baked in or bolted on? security focused scrum teams
A Little About Me...

Mathematician

Certified ScrumMaster (CSM)

Computer Scientist

Certified ScrumProfessional (CSP)

Software Manager

Agile Coach & Trainer

The University of North Carolina Central Alumni Association

AGILE 2014 ORLANDO
exercise

think about your current organization:

✓ who is responsible for security now?
✓ how is it accomplished?
✓ what specifics tasks happen and when do they happen?
**CYBER RISK: WHO MIGHT ATTACK? & WHAT ARE THEY AFTER?**

- **The classic cyber criminal (i.e., organized crime) or skilled individual hackers**
  - Guarded data and personally identifiable information, including health records, for monetary gain

- **The advanced persistent threat, directed by nation-states**
  - Highly sensitive information, including PHI, infrastructural, or strategic information, to gain an economic or technological advantage

- **“Hactivists”**
  - Systemic disruption with political, social, or personal motive, often in the form of high profile protest

*Copyright © 2014 Deloitte Development LLC. All rights reserved. Member of Deloitte Touche Tohmatsu Limited*
assurance is the goal

“Assurance is the level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software” -- DOD

can agile software development methods generate assurance?
60% of employees circumvent security features on their mobile devices.

68% of US companies permit employee-owned devices in the workplace.

80% of CEOs surveyed believe good data protection increases brand or marketplace image.

46% of medical ID theft victims say they know the thief.

77% of UK systems developers surveyed use real production data when developing applications.

51% of CEOs surveyed say their company experiences cyber attacks hourly or daily.

12% of consumers care enough about privacy to take action or suffer an inconvenience.

Global IT security spending has increased 11% per year over the past decade.

94% of healthcare organizations had a breach involving one or more records.

Brazil has the lowest privacy risk profile among 30 countries.
costs of data breaches

Figure 2. The average per capita cost of data breach over two years
Measured in US$

<table>
<thead>
<tr>
<th>Country</th>
<th>FY 2011 (n = 199)</th>
<th>FY 2012 (n = 277)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>BZ*</td>
<td>58</td>
<td>102</td>
</tr>
<tr>
<td>IT</td>
<td>124</td>
<td>125</td>
</tr>
<tr>
<td>JP</td>
<td>132</td>
<td>132</td>
</tr>
<tr>
<td>UK</td>
<td>145</td>
<td>133</td>
</tr>
<tr>
<td>AU</td>
<td>159</td>
<td>168</td>
</tr>
<tr>
<td>FR</td>
<td>188</td>
<td>194</td>
</tr>
<tr>
<td>US</td>
<td>199</td>
<td>191</td>
</tr>
<tr>
<td>DE</td>
<td>199</td>
<td>191</td>
</tr>
</tbody>
</table>
attacker’s interest in applications

Software becomes a target at 3% market share
pillars of information security

“Ensures that only authorized users (confidentiality) have access to accurate and complete information (integrity) when required (availability).” ~ISACA 2008 ~
divergent attitudes

Developers believe security team members do not understand code

Security Team members believe development teams are arrogant
Divergent attitudes

- Security Engineers
  - Confidentiality
  - Integrity

- Developers
  - Availability

AGILE
2014
ORLANDO
What other differences keep development & security from collaborating??
SECURITY IN DEVELOPMENT
### Security and Coding Associations

<table>
<thead>
<tr>
<th>QA Reviews</th>
<th>Security Toll Gates</th>
<th>Integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bugs found, logged, prioritized</td>
<td>Security team members are one part of a project review committee or planning board</td>
<td>Security and risk management are integrated into the dev cycle</td>
</tr>
<tr>
<td>Relies on good relationship between dev and QA</td>
<td>Relies on security being prioritized in a top-down manner from senior leadership</td>
<td>Relies on security team members of developing a core security focus on the part of developers</td>
</tr>
</tbody>
</table>
Waterfall toll gate style is critically important to ensure that security is prioritized during the requirements specification phase and carried out at every phase.
Waterfall security is “breadth first”

Build assurance layer-by-layer

Challenges

• Problem space is very large
• Difficult to prioritize issues
• Loss of traceability from threat agents to source code
• Problems always seem “theoretical” until it’s too late

