Mapping Application Security to Compliance

Ed Adams
CEO
Security Innovation

John Kirkwood
CISO
Security Innovation
Agenda

- **About Security Innovation**
  - Security Drivers and Industry Data
  - Aligning software development with corporate policies and compliance requirements
  - Creating an action plan to identify and remediate gaps between current and best practices
  - Conclusion
About Security Innovation

• Application Security Experts
  – 10+ years research on vulnerabilities
  – Hundreds of assessments on world’s most dominant software
  – Security testing methodology adopted by SAP, Symantec, Microsoft and McAfee
  – Authors of 8 books

• Products, Services & Training
  – Software and Code Assessment
  – SDLC Consulting
  – eLearning

• Helping organizations
  – Build internal application security competency
  – Create a secure, repeatable SDLC
  – Reduce application risk
About Edward Adams & John Kirkwood

Ed Adams, CEO, Security Innovation
- 15+ years experience in the IT security and quality assurance industries
- Sits on board of the National Association of Information Security Groups (NAISG) and Massachusetts North Shore Technology Council (NSTC)
- Maintains a blog with CSO Magazine

John Kirkwood, CISO & Senior Security Strategist, Security Innovation
- Past Global / Chief Information Security Officer for top Global Fortune 500 Companies for the past 8+ years
- Responsible for creating application security and risk management programs
- Work experience of more than 20 years in Financial Services, Retail and Healthcare Industries
- Global experience in the Americas, Europe and Asia
Agenda

• About Security Innovation

  ➢ Security Drivers and Industry Data
  
  • Aligning software development with corporate policies and compliance requirements

  • Creating an action plan to identify and remediate gaps between current and best practices

• Conclusion
Major Cyber Incidents 2011 YTD

Q1
- Honda
  - Automotive
  - 283,000 Unauthorized Data Access
  - $206M Lawsuit
  - 01.01.11

Q2
- Sony
  - Entertainment
  - 40M Cyber Attack
  - $66M Rem.
  - 03.01.11
- Google
  - Technology
  - 100 M+ Unauthorized Data Access
  - $171M Rem.
  - $1 B Lawsuit
  - 04.20.11

Q3
- RSA
  - Global Governmental Coalition
  - "unknown # of systems compromised (possibly 187 member countries)"
  - "Suspicious file transfers"
  - 06.04.11
- BlackBerry
  - Mobile Device
  - "cascade failure" of RIM data systems
  - RIM offers $100 app bundle
  - 06.11.11

Q4
- NASDAQ
  - Stock Exchange
  - 10,000 Clients
  - Data Sharing Service Compromised
  - 02.06.11
- Epsilon
  - Email Marketing
  - Potentially tens of millions of e-mail addresses stolen Unauthorized Data Access
  - 03.01.11
- Lockheed Martin
  - Defense Contractor
  - # unknown RSA SecurID
  - Used to breach LM
  - 05.27.11
- Citibank
  - Financial Services
  - 200,000 Unauthorized Access
  - $2.7M Stolen
  - 06.09.11
- ADP
  - Payroll Processing
  - 1 client w/ 3.5 M users
  - Cyber Attack
  - 06.11.11
- Sutter Health
  - Healthcare
  - 4M healthcare records stolen
  - 10.25.11

Root Cause & the Hackers Target: Insecure Applications

“Today, the “money” is in software applications – that’s where companies process their most sensitive data from Credit card numbers to customer and employee information as well as trade secrets”

-- Forrester

- 90%+ attacks are at the application layer (Verizon Business Study)
- Hacks targeting retail sector have increased 43%, largely due to SQL injection and the use of exploit toolkits (Dell SecureWorks)
- 25% of respondents indicated that meeting compliance objectives is the most effective argument in convincing management to invest in software security (Forrester)
- 92% of attacks were not highly difficult (Verizon Business Study)
Application Security: the Next Frontier of Compliance

- Regulations historically focused on network security, but application security requirements are emerging
  - FISMA & NIST
    require organizations to integrate security assessments into SDLC
  - PCI-DSS
    “secure coding standards”; “..prevent vulnerabilities such as injection flaws”
  - SEC
    Evaluate security risks to determine if disclosure is required
  - Dozens of others

