Digital Security

От анализа кода до пентестов: обеспечение безопасности систем ДБО на всех этапах жизненного цикла

Илья Медведовский
к.т.н., Директор
Digital Security

Жизненный цикл системы ДБО

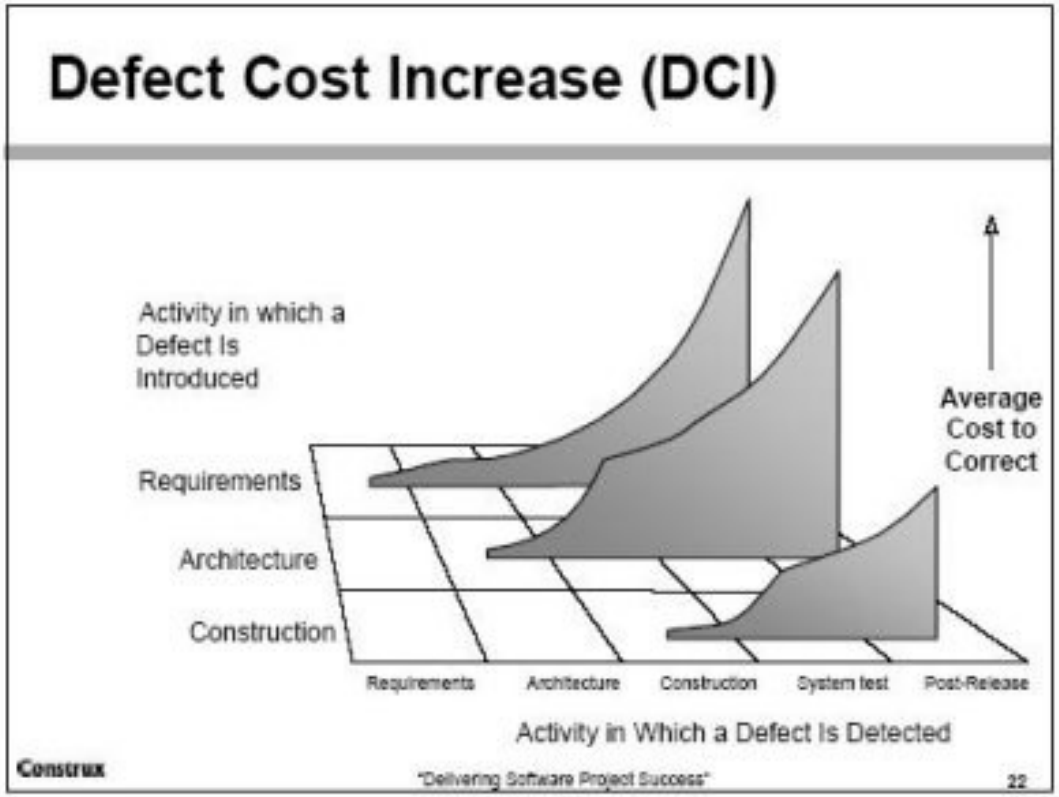
В своей основе системы ДБО не отличаются от других приложений.

Правильный подход уже выработан:
SDLC – Systems development life-cycle

Анализ безопасности ПО должен проводиться на каждом этапе жизненного цикла!

Только в этом случае мы получим приложение приемлемого уровня защищенности.

Трудозатратность исправления



Чем раньше выявлена, тем проще исправить

Жизненный цикл системы ДБО

Планирование
- Требования
- Модель угроз/Модель нарушителя

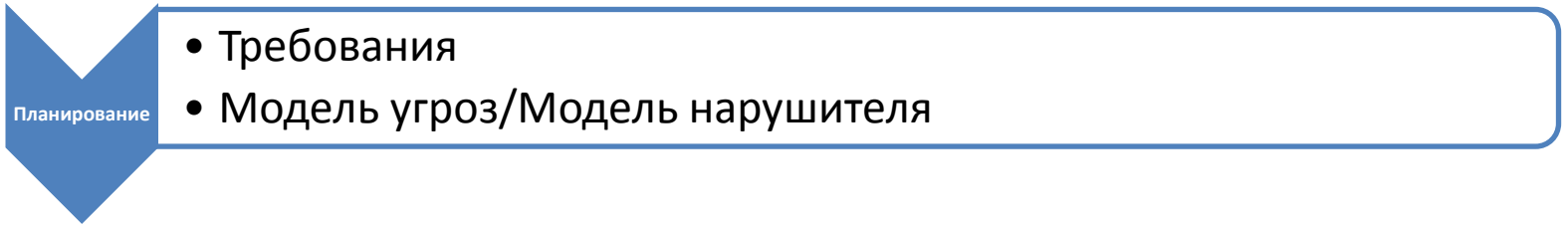
Проектирование
- Анализ архитектуры
- Анализ рисков