Security Requirements
Validation Spec
Security Architecture Review
Code Review
Application Vulnerability Test
External Application Security Assessments
security development lifecycle

Fig 1: Microsoft Security Development Lifecycle as an iterative process
agile development lifecycle
Building assurance “depth first”

Identify most important security concerns
  • and required security mechanisms for these concerns
In periodic security sprints
  • Develop test methods for these security mechanisms
  • Implement/configure/analyze these security mechanisms
  • Ensure mechanisms are being used everywhere properly
cross functional teams...play an important role

- Ensure that security professionals are an integral part of any information system development activities from the initial definition of information security requirements
- Fosters close cooperation between
  - Team members responsible for the design, development, implementation, operation, maintenance, and disposition
  - Information security professionals advising the senior leadership on appropriate security controls needed to adequately mitigate risk and protect critical missions and business functions.
cross functional teams

- Developers
- Testers
- DBAs
- Vulnerability Testers
- Security Engineers
SECURITY ENGINEER OR SECURITY PROFESSIONAL
security engineering

meets functional requirements within cost and schedule constraints

provides sufficient security control to mitigate risks to an acceptable level

this is a never-ending balancing act!!
How do we bring security thought processes up front?

User Stories

Security Requirements

Definition of Done

Abuser Stories

Acceptance & Refutation Criteria

Definition

Acceptance
Security and Privacy Controls for Federal Information Systems and Organizations

JOINT TASK FORCE
TRANSFORMATION INITIATIVE

http://dx.doi.org/10.8807/NIST.SP.800-63r4
AC-9  PREVIOUS LOGON (ACCESS) NOTIFICATION

Control:  The information system notifies the user, upon successful logon (access) to the system, of the date and time of the last logon (access).

Supplemental Guidance:  This control is applicable to logons to information systems via human user interfaces and logons to systems that occur in other types of architectures (e.g., service-oriented architectures). Related controls: AC-7, PL-4.

Control Enhancements:

(1)  PREVIOUS LOGON NOTIFICATION | UNSUCCESSFUL LOGONS
The information system notifies the user, upon successful logon/access, of the number of unsuccessful logon/access attempts since the last successful logon/access.

(2)  PREVIOUS LOGON NOTIFICATION | SUCCESSFUL / UNSUCCESSFUL LOGONS
The information system notifies the user of the number of [Selection: successful logons/accesses; unsuccessful logon/access attempts; both] during [Assignment: organization-defined time period].

(3)  PREVIOUS LOGON NOTIFICATION | NOTIFICATION OF ACCOUNT CHANGES
The information system notifies the user of changes to [Assignment: organization-defined security-related characteristics/parameters of the user’s account] during [Assignment: organization-defined time period].

(4)  PREVIOUS LOGON NOTIFICATION | ADDITIONAL LOGON INFORMATION
The information system notifies the user, upon successful logon (access), of the following additional information: [Assignment: organization-defined information to be included in addition to the date and time of the last logon (access)].

Supplemental Guidance:  This control enhancement permits organizations to specify additional information to be provided to users upon logon including, for example, the location of last logon. User location is defined as that information which can be determined by information systems, for example, IP addresses from which network logons occurred, device identifiers, or notifications of local logons.

References:  None.

Priority and Baseline Allocation:

| Priority | low | Not Selected | MOD | Not Selected | HIGH | Not Selected |
what is done?

Accepted!!
**Story Done**
All code checked in
All developer tests pass
All acceptance tests pass
Help text is written
Product Owner accepted

**Iteration Done**
Product backup complete
Performance tested
Defects fixed or postponed

**Release Done**
Stress tested
Performance tuned
Security validation passes
Disaster recovery plan tested
what should be “done” from a security perspective?

**Story Done**
- All code checked in
- All developer tests pass
- All acceptance tests pass
- Help text is written
- Product Owner accepted

**What’s missing?**
Abuser Stories can help organizations see their products in the same way attackers do.
some examples against a Login Story...

As a hacker hired by a competitor I want to brute force login so that I can gain access to business data.