- Requirements are general and implications non-obvious
  - “Develop according to industry best practices”
    • uh, where can I find those?
  - “Protected information “should not be improperly altered or destroyed
    • Huh???
Agenda

- About Security Innovation
- Security Drivers and Industry Data
  - Aligning software development with corporate policies and compliance requirements
  - Creating an action plan to identify and remediate gaps between current and best practices
- Conclusion
Corporate Application Compliance Framework

aligning development with management policies

- Compliance Standards & Corp. Governance
- Security and Compliance Policies
- Development Processes and Coding Practices
- Compliance Activities
Mapping OWASP Top Ten to PCI DSS

<table>
<thead>
<tr>
<th>PCI DSS Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5.1 Injection flaws, particularly SQL injection. Also consider OS Command Injection, LDAP and XPath injection flaws as well as other injection flaws.</td>
</tr>
<tr>
<td>6.5.2 Buffer overflow</td>
</tr>
<tr>
<td>6.5.3 Insecure cryptographic storage</td>
</tr>
<tr>
<td>6.5.4 Insecure communications</td>
</tr>
<tr>
<td>6.5.5 Improper error handling</td>
</tr>
<tr>
<td>6.5.6 All &quot;High&quot; vulnerabilities identified in the vulnerability identification process (as defined in PCI DSS Requirement 12)</td>
</tr>
<tr>
<td>Note: This requirement is considered a best practice until June 30, 2012, after which it becomes a requirement.</td>
</tr>
<tr>
<td>Note: Requirements 6.5.7 through 6.5.9, below, apply to web applications and application interfaces (internal or external):</td>
</tr>
<tr>
<td>6.5.7 Cross-site scripting (XSS)</td>
</tr>
<tr>
<td>6.5.8 Improper Access Control (such as insecure direct object references; failure to restrict URL access, and directory traversal)</td>
</tr>
<tr>
<td>6.5.9 Cross-site request forgery (CSRF)</td>
</tr>
</tbody>
</table>

OWASP Top 10 – 2010
The Top 10 Most Critical Web Application Security Risks

A1 – Injection
A2 – Cross Site Scripting (XSS)
A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)
A6 – Security Misconfiguration (NEW)
A7 – Failure to Restrict URL Access
A8 – Unvalidated Redirects and Forwards (NEW)
A9 – Insecure Cryptographic Storage
A10 – Insufficient Transport Layer Protection
## PCI DSS mapped to ISO 27001 (and OWASP)

<table>
<thead>
<tr>
<th>PCI DSS</th>
<th>ISO 27001 relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Install and maintain a firewall configuration to protect cardholder data</td>
<td>✓</td>
</tr>
<tr>
<td>2: Do not use vendor-supplied defaults for system passwords and other security parameters</td>
<td>✓</td>
</tr>
<tr>
<td>3: Protect stored cardholder data</td>
<td>✓</td>
</tr>
<tr>
<td>4: Encrypt transmission of cardholder data across open, public networks</td>
<td>✓</td>
</tr>
<tr>
<td>5: Use and regularly update anti-virus software</td>
<td>✓</td>
</tr>
<tr>
<td>6: Develop and maintain secure systems and applications</td>
<td>✓</td>
</tr>
<tr>
<td>7: Restrict access to cardholder data by business need-to-know</td>
<td>✓</td>
</tr>
<tr>
<td>8: Assign a unique ID to each person with computer access</td>
<td>✓</td>
</tr>
<tr>
<td>9: Restrict physical access to cardholder data</td>
<td>✓</td>
</tr>
<tr>
<td>10: Track and monitor all access to network resources and cardholder data</td>
<td>✓</td>
</tr>
<tr>
<td>11: Regularly test security systems and processes</td>
<td>✓</td>
</tr>
<tr>
<td>12: Maintain a policy that addresses information security</td>
<td>✓</td>
</tr>
</tbody>
</table>
### Selected coding practices that contribute to compliance

