработки добавляются требования и шаги, только после осуществления которых можно переходить к следующей стадии.
- План реагирования на инциденты.

Предпосылки возникновения SDL

- 2002–2004 гг.: создание группы реагирования на инциденты в Microsoft.
- Сбор и анализ статистики по ошибкам и исправлениям.
- Выводы:
  - о безопасности нужно думать до и во время создания продукта, а не после;
  - большинство ошибок являются следствием небольшого числа причин;
  - любые внешние данные потенциально опасны.

Принципы безопасной разработки

1. Безопасный дизайн — при разработке архитектуры и дизайна нужно учитывать, что ПО должно защищать себя, информацию, которую оно обрабатывает и противостоять атакам.
2. Безопасность по умолчанию — программа почти наверняка содержит уязвимости, поэтому требуется минимизировать последствия их эксплуатации — минимизация привилегий, ограничение числа пользователей с высоким уровнем доступа.
3. Безопасность при внедрении — инструменты и описания для облегчения конфигурирования ПО и установки обновлений.
4. Взаимодействие — реагирование на обнаруженные уязвимости и помощь для противодействия угрозам.

Примеры циклов безопасной разработки

- Microsoft SDL (Security Development Lifecycle) — системное ПО;
- Cisco SDL (Secure Development Lifecycle) — ПО для сетевого оборудования;
- ГОСТ Р 56939–2016 «Защита информации. Разработка безопасного программного обеспечения. Общие требования».
- PA-DSS — рекомендации к процессам разработки платёжных систем;
- РС БР ИББС-2.6-2014 — рекомендации Банка России для обеспечения информационной безопасности банковских систем.

Microsoft SDL

Microsoft SDL

Обучение и разработка требований

- Обучение по безопасности:
  - все сотрудники;
  - не реже одного раза в год;
  - получение знаний для выполнения остальных этапов.
- Разработка требований:
  - определение требований информационной безопасности и ответственных:
    - по безопасности;
    - по конфиденциальности;
  - оценка рисков:
    - выделение участков кода;
    - назначение приоритета участкам кода.

Microsoft SDL

Проектирование и реализация

- Проектирование:
  - архитектурные требования безопасности:
    - используемые криптографические алгоритмы;
    - настройки межсетевых экранов;
  - анализ и сокращение поверхности атаки:
    - все каналы получения данных извне;
  - моделирование угроз (модель STRIDE):
    - «кто», «что» и «как» может испортить?
- Реализация:
  - использование доверенных средств;
  - отказ от небезопасных функций;
  - статический анализ кода;
  - перекрестная проверка кода.

Microsoft SDL

Проверка, выпуск и реагирование

- Проверка:
  - динамический анализ — мониторинг выполнения, обнаружение повреждения памяти и других проблем;
  - фаззинг — многократный запуск на намеренно повреждённых входных данных;
  - проверка поверхности атаки.
- Выпуск:
  - планы реагирования на инциденты;
  - финальный анализ безопасности;
  - доверенный выпуск.
- Реагирование:
  - выполнение ранее подготовленных планов.

Cisco SDL

Cisco SDL

Требования безопасности и проектирование

- Требования безопасности:
  - внутренние требования ИБ;
  - внешние требования (требования рынка);
  - безопасность сторонних компонент;
  - реестр используемого стороннего ПО;
  - уведомление об уязвимостях и централизованное исправление.
- Проектирование:
  - проектирование с учётом ИБ (обучение, лучшие практики, безопасные компоненты);
  - моделирование угроз (модель STRIDE).

Cisco SDL

Реализация, статический анализ и тестирование

- Реализация:
  - обучение;
  - использование безопасных компонент;
  - проверка кода, статический анализ;
  - использование стандартов безопасного программирования.
- Статический анализ:
  - поиск переполнений буфера;
  - отслеживание распространения внешних данных.
- Тестирование:
  - анализ поверхности атаки;
  - тесты для всех протоколов;
  - тестирование на проникновение (пентестинг).