As a disgruntled employee, I want to brute force the login so that I can steal business data.
role of the security engineer on an agile team

- Contribute to the team’s “Definition of Done”
- Member of the Delivery Team (not a stakeholder)
- Help prioritize security requirements on the product backlog
- Implement and demonstrate security features
- Express security requirements in terms of business value to Delivery Team
- Add acceptance criteria to security relevant functional features
Collaborate & Communicate!
## The Perfect Security Engineer

<table>
<thead>
<tr>
<th>Network Engineer</th>
<th>Operating Systems SME</th>
<th>Systems Engineer</th>
<th>Software Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Routers</td>
<td>• Linux</td>
<td>• Architecture</td>
<td>• Programming Languages</td>
</tr>
<tr>
<td>• Switches</td>
<td>• Unix</td>
<td>• Requirements</td>
<td>• Development methodologies</td>
</tr>
<tr>
<td>• Firewalls</td>
<td>• Trusted operating systems</td>
<td>• Documentation</td>
<td></td>
</tr>
<tr>
<td>• Intrusion Detection Systems</td>
<td>• Windows</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is this too much to ask for???
The Perfect Security Engineer

- Systems Engineer
- Software Developer
- Protocol Expert
- Applications SME
- Integration & Test Engineer
- Operating System SME
- Network Engineer
- Team Builder
at the very least, where can security professionals plug in?

Backlog Refinement Session

Demos and Reviews

Sprint Planning

Offer feedback during sprint
DEVELOPER
writing secure code

Writing Secure Features, 90%

Writing Security Features, 10%
writing secure code is NOT a feature

- Not a User Story
- Doesn’t go in the Product Backlog
- Can’t Get Prioritized In or Out
- Can’t Decide Not to do Security this Sprint
It's a security-driven mindset!!!
## SANS top 25 software coding errors

<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>93.8</td>
<td>CWE-89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
<tr>
<td>2</td>
<td>83.3</td>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
</tr>
<tr>
<td>3</td>
<td>79.0</td>
<td>CWE-120</td>
<td>Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')</td>
</tr>
<tr>
<td>4</td>
<td>77.7</td>
<td>CWE-79</td>
<td>Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')</td>
</tr>
<tr>
<td>5</td>
<td>76.9</td>
<td>CWE-306</td>
<td>Missing Authentication for Critical Function</td>
</tr>
<tr>
<td>6</td>
<td>76.8</td>
<td>CWE-662</td>
<td>Missing Authorization</td>
</tr>
<tr>
<td>7</td>
<td>75.0</td>
<td>CWE-798</td>
<td>Use of Hard-coded Credentials</td>
</tr>
<tr>
<td>8</td>
<td>75.0</td>
<td>CWE-311</td>
<td>Missing Encryption of Sensitive Data</td>
</tr>
<tr>
<td>9</td>
<td>74.0</td>
<td>CWE-434</td>
<td>Unrestricted Upload of File with Dangerous Type</td>
</tr>
<tr>
<td>10</td>
<td>73.8</td>
<td>CWE-807</td>
<td>Reliance on Untrusted Inputs in a Security Decision</td>
</tr>
<tr>
<td>11</td>
<td>73.1</td>
<td>CWE-250</td>
<td>Execution with Unnecessary Privileges</td>
</tr>
<tr>
<td>12</td>
<td>70.1</td>
<td>CWE-352</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>13</td>
<td>69.3</td>
<td>CWE-22</td>
<td>Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')</td>
</tr>
<tr>
<td>14</td>
<td>68.5</td>
<td>CWE-494</td>
<td>Download of Code Without Integrity Check</td>
</tr>
<tr>
<td>15</td>
<td>67.8</td>
<td>CWE-663</td>
<td>Incorrect Authorization</td>
</tr>
<tr>
<td>16</td>
<td>66.0</td>
<td>CWE-829</td>
<td>Inclusion of Functionality from Untrusted Control Sphere</td>
</tr>
<tr>
<td>17</td>
<td>65.5</td>
<td>CWE-732</td>
<td>Incorrect Permission Assignment for Critical Resource</td>
</tr>
<tr>
<td>18</td>
<td>64.6</td>
<td>CWE-676</td>
<td>Use of Potentially Dangerous Function</td>
</tr>
<tr>
<td>19</td>
<td>64.1</td>
<td>CWE-327</td>
<td>Use of a Broken or Risky Cryptographic Algorithm</td>
</tr>
<tr>
<td>20</td>
<td>62.4</td>
<td>CWE-131</td>
<td>Incorrect Calculation of Buffer Size</td>
</tr>
<tr>
<td>21</td>
<td>61.5</td>
<td>CWE-307</td>
<td>Improper Restriction of Excessive Authentication Attempts</td>
</tr>
<tr>
<td>22</td>
<td>61.1</td>
<td>CWE-601</td>
<td>URL Redirection to Untrusted Site ('Open Redirect')</td>
</tr>
<tr>
<td>23</td>
<td>61.0</td>
<td>CWE-134</td>
<td>Uncontrolled Format String</td>
</tr>
<tr>
<td>24</td>
<td>60.3</td>
<td>CWE-190</td>
<td>Integer Overflow or Wraparound</td>
</tr>
<tr>
<td>25</td>
<td>59.9</td>
<td>CWE-759</td>
<td>Use of a One-Way Hash without a Salt</td>
</tr>
<tr>
<td>A1-Injection</td>
<td>Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2-Broken Authentication and Session Management</td>
<td>Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3-Cross-Site Scripting (XSS)</td>
<td>XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4-Insecure Direct Object References</td>
<td>A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-5 Security Misconfiguration</td>
<td>Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6 - Sensitive Data Exposure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A7 - Missing Function Level Access Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A8 - Cross-Site Request Forgery (CSRF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A9 - Using Components with Known Vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A-10 - Unvalidated Redirects and Forwards</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>M1</td>
</tr>
<tr>
<td>M2</td>
</tr>
<tr>
<td>M3</td>
</tr>
<tr>
<td>M4</td>
</tr>
<tr>
<td>M5</td>
</tr>
<tr>
<td>GP1</td>
</tr>
<tr>
<td>GP2</td>
</tr>
<tr>
<td>GP3</td>
</tr>
<tr>
<td>GP4</td>
</tr>
</tbody>
</table>
"Test-Driven Development (TDD) is a style of development where:

