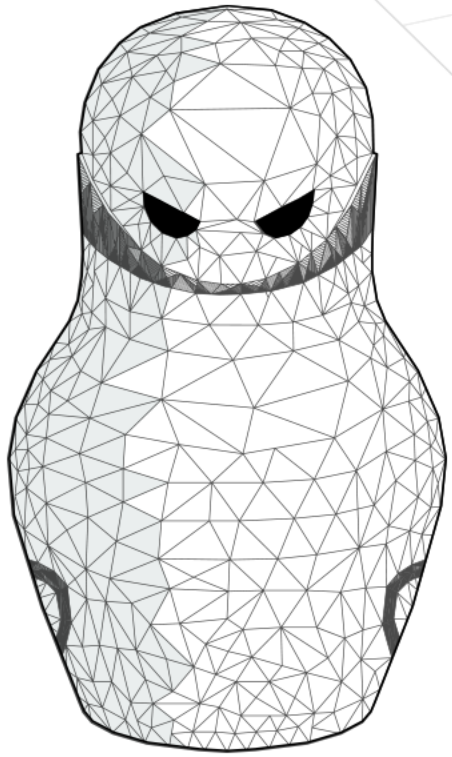


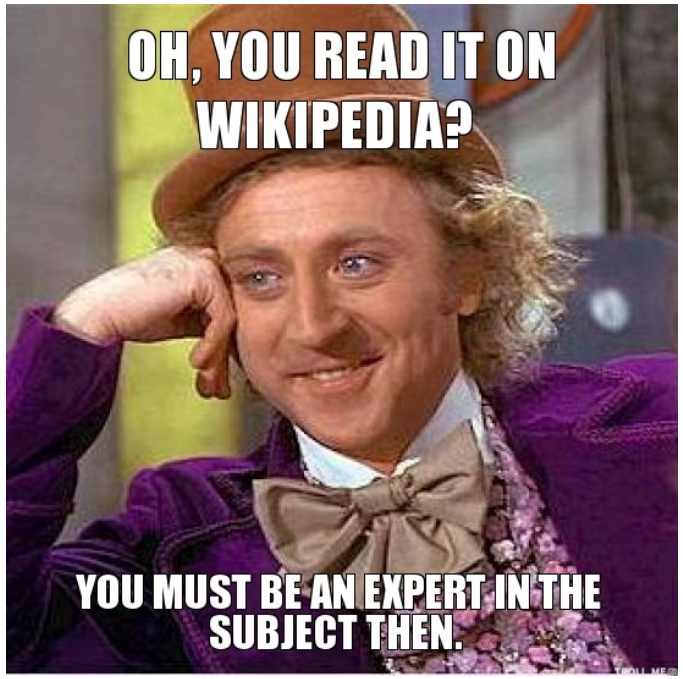
2015 ZERO NIGHTS

Банковский SDL своими руками

Шабалин Юрий

SDLC в классическом понимании

Жизненный цикл ПО — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Этот цикл — процесс построения и развития ПО.
Wikipedia



Этапы:

1. Формирование требований
2. Проектирование
3. Реализация
4. Тестирование
5. Внедрение
6. Эксплуатация и сопровождение

ЦБ РФ: ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ НА СТАДИЯХ ЖИЗНЕННОГО ЦИКЛА АВТОМАТИЗИРОВАННЫХ БАНКОВСКИХ СИСТЕМ

Ожидания руководства ИБ

- Закрытие существующих уязвимостей
- Обнаружение новых уязвимостей
- Повышение грамотности и осведомленности разработчиков
- РЕГЛАМЕНТ безопасной разработки ☹
- Объяснение уязвимостей языком, понятным разработчикам

Что мы имеем

- Изобилие систем собственной разработки (50+)
- Различные команды разработки
- Внешние (неподконтрольные) подрядчики для разработки
- Отсуствие Code Style, Code Review в командах
- Неграмотность разработчиков в области безопасности

Со стороны ИБ есть один человек, который занимается software security и попытками построения SDLC …

