Building Resilient Systems: The Secure Software Development Lifecycle

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

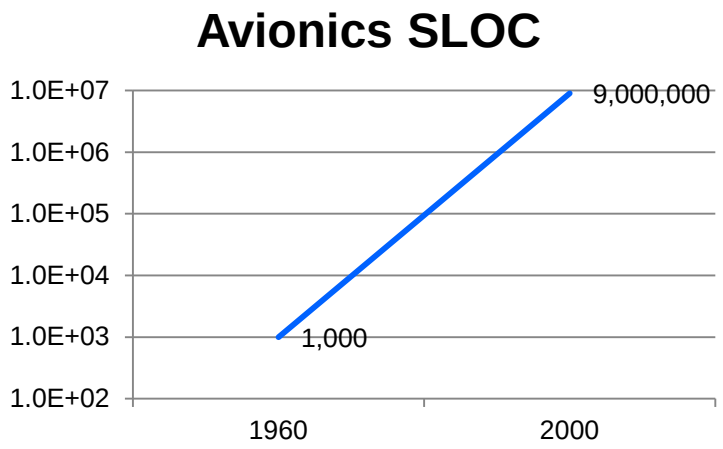
Mark Sherman, PhD
Technical Director, CERT
mssherman@sei.cmu.edu

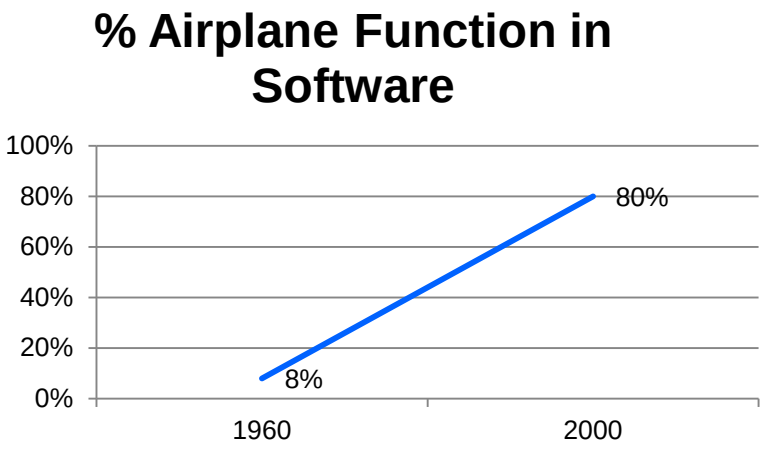
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0004178

Software is advancing function and replacing hardware

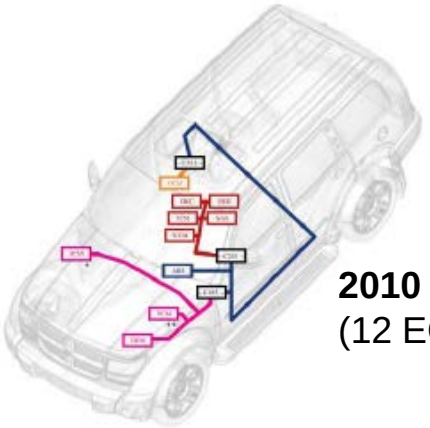
Evolution of avionics size and function from F-4A (1960) to F-35 (2000)





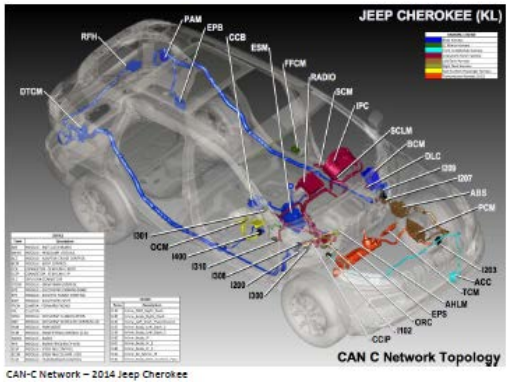
Sources: Final Report, NASA Study on Flight Software Complexity March 2009; Mel Conway, "Tower of Babel and the Fighter Plane," Oct 9, 2013

Vehicle technology following the same path



2010 Jeep Cherokee
(12 ECUs)

2014 Jeep Cherokee
(32 ECUs)



CAN-C Network – 2014 Jeep Cherokee

Common assertion that modern high end vehicles have

- Over 100M lines of code
- Over 50 antennas
- Over 80 ECUs

Sources: Miller and Valasek, A Survey of Remote Automotive Attack Surfaces, http://illmatics.com/remote%20attack%20surfaces.pdf;
https://www.cst.com/webinar14-10-23~?utm_source=rfg&utm_medium=web&utm_content=mobile&utm_campaign=2014series
https://en.wikipedia.org/wiki/Electronic_control_unit

Software vulnerabilities are ubiquitous

Automotive vulnerabilities were predicted

Software Engineering Institute
Carnegie Mellon University

Emerging Technology Domains Risk Survey

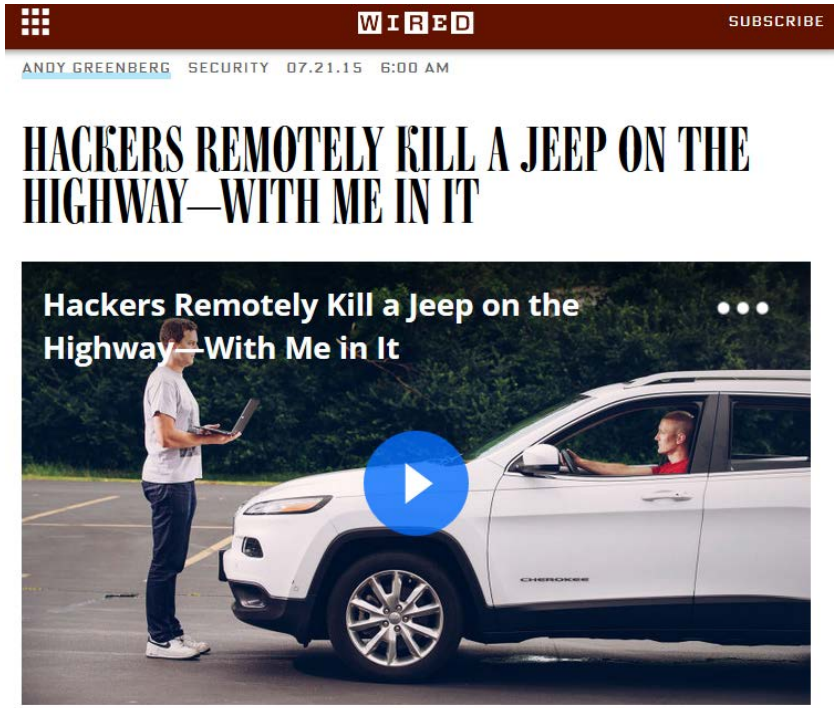
Christopher King
Jonathan Chu
Andrew Mellinger

April 2015

TECHNICAL NOTE
CMU/SEI-2015-TN-003

Source: C. King, et al, Emerging Technology Domains Risk Survey, SEI Technical Note CMU/SEI-2015-TN-003

Automotive vulnerabilities were realized



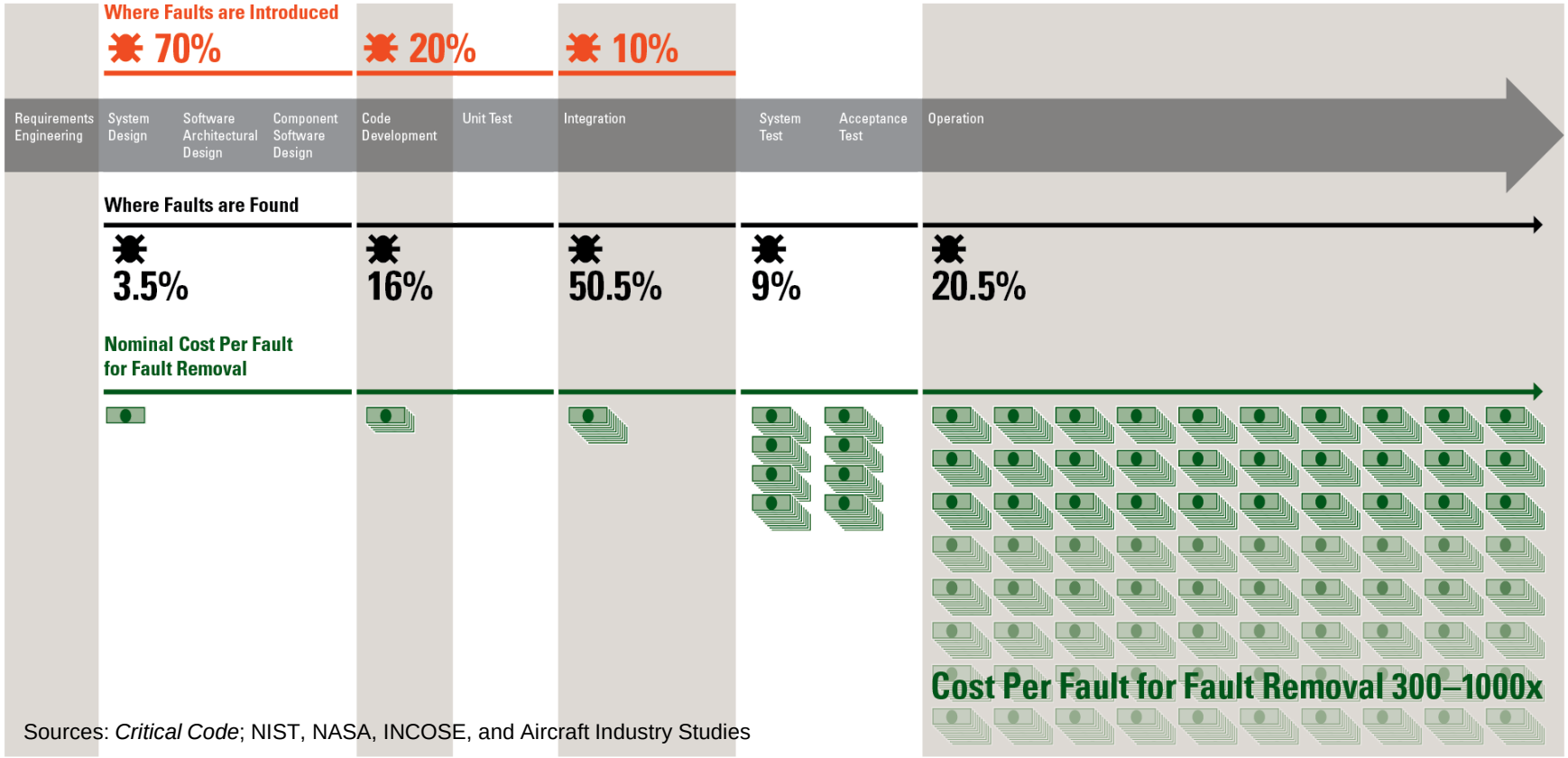
Source: http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