- You maintain an exhaustive suite of Programmer Tests,
- No code goes into production unless it has associated tests,
- You write the tests first,
- The tests determine what code you need to write."
The Test Driven Development Cycle

Write a failing test for new functionality

- RED
- REFACOCTOR
- GREEN

Refactored code must also pass tests, no new functionality

Write just enough code to pass test
Tools in the agile developer’s arsenal...

- Pairing
- Refactoring & TDD
- Decomposition
- Simple design
- Small deliverables
- Collective ownership
- Spikes
static code analysis

Fortify

Veracode

Yasca

Parasoft

AGILE

ORLANDO
Focus on Quality

TESTER
YOU SCHEDULED THE END OF THE TEST PHASE AFTER THE START OF THE PRODUCTION PHASE.

WE'RE FEELING CONFIDENT.

IT'S TOO BAD THAT BEING SMART Doesn'T COME WITH SOME SORT OF GOOD FEELING LIKE THAT.
testing approaches

From the Inside Out

From the Outside In
inside out testing (white box)

Assumes that the applications source code is available to testers

Code can be analyzed statically (tools)

- Coding errors that make applications vulnerable to attack
  - Buffer overflows
  - Format string attacks

“Attack Surface”

- List of inputs
- Resources used by application
Fuzzing

Based on Fault Injections

- Crafted strings used in SQL injection attacks
- Random byte changes in input strings
- Random strings fed as command Line Arguments
Outside In Testing (Black Box)