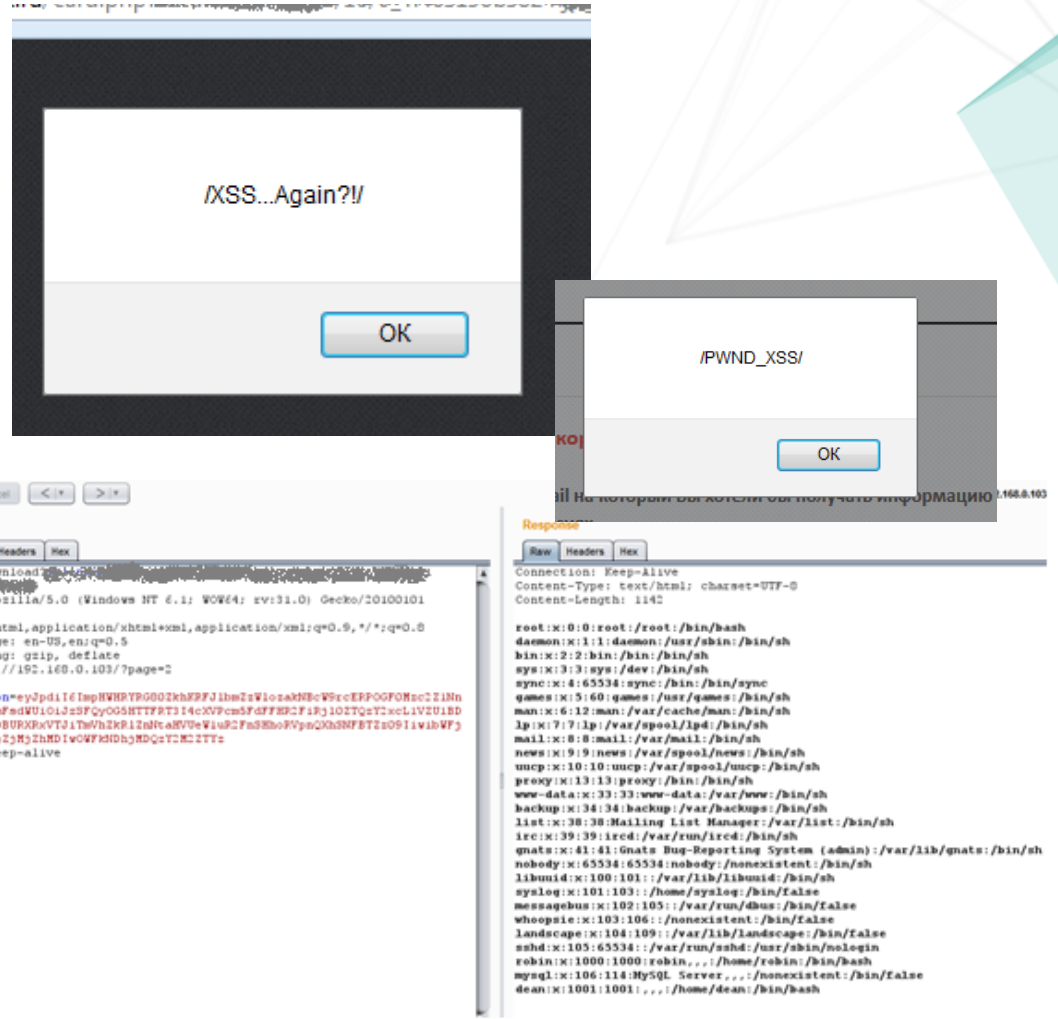
SDL(C)?

Основные проблемы

- Непонимание необходимости устранения уязвимостей
  Зачем устранять, 100 лет так жили? А сколько денег мы сэкономим?
- Непонимание сути уязвимостей вообще
  Да у нас всё зашифровано, всё под SSL, какая ещё SQL-инъекция?
- Нежелание отдавать исходный код безопасникам, традиционное отношение к нам, как к карательно-запретительному органу
- Ожидания разработчиков: обезьяна на инструменте с кнопкой FWD MAIL.