Source: Miller and Valasek, A Survey of Remote Automotive Attack Surfaces, http://illmatics.com/remote%20attack%20surfaces.pdf

Catching software faults early saves money

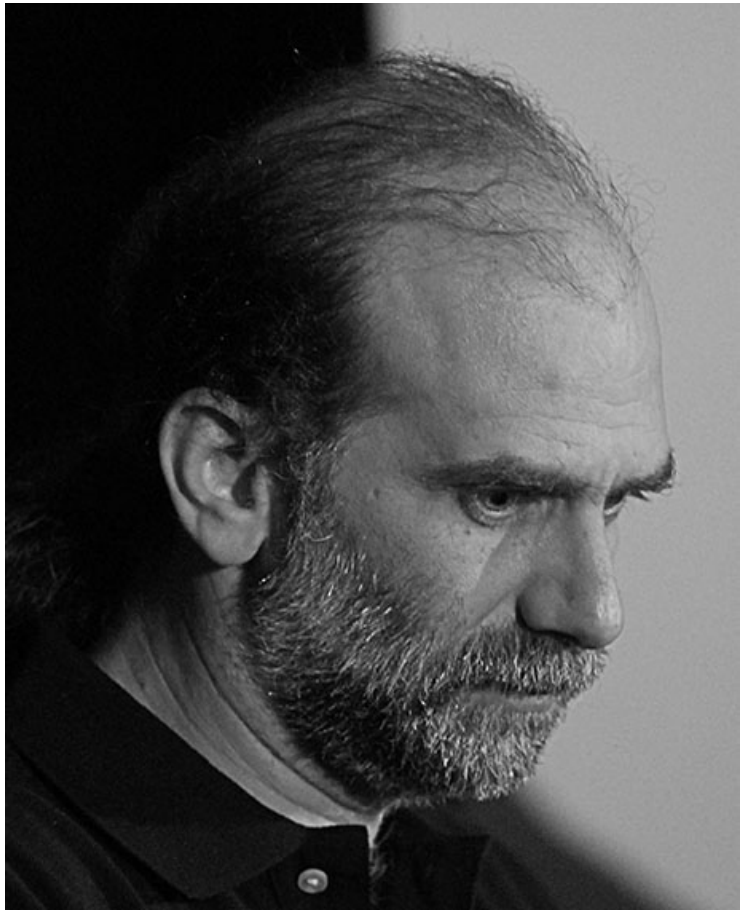
Faults accounts for 30–50% percent of total software project costs

Software Development Lifecycle



Sources: *Critical Code*; NIST, NASA, INCOSE, and Aircraft Industry Studies

An ounce of prevention ….

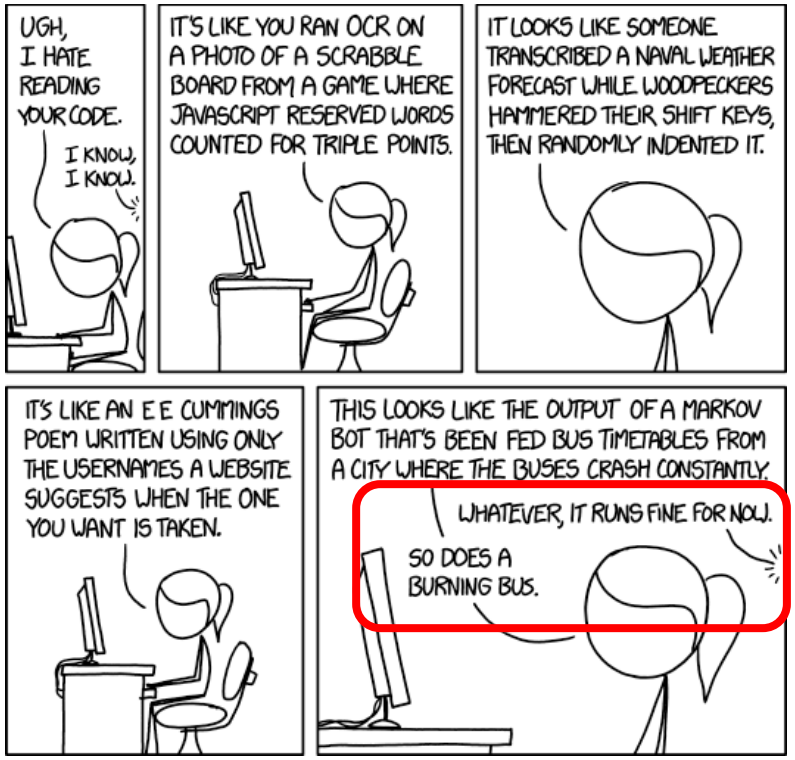


"We wouldn't have to spend so much time, money, and effort on network security if we didn't have such bad software security."

Bruce Schneier in Viega and McGraw, "Building Secure Software," 2001

Source: Washington Post, March 19, 2014, http://www.washingtonpost.com/business/economy/toyota-reaches-12-billion-settlement-to-end-criminal-probe/2014/03/19/5738a3c4-af69-11e3-9627-c65021d6d572_story.html

Need to avoid false sense of security



Source: http://xkcd.com/1695/

An ounce of prevention ….

"We wouldn't have to spend so much time software security."

