DevSecOps или безопасность на скорости DevOps

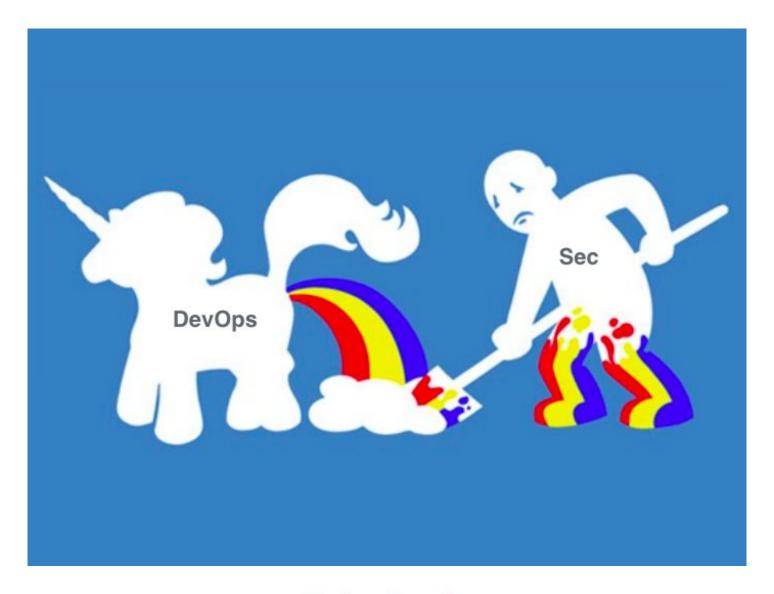
Денис Безкоровайный, генеральный директор ProtoSecurity

https://protosecurity.ruhttps://protosecurity.ru denis@protosecurity.ru



\$ whoami

- Занимаюсь информационной безопасностью более 12 лет
 - ProtoSecurity фокусируется на безопасности и мониторинге производительности веб-приложений
 - Certified Information Systems Auditor, CISA
 - Certified Information Systems Security Professional, CISSP
 - Certificate of Cloud Security Knowledge, CCSK



Почему это касается вас

- Более 50% веб-приложений всегда содержат уязвимости.
- На каждое веб-приложение в среднем приходится 15 уязвимостей, из которых 7 с критическим и высоким риском.
- Из этих 15 уязвимостей около половины будут рано или поздно устранены.
- Устранение каждой из уязвимостей займет в среднем 158 дней.

Не бывает неуязвимых приложений





Почему так?



100 K 1

на 100 разработчиков в среднем 1 специалист по безопасности

Bev€onf

Приоритеты или чего хотят

DevOps

- 1. Увеличить скорость деплоев
- 2. Уменьшить время на устранение сбоев
- 3. Снизить количество проблем при деплоях
- 4. Увеличить частоту деплоев
- 5. Вовлекать Безопасность в процесс разработки на более ранних стадиях
- 6. Ускорить устранение проблем с нормативными требованиями

Security

- 1. Снизить риски
- 2. Выполнить нормативные требования

Типичный цикл

- DevOps часы или дни
- Security недели и месяцы
 - Стандартные инструменты «безопасника» не подходят
 - Ручное управление и процессы
 - Аудит, пентесты, сканирование уязвимостей
- Dev и Ops не сильно вовлечены в процессы безопасности
 - Нет ни знаний, ни мотивации
 - Процессы безопасности не являются частью процесса разработки
 - Инструменты слишком далеки от стандартных инструментов разработчиков

CI/CD глазами Безопасности

- Увеличение скорости = потеря контроля
- Чаще релизы = больше уязвимостей
- Сложнее стэк = больше поверхность атаки



DevSecOps

Принципы DevSecOps

- Безопасность это общая ответственность всех команд
- Автоматизация безопасности для масштабирования

Основные задачи DevSecOps

- Безопасность среды CI/CD
 - Защищенность компонентов, управление учетными записями и ролями и тд
- Безопасность продукта при CI/CD
 - Автоматизированные тесты безопасности, анализ кода и тд
- Автоматизация безопасности
 - Автоматизация обнаружения инцидентов, форенсики и тд

Автоматизация Безопасности

- Меры защиты должны быть автоматизируемы
 - Снижает риск ошибки
 - Часто именно ошибки являются причиной успешных атак в дальнейшем и прочих инцидентов
 - Поддерживает скорость DevOps
- Необходимы API у всех ИБ-продуктов
 - Но сейчас это не так
 - Часто у ИБ есть только GUI
 - Нужен отдельный человек для управления

Нужна новая скорость для

- Обнаружения атак и инцидентов
- Оповещения по ним
- Устранения последствий
- Принятия контрмер и усиления политик
- Расследования/форенсики