Same techniques used by attackers
Security testers need to have the same mindset as attackers
Must be update-to-date on knowledge concerning hardware and software platforms
Usually outsourced to professional security testers
<table>
<thead>
<tr>
<th>White box testing</th>
<th>Black box testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assumes source code is available to tester</td>
<td>Source code not necessarily available</td>
</tr>
<tr>
<td>Code is analyzed statically</td>
<td>Same techniques that hackers would use (security testers need to think like the bad guy)</td>
</tr>
<tr>
<td>&quot;attack surface&quot; (list of inputs &amp; resources used by the application)</td>
<td>Usually outsourced to professional security testers</td>
</tr>
<tr>
<td>Planned with intimate knowledge of the program</td>
<td>Planned without intimate knowledge of the program</td>
</tr>
<tr>
<td>Tests each aspect of the program logic</td>
<td>Based on the specification of the program</td>
</tr>
</tbody>
</table>
strategies for assessing security controls
Incorporate security testing into your overall testing strategy

Add a static code analysis tool

Adopt continuous integration

Add a security vulnerability scanner (such as Nessus)

Add fuzzer tools
the 80/20 rules

80% Vulnerabilities Found
20% Budget

20% Remaining Vulnerabilities
80% Budget (security professionals)
Incorporate security testing into the overall testing strategy

Automate! Automate! Automate!
PROJECT MANAGER
SCRUM MASTER
PRODUCT OWNER
Don’t Hide the Cost of Security!

Explicit:
User registration story implementing a secure connection

Not so Explicit:
Defending against a DoS attack, ensuring scalability becomes a fudge factor across the project
### Security Objectives
- Clarification of goals, objectives, and constraints
- Effectively prioritize and invest accordingly

### Security Spikes
- A spike is a quick experiment for the developer to explore potential solutions.
- A security spike is focused exploring potential security solutions with the goal of reducing risk.

### Security User and Abuser Stories
- A security-focused scenario.
- An existing user story, with a security lens.
- New user story that focuses on a security goal, requirement, or constraint.
- Identify security stories throughout.
Abuser Story Value and Rank May Be Affected By...

- Technology breakthroughs
- More attractive assets
- Counter measures taken in prior iterations
- Better funded adversaries
Abuser stories carry a cost = negative business value

Ranked according to perceived threat to assets

Amount of damage caused by successful attack
STAKEHOLDERS
Capture Risks in Stakeholder Security Stories

As a User...

• I want to be the only one who can access my account, so that I can keep my information private
  • Risk level: HIGH
  • Controls: Authentication and Data Layer Access Control

• I want my personal information encrypted in storage and transit so that it doesn’t get stolen by attackers
  • Risk Level: HIGH
  • Controls: SSL and Encryption

As a User...

• ...I want to be the only who can edit Employee salaries so that I can prevent fraud
  • Risk Level: HIGH
  • Controls: Function Layer Access Control

• I want all security critical actions logged, so that attacks can be noticed and diagnosed
  • Risk Level: MEDIUM
  • Controls: Logging and Intrusion Detection

As a Manager...

As a Business Owner...
THIS CLASS WILL MAKE ME MORE EFFICIENT.

I DON'T WANT YOU TO BE MORE EFFICIENT. YOU'RE WORKING ON A GOVERNMENT CONTRACT AND BILLING BY THE HOUR.

NOW GO BILL THEM FOR THE TIME YOU STOOD HERE AND STARED AT ME LIKE A STUFFED DEER.
THIS WORKSTATION IS STILL LOGGED INTO THE NETWORK.

IT'S TIME TO TEACH SOMEONE A LESSON ABOUT SECURITY.

I'LL JUST GO IN HERE AND...

WARNING! WARNING!

IDIOT ALERT!

GAAAAA!!!

ACTIVATING DEFENSIVE WEDGIE SYSTEM.

I HAVE TO GO. SOME IDIOT VIOLATED MY PERIMETER.

PLEASE MAKE IT STOP.

THEN HOW DO YOU LEARN?
they can help by...

Understanding that business as usual has consequences

Training to help get groups to understand each other

Reprioritize business & security needs

Project divergence rates
THE ORGANIZATION
Every time a new requirement, feature is created, someone should spend time thinking about how that feature might be unintentionally or intentionally abused.
they can help by...

- Commitment to code security
- Evaluate tools that can help code security
- Encourage integration of tools into dev & QA cycles
- Create continuous feedback loop that includes stakeholders
they can help by...