Bruce Schneier in Viega and McGraw, "Building Secure Software," 2001

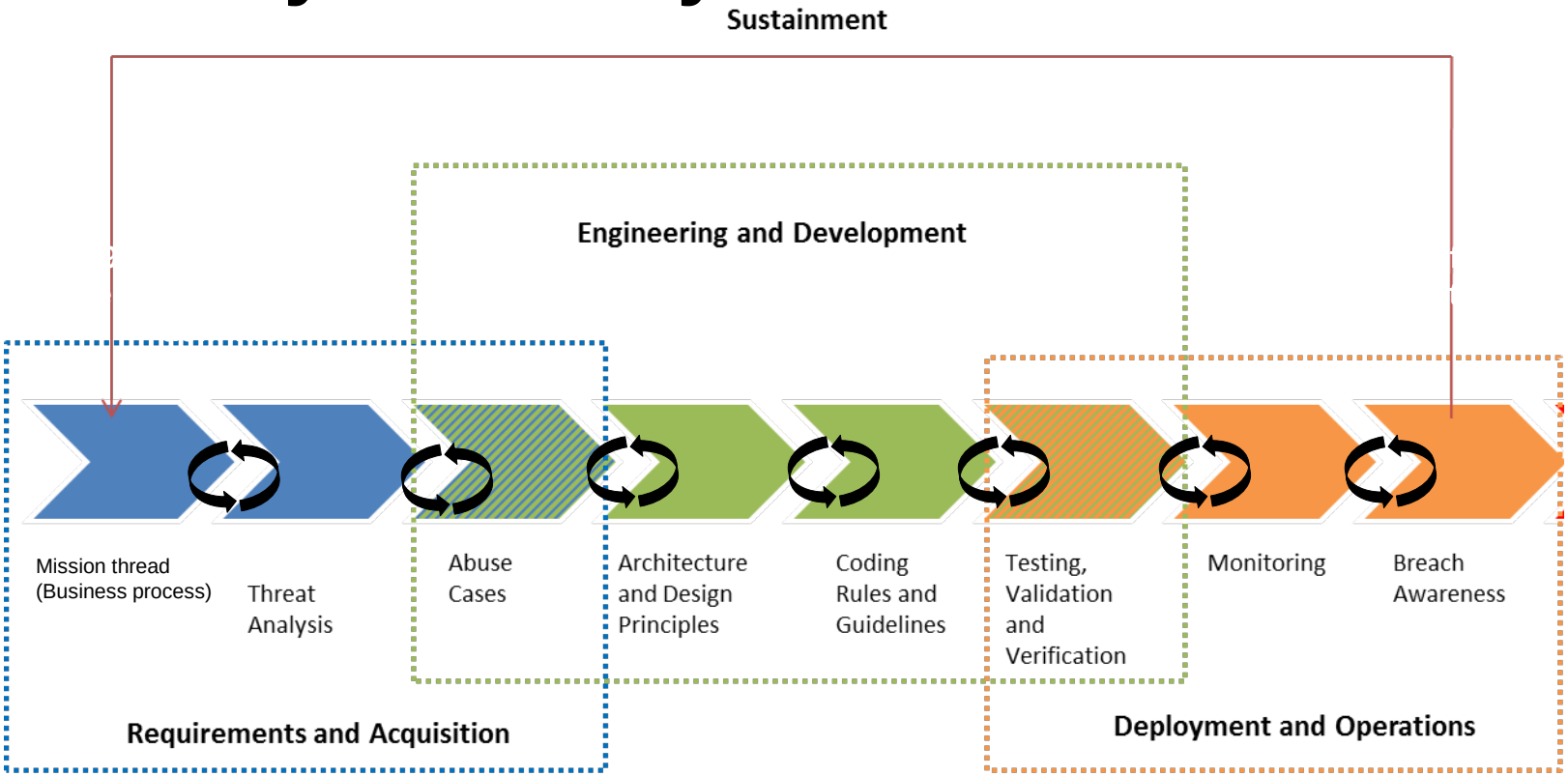
Sections ≡ The Washington Post

Business

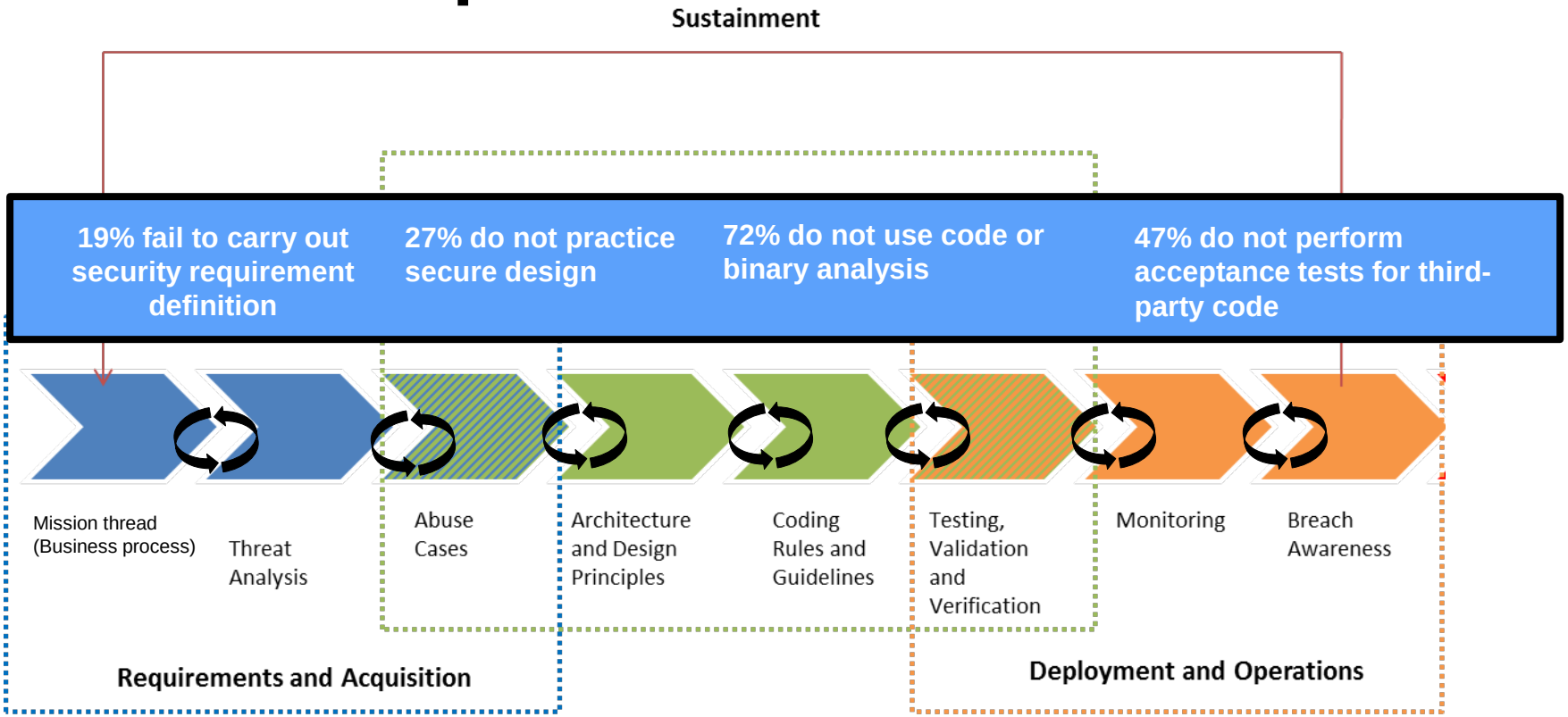
Toyota reaches $1.2 billion settlement to end probe of accelerator problems

Source: Washington Post, March 19, 2014, http://www.washingtonpost.com/business/economy/toyota-reaches-12-billion-settlement-to-end-criminal-probe/2014/03/19/5738a3c4-af69-11e3-9627-c65021d6d572_story.html

Security is a lifecycle issue



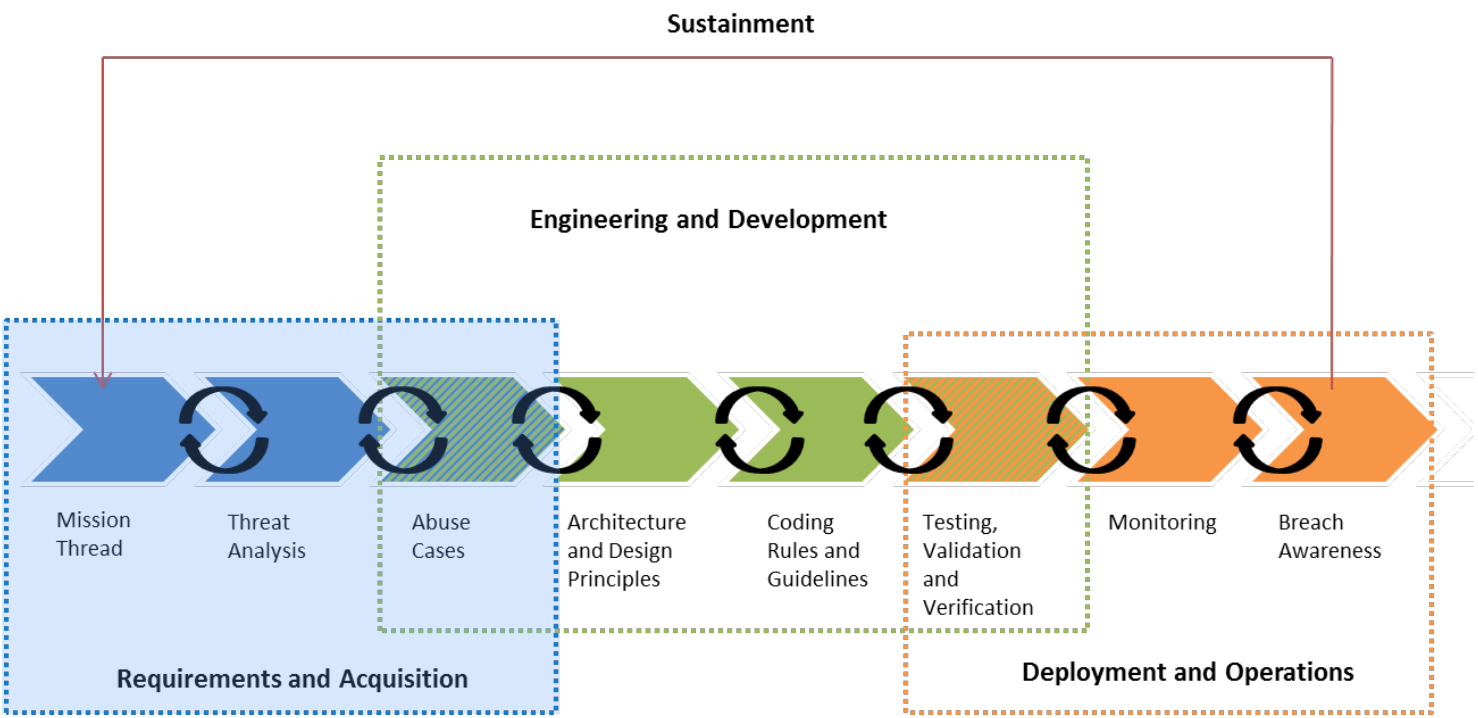
Room for improvement



More than 81% do not coordinate their security practices in various stages of the development life cycle.

Sources: Forrester Consulting, "State of Application Security," January 2011; Wendy Nather, Research Director, 451 Research, "Dynamic testing: Why Tools Alone Aren't Enough, March 25, 2015"

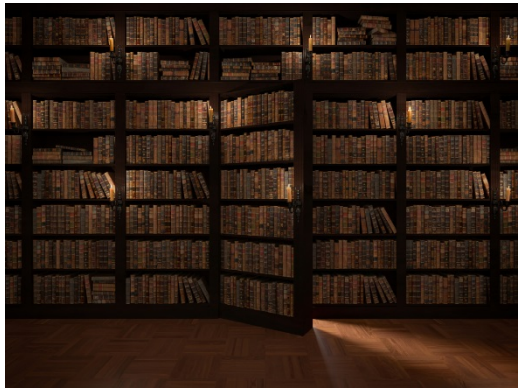
Requirements



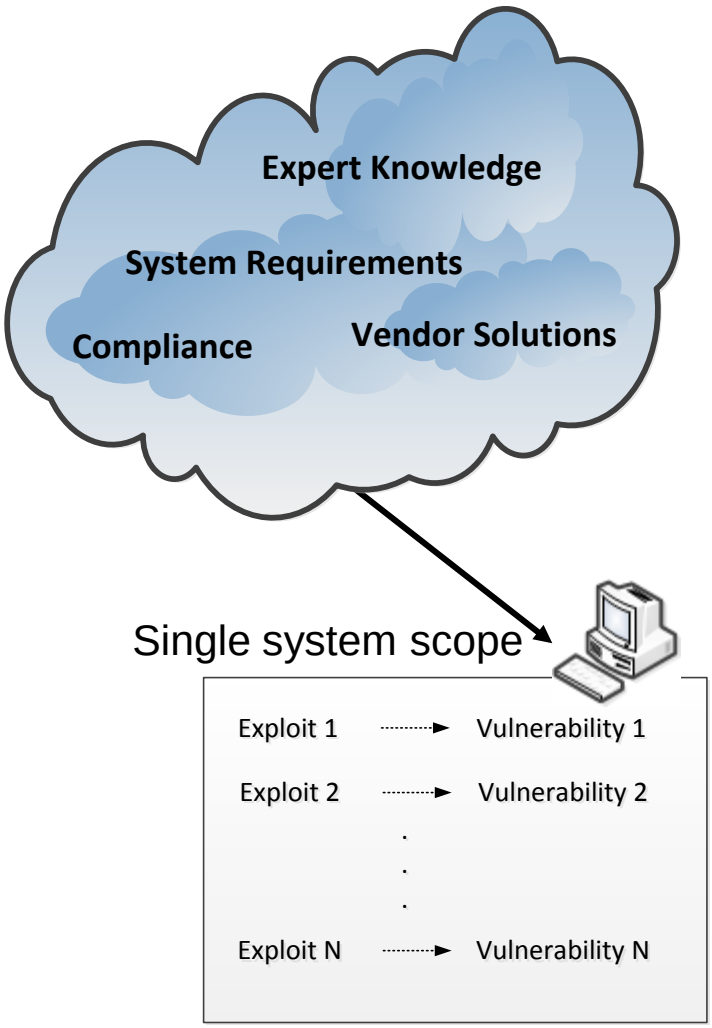
Getting the right requirements: desire for backdoors conflicts with secure operations

