Software Intelligence for Digital Leaders

CAST Security

Communicate - Decide – Measure - Protect – Discover- Improve

Security breaches often cause bad headlines

Facts & Figures

- Estimated average annualized cost of cybersecurity is $11.7M
- 22.7% increase in cost of cybersecurity in a year
- Estimated average number of security breaches each year is 130
- 27.4% increase in average annual number of security breaches
- Forbes - cybercrime will cost approximately $6 trillion per year on average through 2021

True cost of security breaches

The dollar cost of a cybersecurity breach is just the tip of the iceberg

Damaged shareholder and investor perception

Loss of data

Loss of reputation and damaged brand identity

The YAHOO Story

"Yahoo has been through three data breaches in recent years, where nearly two billion accounts were compromised in total. Those breaches may be behind why Verizon is now paying about $4 billion less to purchase the company than was offered just over a year ago." Source - Forbes

The Target Story

"In May 2017, Target paid out a $18.7 million settlement over a large-scale data breach that took place in 2013. The company said that the total cost of the breach was over $202 million." Source - Forbes

The Equifax Story

"Embattled Equifax CEO delivered a public mea culpa this month but it has failed to stop the doomsday headlines and investor exodus. Shares have lost 33% since the disclosure. In fact, this might be one of the worst crisis responses since BP's (NYSE:BP) CEO said, "I want my life back" after the Gulf oil spill." Source – Fox Business

Types of attack and security investment

Annualized Cost for different types of security attack

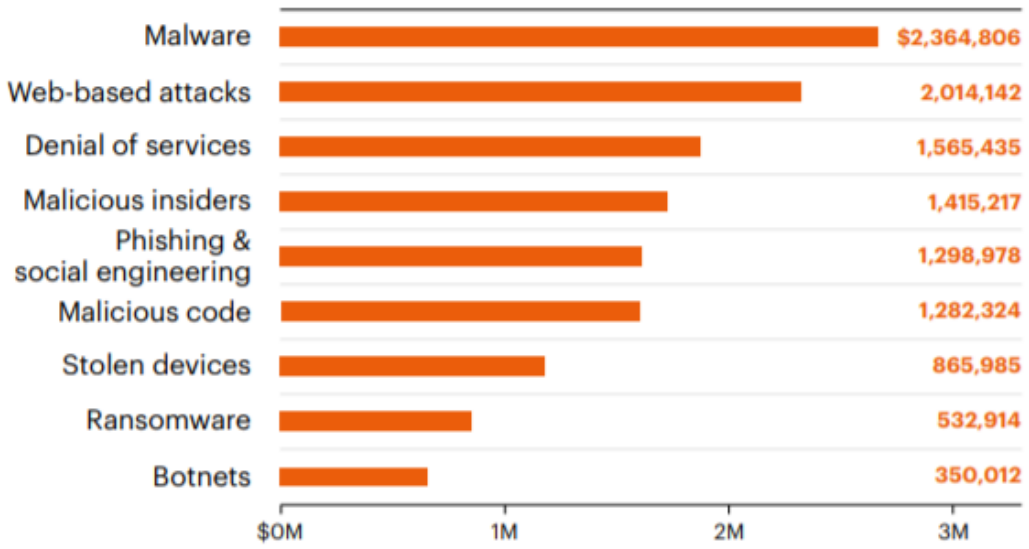
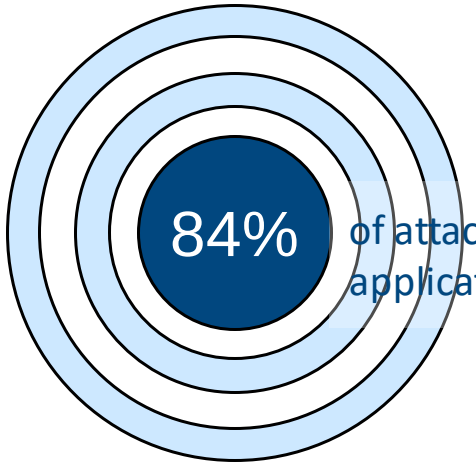


FIGURE 13
Total annualized cyber crime cost for attack types
US$ millions

Legend
Consolidated view
n = 254 separate companies

Source - Accenture

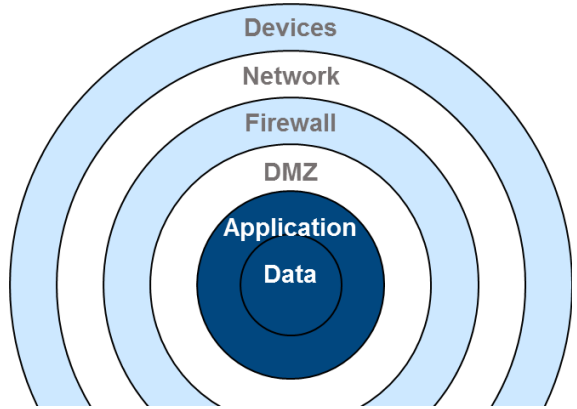
- Security breaches due to web-based attacks, malicious insiders, and malicious code are on the rise and costs due to these breaches are significantly high
- Yet spending on application and data security tends to be lowest



of attacks target application layer

Source: Gartner

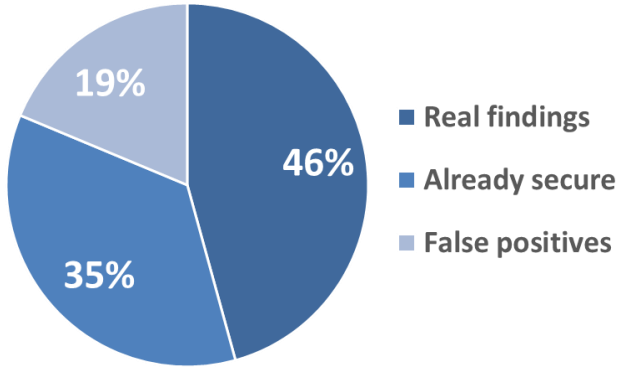
"Up to 70% of CWEs are actually quality defects." Source: SEI



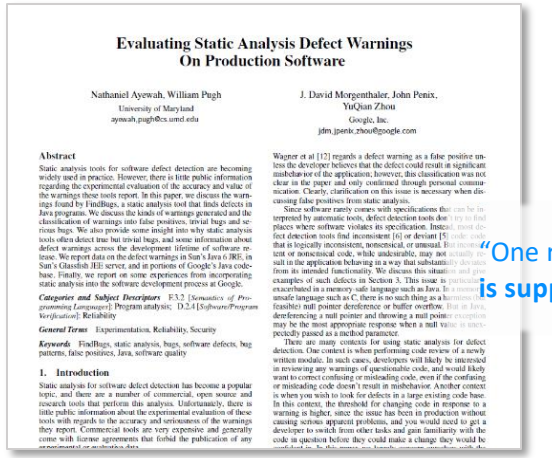
1 : 23 Ratio by which spend on perimeter outstrips application security

