The Need for Confluence

The Essential Role of Incident Response in Secure Software Development
Why do security incidents occur?

What is the root cause?
Faulty software (more often than not)
What is the definition of insanity?

- Year after year
- Thousands upon thousands of incidents
- Same root cause
- What are we doing about it?
- We *talk* about proactive, but do we do it? *Really?*
You can’t bolt security on later

- A room full of firewalls, intrusion detection|prevention systems, etc., will not protect bad software
- We must address the root causes
- Active participation in development
Why aren’t things improving?
Learn from history

• We don’t pay enough attention to our failures
• Consider other engineering disciplines
Lack of knowledge

- Developers tend to not have security knowledge
- Security team tends to not have development knowledge
- “Us” and “them”
We’re overly trusting

- We tend to have misplaced trust in our users
- Sometimes users are malicious
- Sometimes they don’t even try to be
Focus

• Too much attention is paid to functional spec
• Consider what can go wrong as well
Complexity

- Complexity is fighting us every step of the way
- Consider AJAX
Connectivity

- Connectivity is everywhere
- Do you know where your data is?
- Consider mobile users, SOAP, grid computing
Extensibility

- Extensibility isn’t what it used to be
- Who wants a computer that isn’t?
- Is your desktop user privileged?
Old school paradigms

- Old school information security solutions don’t adequately protect the software
- Consider IM, Skype, WiFi, VPNs
Testing isn’t working

- Software testing does not adequately address security
- Penetration testing is not sufficient
So how can we help?

- Deep integration into the development process
- Consider five stages
  - Requirements
  - Design
  - Code
  - Testing
  - Deployment
But first, think positive

- We’re too quick to use negative models
  - Anti-virus products
  - Signature-based IDS
  - Vulnerability scanning

- These are not adequate
  - Think positive validation

- Prove something safe and then allow it
  - All else is evil

- Throughout a system
  - From OS through application

- Prime example
  - Input validation

Copyright© 2008 KRvW Associates, LLC
Part of the team

- Don’t just be a reviewer/auditor
  - Adversarial role can be detrimental
- Be a security consultant to dev
  - Each project
  - Guide and assist the dev team
Requirements

- Help build security requirements
  - Regulatory compliance
  - Abuse/misuse cases
- Guide discussions on what bad things can happen

- Focus on lessons learned
  - Cite similar architectures and failures
- Requirements must be actionable
  - Focus on “must not” actionable statements
Design

- Help conduct design reviews
- Consider available approaches
  - Microsoft’s threat modeling
  - Cigital’s ARA
- Identify and prioritize design weaknesses
  - Based on business risk
- Build mitigations for high priority risks
- Nightmare scenarios can be useful
Code

- Learn the technologies
- Help build prescriptive language guidance
  - Input validation
  - SQL utilization
  - Authentication
  - Session management
- Help build reusable class libraries frameworks
  - Enterprise coding standards
- Code reviews to verify compliance
  - Manual vs. automated
- Look at open projects
  - OWASP’s ESAPI
Testing

- Penetration testing alone is not enough
  - Coverage
  - Internals
- Consider Microsoft’s testing approach
  - Fuzzing
  - Pen testing
  - Dynamic validation

- Results must be meaningful to the dev team
  - Findings by code module
  - Integrate with bug tracking s/w
  - Prescriptive guidance
Deployment

- Verification of safe deployment environment
  - Not just pen testing
  - Host hardening
  - File access controls
  - Event monitoring

- Verify and validate manually and empirically
  - Time to break out the network sniffer

- If WAFs used, help train the WAF on app’s normative behavior
Issues to consider

- Cultural barriers
  - Years of “us and them” may be tough to overcome
  - Developers “allergic” to security
  - Authority to mandate
  - Positive incentive

- Consider small steps towards a goal
  - But first, understand what the goal is

- Measurement helps
  - Track and show improvement
Checklist of things to do

- Read, study, learn
  - Work through OWASP WebGoat exercises
  - Language references
  - See reference list
- Seek dev team
  - Discuss possible roles and responsibilities
- Learn dev environment
  - Bug tracking
  - Process
- Pilot studies
  - Pick a project and dive in
  - Capture lessons learned
  - Constant improvement

Copyright © 2008 KRvW Associates, LLC
Further reading

- OWASP
- Secure Coding Mailing List, [http://www.securecoding.org](http://www.securecoding.org)

Copyright © 2008 KRvW Associates, LLC