<table>
<thead>
<tr>
<th>High-Level Requirement</th>
<th>Standards (Partial List)</th>
<th>Selected Coding Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>SOX, PCI DSS, HIPAA, ISO 27002, HIPAA, GLBA, FFIEC, Basel I I, CA SB 1386, FIPS 199, NIST SP 800-30/ 800-53/800-64</td>
<td>Appropriate use of strong encryption for data in databases.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Encrypting confidential data in memory. No custom or untrusted encryption routines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Encrypting data in motion, especially for wireless transmissions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Masking confidential data that needs to be viewed in part.</td>
</tr>
<tr>
<td>Data integrity</td>
<td>SOX, PCI DSS, ISO 27002, HIPAA, GLBA, FIPS 199, NIST SP 800-30/ 800-53/800-64</td>
<td>Robust integrity checks to prevent tampering with data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input validation and comprehensive error handling to prevent injection attacks, privilege escalation, and other hacking techniques.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output encoding. Use of least privileges.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hashing for confidential data that needs to be validated (e.g. passwords).</td>
</tr>
<tr>
<td>Authentication and access control</td>
<td>SOX, PCI DSS, ISO 27002, HIPAA, II, NIST SP 800-30/ 800-53/800-64</td>
<td>Support for strong passwords &amp; two-factor authentication where appropriate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Role-based access control and revocation of rights, with clear roles mapped to permissions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Locked down file access and database roles. No guest accounts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Passwords and encryption keys encrypted before storage and transmission.</td>
</tr>
<tr>
<td>Logging and auditing</td>
<td>SOX, PCI DSS, ISO 27002, HIPAA, SB 1386, NIST SP 800-30/ 800-53/800-64</td>
<td>Detailed audit trails of users accessing data and resources.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Detailed logging of systems that process sensitive data, including shutdowns, restarts and unusual events. No confidential data exposed in logs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Event logs and audit trails available only to system admins and protected from unauthorized modifications.</td>
</tr>
<tr>
<td>Availability</td>
<td>SOX, ISO 27002, HIPAA, II FIPS 199, NIST SP 800-30/ 800-53/800-64</td>
<td>Code reliability. Failover and redundancy built into applications.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Applications resistant to denial of service attacks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clean up of confidential data in memory and in file systems during failures and shutdowns.</td>
</tr>
<tr>
<td>Change management</td>
<td>SOX, BASEL II, NIST SP 800-53/ 800-64</td>
<td>Source control. Logging of application changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Application change logs accessible only to privileged users and resistant to tampering.</td>
</tr>
</tbody>
</table>
Application Security Continuum

- Secure at the Source
- Defend in Place
- Find and Fix

Results to Dev Team

Ready for re-scan
Aligning Development Activities with Compliance: OWASP and Other Coding Standards

- **OWASP**
  - Maps to and referenced in many industry and regulatory compliance standards and frameworks
    - U.S. FTC and DISA, PCI-DSS
  - Used by many companies
    - NSA: in their developer guidance on web application security
    - Oracle: for developer awareness
    - IBM AppScan: maps source code findings to OWASP Top 10

- **CWE**: most dangerous software weaknesses

- The CERT secure coding standards

- The Microsoft SDL (Secure Development Lifecycle)

- Security Innovation’s TeamMentor™
  - extensive collection of Secure SDLC checklists and code samples
Agenda

- About Security Innovation
- Security Drivers and Industry Data
- Aligning software development with corporate policies and compliance requirements
  - Creating an action plan to identify and remediate gaps between current and best practices
- Conclusion
Assessing your Existing SDLC
Graphical View

1.) Review Org Structure and Team Roles

2.) Analyze Policies & Standards Requirements

3.) Analyze & Aggregate Data

Best Practices

4.) Refine via focused Interviews (usually team leads)

5.) Create Gap Analysis Report with recommendations
### Assessing your Existing SDLC

#### Activity Matrix

<table>
<thead>
<tr>
<th>Activity</th>
<th>Product A</th>
<th>Product B</th>
<th>Product C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Security Objectives</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Apply Security Design Guidelines</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Threat Model</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Security Architecture and Design Review</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Apply Security Implementation Guidelines</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Security Code Review</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Security Penetration Testing</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Apply Security Deployment Guidelines</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Security Deployment Review</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3rd party Security Penetration Test</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Security Incident Response Plan</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Assessing your Existing SDLC

Security Policies

• Security policies
  – are the backbone of your development process
  – without them, many efforts are wasted
    • what good is a scanning tool if it’s use is not required?