ГОСТ Р 56939—2016

- Название: «Защита информации. Разработка безопасного программного обеспечения. Общие требования».
- Рассматриваемые этапы:
  - анализ требований;
  - проектирование архитектуры;
  - конструирование и комлексирование;
  - квалификационное тестирование;
  - инсталляция и поддержка приёмки;
  - решение проблем в процессе эксплуатации;
  - менеджмент документации и конфигурации;
  - менеджмент инфраструктуры среды разработки;
  - менеджмент людских ресурсов.

ГОСТ Р 56939—2016

Анализ требований и проектирование архитектуры

- Анализ требований:
  - разработчик должен определить требования по безопасности;
  - примеры:
    - обеспечение контроля доступа;
    - обеспечение регистрации событий;
    - контроль точности, полноты и правильности входных данных.
- Проектирование архитектуры:
  - моделирование угроз безопасности информации;
  - уточнение проекта с учётом результатов моделирования;
  - конкретная методология моделирования угроз не фиксируется.

ГОСТ Р 56939—2016

Конструирование и комплексирование

- Конструирование и комплексирование:
  - использование при разработке идентифицированных инструментальных средств;
  - создание программы на основе уточнённого проекта архитектуры;
  - создание (выбор) и использование при создании программы порядка оформления исходного кода программы;
  - статический анализ исходного кода программы;
  - экспертиза исходного кода программы.

ГОСТ Р 56939—2016

Квалификационное тестирование; инсталляция и поддержка приёмки

- Квалификационное тестирование:
  - функциональное тестирование программы;
  - тестирование на проникновение (пентестинг);
  - динамический анализ кода программы;
  - фаззинг.
- Инсталляция и поддержка приёмки:
  - обеспечение защиты ПО от угроз безопасности информации, связанных с нарушением целостности, в процессе его передачи пользователю;
  - поставка пользователю эксплуатационных документов.

ГОСТ Р 56939—2016

Эксплуатация и менеджмент

- Решение проблем в процессе эксплуатации:
  - реализация и использование процедуры отслеживания и исправления ошибок ПО и уязвимостей программы;
  - систематический поиск уязвимостей программы.
- Менеджмент документации и конфигурации:
  - реализация и использование процедуры уникальной маркировки каждой версии ПО;
  - использование системы управления конфигурацией ПО.
- Менеджмент инфраструктуры среды разработки:
  - защита от несанкционированного доступа к элементам конфигурации;
  - резервное копирование элементов конфигурации;
  - регистрация событий изменения элементов конфигурации.
- Менеджмент людских ресурсов:
  - периодическое обучение сотрудников;
  - периодический анализ программы обучения сотрудников.

Поверхность атаки

- Сетевая — открытые сетевые соединения, именованные каналы, RPC.
- Локальная — файлы, переменные окружения, реестр.
- Пользовательская — учётные записи:
  - гостевые (анонимные);
  - пользовательские;
  - административные.
- Программная — запущенные сервисы:
  - с пользовательскими правами;
  - с повышенными правами.