Limitations of current code analysis tools

Typical security tools – over 50% of findings are useless



- Unit Level Analysis
- Lack of deeper and contextual understanding of the code.
- Because of that these tools-
  - cannot detect severe issues such as forbidden access to data, lack of input validation, backdoors and insider threats
  - throw out a lot of false positives, and
  - suppresses significant number of true negatives



Evaluating Static Analysis Defect Warnings On Production Software

Nathaniel Ayewah, William Pugh
University of Maryland
aywah,pugh@cs.umd.edu

J. David Morgenthaler, John Penix, YuQian Zhou
Google, Inc.
jdm,jpenix,zhou@google.com

Abstract

Static analysis tools for software defect detection are becoming widely used in practice. However, there is little public information regarding the experimental evaluation of the accuracy and value of the warnings these tools report. In this paper, we discuss the warnings found by FindBugs, a static analysis tool that finds defects in Java programs. We discuss the kinds of warnings generated and the classification of warnings into false positives, trivial bugs and serious bugs. We also provide some insight into why static analysis tools often detect true but trivial bugs, and some information about defect warnings across the development lifetime of software release. We report data on the defect warnings in Sun's Java 6 JRE, in Sun's GlassFish JEE server, and in portions of Google's Java codebase. Finally, we report on some experiences from incorporating static analysis into the software development process at Google.

Categories and Subject Descriptors F.3.2 [*Semantics of Programming Languages*]: Program analysis; D.2.4 [*Software/Program Verification*]: Reliability

General Terms Experimentation, Reliability, Security

Keywords FindBugs, static analysis, bugs, software defects, bug patterns, false positives, Java, software quality

1. Introduction

Static analysis for software defect detection has become a popular topic, and there are a number of commercial, open source and research tools that perform this analysis. Unfortunately, there is little public information about the experimental evaluation of these tools with regards to the accuracy and unforseen consequences of the warnings they report. Commercial tools are very expensive and generally come with license agreements that forbid the publication of any

Wagner et al [12] regards a defect warning as a false positive unless the developer believes that the defect could result in significant misbehavior of the application; however, this classification was not clear in the paper and only confirmed through personal communication. Clearly, clarification on this issue is necessary when discussing false positives from static analysis.

Since software rarely comes with specifications that can be interpreted by automatic tools, defect detection tools don't try to find places where software violates its specification. Instead, most defect detection tools find inconsistent [6] or deviant [5] code: code that is logically inconsistent, nonsensical, or unusual. But inconsistent or nonsensical code, while undesirable, may not actually result in the application behaving in a way that substantially deviates from its intended functionality. We discuss this situation and give examples of such defects in Section 3. This issue is particularly exacerbated in a memory-safe language such as Java. In a memory-unsafe language such as C, there is no such thing as a harmless (but feasible) null pointer dereference or buffer overflow. But in Java, dereferencing a null pointer and throwing a null pointer exception may be the most appropriate response when a null value is unexpectedly passed as a method parameter.

There are many contexts for using static analysis for defect detection. One context is when performing code review of a newly written module. In such cases, developers will likely be interested in reviewing any warnings of questionable code, and would likely want to correct confusing or misleading code, even if the confusing or misleading code doesn't result in misbehavior. Another context is when you wish to look for defects in a large existing code base. In this context, the threshold for changing code in response to a warning is higher, since the issue has been in production without causing serious apparent problems, and you would need to get a developer to switch from other tasks and gain familiarity with the code in question before they could make a change they would be

"One reason that static analysis reports true but trivial issues is that these tools **don't know what the code is supposed to do**. Thus, it can't check that the code correctly implements what it is supposed to do."

50% Security problems are due to design flaws

Flaws of omission

Occurs due to ignorance of a security requirement or potential threat

Ex - store a password in a file without encryption.

