Building Secure Applications

Jeffery Payne, Founder & CEO

jeff.payne@coveros.com

About Coveros

- Coveros helps organizations accelerate the delivery of secure, reliable software

Why Application Security is Important

The State of Application Security

- We are really good at finding bugs
- We are not so great at finding flaws
- We sometimes fix the problems we find
- We mostly never build security in

Finding Bugs

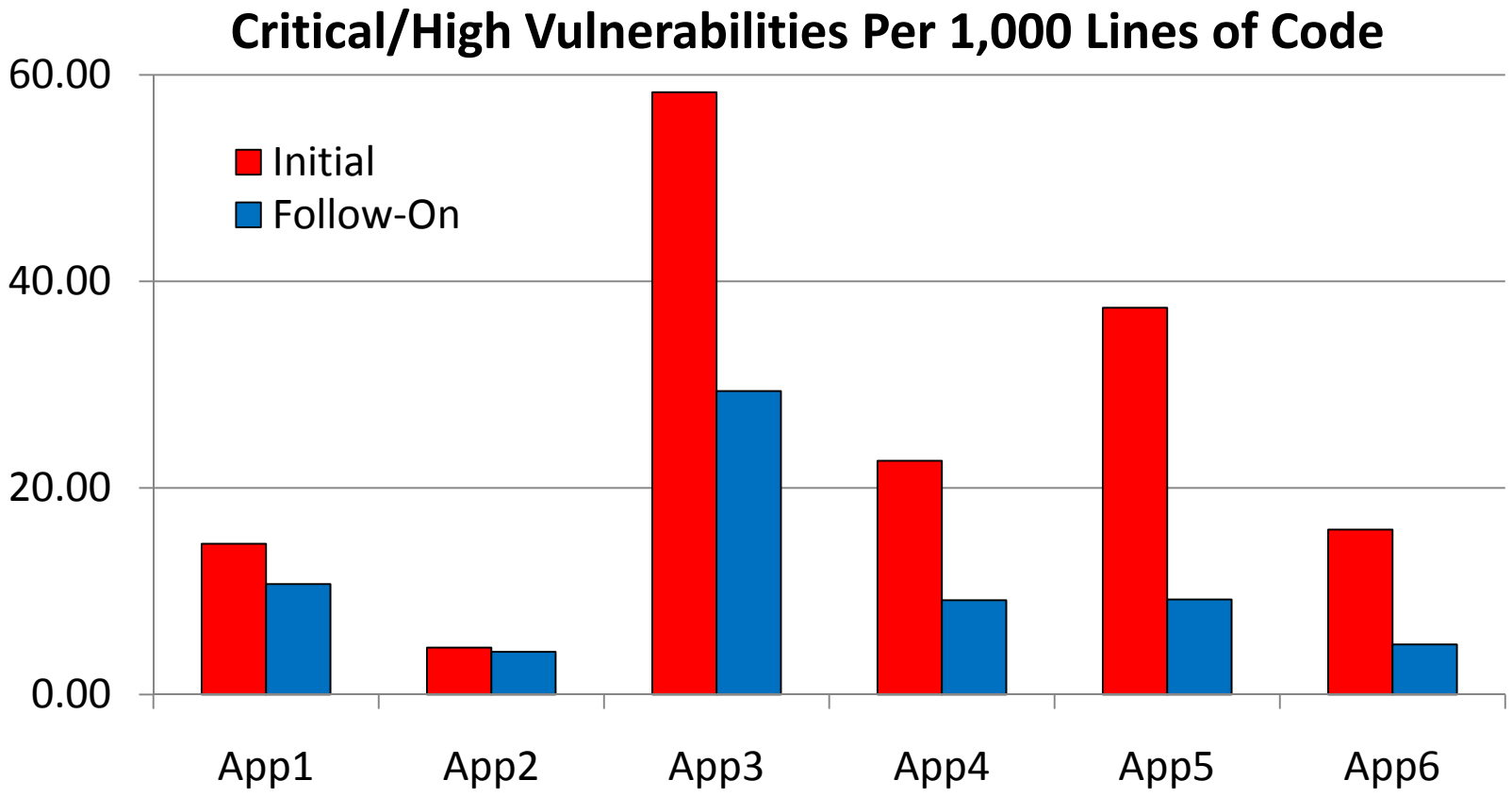
- Common Implementation Errors
  - Buffer overflow
  - Race conditions
  - TOCTOU (time of check to time of use)
  - Unsafe environment variables
  - Unsafe system calls
  - Untrusted input problems

CodeSecure™

Finding Flaws

- Flaws (Design Defects)
  - Misuse of cryptography
  - Compartmentalization problems in design
  - Privileged block protection failure
  - Type safety confusion error
  - Insecure auditing
  - Broken or illogical access control
  - Method over-riding problems