Уменьшение поверхности атаки

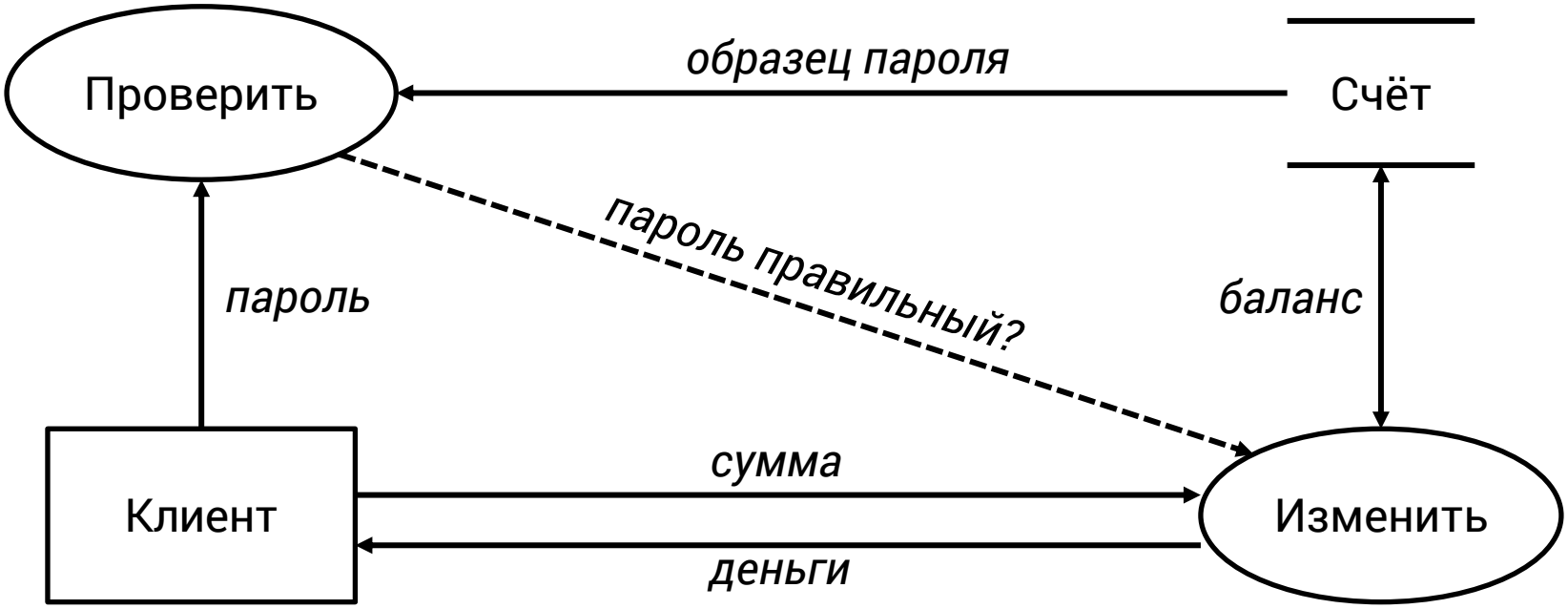
- Уменьшение количества открытых сетевых соединений.
- Отключение ненужных сервисов.
- Запуск сервисов по требованию вместо постоянной работы.
- Контроль доступа вместо анонимного.
- Доступ к критическим компонентам только из локальной сети, не из интернета.
- Изменение настроек в случаях, когда настройки по умолчанию недостаточно безопасны.
- Уменьшение количества кода, обрабатывающего внешние данные.

Модель классификации угроз STRIDE

- S (***s***poofing of identity) — возможность аутентификации под чужой учётной записью;
- T (***t***ampering with data) — возможность изменения информации в обход контроля доступа;
- R (***r***epudiation) — возможность отрицания совершённых действий;
- I (***i***nformation disclosure) — возможность раскрытия информации в обход контроля доступа;
- D (***d***enial of service) — возможность вызвать отказ в обслуживании;
- E (***e***levation of privilege) — возможность получить повышенные права в обход контроля доступа.

Контекст (3A): «активы», «актёры», «действия».

Диаграммы потоков данных



Заключение

- Ошибки лучше обнаруживать как можно раньше.
- Существует много технологий для обеспечения безопасной разработки.
- Их использование усложняет процесс разработки, увеличивает цену и длительность, но это окупается.
- Самые известные циклы безопасной разработки — Microsoft SDL и Cisco SDL. Они сильно похожи, но имеется специфика, связанная с предметной областью ПО.
- В 2016 г. принят ГОСТ, который задаёт требования по разработке безопасного ПО сразу для всей отрасли.

Литература к лекции

Основные источники

1. ГОСТ Р ISO/МЭК 12207–2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств.
2. Общие сведения о реализации процесса Microsoft SDL.
3. Building Trustworthy Systems with Cisco Secure Development Lifecycle.
4. ГОСТ Р 56939—2016. Защита информации. Разработка безопасного программного обеспечения. Общие требования.