Reducing Software Vulnerabilities
The “Vital Few” Product and Process Metrics

NDIA
20th Annual Systems Engineering Conference
Girish Seshagiri
19735
Topics

• Why We Are Here
• Ubiquitous Software Defects, “Patch and Pray”, What is at Stake
• Unique Point in Time
• State of the Practice – Software Development
• Managing Secure Software Development
• “Vital Few” Metrics
• Case Study
Main Points

- **Defective software** is insecure
  - “If you have a quality problem, you have a security problem”

- **Cannot rely on testing alone** to find and remove software defects
  - Common misconception – “if it passes test, it must be OK”
  - Root cause of “Deliver now, Fix later” culture, technical debt, increase in total ownership cost in many agile projects

- Move from reactive to proactive – **threat detection to threat prevention**

- **Reducing vulnerabilities** - number one goal for every agile software team
  - Manage software by managing software quality

- **Need transformation** – organization, team, individual
  - Combine high maturity CMMI Maturity Level 5 with agile team process to create secure software by self-managed teams and empowered developers with certifications
Why We Are Here

• Increasing number of cyber attacks against critical infrastructure
• Current cybersecurity measures are reactive
• 90% of attacks are successful by exploiting defects in software
• Software developers not trained to deliver software with fewer vulnerabilities
• 1.5 million cybersecurity jobs currently unfilled
Why We Are Here
Personal Identity Breaches

<table>
<thead>
<tr>
<th>Breaches</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total breaches</td>
<td>1,523</td>
<td>1,211</td>
<td>1,209</td>
</tr>
<tr>
<td>Breaches with more than 10 million identities exposed</td>
<td>11</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>Total identities exposed</td>
<td>1.2B</td>
<td>564M</td>
<td>1.1B</td>
</tr>
<tr>
<td>Average identities exposed per breach</td>
<td>805K</td>
<td>466K</td>
<td>927K</td>
</tr>
</tbody>
</table>

In the last 8 years more than 7.1 billion identities have been exposed in data breaches.
Why We Are Here

Browser Vulnerabilities

![Browser Vulnerabilities Chart]

Source: Symantec Internet Security Threat Report April 2017
### Why We Are Here

**Websites with Vulnerabilities**

<table>
<thead>
<tr>
<th>Year</th>
<th>-1% pts</th>
<th>+2% pts</th>
<th>-2% pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>76%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td></td>
<td>78%</td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td></td>
<td></td>
<td>76%</td>
</tr>
</tbody>
</table>
Patch and Pray - 1

Microsoft Patch Tuesday

On Tuesday, September 12, Microsoft released fixes for more than 80 security issues in multiple products, including Windows, Office, Microsoft .NET Framework, Flash, Internet Explorer, and Edge.
Patch and Pray - 2

Apache Struts Vulnerability Exploited in Equifax Breach

Equifax has acknowledged that the massive breach that exposed personal information of as many as 143 million people was due to a failure to apply a patch for a vulnerability in Apache Struts. A patch for the flaw was released on March 6, 2017. The Equifax breach occurred in "mid-May" 2017.
Adobe Security Updates

Adobe has released updates to address security issues in Flash Player, ColdFusion, and RoboHelp for Windows. The Flash updates, available for Windows, Mac, Linux, and Chrome OS, address two critical memory corruption flaws. The ColdFusion update includes fixes for four flaws, and the RoboHelp update fixes two flaws.
Patches dominate the Top of the News this week.

Bill Murray speaks for many when he writes: "The cost to tolerate or remediate a design, recording, or coding error goes up exponentially with the time to its discovery. **There is something fundamentally wrong with an industry in which the toleration, indeed the institutionalization, of late discovery and remediation of error is as it is in ours.**"

And John Pescatore offers a path toward fixing that fundamental flaw in the software industry, describing a future where larger buyers of software (government, for example, along with the Business Roundtable) set a much higher bar for application security testing at multiple stages of the development life cycle, both for custom software they develop and for every package they buy, with **substantial contractual penalties for vendors who fail.**
What is at Stake - 1

Some US States Are Going Back to Paper Ballots

In the wake of rising concerns about the security of electronic voting systems, several US states are returning to the use of paper ballots for their elections. Georgia will pilot a paper-ballot system in elections this fall.