• Questions to ask yourself
  – do you have a formal development process with well-defined phases and activities?
  – do you have a dedicated security team?
  – do you have corporate security and compliance policies?
  – how is the development team made aware of security policies?
  – how does the development team access security policies?
  – how does your development team interact with company security policies (governance, compliance, etc)?
Assessing your Existing SDLC

Requirements & Design Phase

- **Security activities**
  - security requirements objectives
  - threat modeling
  - design best practices & design reviews

- **Questions to ask yourself:**
  - do you gather security objectives?
    - How are they stored? How are they mapped to the rest of the design process?
  - do you have a set of design best practices that you employ for security?
    - How do you ensure architects are using them?
    - How do you revise and improve them over time?
  - does your team conduct security architecture and design reviews?
    - How often? Do you use checklists to drive the process?
    - How are the results tracked and used to improve the design?
  - does your team create threat models for your application’s architecture/design?
    - When? Is it updated over time?
    - How is it used to improve the design, implementation and testing?
Assessing your Existing SDLC

Implementation Phase

• Implementation phase security activities
  – development best practices
  – security code reviews

• Questions to Ask
  – does your team use a formalized set of security coding best practices?
  – what type of code scanning tools do you use?
  – do you perform code reviews against security best practices?
    • How often? What is the process?
    • Do you have a set of checklists that can use drive the review process?
    • How are the results tracked and used to improve the implementation?
Assessing your Existing SDLC
Verification Phase

• Verification phase security activities
  – abuse case definition
  – penetration testing

• Questions to ask:
  – does your team conduct 3rd party or internal penetration tests?
    • How often are they performed?
    • Do you prioritize attack paths based on a threat model?
    • Do you have a set of vulnerabilities, unique to your system, that you test against?
    • How are the results tracked and used to improve the implementation?
  – are your testers/QA trained on the latest attack trends and test techniques?
  – do you use security testing tools? Does your team know how to use them effectively?
    • Web scanners such as AppScan or WebInspect
    • File and network fuzzers
    • etc
Assessing your Existing SDLC

Release & Response Phase

• Release/response phase security activities and preparedness
  – security deployment review
  – security attack response
  – patching processes

• Questions
  – does your team use a formalized set of security deployment best practices?
  – do you have a security incident response plan?
  – do you use network scanning tools such as Nessus?
  – do you have a set of deployment best practices that you employ for security?
    • How are they stored? Do you ensure your developers are using these?
    • How do you revise and improve these best practices over time?
  – do you review the deployment for security best practices before deployment?
    • How often are inspections performed? Do you use checklists to drive the process?
    • How are the results tracked and used to improve the deployment?
Use assessments from previous phases. For each high-priority area:
- review the major risk management strategies
- identify appropriate control options
- describe necessary modifications to compliance activities
- Identify which activities/controls will yield biggest “bang for the buck”

Use results to construct a phased software risk remediation roadmap

Select tools and partners that can help implement

Sequencing is critical
- introduce baseline guidance for all first
- work with security champions; develop them as mentors
- beware not to invest in new tools too soon

Measure progress relative to compliance & security objectives/requirements
- adjust as corporate priorities, threat patterns, compliance standards change
Agenda

- About Security Innovation
- Security Drivers and Industry Data
- Aligning software development with corporate policies and compliance requirements
  - Creating an action plan to identify and remediate gaps between current and best practices
- Conclusion
SDLC Compliance Assessment for Energy Client

High-Level findings

- Good software engineering processes but lacks coordinated formalized security engineering processes
  - teams not trained on security principles
  - no standards for secure design or implementation
  - no architecture, design or code reviews focused on security
  - no threat modeling being done to assess risk
    - team unable to prioritize efforts or mitigate threats repeatedly