Fixing the Problem – One DoD Initiative



But there are 1,000's of apps … do the math

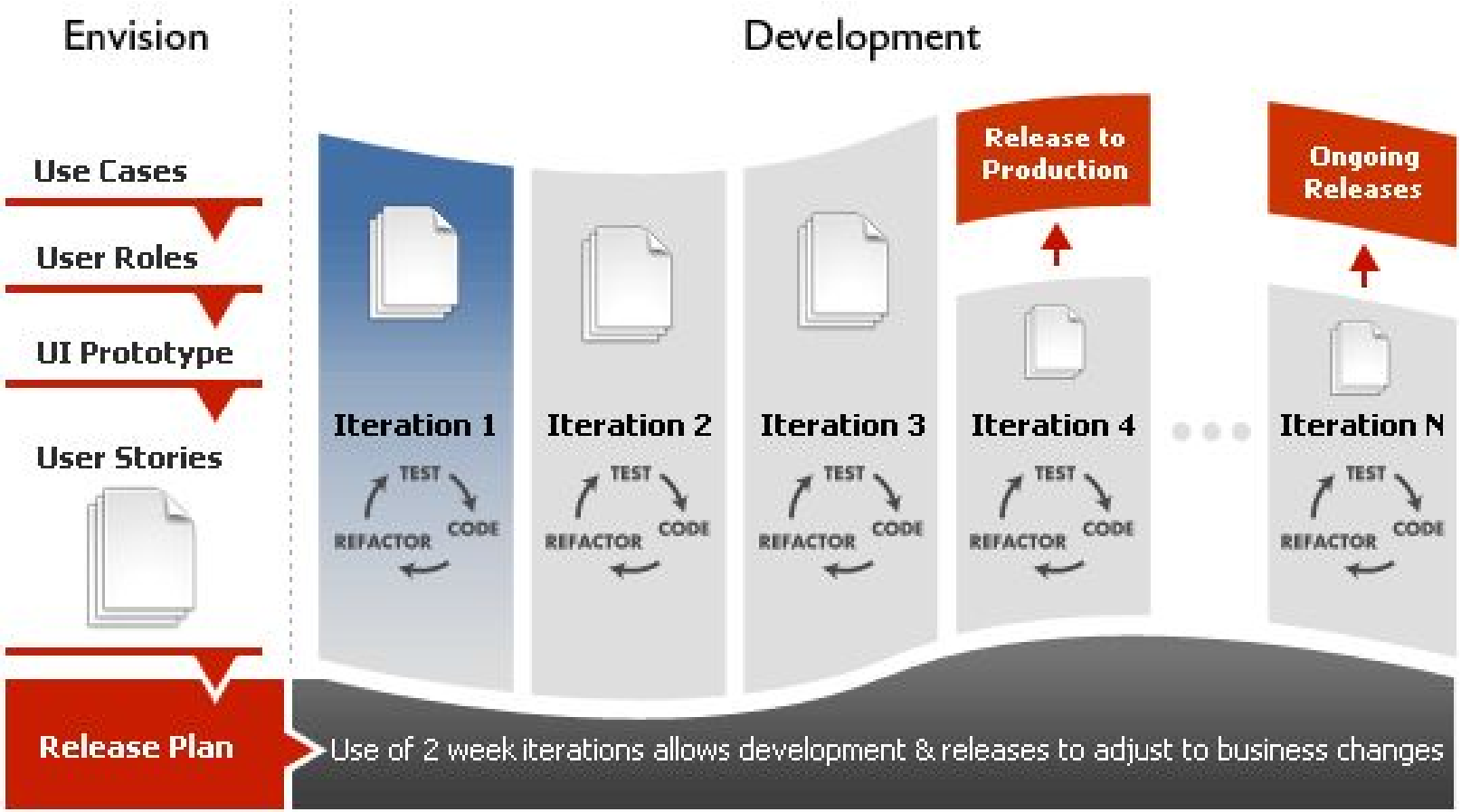
Agile and Security

- There is a widely held belief that secure applications cannot be developed using agile practices.

- This belief comes from:
  - Misunderstanding agile (aka myths about agile)
  - Misunderstanding application security
  - An excuse not to change

- We have successfully built many, many security-critical applications using agile

SecureAgile™ Development Process



Assures time-to-market while achieving security objectives

SecureAgile™ Security Practices

- Misuse/Abuse Cases
- Security Stories
- Defensive Design and Programming
- Continuous Software Security Assurance
  - Architectural risk analysis
  - Code analysis
  - Security and penetration testing

Misuse / Abuse Case Development

- Purpose: Define the possible mechanisms an adversary might exploit to compromise your system

- Approach:
  - Misuse cases are extensions to use cases that highlight ways in which the system might be misused accidentally
  - Abuse cases are extensions to use cases that highlight ways in which the system might be abused on purpose

- Results:
  - Insight into potential abuses that can be avoided and also tested for later