FDA Approves Pacemaker Patch, Announces Recall of Abbott/St. Jude Medical Devices

The US Food and Drug Administration (FDA) has announced a recall of more than 450,000 pacemakers because they require a firmware update to address several security issues. The recall applies to several models of pacemakers manufactured by Abbott, formerly known as St. Jude Medical. Patients must visit their doctor's office where the update can be installed while the device is in backup mode. The flaws could be exploited to gain unauthorized access to vulnerable devices and issue commands to modify the pacemaker’s settings and functionality.
What is at Stake - 2

Car Safety Vulnerability Lies in the Way CAN Handles Error Messages

A vulnerability in the Controller Area Network (CAN) that exists in most new automobiles could be exploited to shut down components of the car, including safety systems. Any component connected to the car's CAN bus could be affected. The issue is not one that can simply be patched because it lies in the CAN bus messaging protocol standard. Components that send too many error messages are disconnected from the CAN, so if attackers can spoof error messages to appear to be coming from a targeted component, that component could be shut off from the CAN.
What is at Stake -3

National Infrastructure Advisory Council Report - A Pre 9-11 Moment

"There is a narrow and fleeting window of opportunity before a watershed, 9/11-level cyber attack, [for the nation] to organize effectively and take bold action," said the US National Infrastructure Advisory Council report. The report lists 11 recommendations, including "establish separate, secure communications networks specifically designated for the most critical cyber networks; ... identify best-in-class scanning tools and assessment practices; ... [and] establish clear protocols to rapidly declassify cyber threat information."
Unique Point in Time – 1

• Software is the most important technology contributing to global high standards of living

• Cyber-attacks are increasing; 90% of attacks result from exploiting defects in software

• The cybersecurity spend estimated to grow to $170 billion by 2020

• 50 billion devices coming online during the next 5 years (IOT)

• Government is dependent on legacy code which is insecure
Unique Point in Time – 2

- GAO has added the Management of Information Technology (IT) Acquisitions and Operations to high risk list
- Many agencies are convinced that agile/Scrum is the silver bullet solution
- Majority of the government’s IT spend is for operations & maintenance; majority of the maintenance spend is for corrective maintenance
- There is no evidence that software project management capability has improved during the past 50 years
Software Engineering’s Persistent Problems – 1

• Exponential rise in cybersecurity vulnerabilities due to defective software
• Unacceptable cost, schedule, and quality performance of Enterprise Resource Planning (ERP) and legacy systems modernization projects
• The number one cost driver in software projects – cost of finding and fixing bugs (i.e. scrap and rework)
• Arbitrary and unrealistic schedules leading to a culture of “deliver now, fix later”
Software Engineering’s Persistent Problems – 2

• Inability to scale software engineering methods even for medium size systems
• Lack of understanding of the impact of variation in individual productivity
• Absence of work place democracy and joy in work
Managing for Secure Software Development – 1

• Staff the projects with team members who are formally trained in estimating, planning and tracking, and measuring and managing quality

• Start each project right with a jelled team capable of making disciplined commitments that the team knows it can meet

• Ensure team members are trained to conduct personal and peer reviews of all design and code artifacts in order to put the highest-quality code components into test, striving for 90 – 100% defect-free components with zero cybersecurity vulnerabilities
Managing for Secure Software Development – 2

• Ensure the review checklists incorporate specific checks for the types of vulnerabilities identified within “The OWASP Top Ten and CWE/SANS Top 25 Most Dangerous Programming Errors”

• Support the teams in collecting, analyzing, and reporting product and process data—size, effort, schedule, and quality

• Require teams to report status weekly
  • Precise and accurate data on planned versus actual size, effort, schedule, earned value and defects injected and removed
CISQ Quality Characteristic Measures

What Attributes Should Be Measured?

**CISQ Quality Characteristic Measures**

- **Security**: Ability to prevent unauthorized intrusions and data theft
- **Reliability**: Ability to avoid outages and to recover operations quickly
- **Performance Efficiency**: Ability to avoid response degradation, resource overuse
- **Maintainability**: Ability to understand and modify software quickly

**Business outcomes**

- Damages, customer confidence
- Damages, lost revenue, customer loss
- Lost customers, operating cost
- Cost of ownership, time to market

---

http://it-cisq.org/standards/quality_characteristic_measures

Copyright © 2014, Consortium for IT Software Quality, Inc. All Rights Reserved
CISQ Quality Characteristic Violations

CISQ Measures Violations

<table>
<thead>
<tr>
<th>CISQ Quality Characteristic Measures</th>
<th>Example architectural and coding violations composing the measures</th>
</tr>
</thead>
</table>
| Security 22 violations (Top 25 CWEs)  | • SQL injection  
• Cross-site scripting  
• Buffer overflow |
| Reliability 29 violations            | • Empty exception block  
• Unreleased resources  
• Circular dependency |
| Performance Efficiency 15 violations | • Expensive loop operation  
• Un-indexed data access  
• Unreleased memory |
| Maintainability 20 violations        | • Excessive coupling  
• Dead code  
• Hard-coded literals |
Principles for Successful Metrics Program