Flaws of commission

Design decision which can lead to undesirable consequences

Ex – client side authentication

Flaws of realization

The design is correct, but implementation suffers from coding mistakes

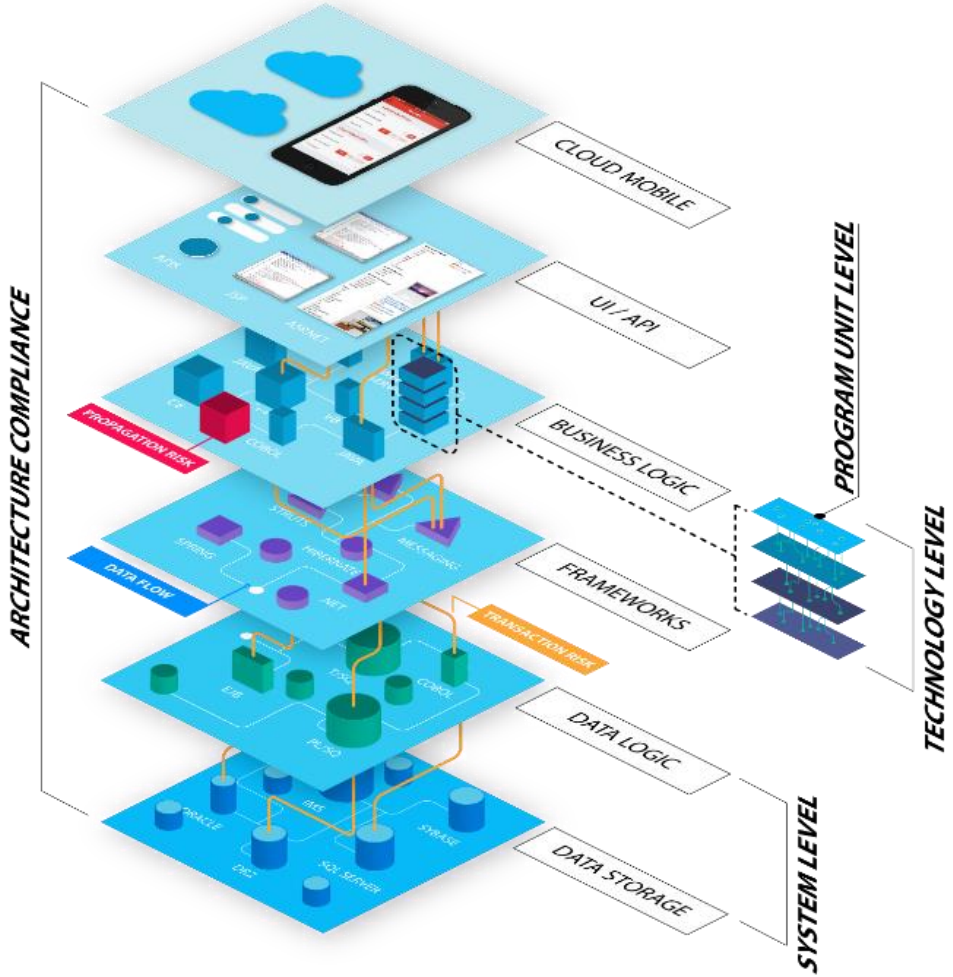
Ex – input sanitization

IEEE Advancing Technology for Humanity

"Architectural flaws are results of inappropriate design choices in early stages of software development, incorrect implementation of security patterns, or degradation of security architecture over time."

A new technology applied to app security

Full Context Analysis



For over 20 years CAST has been developing Full Context Analysis the most advanced software analysis technology in the industry

It has now been tuned to solve the biggest application security and data integrity problems

CAST can see all the business transactions end-to-end – a unique capability that enables:

- Blueprints that inform security designers
- Architecture-based prioritization of findings
- Establishing specific architectural constructs
- Consistent, reliable measurement to standards

Securing Your Applications in 4 Steps

1 Design Security In

2 Secure Architecture Governance

3 Contextual Software Analysis

Security Tool Code Scans

4 Measure Security Risk

Address the blind spots in organizations security strategies

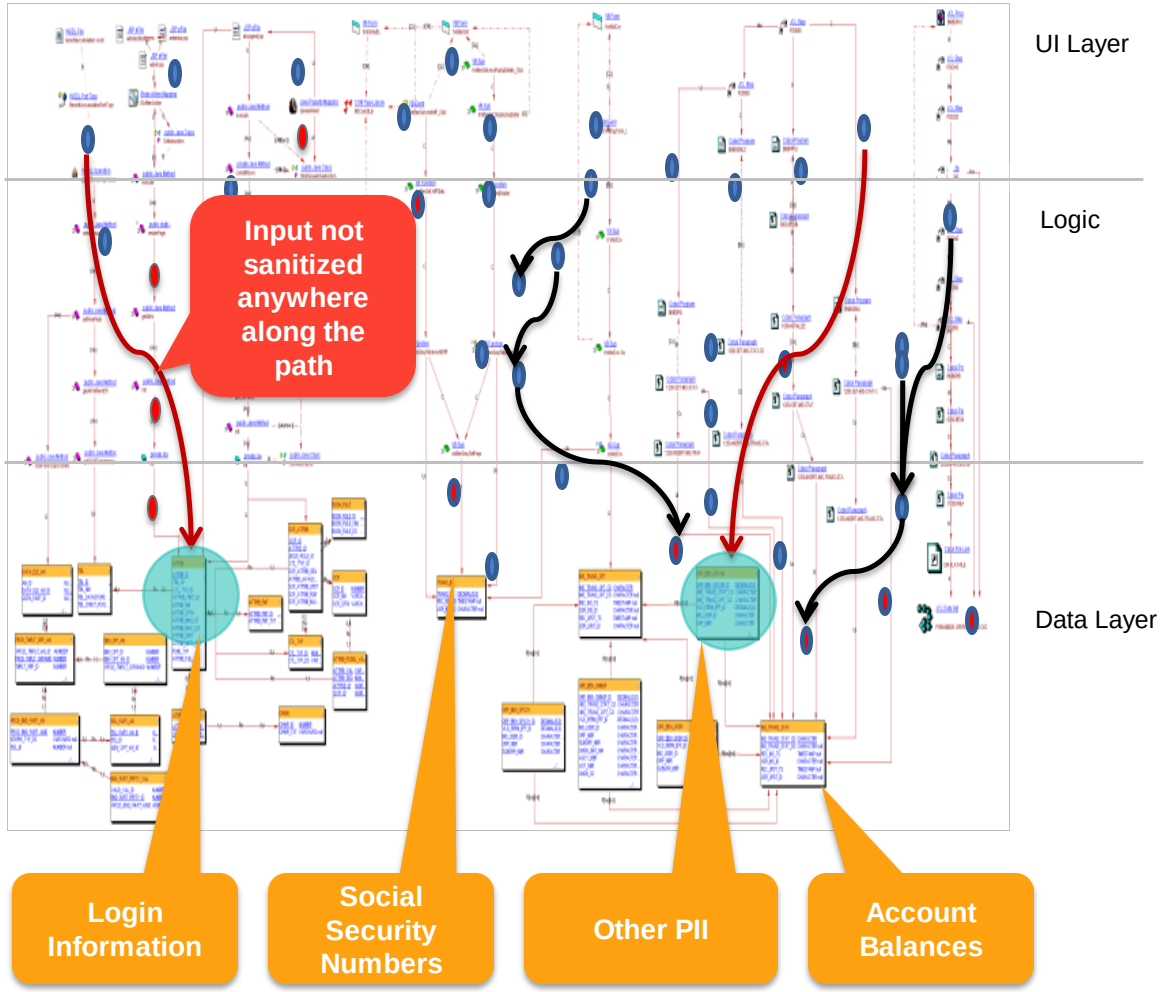
1 Design security in

- Create blueprints for proactive threat analysis – to see “as is” architecture and automated inspection to understand what connects to sensitive data structures
- Identify data call pathways that are safe and which are intrinsically vulnerable to attack
- Understand what controls are needed to prevent common attacks
- Establish new architectural constructs to keep the most sensitive data secure

“Relying on developers for secure code is only superficial application security.”

