Building Secure Software for Mission Critical Systems

Mark Sherman, PhD
Technical Director, CERT

Software Solutions Conference 2015
November 16–18, 2015

Agenda

- **State of software**
- **Building software: the Secure Software Development Lifecycle**
  - Requirements
  - Development
  - Operations
- **Review**

“Software is eating the world”

Marc Andreessen

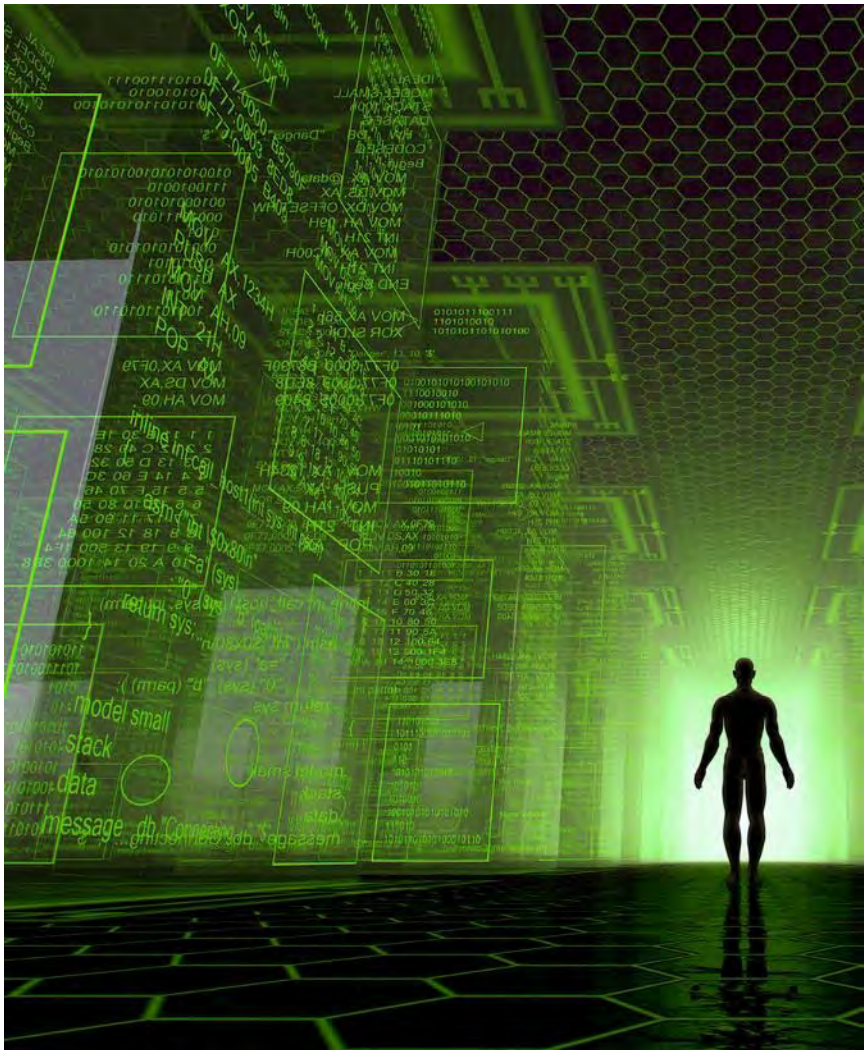
Wall Street Journal

Aug 20, 2011

Software is the new Hardware

Source: http://www.wsj.com/articles/SB10001424053111903480904576512250915629460

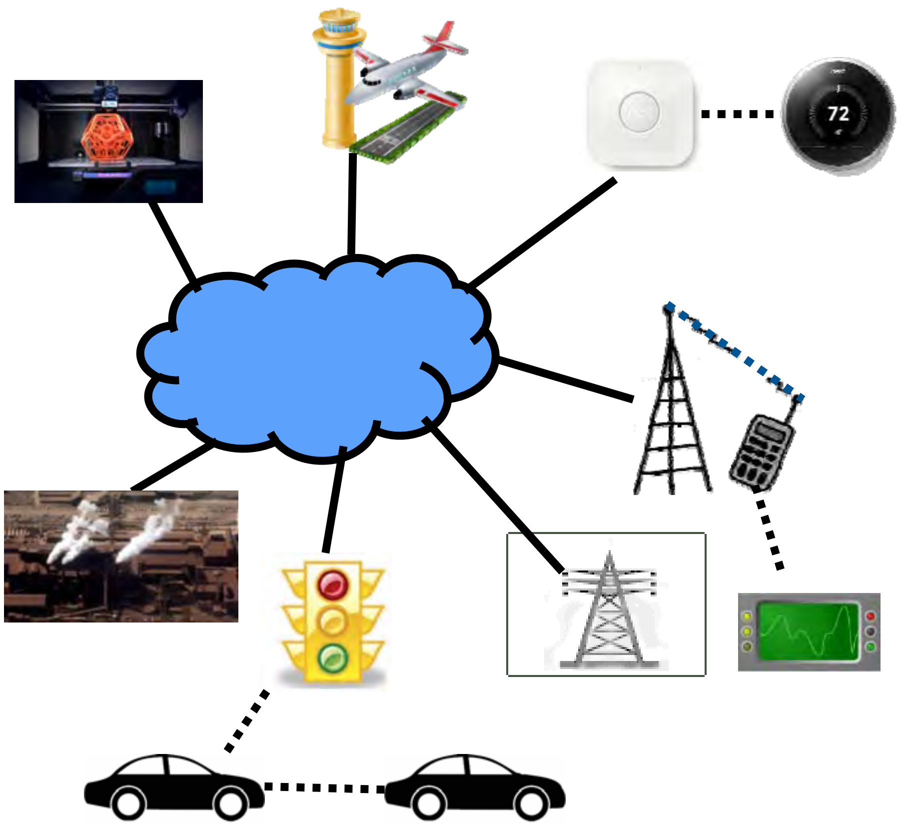
Software is the new hardware – IT



IT moving from specialized hardware to software, virtualized as

- Servers: virtual CPUs
- Storage: SANs
- Switches: Soft switches
- Networks: Software defined networks

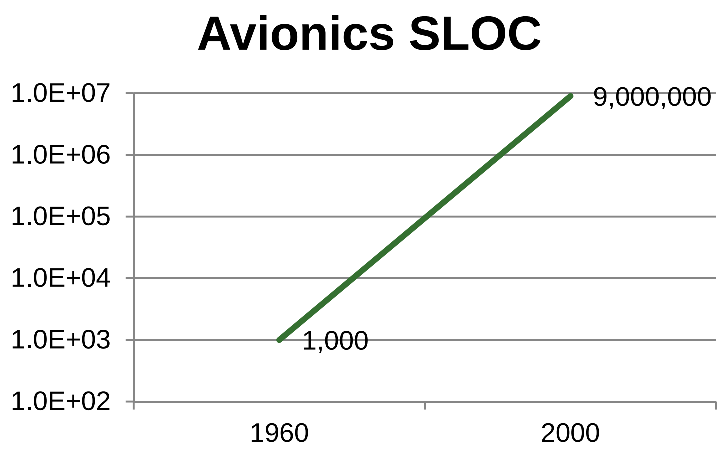
Software is the new hardware – cyber physical

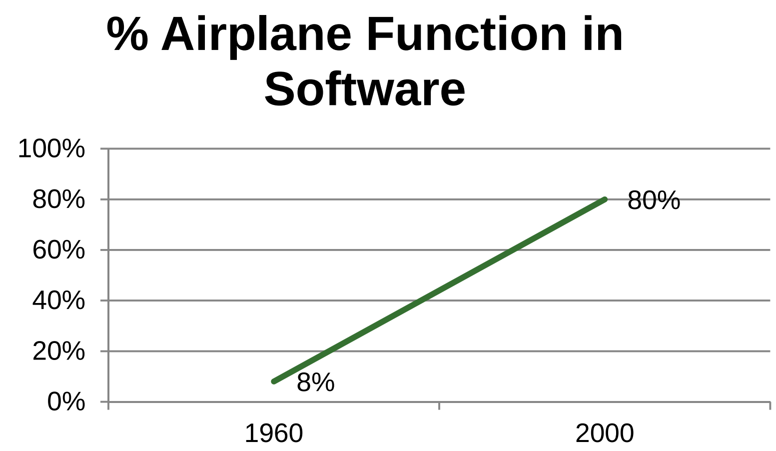


- Cellular
  - Main processor
  - Base band processor (SDR)
  - Secure element (SIM)
- Automotive
  - Autonomous vehicles
  - Vehicle to infrastructure (V2I)
  - Vehicle to vehicle (V2V)
- Industrial and home automation
  - 3D printing (additive manufacturing)
  - Autonomous robots
  - Interconnected SCADA
- Aviation
  - Next Gen air traffic control
- Smart grid
  - Smart electric meters
  - Smart metering infrastructure
- Embedded medical devices

Complex software is business and mission critical

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

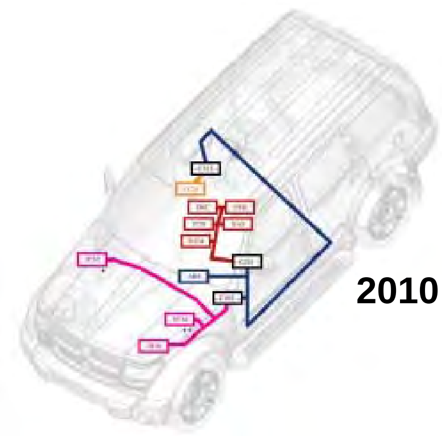




Sources: Final Report, NASA Study on Flight Software Complexity

March 2009; Mel Conway, "Tower of Babel and the Fighter Plane," Oct 9, 2013

Vehicle technology following the same path

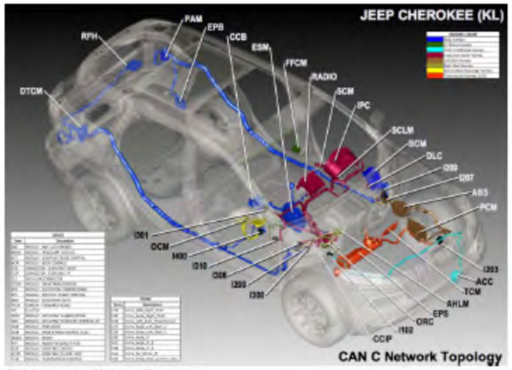


2014 Jeep Cherokee

(32 ECUs)