Разработка
- Статический анализ кода

Тестирование
- Динамический анализ кода/Фаззинг

Внедрение
- Тесты на проникновение

Планирование

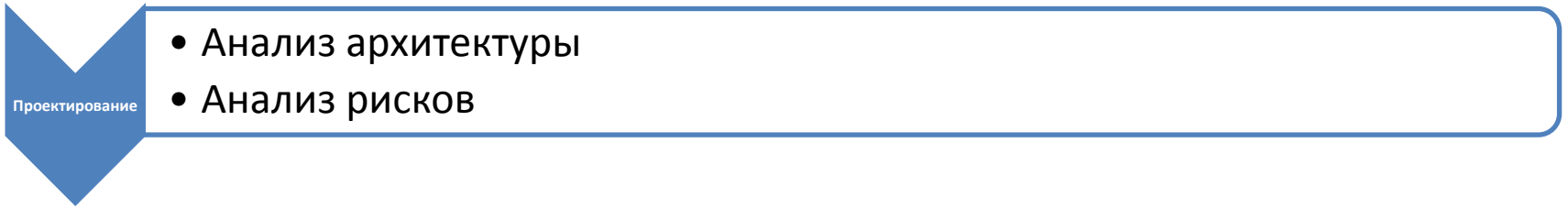


- По некорректному ТЗ (Требованиям) невозможно создать защищенную архитектуру будущего приложения, если ТЗ противоречит основам безопасности.

- Хорошее ТЗ позволит заранее избавиться от ряда проблем с безопасностью.

➢Необходим контроль со стороны специалистов в области безопасности.

Проектирование



- Выбор некорректной архитектуры приложения, даже при правильном программировании и качественном внедрении, приведет к тому, что злоумышленник сможет скомпрометировать Систему.

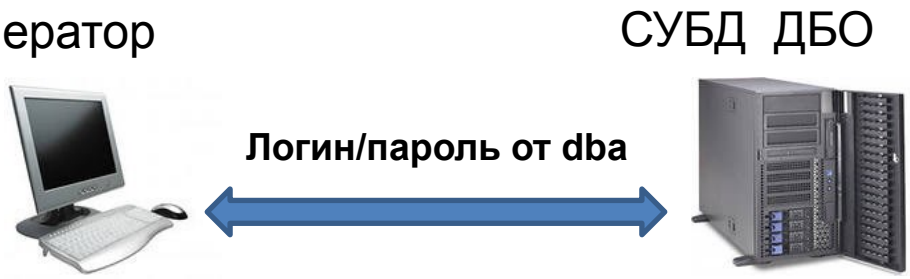
➢Пример BSS (APM)

Проектирование

Проектирование

- Анализ архитектуры
- Анализ рисков

➢Пример BSS (АРМ)



1) Оператор входит под привилегированной учетной записью СУБД.
2) Оператор ограничен на клиентской стороне возможностями в интерфейсе АРМ.
3) Зашифрованный пароль хранится в текстовом файле
4) Пароль зашифрован, но ключ единый и прошит во всех приложениях BSS