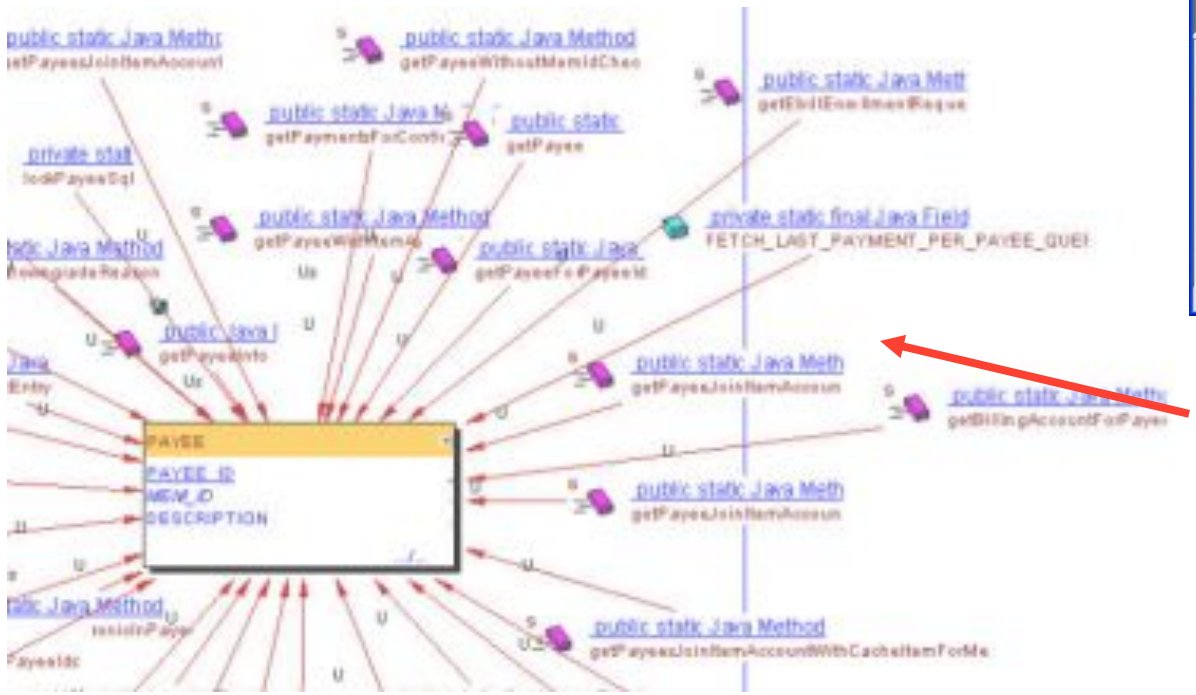
- CISO of large insurance company

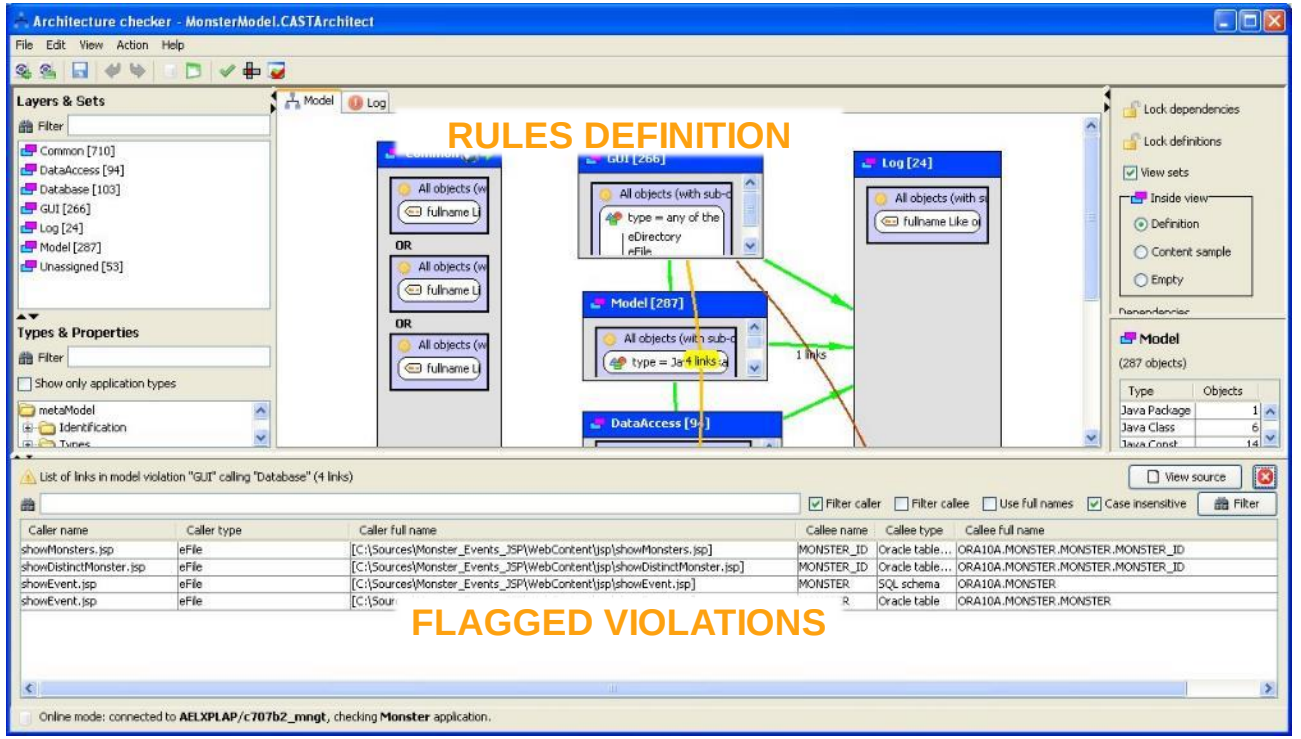
Data Call Flow Diagram: Red ovals are data objects that perform data calls but need investigation. Red lines are paths that are vulnerable



2 Secure architecture governance

- Establish architectural rules specific to your environment
- Measure and enforce development adherence to architectural rules
- Find violations immediately, or during regular in-cycle scans by developers





Avoid this!