Helpful capability

Backdoor vulnerability



Need for multisystem risk analysis



Risk analysis is focused on a single system

- Standalone (i.e., single system) models have been developed
- Risk analysis considers the exploit of an individual vulnerability within a single system

Security risk identification techniques do not consider:

- Compositions of multiple vulnerabilities
- Cross-system security events/risks
- Impacts beyond the exploit of a single system (to the intended service and organization)

Need for systematic, multiple system evaluations

- Notation for expressing a security events and risks
- Take into account all context: operational **and** physical, data, workflow, stakeholder, network views

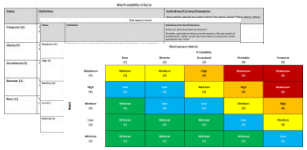
Security Engineering Risk Analysis

1. Establish operational context.
2. Identify risk.
3. Analyze risk.
4. Develop control plan.

Process Thread Worksheet

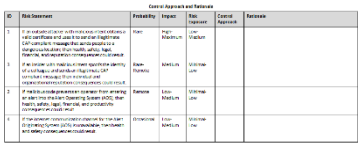
Risk Identification Worksheet

Risk Evaluation Criteria



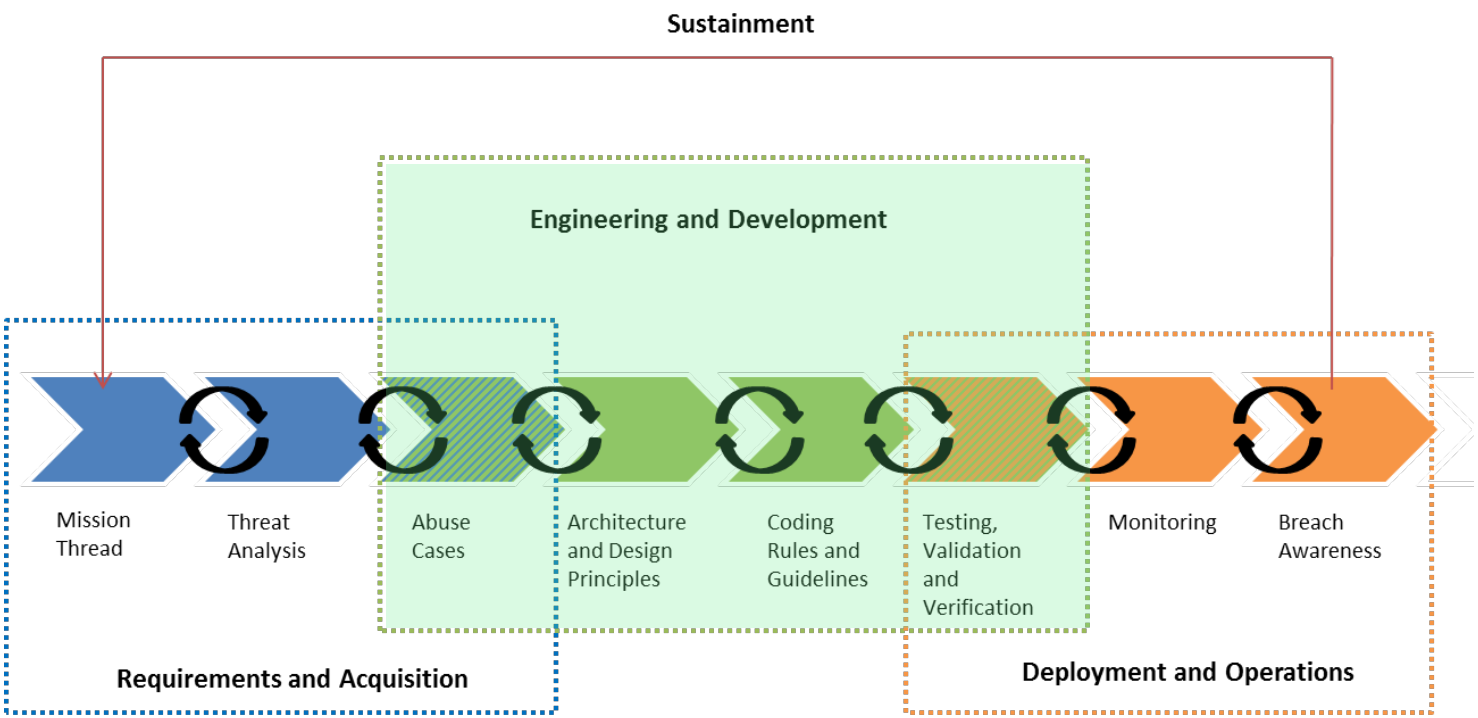
Risk Analysis Worksheet

Control Approach Worksheet

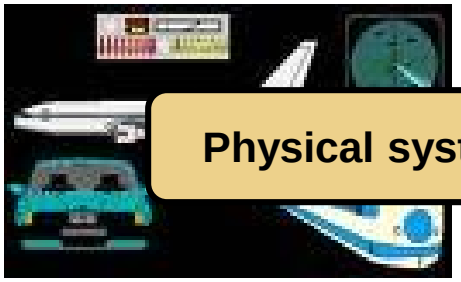


Control Plan Worksheet

Engineering and Development



Architecture Analysis & Design Language (AADL)