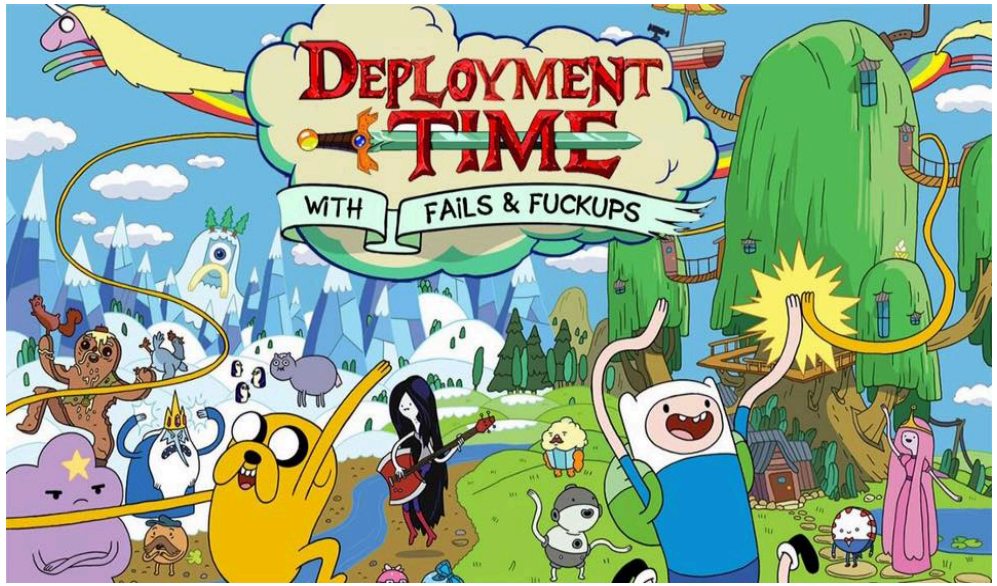
Чем убеждать

- Эксплуатация уязвимостей (скриншоты, описание реализации)
- Внятное описание проблем и последствий эксплуатации уязвимостей
  Понимается куда лучше, чем УГРОЗЫ КОНФИДЕНЦИАЛЬНОСТИ, НАРУШЕНИЕ ЦЕЛОСТНОСТИ, ДОСТУПНОСТИ, !!1!!1!
- Разъяснение, как именно нужно закрыть и что сделать, чтобы такая уязвимость не появилась снова.



Чем завоевать доверие

- Понимание процесса разработки изнутри
  Понимание проблем, ошибок, боли программистов. Как внедрить систему с минимальными потерями...
- Выстраивание дружественных отношений вместо регламентных
  Максимальное встраивание в процесс разработки, использование существующего иструментария (CI, Nightly Builds, Bug trackers)
- Выступаю "переводчиком" с языка отчётов по пентесту, оповещений CERT, писем “I HACK U GIMME MONEY”
  Объяснить, что имелось ввиду, как и где это поправить, что сделать, чтобы не повторялось в дальнейшем



Инструменты

Инструменты стандартны и не интересны*
Fortify, Burp, Metasploit, SQLmap, Acunetix, AppScan ...

FORTIFY®

AppScan
IBM. Rational.

BURPSUITE
PROFESSIONAL

sqlmap
Automatic SQL injection and database takeover tool

*Главное **не** использовать их из коробки там, где это возможно

Результат

- Закрыли кучу критических уязвимостей, через которые могли бы поломать клиентов и сервисы.
- Постоянный мониторинг изменения кода и добавления нового функционала
- Мониторим попытки эксплуатации исправленных уязвимостей на SIEM
- Что нельзя закрыть или оперативно исправить – закрываем виртуальными патчами на WAF
- Повышаем уровень доверия разработчиков к ИБ