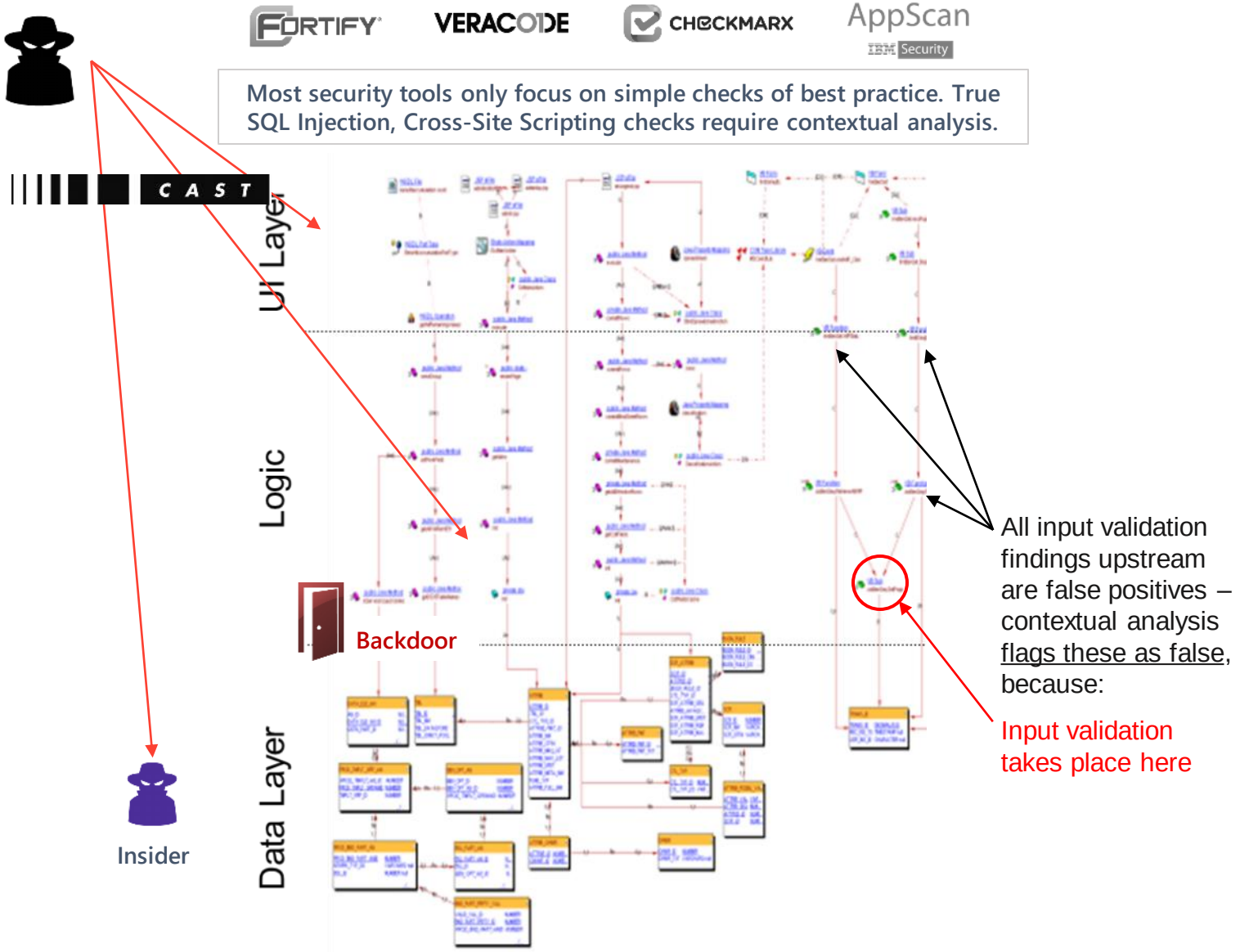
"Architectural analysis is the most important capability missing in almost all IT environments."

- Gary McGraw, CTO, Cigital

3 Contextual software analysis

FORTIFY VERACODE CHECKMARX AppScan IBM Security

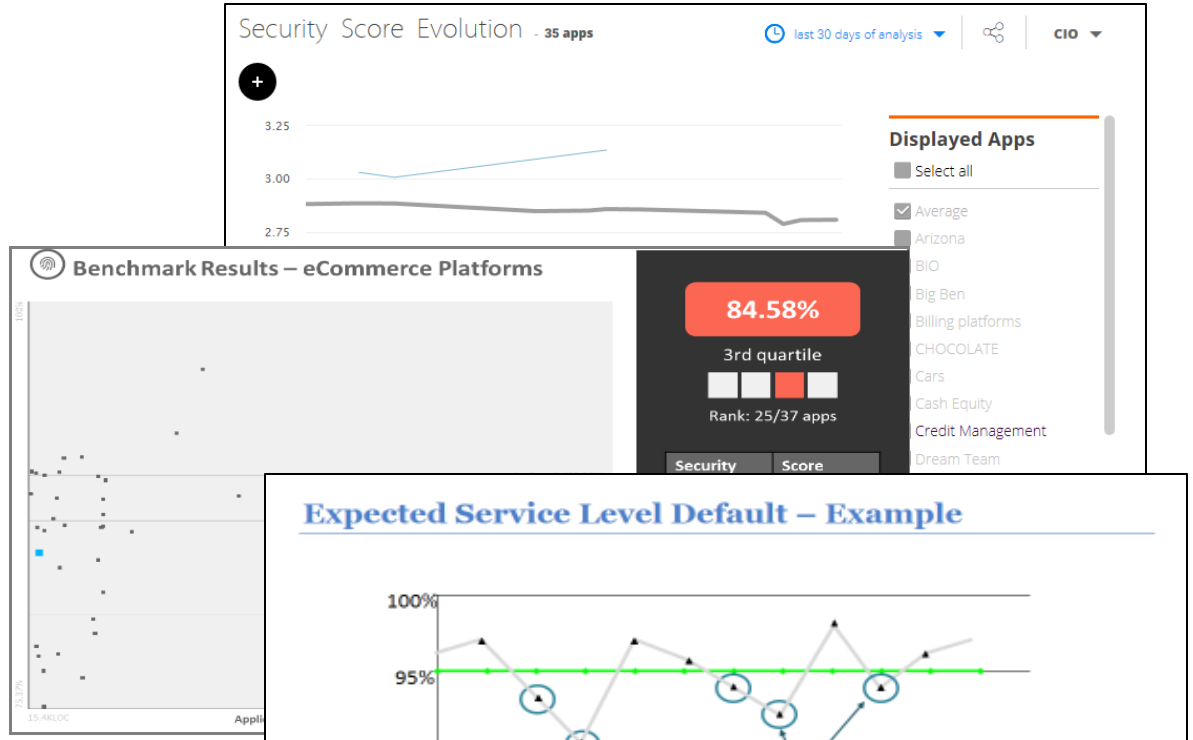
Most security tools only focus on simple checks of best practice. True SQL Injection, Cross-Site Scripting checks require contextual analysis.