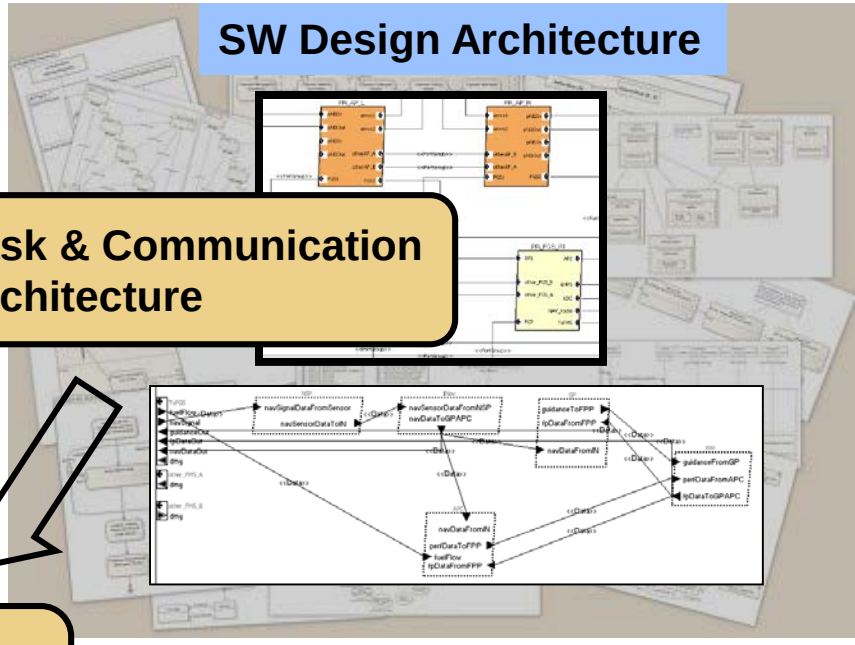


Physical system

Command & Control

Task & Communication Architecture

SW Design Architecture



Deployed on

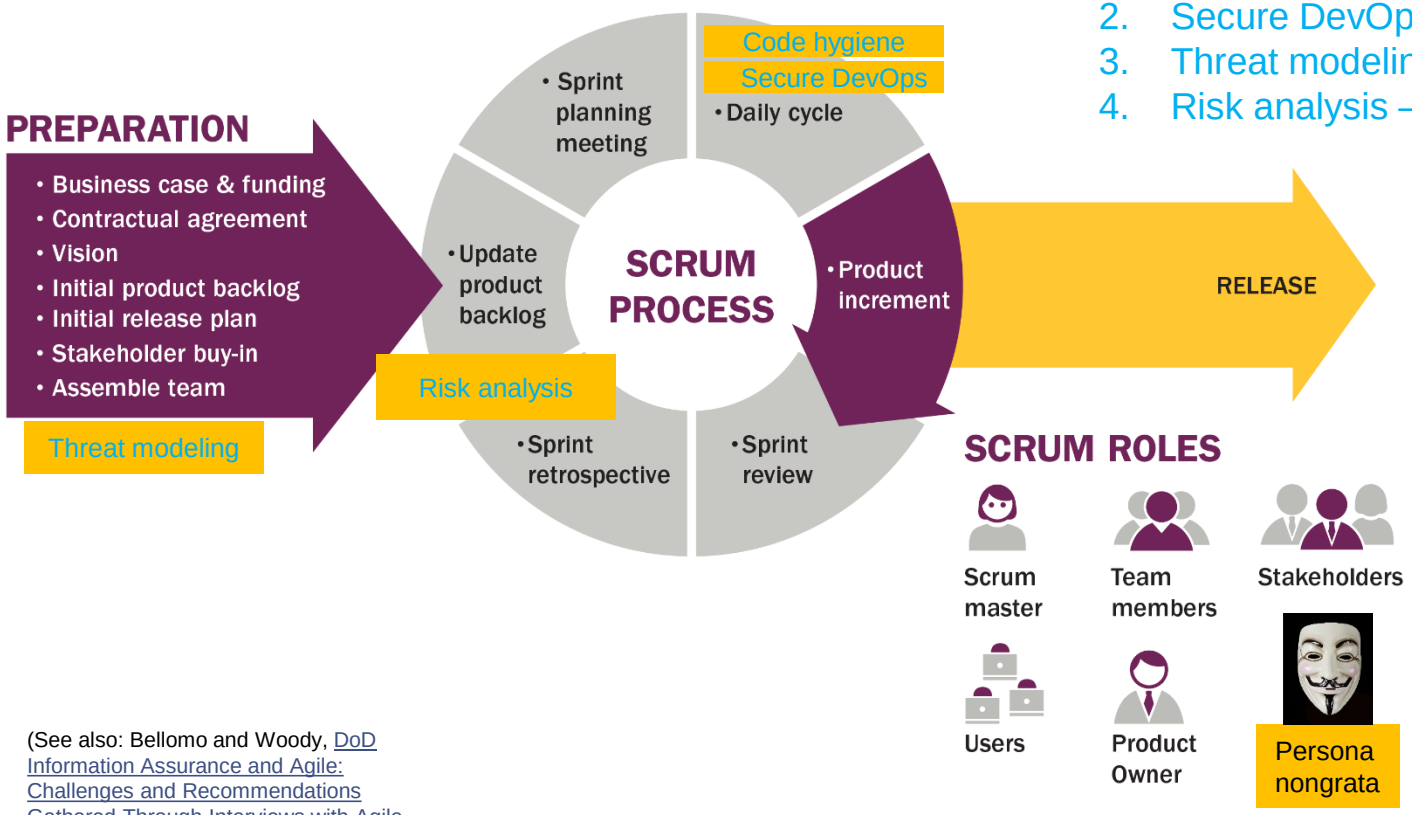
Physical interface

Distributed Computer Platform

AADL Addresses Increasing Interaction Complexity and Mismatched Assumptions

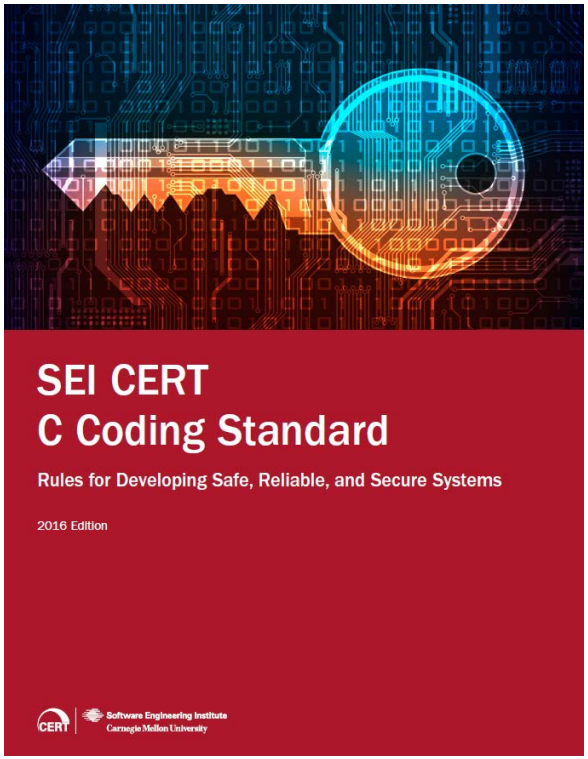
Integrating security into Agile development

1. Code hygiene – introduce secure coding
2. Secure DevOps – include security tools
3. Threat modeling – represent a new role
4. Risk analysis – prioritize in backlog



(See also: Bellomo and Woody, DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers (http://repository.cmu.edu/cgi/viewcontent.cgi?article=1674&context=sei)

Coding rules



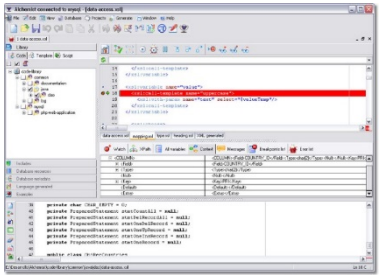
- Collected wisdom of programmers and tools vendors
  - Fed by community wiki started in Spring 2006
  - 1,576 registered contributors
- Basis for ISO Standard
- Downloadable 2016 edition

Download link:
http://www.cert.org/secure-coding/products-services/secure-coding-download.cfm

Adoption of secure coding rules

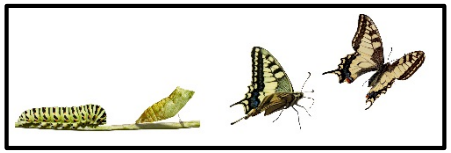
Training

Integrated development environments



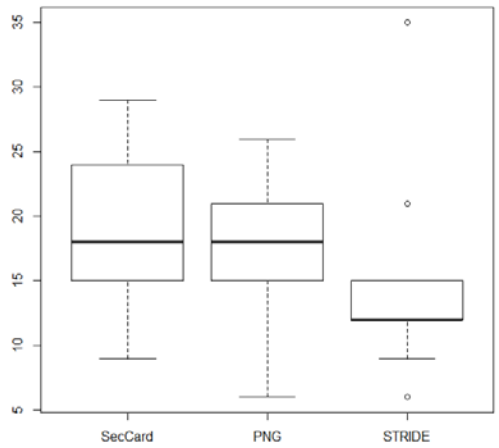
Batch analyzers

Automated transformation and remediation



Threat modeling methodologies vary in coverage

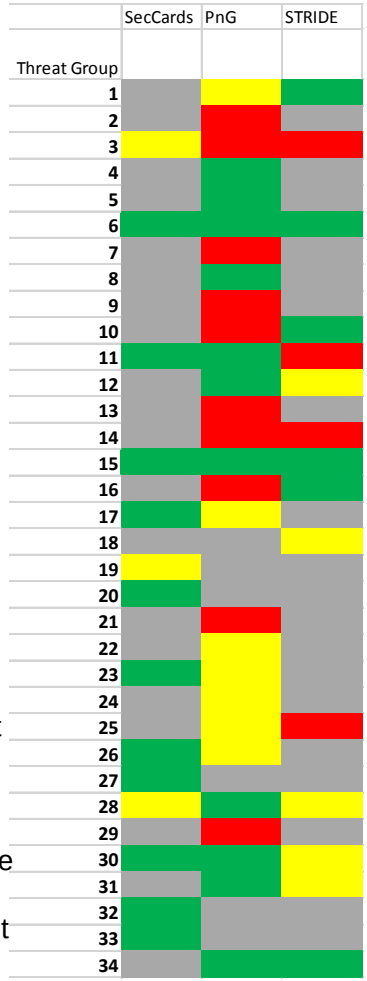
Team Effectiveness – within all threat types reported for Aircraft Scenario



Take-aways:

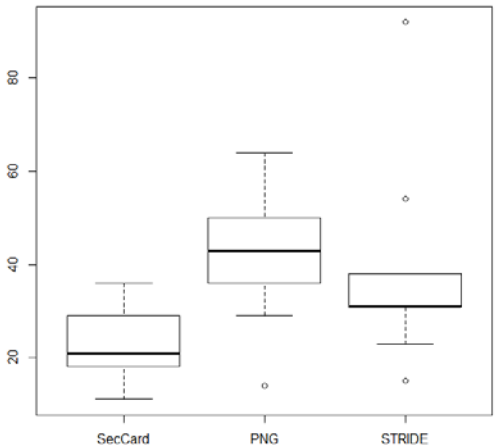
- The "new" approaches perform similarly, finding just about 20% of the relevant threat types on average. Both performed much better than the state of the practice approach, STRIDE.
- STRIDE has the most outliers – leaves more up to the experience of the teams?
- STRIDE has a longer tail above the mean. It seems to get most teams to a certain floor level of performance but allows more teams to do better?

Threat Type Coverage



Source: Forrest Shull, et al, discussed in SEI Research Review 2016

Team Effectiveness – within threat types reported by that TMM



Take-aways:

- PNG misses several classes of threats altogether. However, for the classes it DOES find, teams are consistently more likely to find them.
- SecCards can find a greater variety of threats – but each team is likely to report only a small subset, with lots of variation among teams.
- STRIDE is somewhere in between. Like PNG, it seems to allow more consistent results for the threats that it does find. The distribution is skewed to have a long tail above the mean, meaning that there are some types of threats that truly are found by almost all teams applying STRIDE.

Embedded Systems (ES) represent new classes of vulnerabilities and risk

Characteristics of Embedded Systems

ECU designed for a specific purpose – architecture could be unique for each embedded system

Size, Weight, Power and Latency concerns

- Watchdog and filtering processes may not fit in operational envelop

Designing ES and code is a special field

- Subject matter expertise of unique system
- Autonomous systems have physical resources, navigation needs and Safety-Critical Real-time OS
- Intermittent communications and multiple command-and-control masters
- Embedded firmware, unique internal busses & controllers
- Can require specific skills at the bit and clock cycle level

ES cyber defenses differ from PC/IT network cyber defenses

Network-centric cyber defenses have limited applicability to embedded systems

- Virus definitions and operating guidelines don't always apply
- Centralized account control not possible
- Network tools and assessment techniques have limited relevance to embedded systems architecture and interfaces

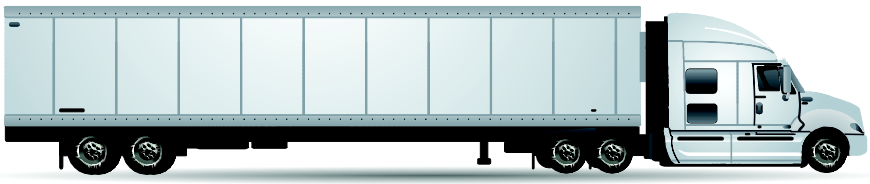
Threat Mitigation for ES and PC/IT are quite different

- Larger number and more diverse attack surfaces
- Back doors (maintenance), hardcoded credentials, insecure protocols, unplanned connectivity and upgrades

Programming for security is not the same as programming for safety

Safety strategy	Security view
Rely on physical models in fault trees	Attackers do not obey the laws of physics
Redundancy mitigates single failures	Attackers are not independent events
Shared, global state provides single world-view	Attackers use leaked information beyond intended purposes
Shared service containers to meet space, power and weight constraints	Coupled services enable denial of service attacks
Microcontrollers and air gaps implement boundaries	Side channels open vulnerabilities