2010 Jeep Cherokee

(12 ECUs)



CAN-C Network – 2014 Jeep Cherokee

Common assertion that modern high end vehicles have

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

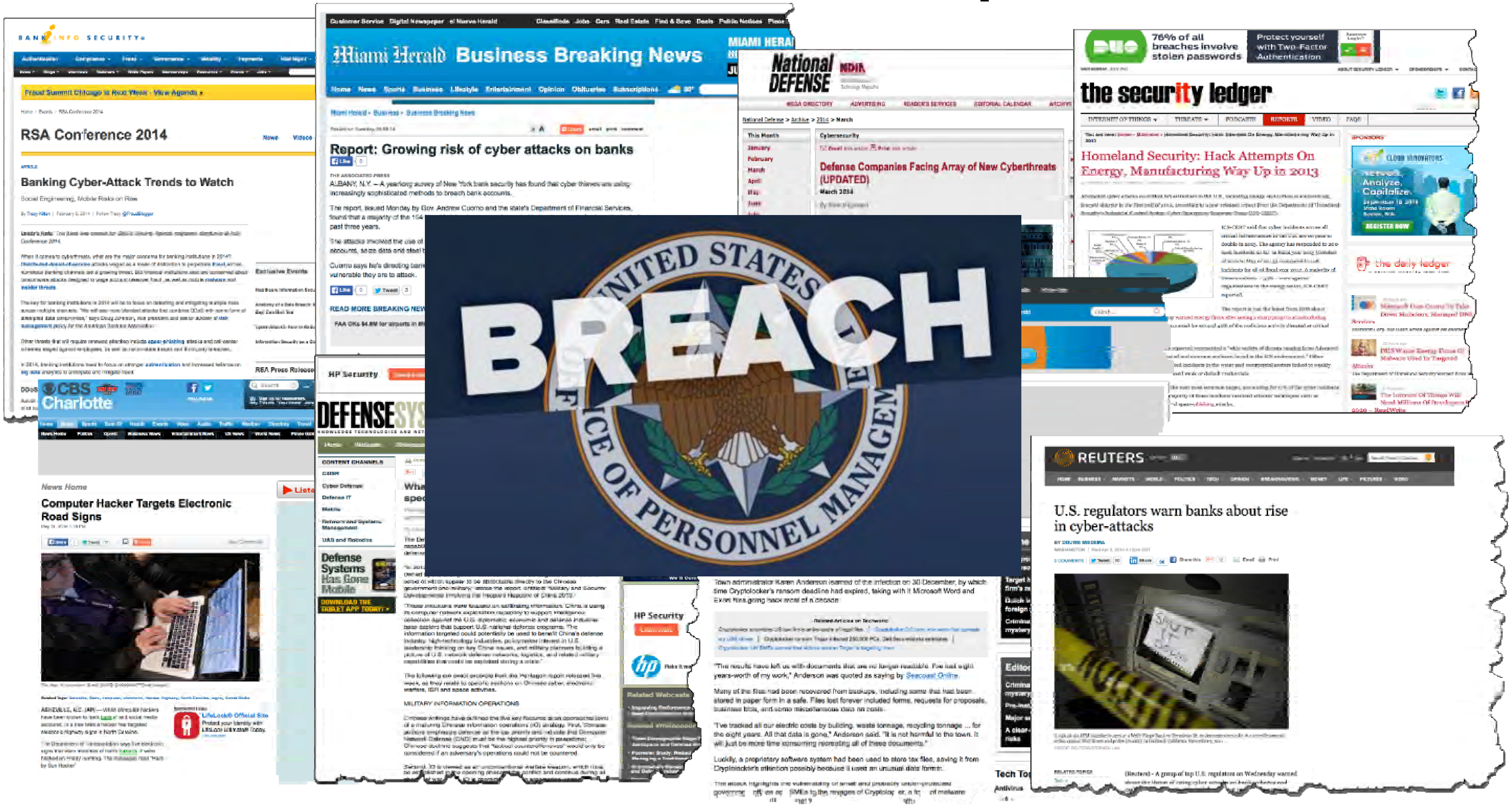
Sources: Miller and Valasek, A Survey of Remote Automotive Attack Surfaces, http://illmatics.com/remote%20attack%20surfaces.pdf;

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

MSS7

Software vulnerabilities are ubiquitous



Slide 9

MSS7 New slide before 14 -- what's the external view why we need to get better and fast
Mark S. Sherman, 7/13/2015

Cyber attacks on physical systems

Steelworks compromise causes massive damage to furnace.

One of the most concerning was a targeted APT attack on a German steelworks which ended in the attackers gaining access to the business systems and through them to the production network (including SCADA). The effect was that the attackers gained control of a steel furnace and this caused massive damages to the plant.

Dragonfly attacks a dozen companies

The Dragonfly hacker group attacked a number of companies' SCADA systems and installed the malware 'Havex'. This was used to gather information about the systems. No damage was done, because the compromise was detected and removed before the hackers had completed the observation and intelligence gathering phase.

Sources: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2014.pdf?__blob=publicationFile;

http://www.resilienceoutcomes.com/state-ict-security/

Sections ☰ The Washington Post

Business

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

[GREENE BROILLET & WHEELER, LLP]
WHERE SUCCESS IS A TRADITION®

Toyota Sudden Acceleration Defect Case: $1.1 Billion Settlement

Target on Friday revised the number of customers whose personal information was stolen in a widespread data breach during the holiday season, now reporting a range of 70 million to 110 million people.

The stunning figure represents about a third of all American adults at the low end, and is nearly three times as great as the company's original estimate at the upper end. The theft is one of the largest ever of retail data.

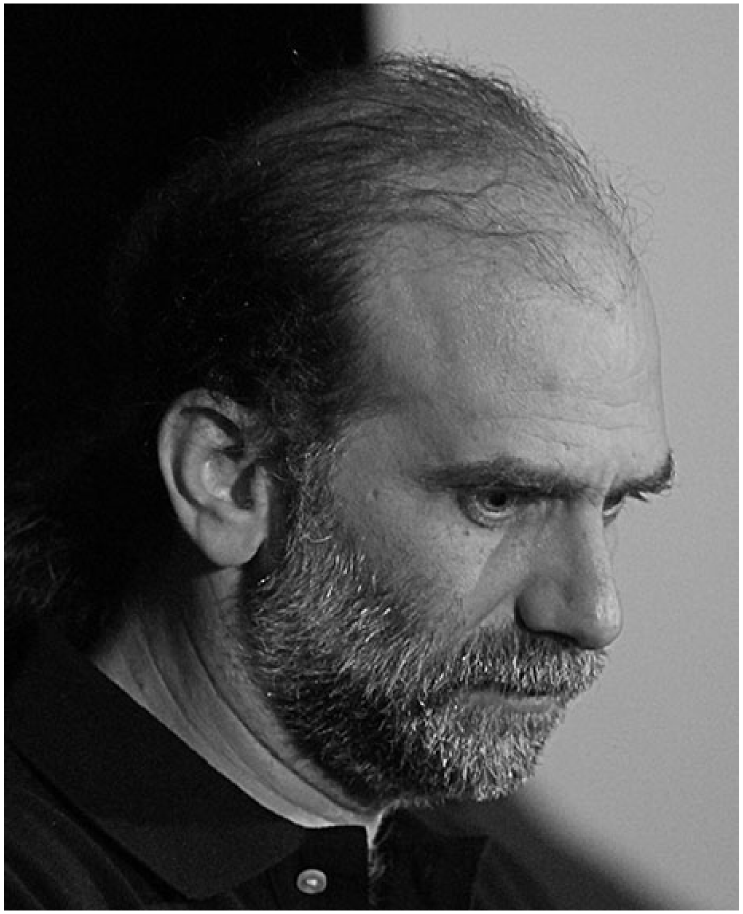
Source: New York Times, Jan 10, 2014

Average cost in a breach:

US$188 per record

Source: Ponemon Institute, "2013 Cost of Data Breach Study: Global Analysis", May 2013

An ounce of prevention ….



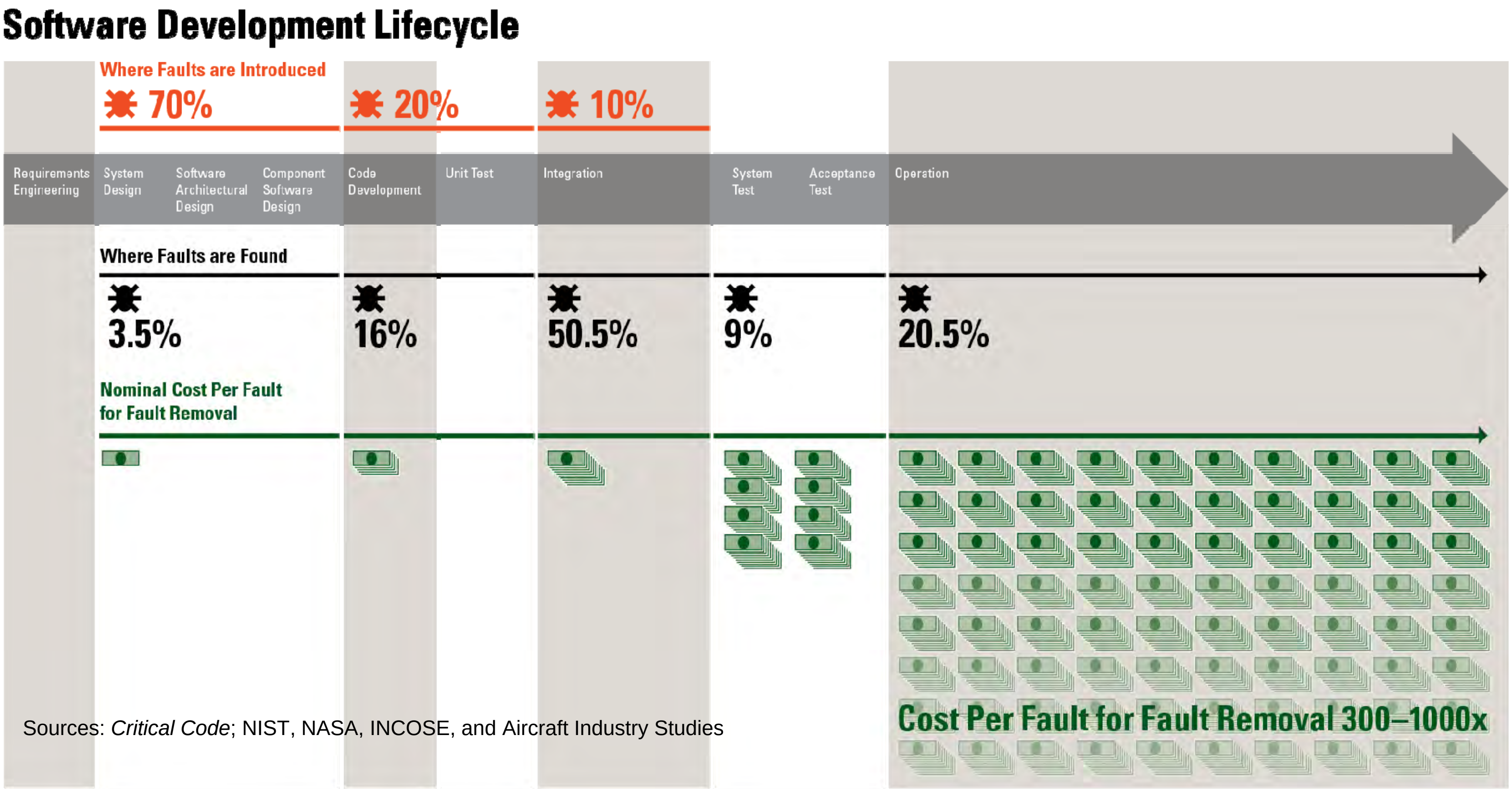
"We wouldn't have to spend so much time, money, and effort on network security if we didn't have such bad software security."