- Team is making lots of mistakes, both in design & development
  - resulted in many exploitable vulnerabilities in critical applications
  - likely indicative of all the company’s applications

- Would fail PCI and ISO audits
Case Study: SDLC Assessment for Energy Client

security activity recommendations

How security engineering activities can be layered into a traditional software development process

<table>
<thead>
<tr>
<th>Core</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td></td>
</tr>
<tr>
<td>Requirements and Analysis</td>
<td>Security Objectives</td>
</tr>
<tr>
<td>Functional Requirements</td>
<td></td>
</tr>
<tr>
<td>Non Functional Requirements</td>
<td></td>
</tr>
<tr>
<td>Technology Requirements</td>
<td></td>
</tr>
<tr>
<td>Architecture and Design</td>
<td>Design Guidelines for Security</td>
</tr>
<tr>
<td>Design Guidelines</td>
<td></td>
</tr>
<tr>
<td>Architecture and Design Review</td>
<td></td>
</tr>
<tr>
<td>Threat Modeling</td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td>Code Review for Security</td>
</tr>
<tr>
<td>Unit Tests</td>
<td></td>
</tr>
<tr>
<td>Code Review</td>
<td></td>
</tr>
<tr>
<td>Daily Builds</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>Security Testing</td>
</tr>
<tr>
<td>Integration Testing</td>
<td></td>
</tr>
<tr>
<td>System Testing</td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td>Deployment Review for Security</td>
</tr>
<tr>
<td>Deployment Review</td>
<td></td>
</tr>
<tr>
<td>Maintenance</td>
<td></td>
</tr>
</tbody>
</table>
A **threat model** would have exposed key assets for protection and design-level mitigations could have been created.

A **design review** would have checked that each design mitigation was placed in the architecture properly.

**Secure implementation best practices** and **security focused code reviews** would ensure:
- the development of input and data validation routines
- the proper use of output encoding and cryptography

A **penetration test before** deployment could discover issues that fell through the cracks in the early phases of development.

All of the above would have been steps toward compliance
- (not to mention making their software systems more secure 😊)
Case Study: SDLC Assessment for Energy Client

training recommendations

Requirements & Analysis
• How to Define Security Objectives (ENG 111)

Architecture & Design
• Fundamentals of Secure Architecture (DES 101)
• Architecture Risk Analysis & Remediation (DES 212)
• Creating Secure Application Architecture (DES 311)
• Intro to Threat Modeling (ENG 301)

Code Implementation
• Classes of Security Defects (TST 201)
• Fundamentals of Secure Development (COD 101)
• Understanding Secure Code -.NET 4.0 (COD 214)
• Creating Secure Code - ASP.NET (COD 311)
• How to Perform a Code Review (ENG 401)

Testing
• Fundamentals of Security Testing (TST 101)
• How to Test for OWASP Top Ten (TST 211)

Everyone: Fundamentals of Application Security (AWA 101)
recommendations for requirements gathering

• When accepting a change request or a new requirement, examine each requirement for security and compliance impact
  – if there will be a security impact, track it as a new security requirement
  – evaluate if there needs to be additional security requirements in place to mitigate added risk
  – requirements management tools can help here (esp. with traceability)

• Before moving on to application design, security objectives and requirements must be defined
  – review security objectives to ensure they are appropriate for the functional requirements and application scenario
  – determine security objectives based upon data asset classification

• All security requirements should be tracked in the requirements management tool
  – they had one already; just weren’t using it for security
Case Study: SDLC Assessment for Energy Client

Design Recommendations

- Use the TeamMentor security guidelines to apply security design best practices

- Perform a security architecture and design review before coding starts
  - Use the TeamMentor security checklists to drive the review, provide usable guidance, and document for compliance audits

- Create a threat model on your application’s design before coding begins
  - Ensure asset classification and to help prioritize threats
Implementation Recommendations

- Use TeamMentor security guidelines to apply security implementation best practices
  - “Secure Coding Standards” requirement in PCI