- Education & training: developers, testers, customers
- User/abuser story-driven process
- Enterprise architecture decisions
- Organizational adoption of threat modeling
Making Your Organization More Agile

Standard controls
• Without these, security in Agile is simply too hard

Template information
• Threat models
• Stakeholder security stories
• Test cases

On demand application security guidance
• Role based guidance
Software needs to be...

Fast
Agile

But also...

Harsh
Unfriendly

The Rugged Manifesto

I am rugged and, more importantly, my code is rugged.

I recognize that software has become a foundation of our modern world.
I recognize the awesome responsibility that comes with this foundational role.
I recognize that my code will be used in ways I cannot anticipate, in ways it was not
designed, and for longer than it was ever intended.
I recognize that my code will be attacked by talented and persistent adversaries who
threaten our physical, economic and national security.

I recognize these things – and I choose to be rugged.

I am rugged because I refuse to be a source of vulnerability or weakness.
I am rugged because I assure my code will support its mission.
I am rugged because my code can face these challenges and persist in spite of them.
I am rugged, not because it is easy, but because it is necessary and I am up for the
challenge.
Exercise

Think about your current organization:

• What will you add to each of the following activities and who will do them?
  - Release or Project Planning
  - Sprint Planning
  - Execution (during the sprint)
  - Closing the sprint
  - Stabilizing a release
Security in Agile

Application Security Assurance Review

1. Define Project
2. Story Finding/Initial Estimation
3. High Level Planning
4. Begin iteration N
5. Write Story and Scenario
6. Implement functionality and acceptance tests
7. Deploy
8. Quality Assurance
9. Stories Left?
10. System Testing
11. RELEASE

Threat Model

Stakeholder Security Stories

Periodic Security Sprints
Session Feedback

Please provide feedback on this session!

You can do so in 3 ways:

1. Visit this session on the Mobile App. Click Session Feedback.
2. Scan the unique QR Code for this session located at the front and back of the room.
3. Visit the unique URL for this session located at the front and back of the room.

Thank you for providing your feedback 😊
bibliography


NIST SP 800-53
BACKUP SLIDES
Periodic Security Sprints

Build/integrate related security controls
  • Implement highest priority related security controls first

Examples
  • Authentication, Sessions, Authorization
  • Validation, Canonicalization, Encoding
  • Error Handling, Logging, Intrusion Detection
Implement Stakeholder Security Stories

- Security stories are implemented just like other stories
  - Test-Driven Development
    - Unit test cases come before the code
  - Continuous reviews and inspection
    - Pair programming
    - Constant informal reviews
  - Continuous integration

- Avoid common vulnerabilities
  - Security controls are missing
  - Security controls are not used in all the right places
  - Security controls are incorrect
  - Security controls are misconfigured
  - Security controls are not used properly
Test Cases for Security Controls

Security ‘requirements’ are defined by developing test cases.

- Unit tests can test both positive (functional) and negative (not broken) aspects of security mechanisms.
- Tests are repeatable, providing full regression testing.
- But not true penetration testing or analysis.

Real experience with test driven development.

- The OWASP Enterprise Security API.
- 600+ test cases, rerun every time a change is made.

Results in significant increase in assurance.
Test Cases for Security Stories

- Functional test cases
  - Typical unit testing
  - Verify presence and proper function of security control
  - May include simple tests with a browser

- Security test cases
  - Check for best practices
  - Test for common pitfalls

- Test cases provide strong assurance evidence

- Security Testing
  - Verify that functional and security tests were performed
  - Provide additional specialized security testing expertise
Perform Agile Security Reviews

Security Sprints: Verify all are in place and complete

- Threat model
- Security stories
- Security controls
- Test cases
- Test results

Notice: Most are standard agile artifacts, not just add-on security deliverables

Application code review and penetration testing

- Added for critical applications to increase assurance
- Manual (tool supported), automated, or both
- In security sprints and/or predeployment testing
Example: Using Agile to Solve XSS

- What are the risks associated with XSS?
  - What is the likelihood and impact of each risk?
  - Who is affected?

- What stories can you create around those risks?
  - What controls are required?
  - Which stories do you implement first?

- How do you test proper implementation?
  - What test cases would be appropriate?

- Can multiple security stories be implemented in the same sprint?