Machine-learning based systems increase exposures



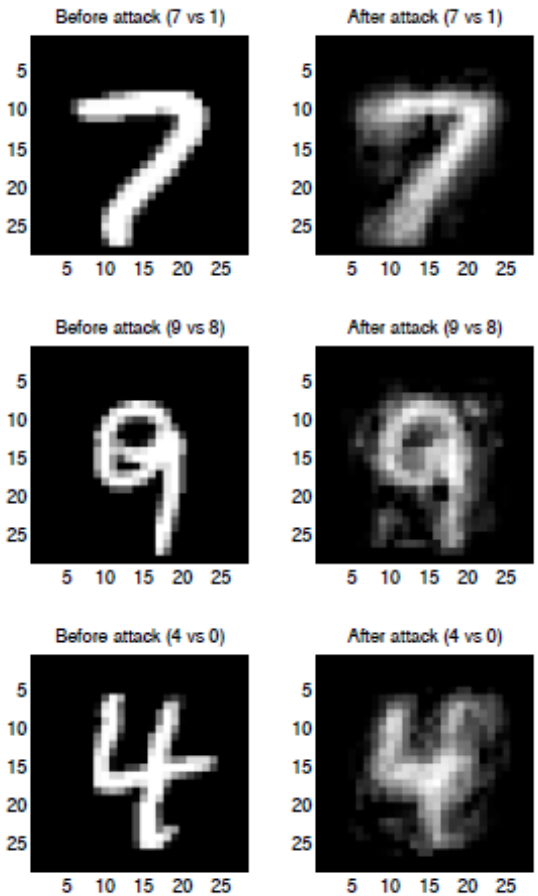
Operations are driven by sensor data with large ranges of values

Decision making is based on “trained” models of behaviors

Conventional analysis of code of modest help

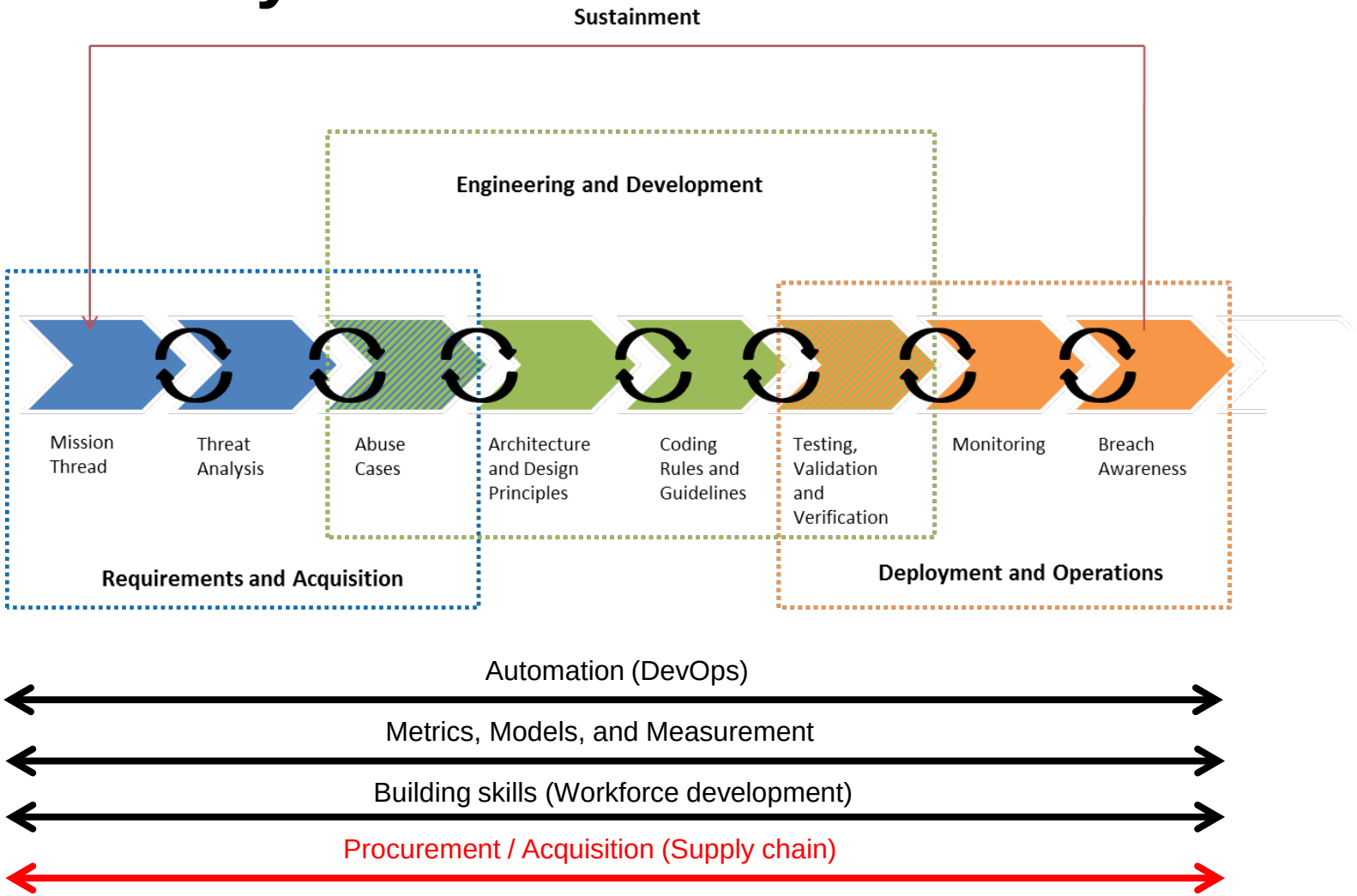
Understand the limits of training

Recognizing and recovering poisoned systems



- “Chaff” and “noise” can emerge as vulnerabilities
- Defensive strategy based on “it is difficult to lie at scale”
- Tactics include consistency checks, such as
  - Multiple models in a single unit
  - Coordination among units
  - Coordination with environment

Cross lifecycle issues



Evolution of software development

Custom development – context:

- Software was limited
  - Size
  - Function
  - Audience
- Each organization employed developers
- Each organization created their own software

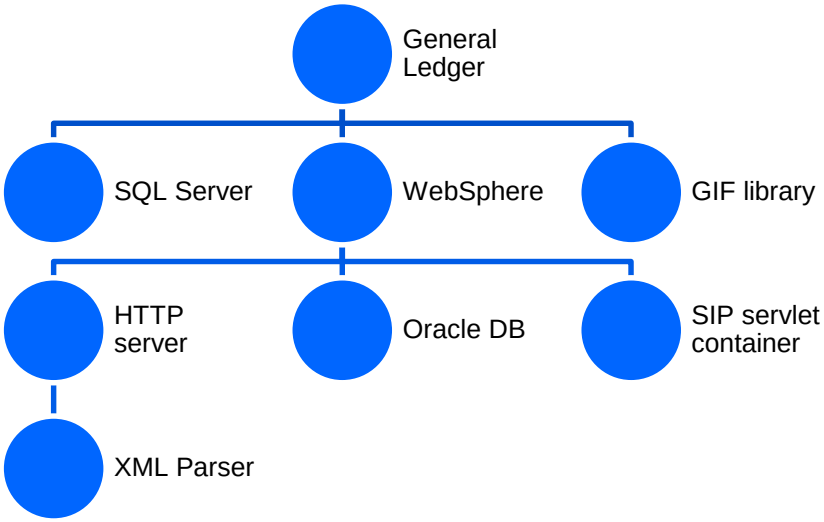
Supply chain: practically none

Shared development – ISVs (COTS) – context:

- Function largely understood
  - Automating existing processes
- Grown beyond ability for using organization to develop economically
- Outside of core competitiveness by acquirers

Supply chain: software supplier

Development is now assembly



Collective development – context:

- Too large for single organization
- Too much specialization
- Too little value in individual components

Supply chain: long

Note: hypothetical application composition

Vulnerabilities emerge in existing code

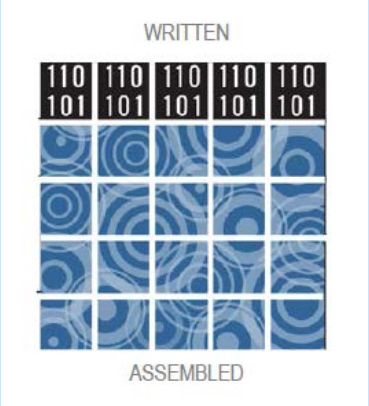
Defects in functionality found early and in new code

Vulnerabilities found in legacy code and late (“honeymoon effect”)

New operating environments are a major cause of vulnerabilities

Clark, Frei, Blaze, Smith, “Familiarity Breeds Contempt: The Honeymoon Effect and the Role of Legacy Code in Zero-Day Vulnerabilities,” ACSAC ’10 Dec. 6-10, 2010, p. 251-260.”