• Metrics derived from business and team goals
  • What do we need to know to make sure we are achieving the goals?
• Software teams must be focused on the “vital few” metrics and not the “trivial many”
• Size metric is essential to normalize data
• Operational definitions, precision, and accuracy of data are of utmost importance
• Unless data is collected and used by trained team members, it won’t be useful
• Never use defect data for performance appraisal
## Operational Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>% schedule deviation</td>
<td>Actual schedule divided by planned schedule (by months) times 100 minus 100</td>
</tr>
<tr>
<td>Acceptance test defect</td>
<td>Defect found during the acceptance test of the product. This defect may be identified by the customer as well as the ISHPI team. Acceptance test period starts after the completion of system test of the product and ends when the code is deployed to production.</td>
</tr>
</tbody>
</table>
| Cost of Quality (COQ)       | COQ = \( \frac{\text{effort in appraisal tasks} + \text{effort in prevention tasks} + \text{effort in failure tasks}}{\text{Total effort towards the commitment (including project management)}} \times 100 \)  
  • Appraisal tasks: Personal reviews, peer reviews, and first-time test execution  
  • Prevention tasks: Training, postmortems, and causal analysis  
  • Failure tasks: Analyzing and fixing defects found in reviews and testing |
| Defect density              | The number of defects identified in a product divided by the size of the product component, expressed in standard measurement terms for that product (e.g., 1000 lines of new and changed code) |
| Earned value                | Percentage of effort for tasks completed divided by total effort of the tasks                                                                                                                              |
| First Time Right (in acceptance test) | Deployed software changes, accepted by the customer the first time, with no further rework                                                                                                               |
| Peer review yield           | The percentage of the total defects that are found and removed in peer review                                                                                                                              |
| Personal review yield       | The percentage of the total defects that are found and removed in personal review                                                                                                                           |
## Recommended Team Base Measures

<table>
<thead>
<tr>
<th>Measure</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Lines of code</td>
</tr>
<tr>
<td>Task time</td>
<td>Minutes</td>
</tr>
<tr>
<td>Defects</td>
<td>Number</td>
</tr>
<tr>
<td>Task Completion</td>
<td>Yes/No</td>
</tr>
</tbody>
</table>
"Vital Few" Metrics to Reduce Vulnerabilities

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>Leading Indicator?</th>
<th>Lagging Indicator?</th>
<th>Organization</th>
<th>Project</th>
<th>Increment</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planned vs. actual size</td>
<td>Y</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Planned vs. actual effort</td>
<td>Y</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Planned vs. actual schedule</td>
<td>Y</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Planned vs. actual earned value</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Planned vs. actual defect profile</td>
<td>Y</td>
<td>Y</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Total Cost of Quality (COQ)</td>
<td>Y</td>
<td>Y</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>First Time Right in acceptance test</td>
<td>Y</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Acceptance test defect density in delivered code</td>
<td>Y</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Decision Process to Reduce Vulnerabilities

• Size, effort, schedule deviation
  • Review process adherence and test and rework due to poor quality

• Earned value deviation
  • Review hours worked against plan, task dependencies
  • “What if” scenario using process performance model

• Planned versus actual defect profile
  • Review personal and peer review yields – design, code
  • Estimate defect counts and defect densities for downstream steps using quality projection model
Project Management

Metrics SW

Agile Milestone Tracking

Senior Management View

Agile Project Team View

Release Burndown

Release & Sprint Velocity

Issues / Risks

Daily Burndown
Organization Capabilities – 1

Schedule Deviation – Development Phases

Schedule Deviation (Avg=2.8720, UCL=33.2279, LCL=-27.4838)

Before TSP Introduction

After TSP Introduction

After High Velocity Development Introduction

Project/Increment Start Date

04/24/98  11/23/98  06/21/99  10/03/99  12/06/99  06/04/00  03/19/01  05/22/01  06/07/02  05/12/03  04/04/05  10/19/05  12/11/05  10/01/09  10/11/10  09/03/12  04/01/13  08/23/16

© Ishpi Information Technologies, Inc. (DBA ISHPI). All Rights Reserved
Organization Capabilities – 2

Effort Deviation – Development Phases

Effort Deviation (Avg=10.9516, UCL=56.3014, LCL=-34.3982)