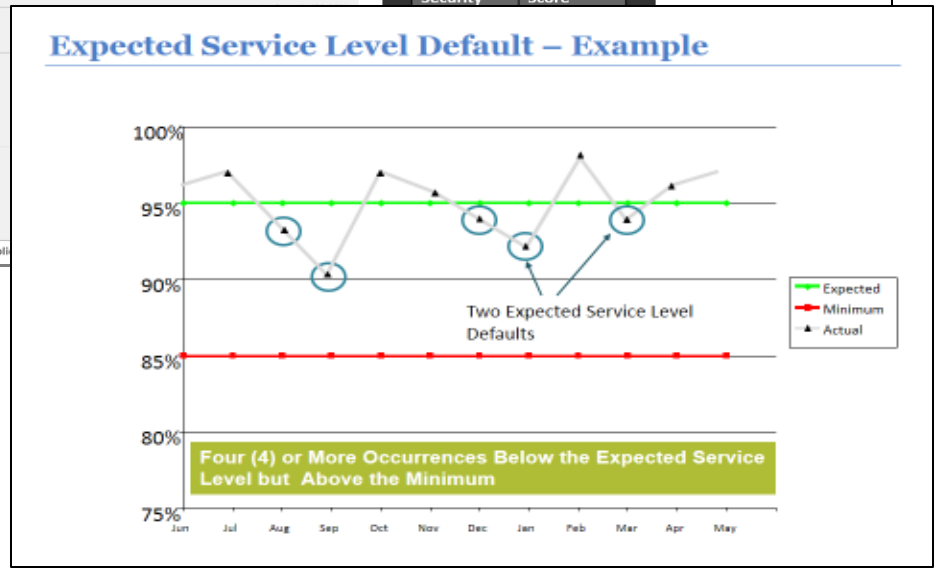


- Contextual Software Analysis finds flaws that traditional application security tools can't catch: Forbidden access to data, lack of input validation, backdoors and insider threats.
- Current security analysis tools review code at the unit level to ensure programming best practices are followed. Without contextual analysis current tools:
  - Miss important problems
  - Provide way too many findings that are irrelevant

4 Measure Security Risk

- CAST provides a calibrated quality model that scores application security in a reliable, consistent way and delivers insight to management that enables:
  - Measure and trend level of software security
  - Provide benchmarks to industry
  - Sourcing governance
  - Estimate the security debt of critical applications
- Compliance to Standards