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; http://www.greene-broillet.com/Articles/Toyotasuddenacceleration.shtml

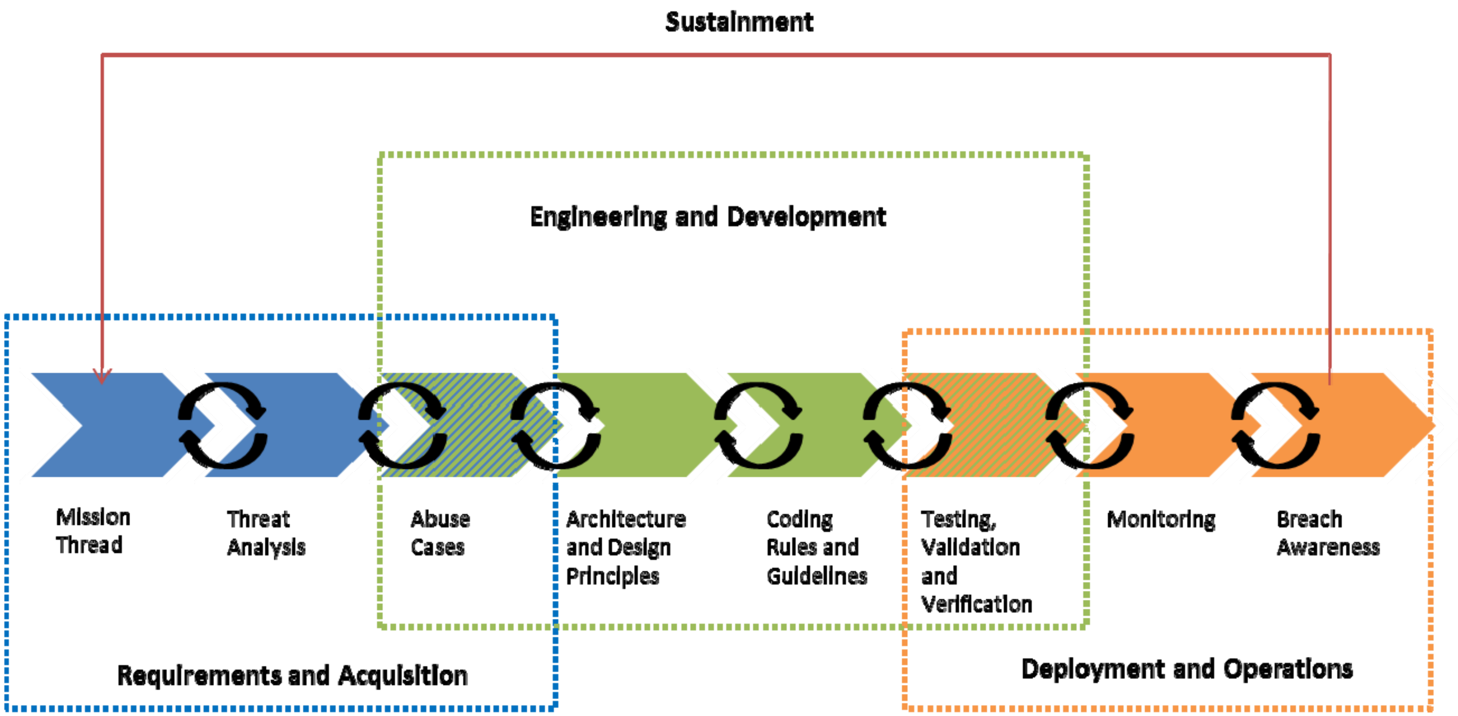
Catching software faults early saves money