Before TSP Introduction

After TSP Introduction

After High Velocity Development Introduction

Project/Increment Start Date

04/01/98  06/26/98  11/21/98  06/21/99  12/09/99  06/04/00  06/11/00  04/23/01  07/03/01  06/07/02  01/20/03  10/06/03  04/18/05  05/08/06  03/04/07  07/19/10  10/01/11  09/03/12  04/29/13  07/01/16
Organization Capabilities – 3

Acceptance Test Defect Density – Maintenance/Enhancement

AT Defects Per KLOC (Avg=0.2606, UCL=1.1990, LCL=-0.6779)

After High Velocity Development Introduction

AT Defects Per KLOC

Project Start Date

10/07/02 05/12/03 04/04/05 12/19/05 05/15/06 03/04/07 07/19/10 10/01/11 05/14/12 09/01/12 10/01/12 04/01/13 07/01/13 04/29/14 06/29/15
Organization Capabilities – 4

Defect Injection and Removal Profile

**Actual Defect Injection and Removal Profile**

- **Development Step**: CD, CDPR, CDI, CDIW, CC, CCPR, CCI, CCIW, CT, CTW, INT, ITST, ITSTW, STE, STEW, ATE, ATEW, PD
- **Defects/KLOC**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- **Defects Injected**
- **Defects Removed**
Organization Capabilities – 5

Total Cost of Quality

- **Appraisal**: Personal reviews, Peer reviews, All levels of first-time test execution
- **Failure**: Rework from reviews, Rework from tests, Regression testing

**TOTAL COQ (Avg=0.3280, UCL=0.5800, LCL=0.0760)**
## Organization Capabilities Summary

### Industry, Best in Class, Organization

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>Industry Average</th>
<th>Best in Class Average</th>
<th>Org. Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule deviation</td>
<td>&gt; 50%</td>
<td>6%</td>
<td>&lt; 3%</td>
</tr>
<tr>
<td>Acceptance test defects in delivered product (100,000 Source Lines of Code)</td>
<td>&gt; 100</td>
<td>55</td>
<td>&lt; 27</td>
</tr>
<tr>
<td>% of design and code inspected</td>
<td>&lt; 100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Customer’s time to accept 100,000 LOC product</td>
<td>&gt; 4 months</td>
<td>3.4 weeks</td>
<td>&lt; 5 weeks</td>
</tr>
<tr>
<td>% of defects removed prior to system test</td>
<td>&lt; 60%</td>
<td>70 – 95%</td>
<td>&gt; 85%</td>
</tr>
<tr>
<td>% of development time in rework fixing system test defects</td>
<td>&gt; 33%</td>
<td>2.7 – 10%</td>
<td>&lt; 10%</td>
</tr>
<tr>
<td>Cost of quality</td>
<td>&gt; 50%</td>
<td>29%</td>
<td>&lt; 33%</td>
</tr>
</tbody>
</table>
Customer Benefits of Optimizing Process and Agile Development

• Frequent incremental deliveries that meet customer business goals
• Dramatically reduces and practically eliminates cyber security incidents attributable to poor quality software code
• Reduces software operations & maintenance (O&M) costs by more than half
• Fewer bugs to fix in production software making available a larger percentage of O&M spend for new features and enhancements
• Enables lifetime warranty
Main Points

• **Defective software** is insecure
  • “If you have a quality problem, you have a security problem”

• **Cannot rely on testing alone** to find and remove software defects
  • Common misconception – “if it passes test, it must be OK”
  • Root cause of “Deliver now, Fix later” culture, technical debt, increase in total ownership cost in many agile projects

• Move from reactive to proactive – **threat detection to threat prevention**

• **Reducing vulnerabilities** - number one goal for every agile software team
  • Manage software by managing software quality

• Need **transformation** – organization, team, individual
  • Combine high maturity CMMI Maturity Level 5 with agile team process to create secure software by self-managed teams and empowered developers with certifications
Takeaways/“ASKs”

• Act with a **sense of urgency** to move from reactive “patch and pray” to proactive secure software development

• Begin **transformation** of organization, teams, and individuals to consistently deliver software which is secure from cyber attacks

• Manage secure software development by **managing software quality**

• Focus on the “**vital few**” metrics and not the “trivial many”

• Support teams in **reducing vulnerabilities** as the number one project goal

• Develop industry/government/academic coalition led by industry to address cybersecurity “**skills gap**” and talent pipeline
References

NIST Publication NISTIR 8151
Dramatically Reducing Software Vulnerabilities
Report to the White House Office of Science and Technology Policy

Barti Perini, Stephen Shook and Girish Seshagiri
Reducing Software Vulnerabilities – The Number One Goal for Every Software Development Organization, Team, and Individual
Contact

Girish Seshagiri

girish.seshagiri@ishpi.net

(703) 426-2790