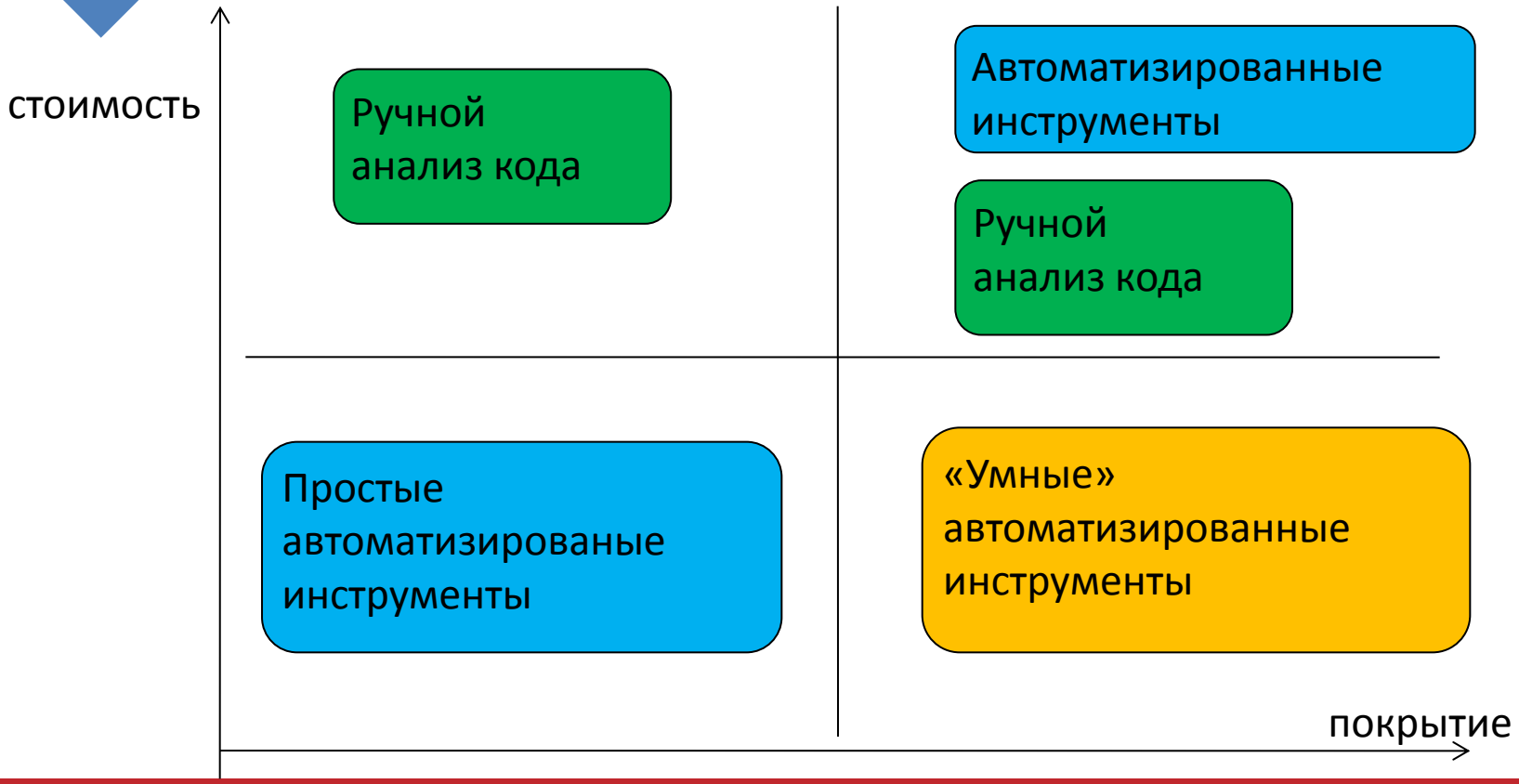
Итог: **Оператор = > Администратор СУБД ДБО = > Возможность украсть деньги у любого клиента банка.**

Причина: Отсутствие анализа архитектуры на стадии проектирования.