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

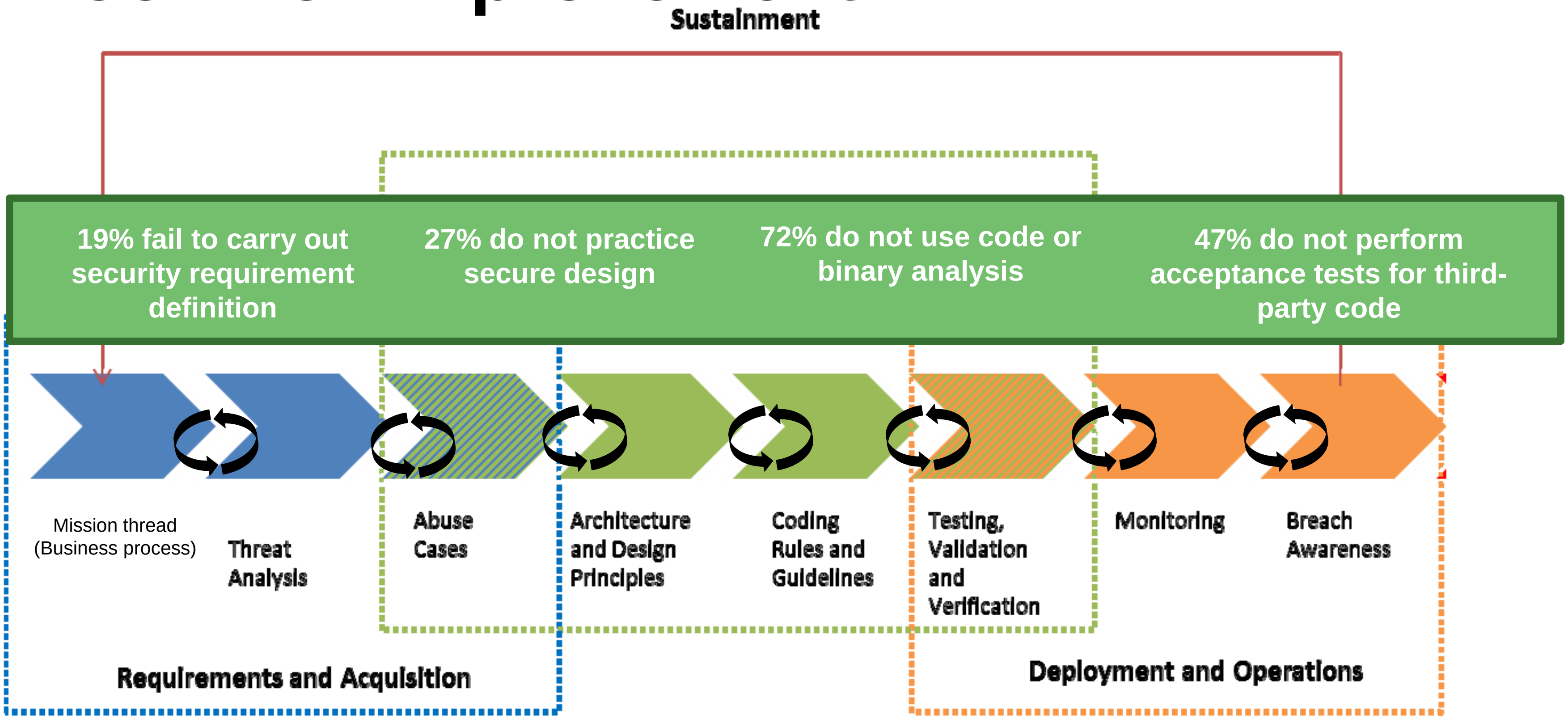


Sources: *Critical Code*; NIST, NASA, INCOSE, and Aircraft Industry Studies

Security is a lifecycle issue



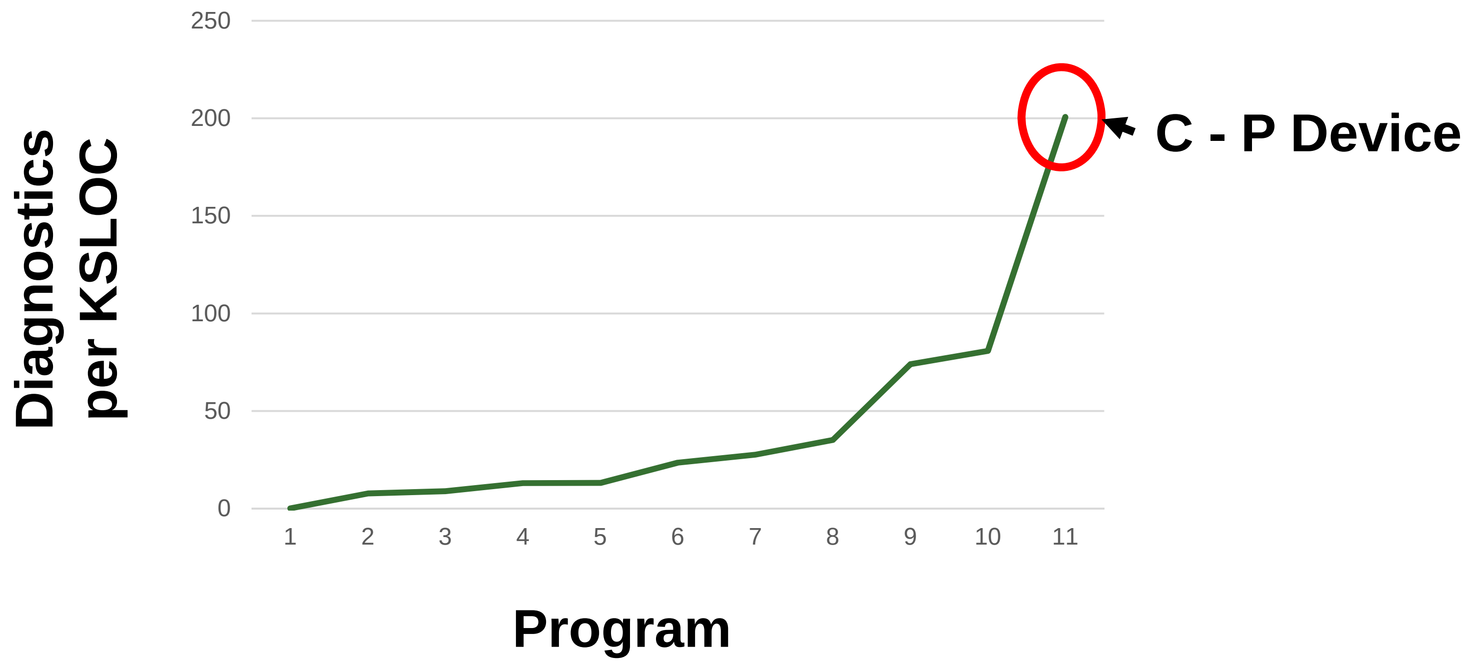
Room for improvement



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

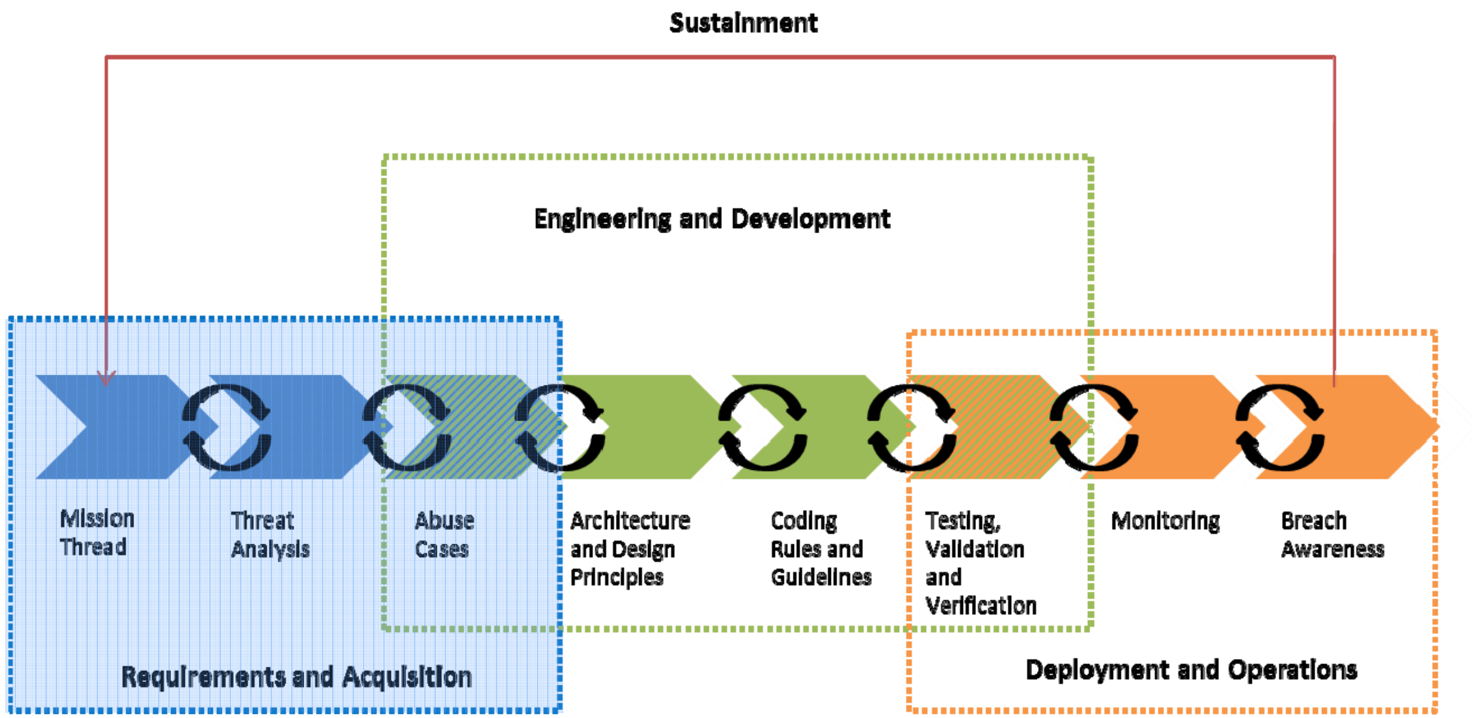
Sources: Forrester Consulting, "State of Application Security," January 2011; Wendy Nather, Research Director, 451 Research, "Dynamic testing: Why Tools Alone Aren't Enough, March 25, 2015"

There is a wide range of application security quality

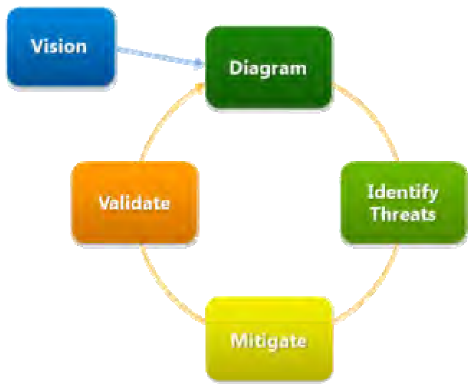


Source: CERT sample of evaluated programs

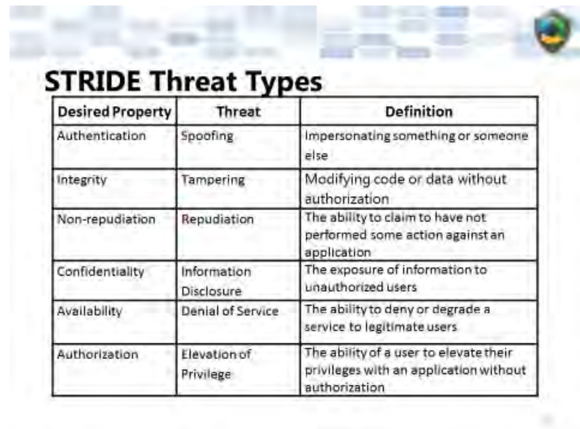
Requirements



Threat analysis tools help derive abuse and misuse cases



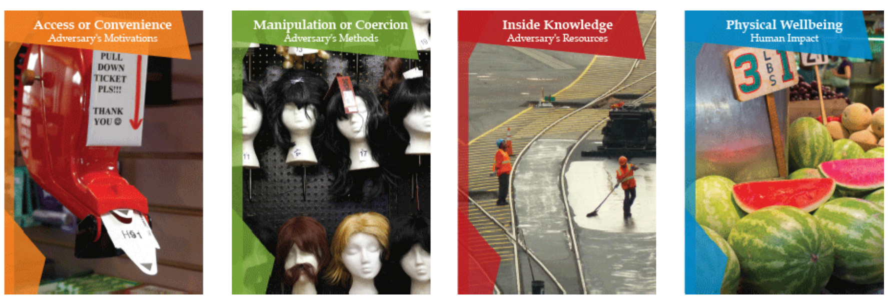
Microsoft SDL Threat Modeling Tool



STRIDE Threat Types

Desired Property	Threat	Definition
Authentication	Spoofing	Impersonating something or someone else
Integrity	Tampering	Modifying code or data without authorization
Non-repudiation	Repudiation	The ability to claim to have not performed some action against an application
Confidentiality	Information Disclosure	The exposure of information to unauthorized users
Availability	Denial of Service	The ability to deny or degrade a service to legitimate users
Authorization	Elevation of Privilege	The ability of a user to elevate their privileges with an application without authorization

Microsoft STRIDE Threat Types



Denning, Friedman, Kohno
The Security Cards: Security Threat Brainstorming Toolkit

Jane Cleland-Huang's Persona non Grata
http://www.infoq.com/articles/personae-non-gratae

Embedded systems represent new classes of vulnerabilities

Embedded systems have different characteristics than IT systems

- Size, weight, power and latency concerns
- Unique architectures of embedded controllers
- Bit and clock cycle level operations
- Physical resources with real time sensors
- Safety-Critical Real-time OS
- Intermittent communications
- Multiple command-and-control masters
- Embedded firmware,
- Unique internal busses & controllers
- Developers are engineers, not software developers

Security approaches for IT systems do not cover embedded system security

Responses to attack surfaces and threat models not generally reusable

Virus definitions and operating guidelines do not always apply

Firewalls and IDS/IPS of limited value

Centralized account control not possible

Network tools and assessment techniques unaware of embedded systems architecture and interfaces

Larger number of attack surfaces

More diverse attack surfaces

Maintenance backdoors

Hardcoded credentials

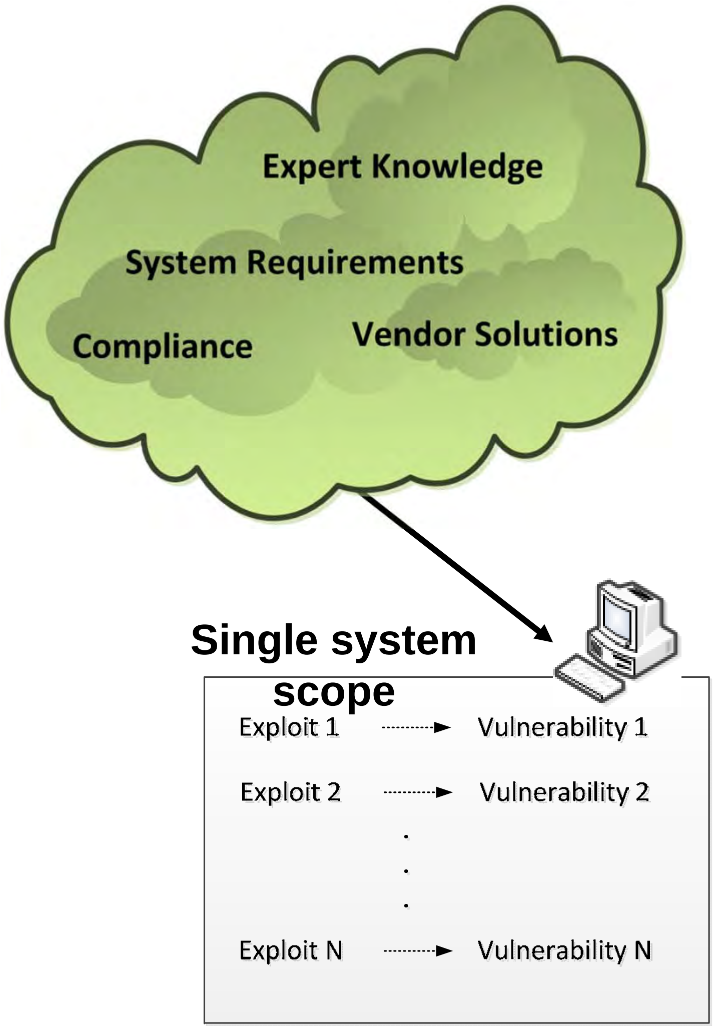
Unique and insecure protocols

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 improves behavior	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