Что можно (и нужно) автоматизировать

- Выявление уязвимостей инфраструктуры
- Выявление уязвимостей приложений
- Управление учетными записями и ролями
- Управление секретами, паролями, ключами, токенами и тд.
- Управление политиками FW

Security as code

- Фиксация политик безопасности и тестов в коде
 - доступ, конфигурации, логгирование и тд
- Автоматическая проверка соответствия на всех этапах CI/CD
- Security тесты как необходимая часть тестирования продукта
- Обновление, версионность и контроль
- Портативность политик
 - Не нужно быть безопасником, чтобы использовать

Рекомендации по управлению конфигурациями

- Процессы для автоматического сканирования всех образов систем, включая ОС, приложения и контейнеры для поиска ошибок конфигураций и уязвимостей
 - в том числе в open source software
- Релиз должен быть остановлен автоматически, если есть критические уязвимости
- Удаление всех ненужных модулей, применение политик настройки безопасности систем для соответствия лучшим практикам
- Чем проверять
 - InSpec, Serverspec, Amazon Inspector и др.



Безопасность продукта при CI/CD



Тестирование безопасности приложения DAST, SAST, IAST

- Необходимо автоматически проверять код и готовые приложения с помощью DAST, SAST и IAST
- Проверки должны встраиваться в обычную процедуру тестирования
 - Экономит ресурсы разработчиков (проще исправлять)

Проблемы с тестированием кода и приложений

- Ложные срабатывания
- Длительное время сканирования
- Обоснование и обсуждение находок
- Скорость устранения сотни уязвимостей

Что необходимо

- Инкрементальные сканирования
- Быстрые циклы проверки и обработки
- Отсев ложных срабатываний
- Использование существующих инструментов коммуникации

Пример SAST в SDLC

- 1. Инкрементальное сканирование нового кода в репозитории
- 2. Внешние Security-аналитики отсеивают ложные срабатывания сканера
- 3. Все актуальные находки приоритезируются
- 4. Создаются тикеты в баг-трекинге по каждой уязвимости
- 5. Повторять каждую неделю

Пример SAST в SDLC - еще быстрее

- 1. Инкрементальное сканирование нового кода в репозитории
- 2. Сканер ранжирует находки автоматически и добавляет прямо в код комментарии по уязвимостям
 - Не нужен новый пользовательский интерфейс
- 3. Повторять каждую ночь

Что изменить?

Что изменить? Роль Security специалиста

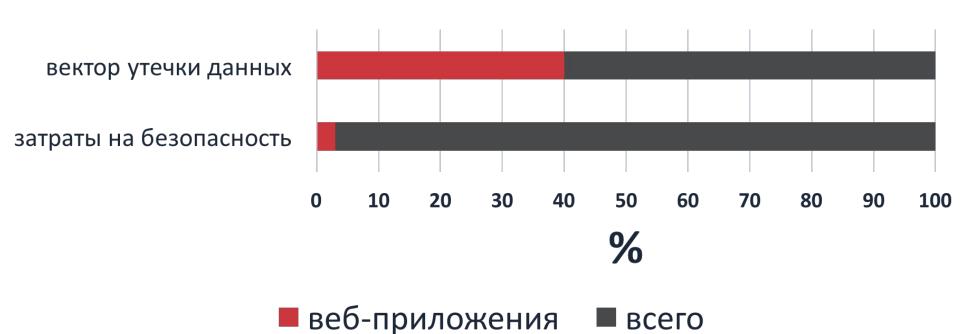
- Переход от «делаю сам» к «делаем вместе»
- Вместе с Operations определять политики, которые затем могут быть применены к системам автоматически, даже другими подразделениями

Что изменить? Навыки Dev и Ops специалистов

- Разработчикам НЕ нужно становиться безопасниками
 - Это невозможно
- Но нужно знать основы безопасной разработки
- Необходимо обучение:
 - Основы безопасной разработки
 - Типовым ошибкам/уязвимостям и как их избежать (не только OWASP Top 10)
 - Методам атак и как защищаться
 - Реальные примеры из реальной жизни
 - На тех языках, которые используются у вас
 - Интерактивные задания, а не скучные лекции

Что изменить? Инструменты безопасности

- 100% функций должны быть доступны через API
- Поддержка систем управления (Chef, Puppet, Salt и др)
- Поддержка контейнеров
- SAST и DAST сканеры должны поддерживать инкрементальное сканирование



Источник.

Общие рекомендации

- Командная работа
 - Безопасность приложений и систем общая ответственность
- Отслеживание состояния и метрик
- Мотивация на производство кода с меньшим количеством уязвимостей
 - Степень безопасности кода = степень качества кода



Итого

"Архитекторы информационной безопасности должны встраивать безопасность в разные точки DevOps процессов совместным образом, который практически прозрачен для разработчиков, сохраняет командную работу, гибкость и скорость DevOps и гибкость среды разработки"