Разработка

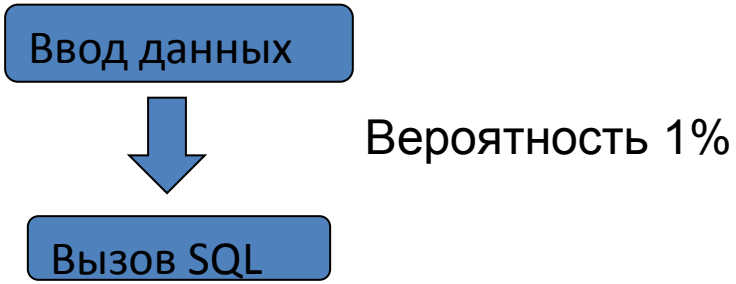
Разработка

- Статический анализ кода

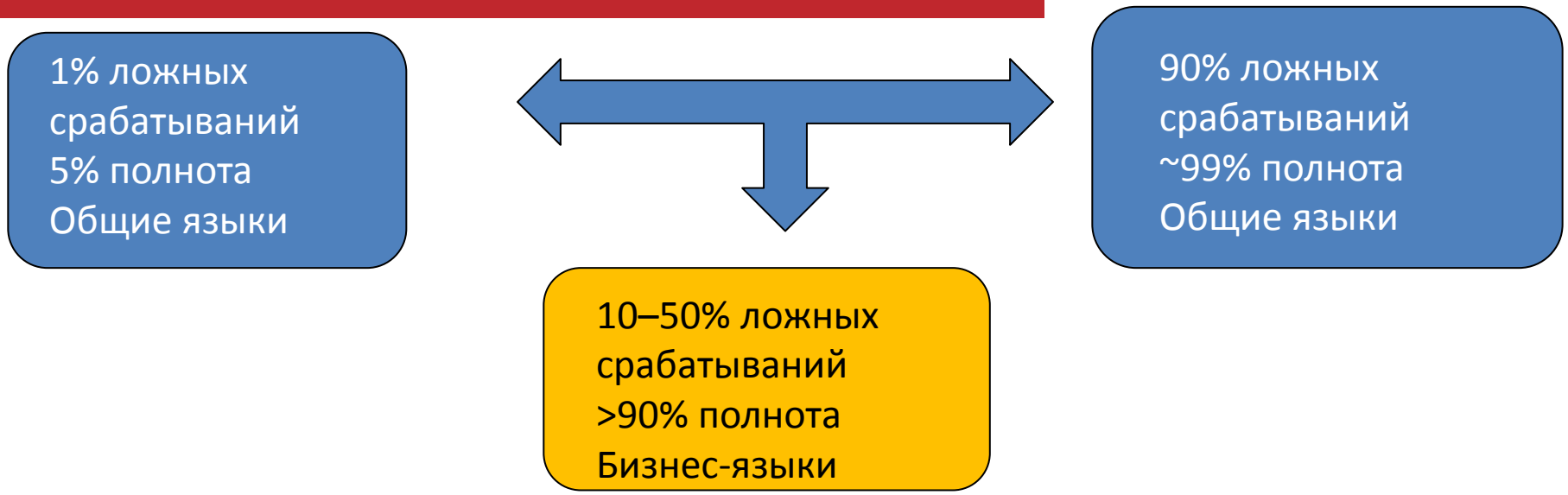


Сигнатурный анализ

- Ищет совпадения сигнатур в коде
- Высокая скорость работы
- Недостатки:
  - Много ложных срабатываний при поиске уязвимостей
  - Не учитывается структура программы
  - Не находит уязвимости инъекции кода и SQL



Правильный подход



- Настройка под конкретный язык
- Градация уязвимостей по вероятности

80% 50% 20%

Тестирование

Тестирование

- Динамический анализ кода/Фаззинг

Фаззинг доступен как разработчику, так и злоумышленнику.
Кто из них первый займется этим? Явно не разработчик.

Пример:
Digital Security. “Результаты исследования безопасности банк-клиентов российских производителей за период 2009-2011 гг.” и ранее на конференции PCI DSS RUSSIA

БСС, Р-Стайл, Степ АП, ЦФТ, Сигнал КОС и др.
Во всех были выявлены уязвимости типа “memory corruption” (с возможностью выполнения произвольного кода).
Найдены они методами фаззинга, с помощью бесплатных стандартных средств.

Жизненный цикл системы ДБО

Внедрение

- Тесты на проникновение

В идеальном мире они нужны в основном для выявления ошибок конфигурации, логических уязвимостей ПО, а также для общей контрольной проверки.
Пример:
1) Встроенная система логирования событий, хранящая в себе все карточные данные клиентов ДБО, доступная анонимно из сети Интернет.
2) Неподключенная защита от CSRF-атак, позволяющая проводить нелегальные действия в системе от имени пользователя.

Фактически также включают:

- Динамический анализ кода/Фаззинг
- Анализ архитектуры

Позволяют понять реальный уровень защищенности системы ДБО.

Digital Security

Digital Security в Москве: (495) 223-07-86
Digital Security в Санкт-Петербурге: (812) 703-15-47

www.dsec.ru
info@dsec.ru