- Perform a security code review before each check-in
  - can be implemented with buddy code reviews as well as with the occasional group code review for knowledge sharing
  - use the TeamMentor security checklists to drive the review

- Require that Visual Studio Code Analysis is turned on and all errors and warnings are handled before each check-in
Case Study: SDLC Assessment for Energy Client

Verification Recommendations

- Use the security objectives and the threat model to build a security penetration test plan
  - can write tests before code is written

- Complete internal penetration testing before deployment
  - document for PCI and ISO requirements

- Complete a 3rd party penetration test on Internet facing applications before deployment
  - document for PCI and ISO requirements
Case Study: SDLC Assessment for Energy Client

Release Recommendations

- Perform a security deployment review, including configuration settings, before deploying
  - Use TeamMentor security checklists to drive review

- Ensure there is a security incident response plan in place
  - should include severity levels for potential vulnerabilities, escalation plans and engineers on call

- Where there is an incident response plan in place, this can be used as the basis for the security incident response plan
Case Study: SDLC Assessment for Energy Client

Recommended rollout sequencing

- **Security Objectives**
  - if you don’t know the security it’s difficult to be successful with any other activities

- **Architecture and Design Review for Security**
  - bugs introduced in the design phase are the most expensive to deal with later

- **Threat Modeling**
  - Drives the test plans and ensures that you address most critical threats

- **Code Review for Security**
  - implementation bugs are the most common
  - can save you later rework or help avoid costly exploits

- **Security Review for Deployment**
  - even an effective process can be undone by a configuration error during deployment

- **Design Guidelines for Security**
  - adopting proven design principles ensures your application is secure from the start

- **Security Testing**
  - used to validate designed mitigations and ensure nothing slipped through the cracks
Agenda

- About Security Innovation
- Aligning software development with corporate policies and compliance requirements
- Creating an action plan to identify and remediate gaps between current and best practices

➤ Conclusion
Conclusion

- Most regulations, frameworks, and compliance mandates revolve around the same key, best practices for secure development
  - Simple tools like spreadsheets can help you organize these with controls for rows and activities for columns
    - helps visualize impact of single activity on multiple compliance requirements

- Rolling out a repeatable SDLC that integrates key security and compliance activities:
  - Ensures future requirements will have little impact on existing efforts
  - Allows you to maintain a “big picture” view to software development and IT teams

- Secure development has benefits beyond compliance
  - 50% of software vulnerabilities prior to production can reduce configuration management and incident response costs by 75 percent (Gartner)
  - Audit costs and “re-do” expenses dramatically reduced
  - over 70% of all exploits take advantage of known and common vulnerabilities
Repeatable, Secure Development Works
A look at the Microsoft SDL

Total Vulnerabilities Disclosed 12 Months After Release

- Windows® XP: Before SDL 119, After SDL 66
- Windows Vista®: Before SDL 400, After SDL 242
- OS I: Before SDL 242, After SDL 157

45% reduction in Vulnerabilities

Total Vulnerabilities Disclosed 36 Months After Release

- SQL Server® 2000: Before SDL 34, After SDL 3
- SQL Server 2005: Before SDL 157, After SDL 187
- Competing commercial DB: Before SDL 119, After SDL 66

91% reduction in Vulnerabilities

Consistent, well documented security practices during all phases of a development project will not only facilitate compliance, but result in fewer vulnerabilities.
How Security Innovation can Help

TeamProfessor eLearning
- Security Awareness for Staff
- Technical Training for Dev/IT Teams
- ASP.Net, Java, C/C++, .Net, Windows, C#, JRE
- PCI-DSS for Developers; Intro to PCI-DSS
- How to Test for the OWASP Top Ten

TeamMentor: Secure Development Guidance System
- 3,5000 Code snippets, how-to’s, checklists, attacks, principle
- Guidance views for OWASP, PCI-DSS
- http://teammentor.securityinnovation.com

Application Risk Consulting
- Software & Code Assessment
- Enterprise Application Risk Ranking
- SDLC Assessment (and mapping to compliance requirements!)

getsecure@securityinnovation.com