OWASP Open Web Application Security Project

CWE

PCI DSS COMPLIANT

DISA

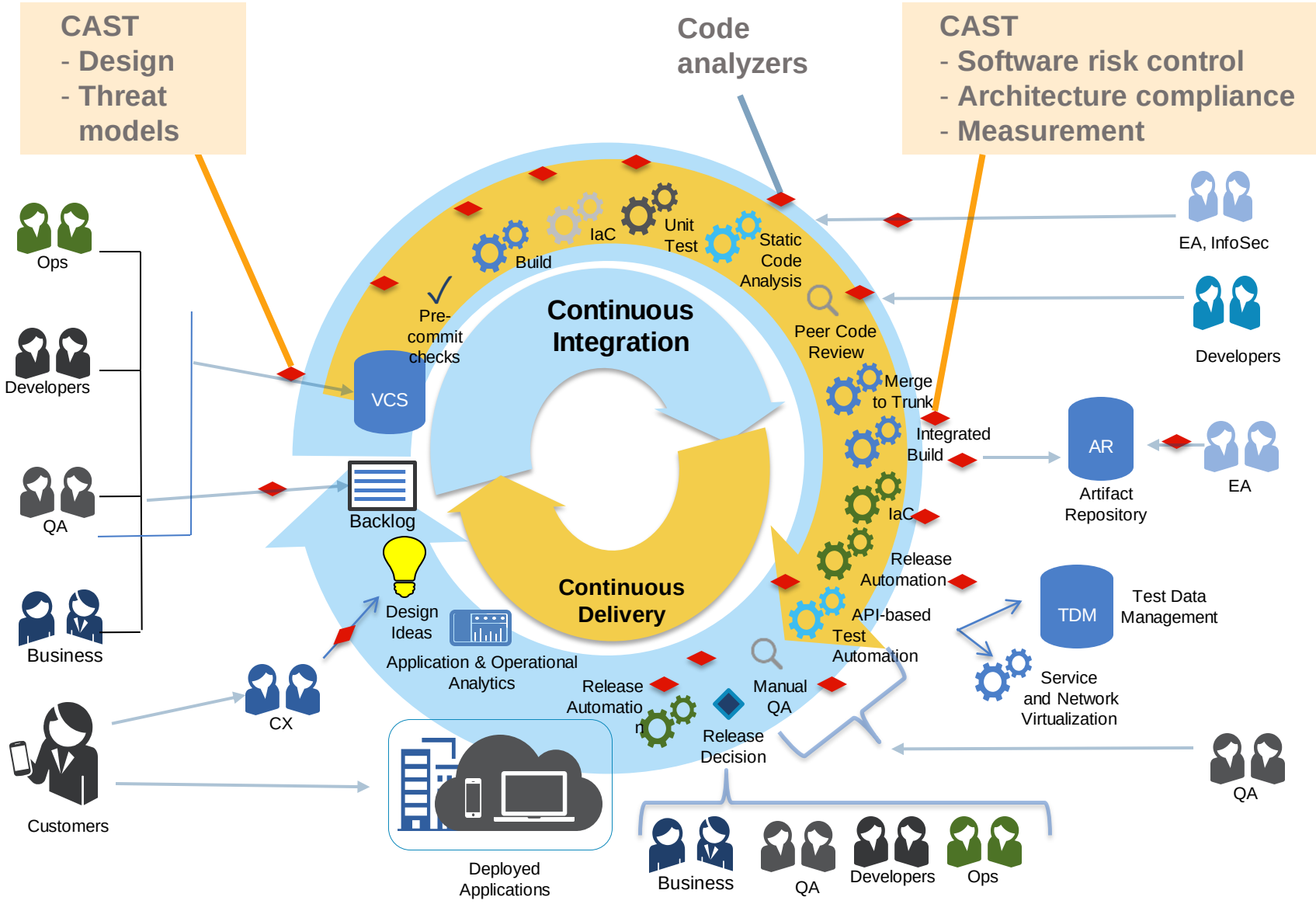
CISQ

Software Engineering Institute Carnegie Mellon

OMG OBJECT MANAGEMENT GROUP

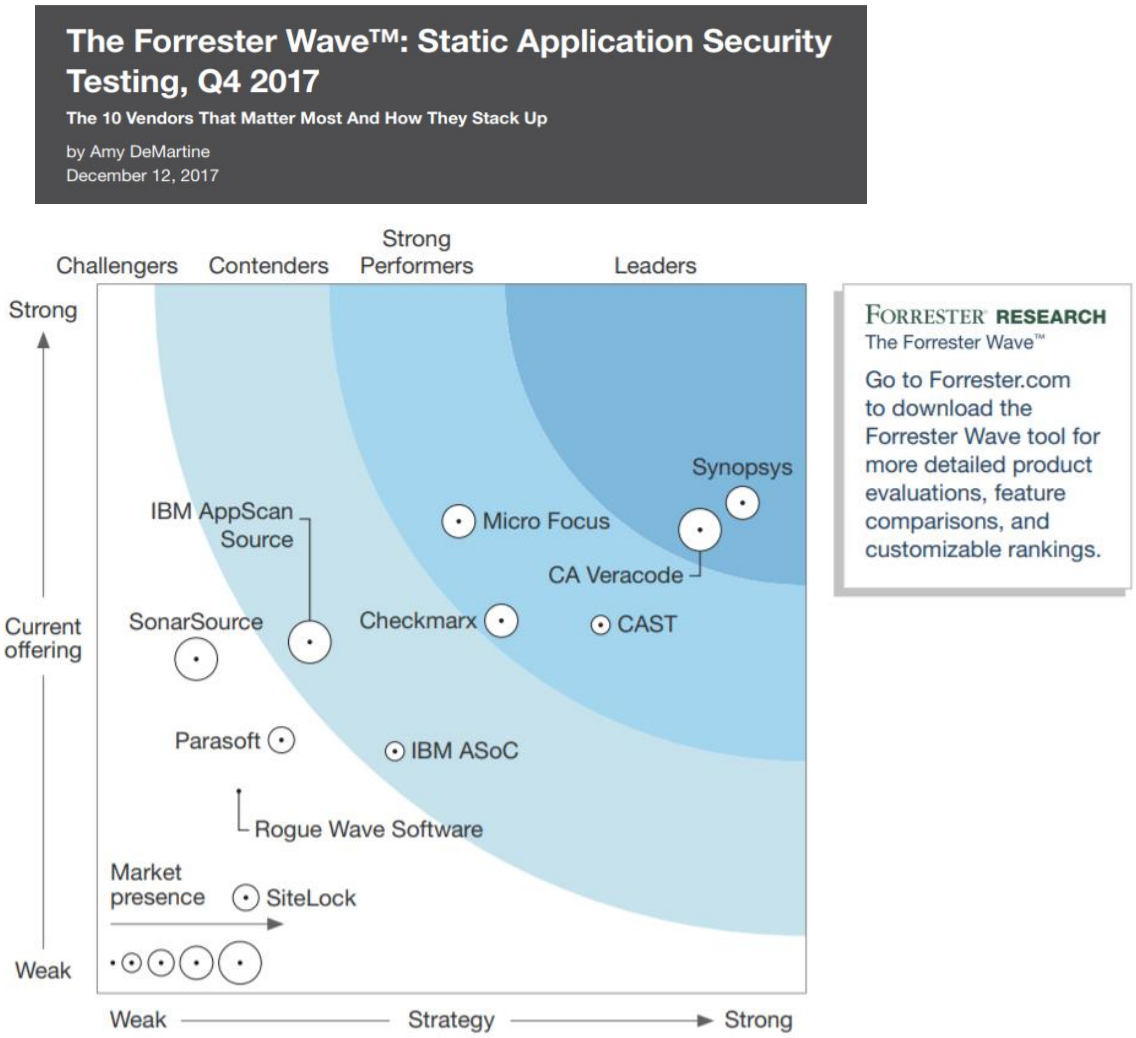
Where CAST fits in the CI/CD SDLC

- CAST AIP fits into the automated CI/CD loop
- Typically CAST runs are not daily, but weekly or bi-weekly
- Main areas of focus:
  1. Design based on understanding “as is” architecture
  2. Contextual analysis and measurement
  3. Continuous monitoring of potential security breaches



Source: Forrester Research & CAST

Industry Recognition



Key Recognitions –

CAST Named A Strong Performer

Received Top Score For "Accuracy" Of Findings

And Perfect Score In Source Code Language Support

"The architectural assessment of design consequences (on software performance, stability, adaptability, maintainability, and security vulnerabilities) is an area in which CAST excels and successfully differentiates from static analyzers." - **Melinda Ballou, IDC**

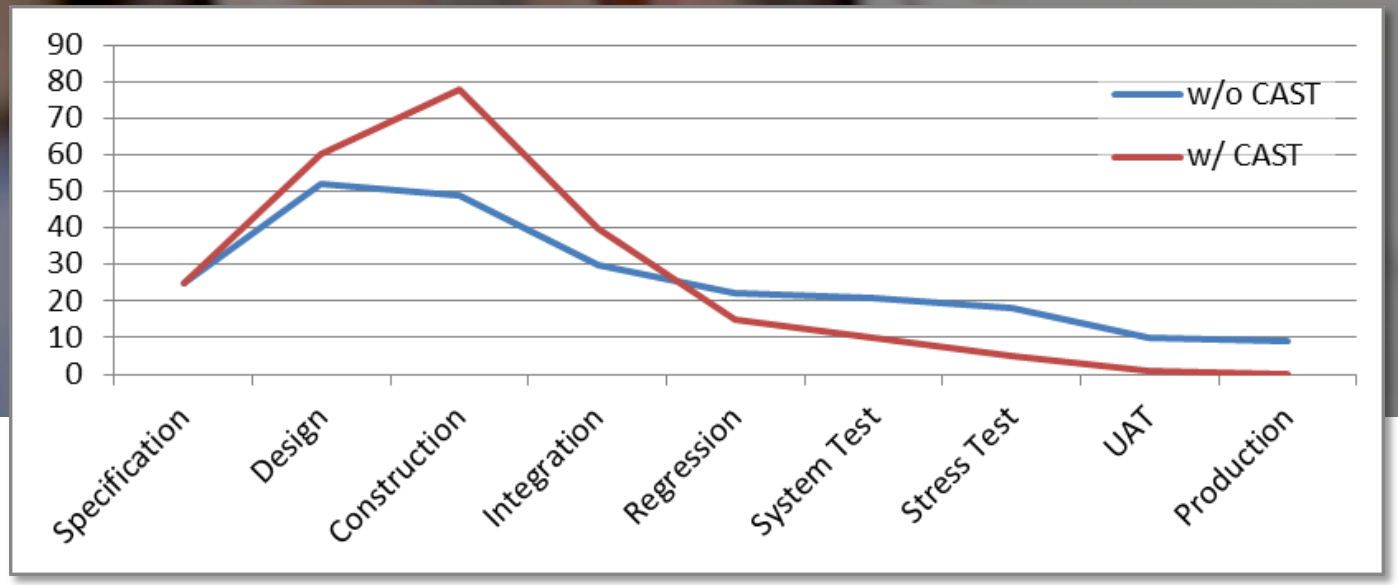
Customer References

*"[CAST's] holistic system approach, looking at the architecture, transactions, control, and data flow across multiple technologies, may be **very beneficial**, with numerous engineering studies showing that bad software engineering practices in the ways components are interrelated and interact…account for only 10% of total defects, but can lead to 90% of production issues."* - **Ovum**