Security Stories

- Purpose: Document the non-functional security requirements associated with the system
- Approach
  - “User shall not …” nomenclature
- Purpose
  - Assure all explicit security requirements are documented to aid secure development and testing activities

Defensive design and coding

- Incorporation of security controls into software design and code
  - Security frameworks like OWASP ESAPI
- Use of vetted components
  - Libraries of secure components
- Examination of design / code looking for realization of architectural risks and misuses / abuses

Software Assurance

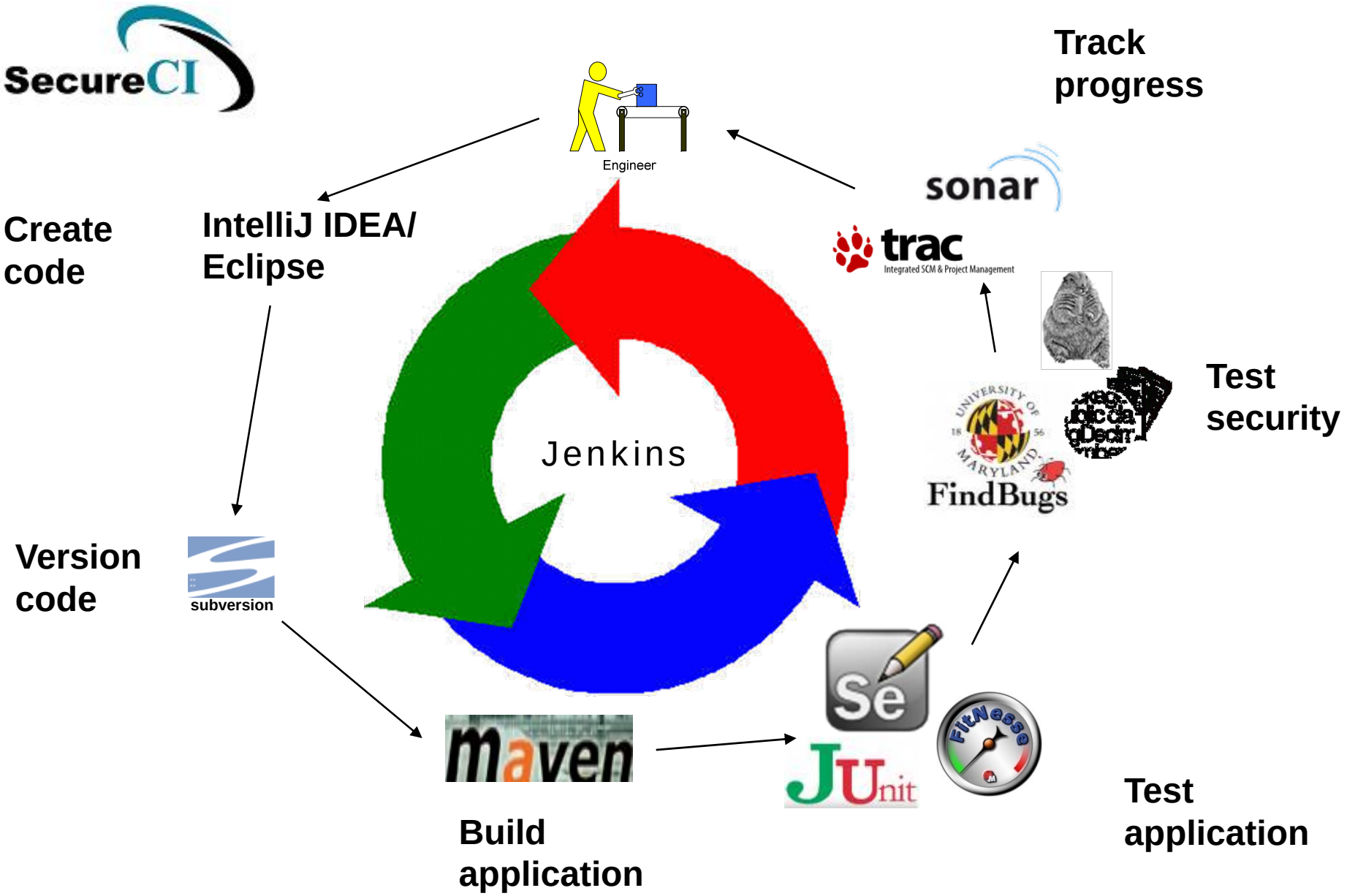
- Architectural risk analysis
  - Assess architecture against threat model, attack patterns, known weaknesses
- Secure code review
  - Both automated and manual
- Security testing
  - Risk-based testing
  - Testing of security functionality
- Penetration testing
  - Performed during the release process

Continuous Integration

- Automation of build, test, deploy process
  - Check-in builds / tests
  - Nightly code integrations and regression tests
  - Automated promotion between test stages
  - Automated notification of build failures

- A critical capability to have when building software using agile … and supports security analysis

- Integration of code analysis and automated security testing can result in identification of security issues early in the process



Thank You