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)

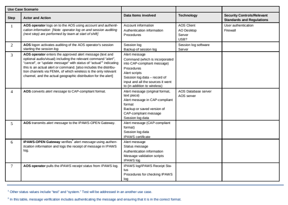
Need for systematic, multiple system evaluations

- Notation for expressing a security events and risks
- Take into account all context

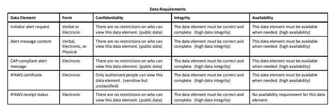
Security Engineering Risk Analysis approach

Comprehensive context

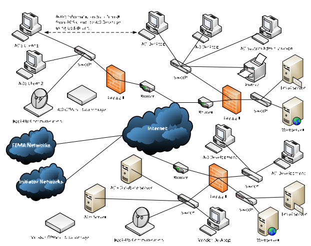
Use-Case View



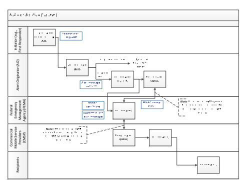
Data View



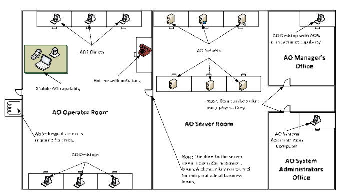
Network View



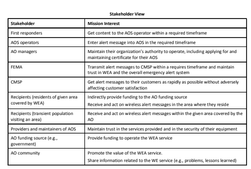
Workflow View



Physical View



Stakeholder View

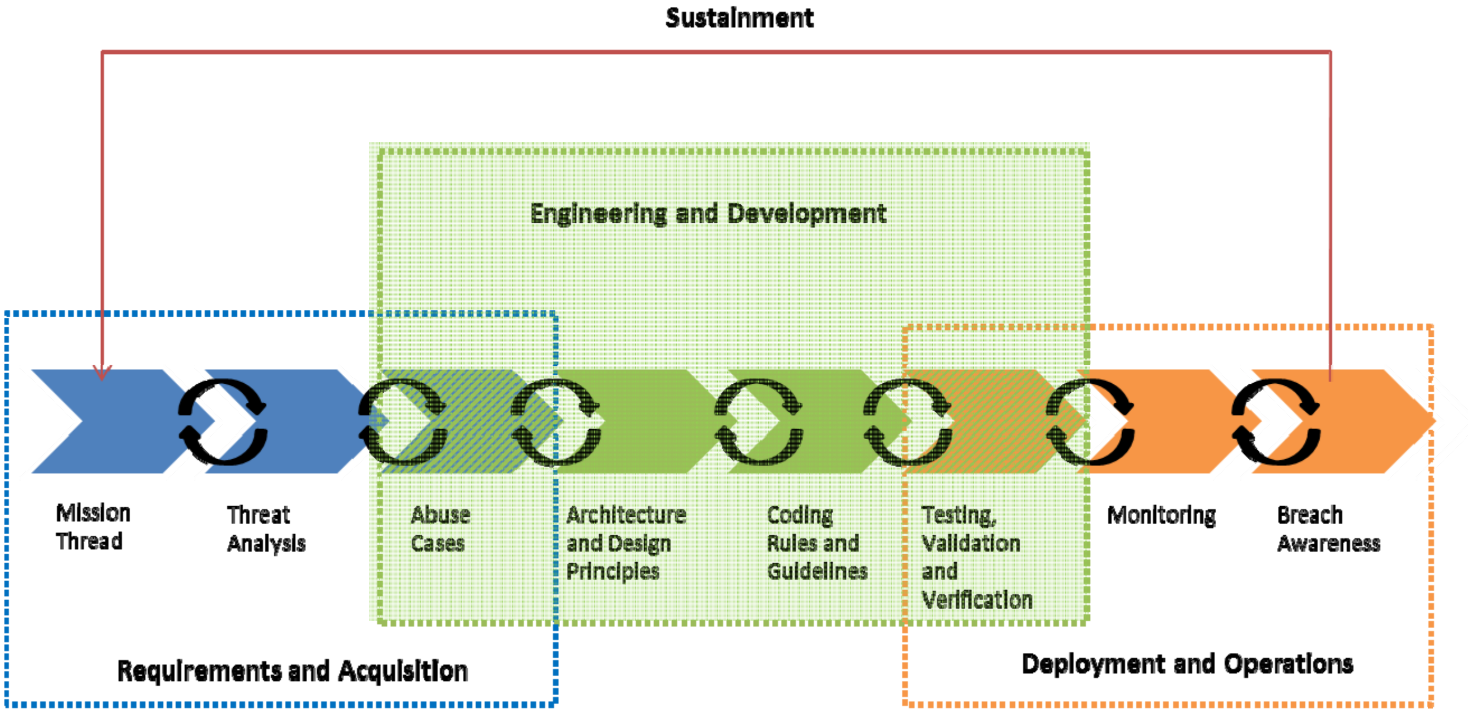


Determining actions

- Establish threat model
- Determine common system view
- Inspect connections between systems
- Evaluate
  - Consequences
  - Likelihood
  - Risk

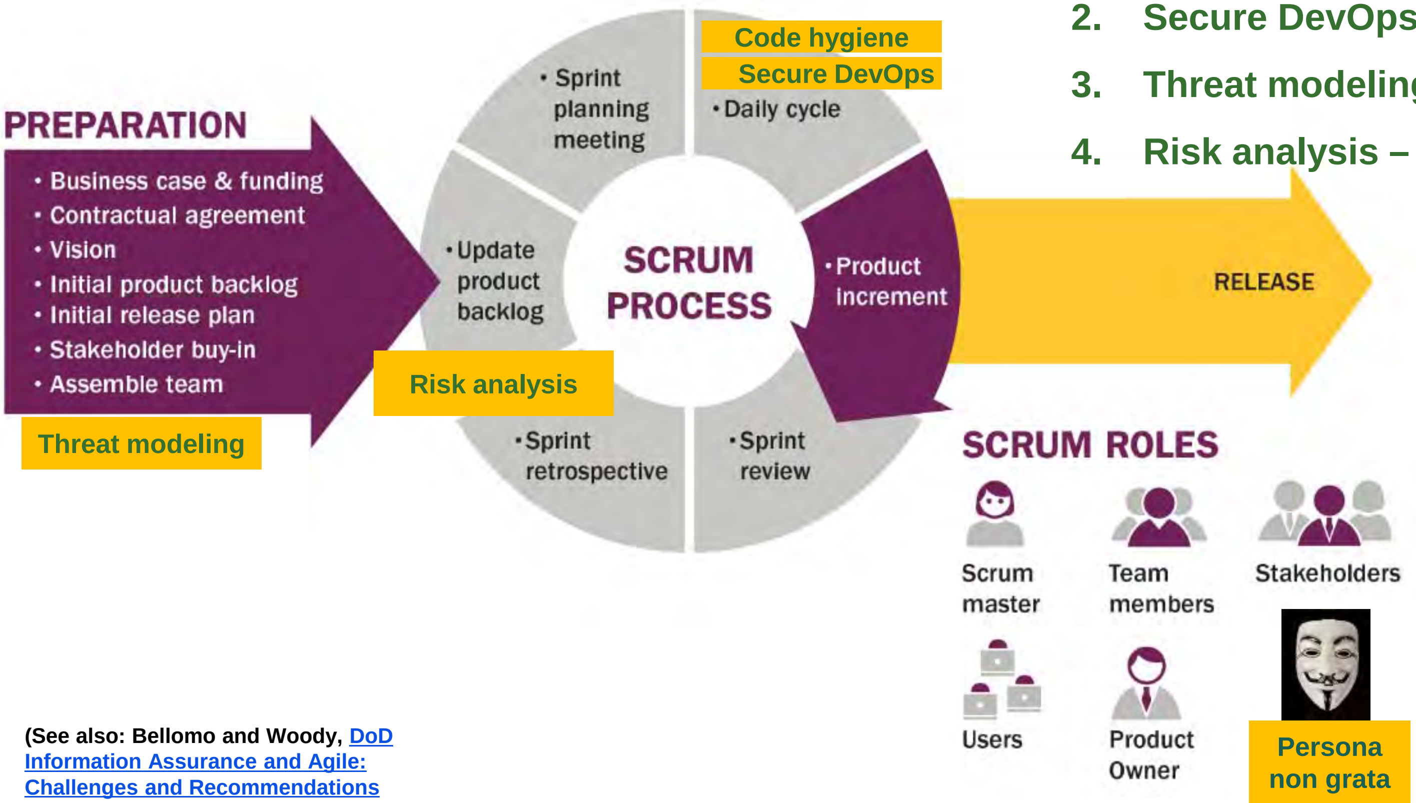
http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=427321

Development



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

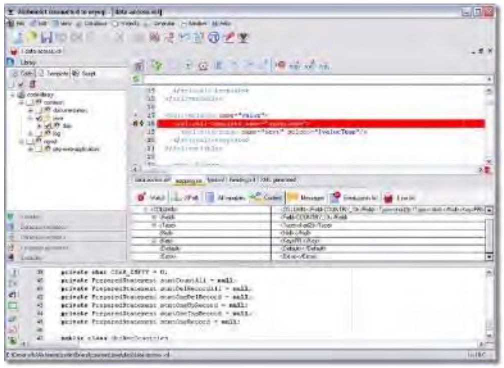


(See also: Bellomo and Woody, DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers (http://repository.cmu.edu/cgi/viewcontent.cgi?article=1674&context=sei)