"Information Assurance is more than Security. We found that the foundation of secure software is quality software. Software Assurance is 5 parts Structural Quality with 2 parts Software Security."
John Keane, SCQC Director, U.S. Military Health System

Securing medical records & patient data

Healthcare provider saved $1.4 M on Post-Production Issues in first year



- Electronic Medical Records (EMR) system introduced several to manage +20 million patients worldwide
- Implemented CAST and found **critical architectural flaws** that evaded traditional testing tools (Sonar + Fortify)
- Discovered true source of security flaws while eliminating false positives

12% decrease in security flaws

35% decrease in false positives

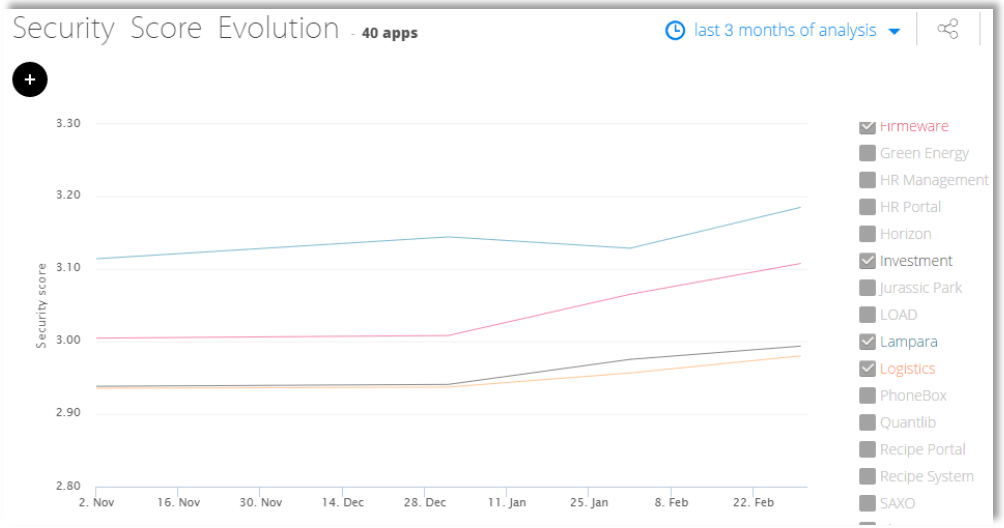
Shift flaw detection left

"If you have a quality problem, then you have a security problem."

Software Engineering Institute
Carnegie Mellon

CAST found the critical security flaws that others missed

Health Insurer Finds Difficult to Detect Security Defects and Enables Continuous Improvement



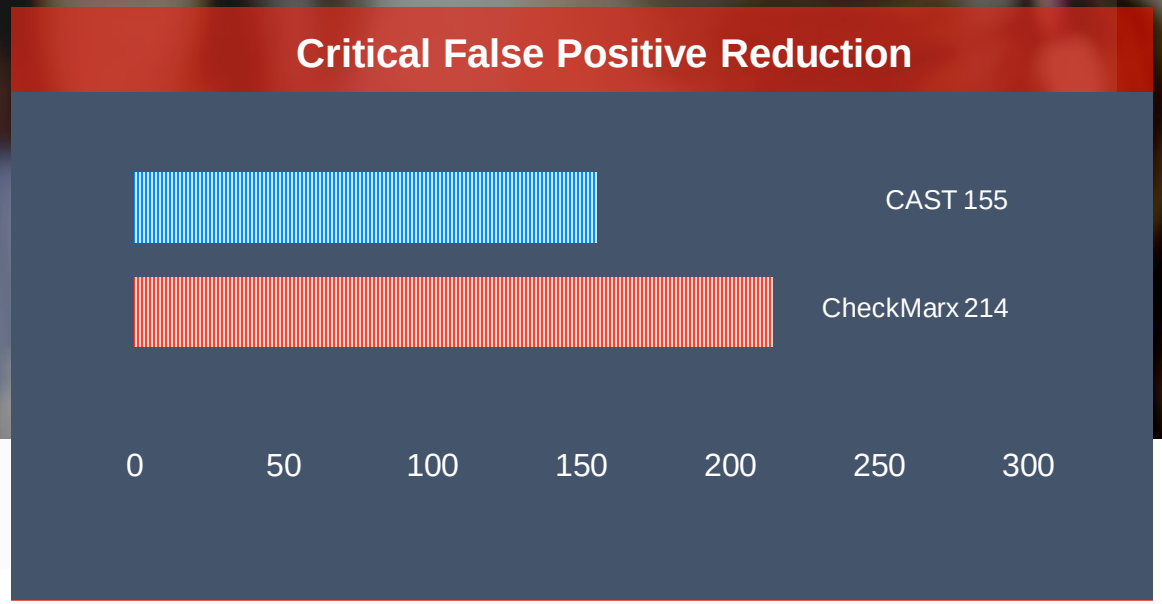
Measure of security across the portfolio to identify trends and take early action before production

CAST found violations of these critical CWE rules	
CWE-79	Avoid cross-site scripting DOM vulnerabilities
CWE-73	Avoid file path manipulation vulnerabilities
CWE-89	Avoid SQL injection vulnerabilities
CWE-117	Avoid Log forging vulnerabilities

Several CWE Top 25 rules that were missed by other tools

CAST improved result and reduced false positives

Deployed CAST as the firmwide static security analyzer in complement to dynamic security testing tool



- World's 2nd largest payment protection services provider with workforce of 10,000 in 36 countries
- Implemented CAST as part of automated testing process and reduced false critical flaws found by CheckMarx
- Discovered true source of security flaws while eliminating false positives

CAST vs CheckMarx

35% increase in security flaws found

27% decrease in false critical violation

CAST found critical security flaws

Global Financial Services Secured Their Internet Banking Business By Addressing Injection Flaws Previous Unknown

- Global financial services provider with 8500+ branches and 147000+ employees across 17 countries
- Integrated CAST as part of the firm's security, quality, and productivity programs
- Incorporated CWE security rules to meet multi-national regulations and compliance

Productivity analysis on over **1600** mixed technology applications

Quality and security analysis on over **200** Cobol and Java applications

Expand technology coverage to **ABAP, J2EE, Mainframe, SQL, VB.NET**