Substantial open source contained in supply chain



- 90% of modern applications are assembled from 3rd party components
  - At least 75% of organizations rely on open source as the foundation of their applications
- Most applications are now assembled from hundreds of open source components, often reflecting as much as 90% of an application

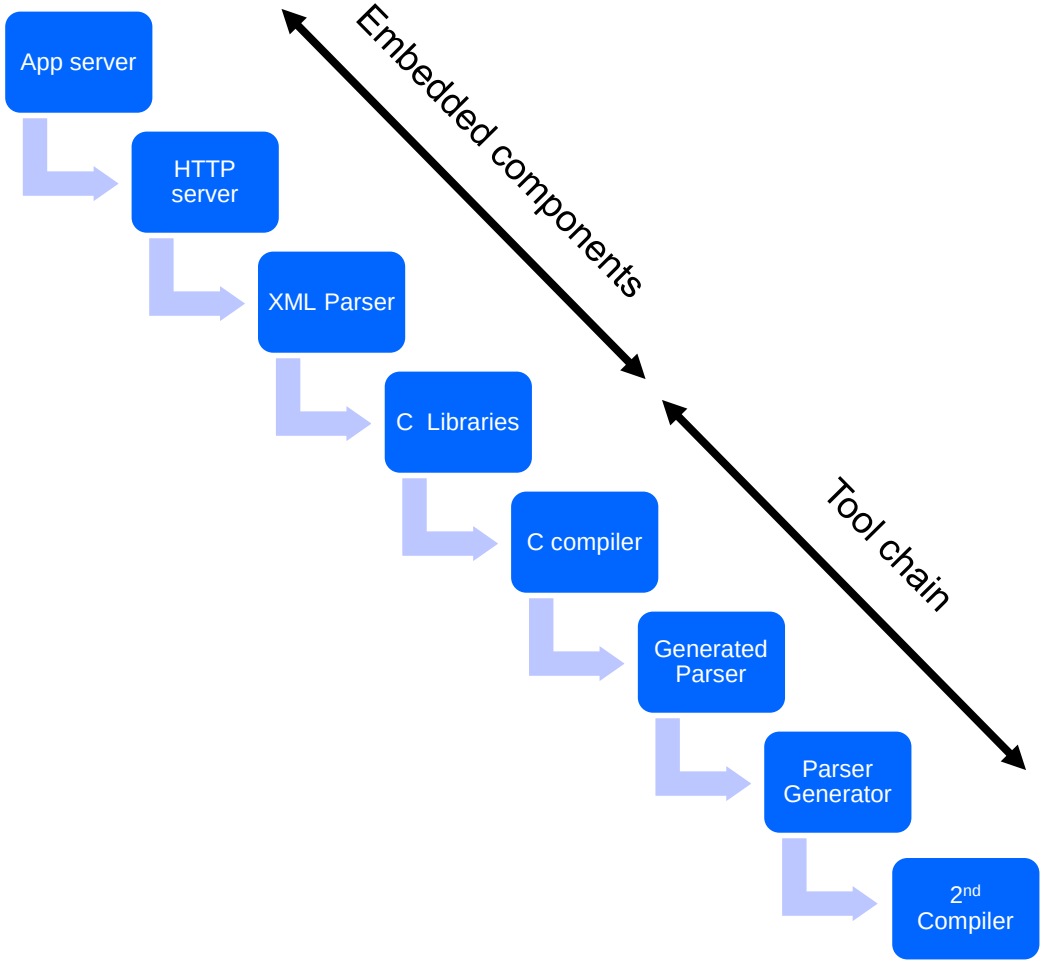
Distributed development – context:

- Amortize expense
- Outsource non-differential features
- Lower acquisition (CapEx) expense

Supply chain: opaque

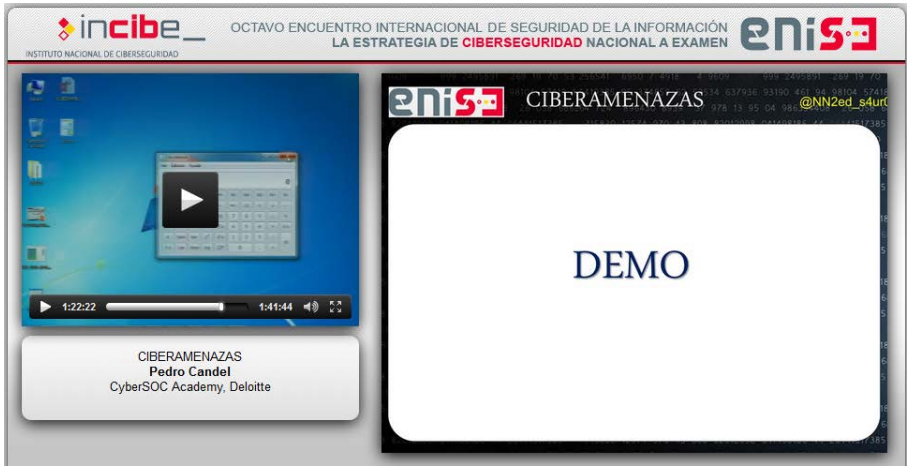
Sources: Geer and Corman, “Almost Too Big To Fail,” ;login: (Usenix), Aug 2014; Sonatype, 2014 open source development and application security survey

Open source supply chain has a long path



Corruption along the supply chain is easy

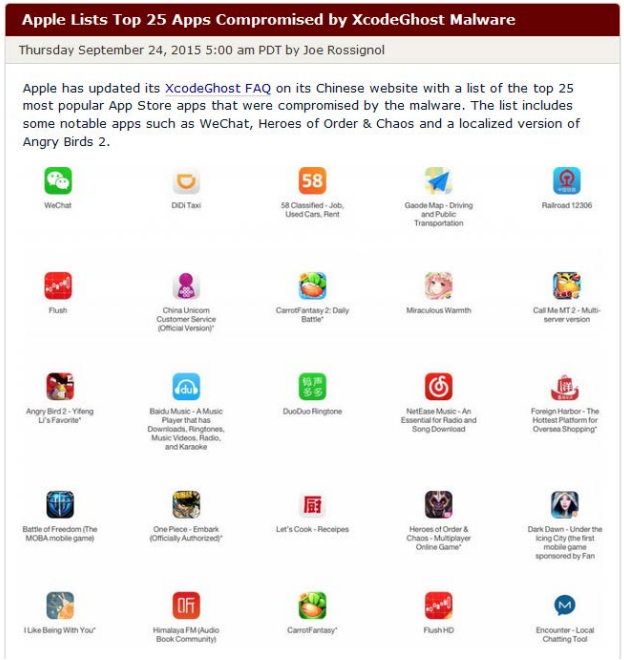
Unexpected or unintended behaviors in components



Knowledgeable analysts can convert packaged binary into malware in minutes

Sources: Pedro Candel, Deloitte CyberSOC Academy , Deloitte
http://www.8enise.webcastlive.es/webcast.htm?video=08; http://www.microsoft.com/Products/Games/FSInsider/freeflight/PublishingImages/scene.jpg;
https://www.withfriendship.com/user/mithunss/easter-eggs-in-microsoft-products.php

Corruption in the tool chain already exists



Apple Lists Top 25 Apps Compromised by XcodeGhost Malware

Thursday September 24, 2015 5:00 am PDT by Joe Rossignol

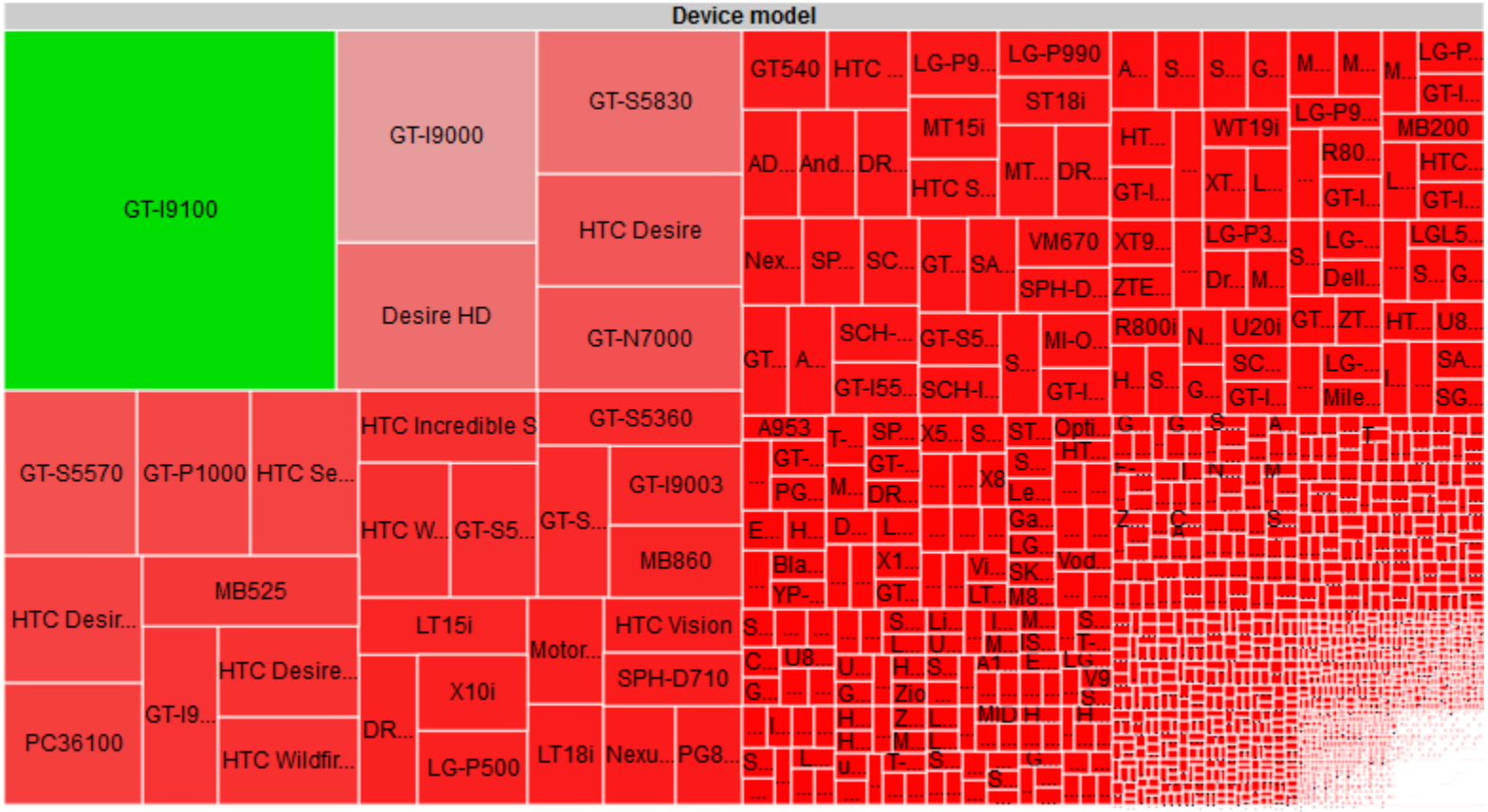
Apple has updated its XcodeGhost FAQ on its Chinese website with a list of the top 25 most popular App Store apps that were compromised by the malware. The list includes some notable apps such as WeChat, Heroes of Order & Chaos and a localized version of Angry Birds 2.