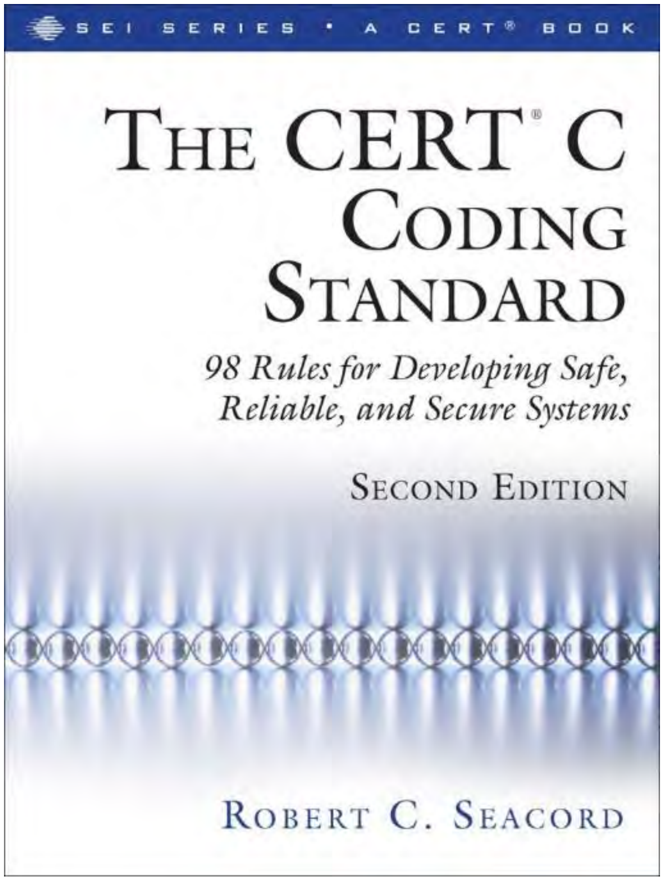
Adoption of secure coding rules

Training

Integrated development environments



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

Learning from rules and recommendations

Rules and recommendations in the secure coding standards focus to improve behavior

The "Ah ha" moment:

Noncompliant code examples or antipatterns in a pink frame—do not copy and paste into your code

Noncompliant Code Example

In this example, the FormatMessage() function allocates a buffer and stores it in the buf parameter. From the documentation of FORMAT_MESSAGE_ALLOCATE_BUFFER [MSDN]:

> The function allocates a buffer large enough to hold the formatted message, and places a pointer to the allocated buffer at the address specified by lpBuffer. The lpBuffer parameter is a pointer to an LPTSTR; you must cast the pointer to an LPTSTR (for example, (LPTSTR)&lpBuffer). The nSize parameter specifies the minimum number of TCHARs to allocate for an output message buffer. The caller should use the LocalFree function to free the buffer when it is no longer needed.

Instead of freeing the memory using LocalFree(), this code example uses GlobalFree() erroneously.

```
LPTSTR buf;
DWORD n = FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
                        FORMAT_MESSAGE_FROM_SYSTEM |
                        FORMAT_MESSAGE_IGNORE_INSERTS, 0, GetLastError(),
                        LANG_USER_DEFAULT, (LPTSTR)&buf, 1024, 0);
if (n != 0) {
  /* Format and display the error to the user */

  GlobalFree(buf);
}
```

Compliant Solution

The compliant solution uses the proper deallocation function as described by the documentation.

```
LPTSTR buf;
DWORD n = FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
                        FORMAT_MESSAGE_FROM_SYSTEM |
                        FORMAT_MESSAGE_IGNORE_INSERTS, 0, GetLastError(),
                        LANG_USER_DEFAULT, (LPTSTR)&buf, 1024, 0);
if (n != 0) {
  /* Format and display the error to the user */

  LocalFree(buf);
}
```

Compliant solutions in a blue frame that conform with all rules and can be reused in your code

Secure Coding in C/C++ Training

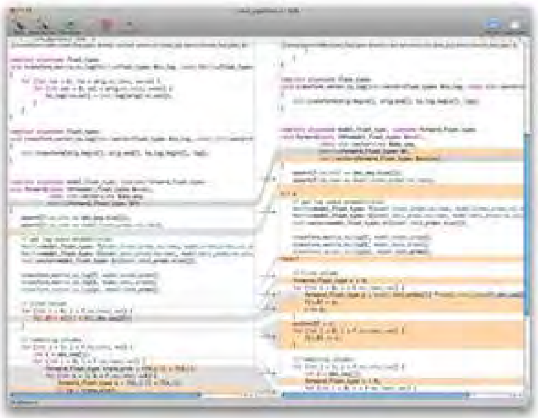
The Secure Coding course is designed for C and C++ developers. It encourages programmers to adopt security best practices and develop a security mindset that can help protect software from tomorrow's attacks, not just today's.

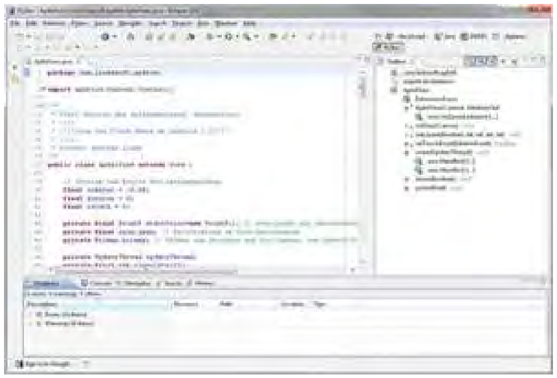
Topics

- String management
- Dynamic memory management
- Integral security
- Formatted output
- File I/O

Additional information at ttp://www.sei.cmu.edu/training/p63.cfm

Tools encourage application of secure coding





Moving rules into IDE improves application of secure coding

- Early feedback corrects errors on introduction
- Exceptions are understood in context
- Feedback improves developer skill

Target Clang static analyzer (C based languages)

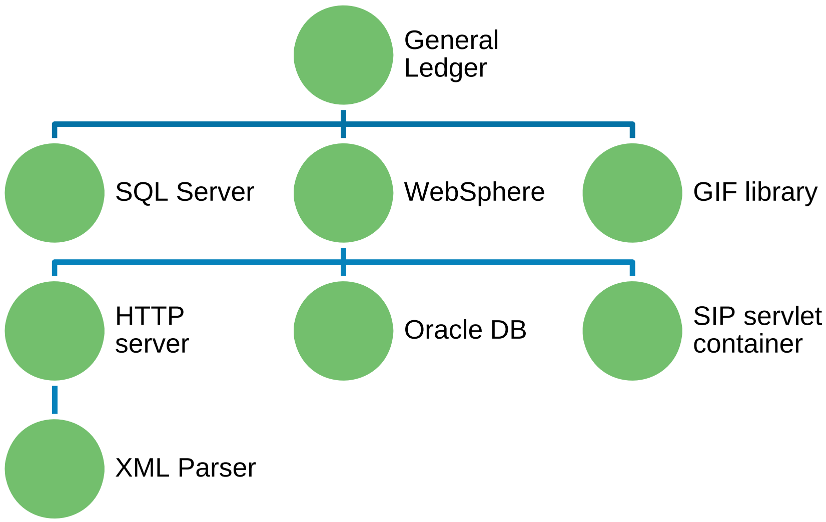
- Widely used open source front end for popular compilers
- Integrated into Apple's Xcode IDE

Target FindBugs (Java)

- Integrated into Eclipse and JDeveloper

Software depends on a supply chain of components

Development is largely assembly

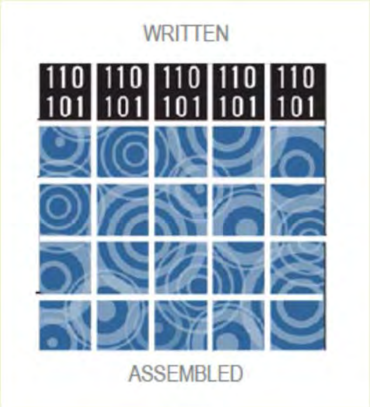


Collective development – context:

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

Note: hypothetical application composition

Substantial open source contained in supply chain



- 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

Source: Sonatype, 2014 Sonatype Open Source Development and Application Security Survey;