  Некоторые команды сами приходят к нам и просят проверить приложение

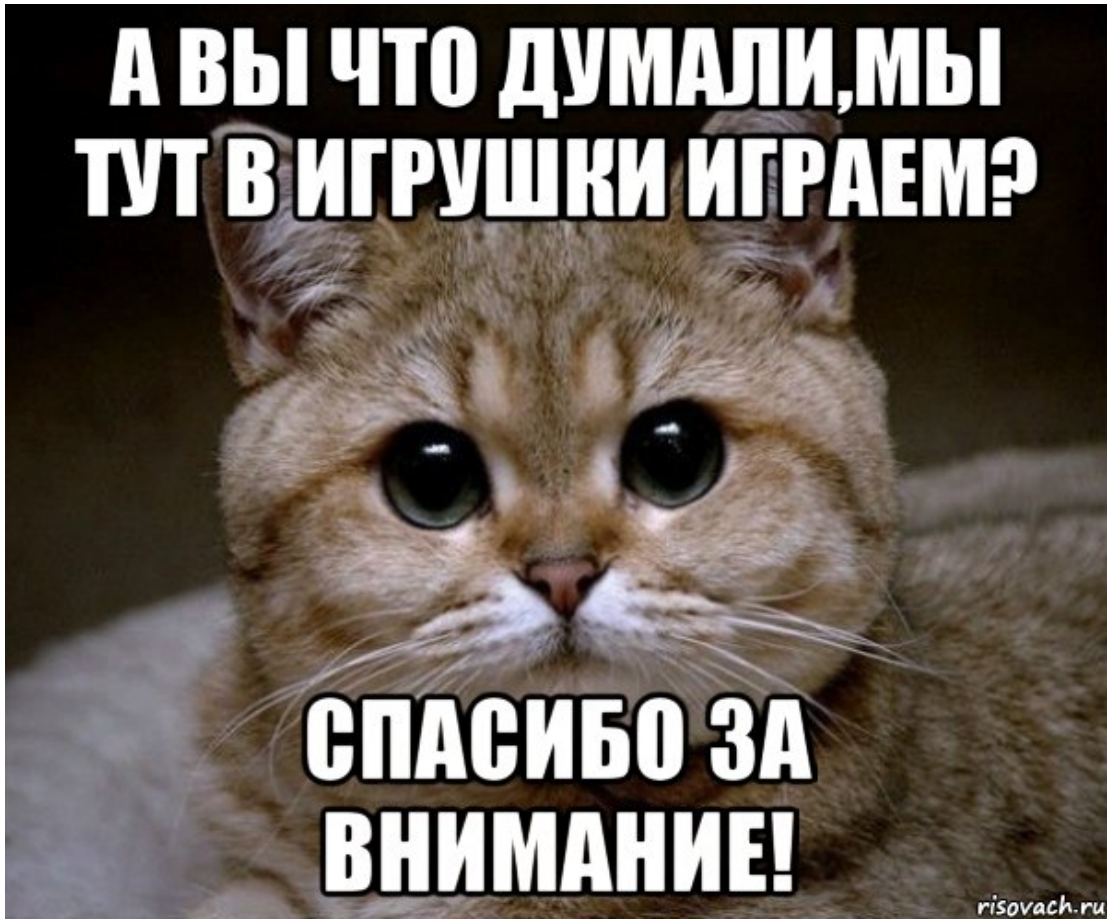
Текущий статус

- Регулярно анализируем исходный код приложений
- Добиваем внешних разработчиков на предоставление кода существующих систем
- Навязываем обязательный аудит исходников перед приёмкой приложения
- Сканирование стороннего кода при добавлении в проекты
- Периодически пентестим ресурсы Банка и помогаем исправлять выявленные уязвимости

Дальнейшие планы

- Развитие направления Application Security
- Полноценное включение в процесс динамических анализаторов и Web-сканеров

Вопросы?



*Special thanks to Mona for providing images

Yury.Shabalin@gmail.com https://www.linkedin.com/in/YuryShabalin