- WeChat
- DiDi Taxi
- 58 Classified - Job, Used Cars, Rent
- Gaode Map - Driving and Public Transportation
- Railroad 12306
- Flush
- China Unicom Customer Service (Official Version)*
- CarrotFantasy 2: Daily Battle*
- Miraculous Warmth
- Call Me MT 2 - Multi-server version
- Angry Bird 2 - Yifeng Li's Favorite*
- Baidu Music - A Music Player that has Downloads, Ringtones, Music Videos, Radio, and Karaoke
- DuoDuo Ringtone
- NetEase Music - An Essential for Radio and Song Download
- Foreign Harbor - The Hottest Platform for Overseas Shopping*
- Battle of Freedom (The MOBA mobile game)
- One Piece - Embark (Officially Authorized)*
- Let's Cook - Receipes
- Heroes of Order & Chaos - Multiplayer Online Game*
- Dark Dawn - Under the Icing City (the first mobile game sponsored by Fan
- I Like Being With You*
- Himalaya FM (Audio Book Community)
- CarrotFantasy*
- Flush HD
- Encounter - Local Chatting Tool

- XcodeGhost corrupted Apple's development environment
- Major programs affected
  - WeChat
  - Badu Music
  - Angry Birds 2
  - Heroes of Order & Chaos
  - iOBD2

Sources: http://www.macrumors.com/2015/09/24/xcodeghost-top-25-apps-apple-list/
http://www.itntoday.com/2015/09/the-85-ios-apps-affected-by-xcodeghost.html

Versions of Android illustrate open source fragmentation



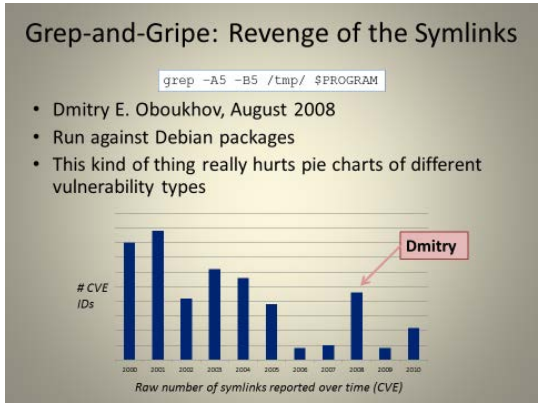
Source: http://opensignal.com/reports/fragmentation.php

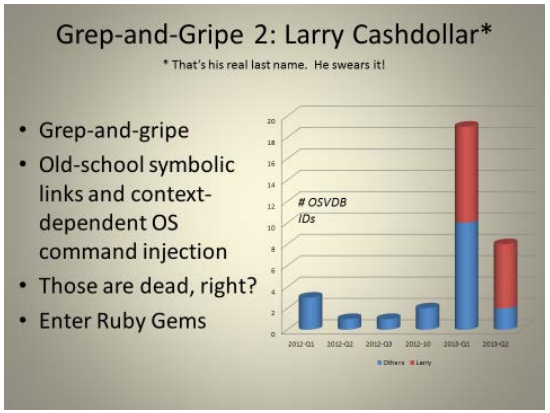
.

Open source is not secure

Heartbleed and Shellshock were found by exploitation

Other open source software illustrates vulnerabilities from cursory inspection





Sources: Steve Christey (MITRE) & Brian Martin (OSF), Buying Into the Bias: Why Vulnerability Statistics Suck, https://media.blackhat.com/us-13/US-13-Martin-Buying-Into-The-Bias-Why-Vulnerability-Statistics-Suck-Slides.pdf; Sonatype, Sonatype Open Source Development and Application Security Survey; Sonatype, 2016 State of the Software Supply Chain; Aspect Software "The Unfortunate Reality of Insecure Libraries," March 2012

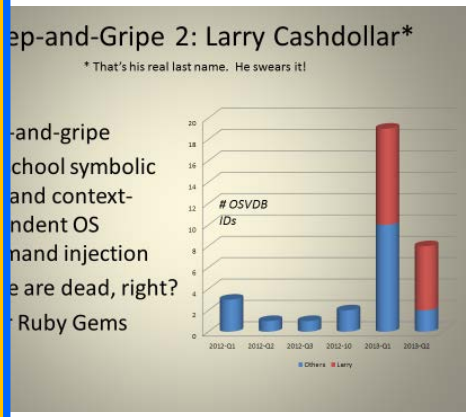
Open source is not secure

Heartbleed and Shellshock were fou
by exploitation

Other open source software illustrates vulnerabilities from cu
inspection

1.8 billion vulnerable open source components downloaded in 2015

26% of the most common open source components have high risk vulnerabilities



Sources: Steve Christey (MITRE) & Brian Martin (OSF), Buying Into the Bias: Why Vulnerability Statistics Suck, https://media.blackhat.com/us-13/US-13-Martin-Buying-Into-The-Bias-Why-Vulnerability-Statistics-Suck-Slides.pdf; Sonatype, Sonatype Open Source Development and Application Security Survey; Sonatype, 2016 State of the Software Supply Chain; Aspect Software "The Unfortunate Reality of Insecure Libraries," March 2012

Reducing software supply chain risk factors

Software supply chain risk for a product needs to be reduced to acceptable level

Supplier Capability	*Product Security*	*Product Distribution*	*Operational Product Control*
Supplier follows practices that reduce supply chain risks	Delivered or updated product is acceptably secure	Methods of transmitting the product to the purchaser guard again tampering	Product is used in a secure manner

Assessment: Software Assurance Framework

What

- Defines software assurance practices for acquiring and developing assured software products

Why

- Improve software assurance practices in acquisition programs
- Enhance software assurance services provided by third parties

Benefits

- Establish confidence in a program's ability to acquire software-reliant systems across the life cycle and supply chain
- Reduce cybersecurity risk of deployed software-reliant systems

Nine Practice Areas

Focus

1. Governance Infrastructure Practices — **Governance Infrastructure**

2. Materiel Solution Analysis (MSA) Practices (Alternatives)
3. Technology Development (TD) Practices (Gaps)
4. Engineering and Manufacturing Development (EMD) Practices (Pilot)
5. Production and Deployment (PD) Practices (Production)
6. Operations and Support (O&S) Practices

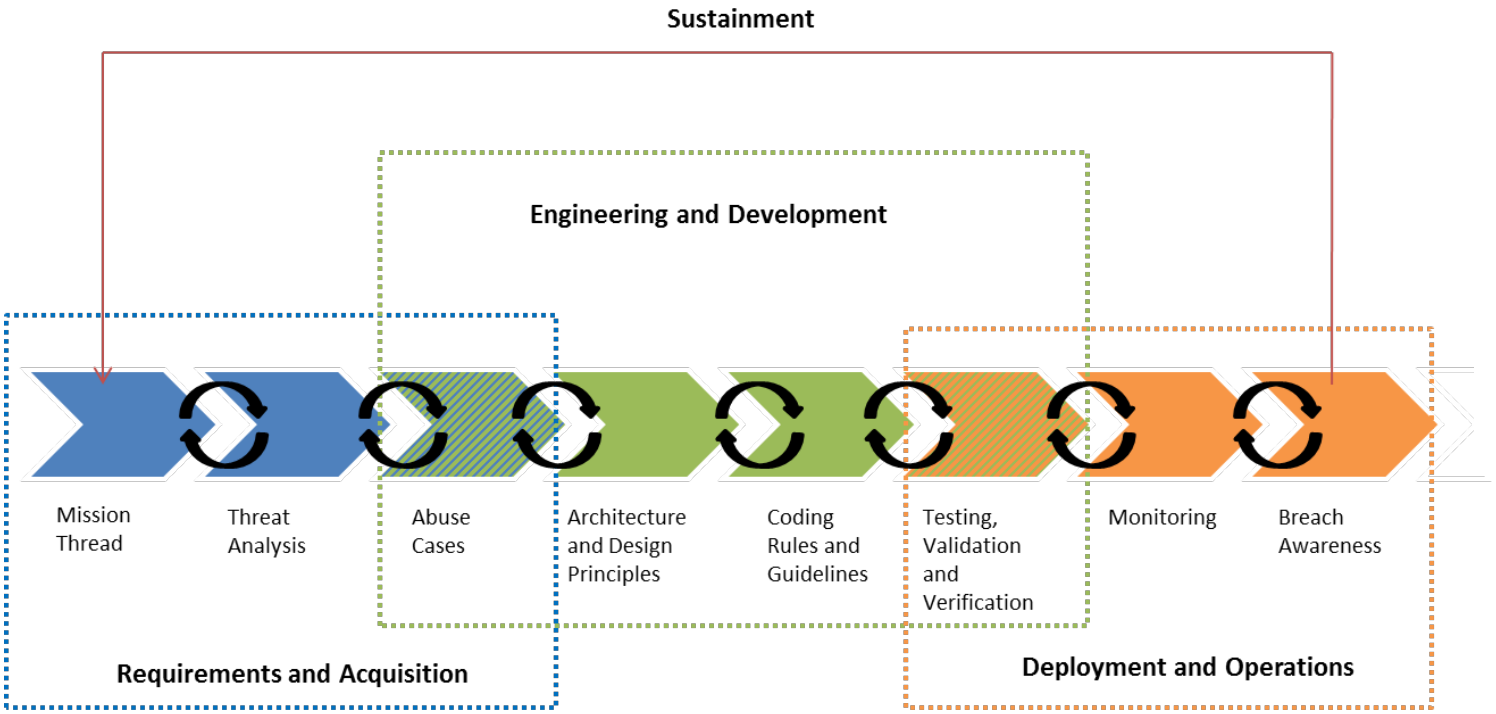
— **Acquisition Lifecycle Assurance**

7. Secure Software Development Practices
8. Secure Software Operation Practices

— **Software Security**

9. Software Security Infrastructure Practices — **Software Security Infrastructure**

Security is a lifecycle issue



Contact Information

Mark Sherman

(412) 268-9223

mssherman@sei.cmu.edu

Web Resources (CERT/SEI)

http://www.cert.org/

http://www.sei.cmu.edu/

Software Engineering Institute | Carnegie Mellon University