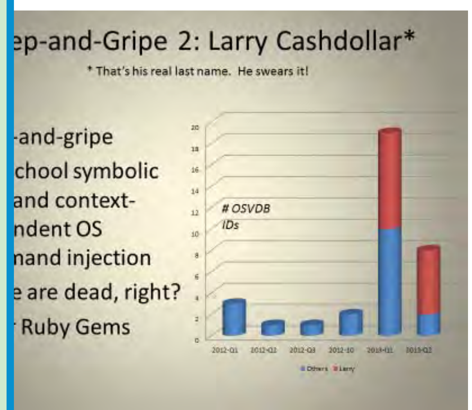
Open source is not secure

Heartbleed and Shellshock were found by exploitatio

Other open source software illustrates vulnerabilities from cursory inspection

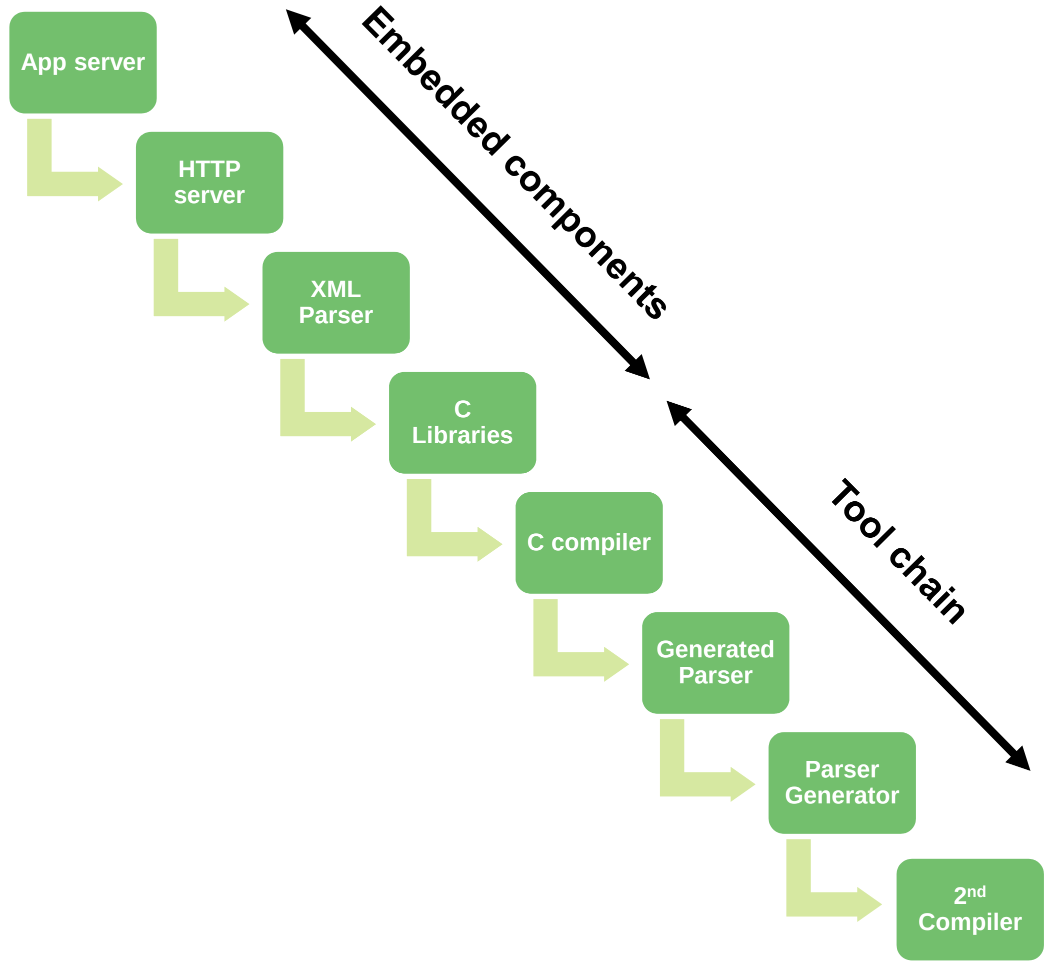
46 million vulnerable open source components downloaded annually

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

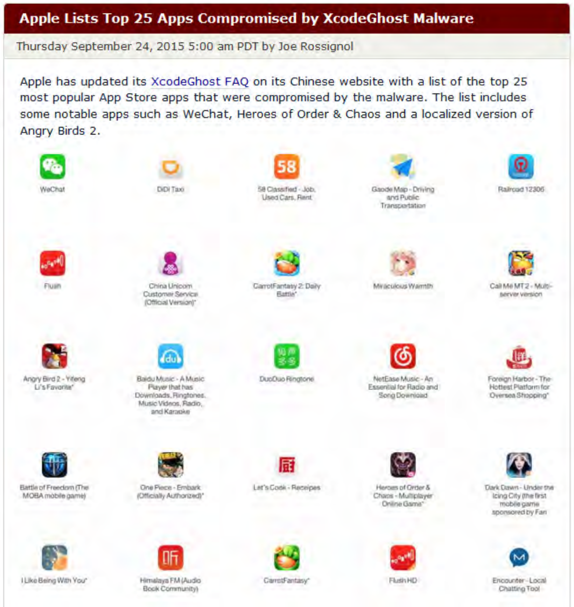


Sources: Steve Christey (MITRE) & Brian Martin (OSF), Buying Into the Bias: Why Vulnerability Statistics Suck, https://media.blackhat.com/us-13/US-13-Martin-Buying-Into-The-Bias-Why-Vulnerability-Statistics-Suck-Slides.pdf; Sonatype, Sonatype Open Source Development and Application Security Survey; Aspect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

Open source supply chain has a long path



Corruption in the tool chain already exists



Apple Lists Top 25 Apps Compromised by XcodeGhost Malware

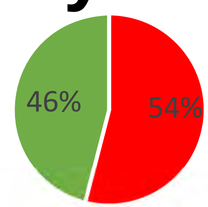
Thursday September 24, 2015 5:00 am PDT by Joe Rossignol

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.

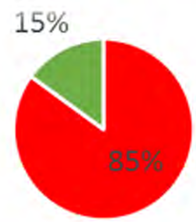
- XcodeGhost corrupted Apple's development environment
- Major programs affected
  - WeChat
  - Badu Music
  - Angry Birds 2
  - Heroes of Order & Chaos
  - iOBD2

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

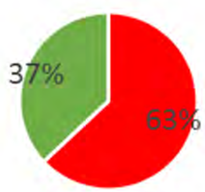
Existing Customer Premise Equipment (SOHO) typically vulnerable



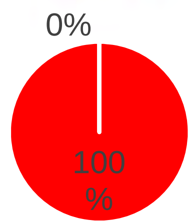
54% of tested routers are vulnerable to cross-site request forgery (CSRF)



85% of tested routers use non-unique default credentials



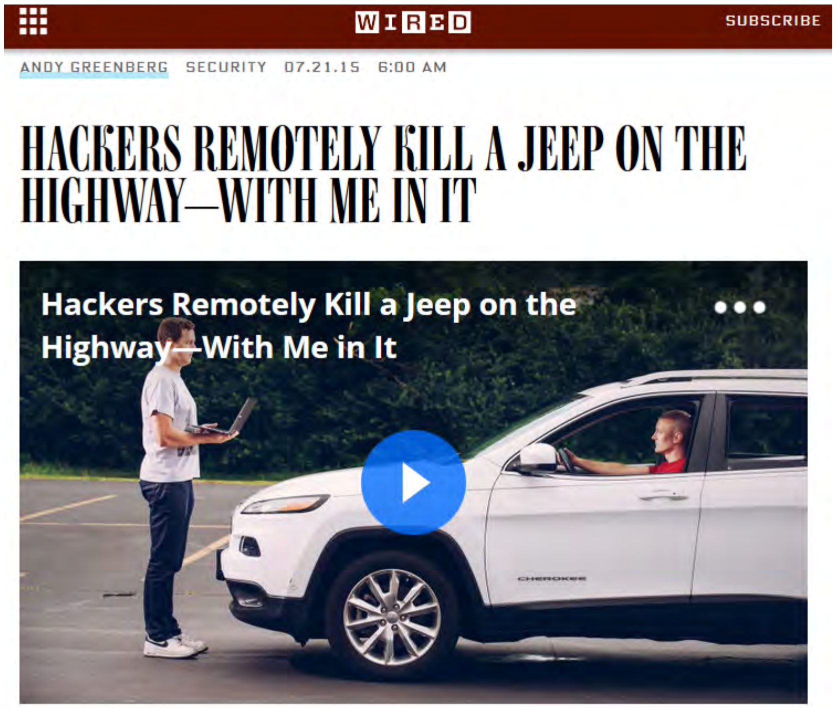
63% of tested routers are vulnerable to DNS spoofing attacks



100% of router firmware use BusyBox versions from 2011 or earlier and embedded Linux kernel versions from 2010 or earlier

Source: Land, J. "Systemic Vulnerabilities in Customer-Premises Equipment Routers," unpublished white paper, 2015

Connecting automotive systems to internet opens system to attack



Extending systems opens vulnerabilities not anticipated

- Optimizations performed assuming one attack method
- Assumptions no longer hold with additional integrations

Studies suggest that new operational environments are a leading cause for introducing new vulnerabilities in existing systems.

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

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."

Need to manage software supply chain

Software Supply Chain risk for a product **needs to be** reduced to acceptable level

Supplier Capability

Supplier follows practices that reduce supply chain risks

Product Security

Delivered or updated product is acceptably secure.

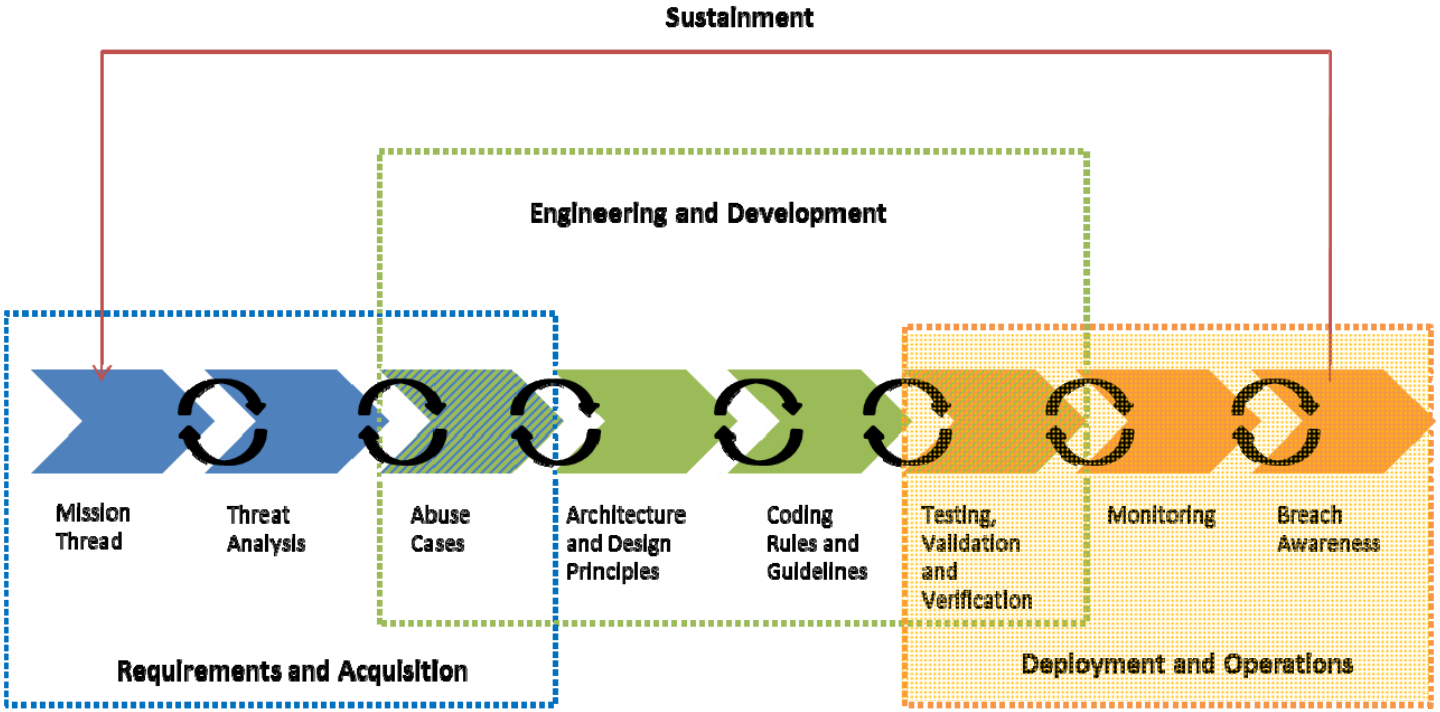
Product Distribution

Methods of transmitting the product to the purchaser guard again tampering

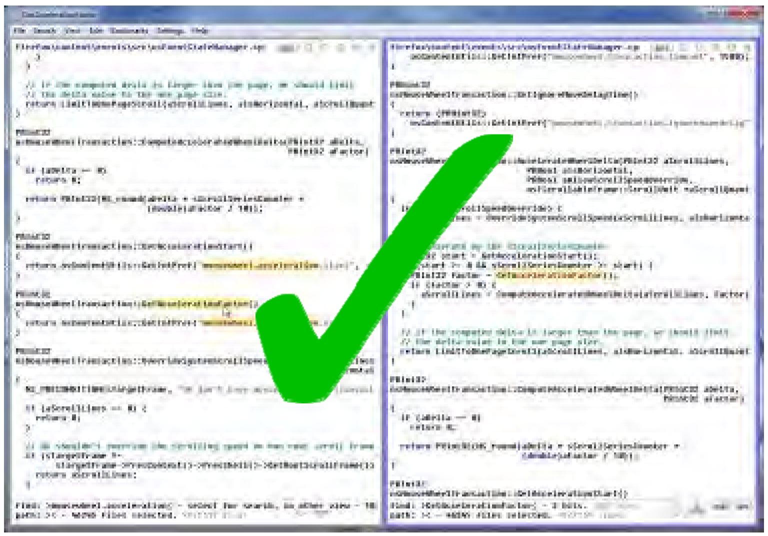
Operational Product Control

Product is used in a secure manner.

Deployment and operations



Static Testing – Source code analysis tools



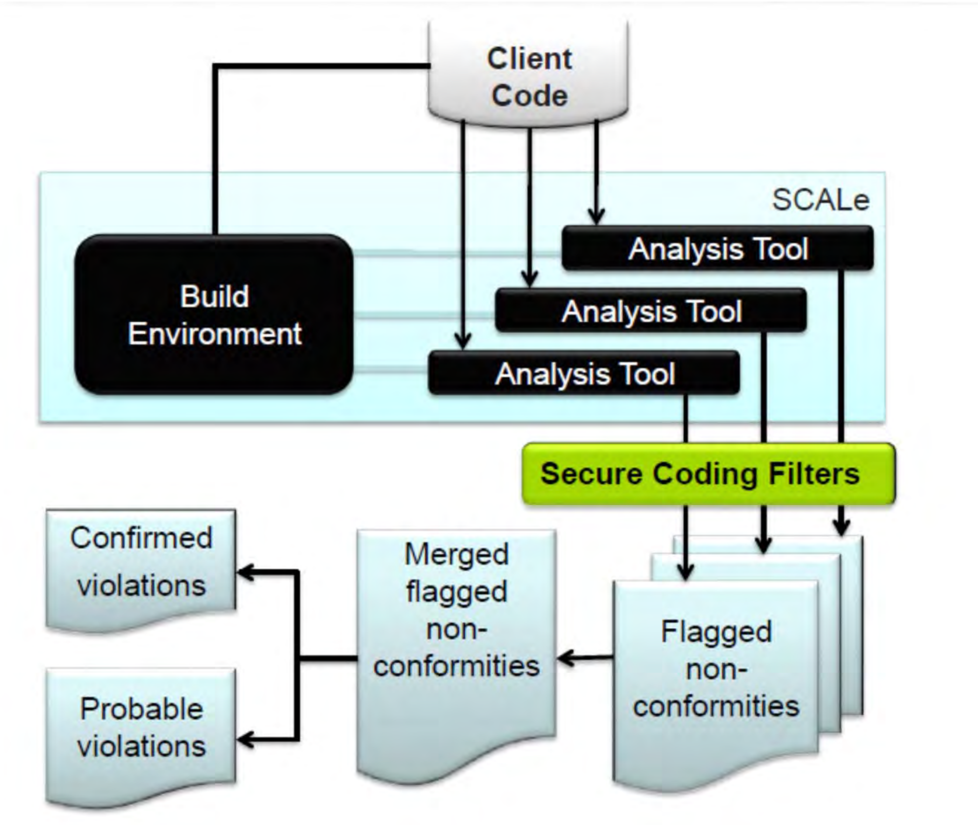
Secure Code Analysis Laboratory (SCALe)

- C, C++, Java, PERL, Python, Android rule conformance checking
- Thread safety analysis
- Information flows across Android applications
- Operating system call flows

Static testing optimization

- SCALe set up
- SCALe filters and visualizer
- Tool conformance and capability testing
- Multitool integration and statistical optimizer

SCALe Multitool evaluation



Improve expert review productivity by focusing on high priority violations

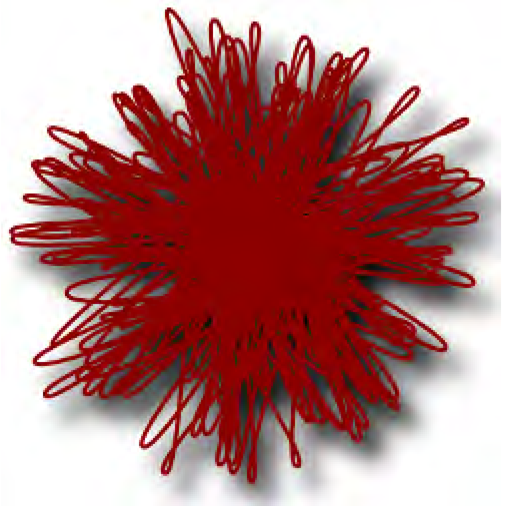
Filter select secure coding rule violations

- Eliminate irrelevant diagnostics
- Convert to common CERT Secure Coding rule labeling

Single view into code and all diagnostics

Maintain record of decisions

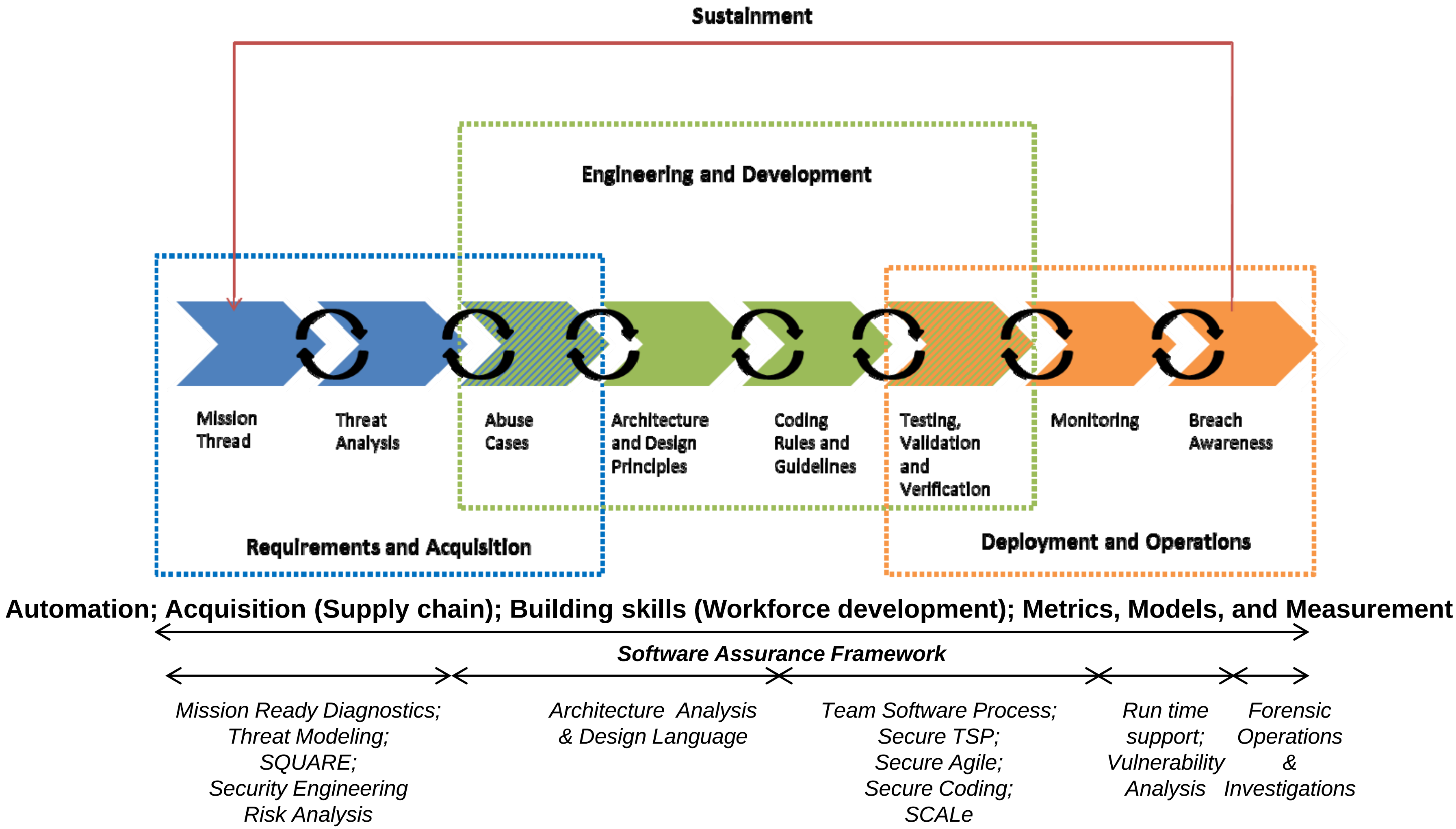
Dynamic testing and evaluation – fuzzing



Fuzz testing of attack surfaces

- Based on techniques used in CERT's Basic Fuzzing Framework (BFF)
- mutational fuzzing
- machine learning and evolutionary computing techniques
- adjusts its configuration parameters based on what it finds (or does not find) over the course of a fuzzing campaign

Review: Secure Software Development Lifecycle



Contact Information

Mark Sherman

(412) 268-9223

mssherman@sei.cmu.edu

Web Resources (CERT/SEI)

http://www.cert.org/

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