Vulnerability Management in an Application Security World

OWASP San Antonio

January 29th, 2009
Agenda

• Background
• A Little Bit of Theatre
• You Found Vulnerabilities – Now What?
• Vulnerability Management – The Security Perspective
• Defect Management – The Development Perspective
• Making it Work
• Case Studies
• Questions
Background

• Dan Cornell
  – Principal at Denim Group [www.denimgroup.com](http://www.denimgroup.com)
  – Software Developer: MCSD, Java 2 Certified Programmer

• Denim Group
  – Texas-based consultancy
  – Application Development
    • Java and .NET
  – Application Security
    • Assessments, penetration tests, code reviews, training, process consulting
A Little Bit of Theatre

• This is a one-act play entitled: “We Found Some Vulnerabilities”

• Need a volunteer
You Found Vulnerabilities – Now What?

• Security Industry is too focused on finding vulnerabilities
  – Especially in application security this typically isn’t hard
• Finding vulnerabilities is of little value
• Fixing vulnerabilities is actually valuable
• Mark Curphey: Are You a Builder or a Breaker
  – http://securitybuddha.com/2008/09/10/are-you-a-builder-or-a-breaker/
Vulnerability Management – The Security Perspective

- **Steps:**
  - Policy
  - Baseline
  - Prioritize
  - Shield
  - Mitigate
  - Maintain

So How Are We Doing?

• Policy
  – Does your organization have policies for Application Security?
  – Or is your policy “Use SSL and do the OWASP Top 10”?

• Baseline
  – What are your organization’s testing strategies?
  – Hopefully not “Run scanner XYZ the day before an application goes into production”
  – Also – do you actually know how many applications you have in production?

• Prioritize
  – How do you determine the business risk?
  – Critical, High, Medium, Low often does not account for enough context
  – To defend everything is to defend nothing
So How Are We Doing? (continued)

• Shield
  – *Have you deployed technologies to help protect you in the interim?*
  – *WAFs, IDS/IPF*

• Mitigate
  – *Do your developers know what the actual problems are?*
  – *Do your developers know how to fix them?*
  – *When are these vulnerabilities going to be addressed and when do they go into production?*

• Maintain
  – *Web applications are dynamic – what is the ongoing testing strategy?*
Defect Management – The Developer Perspective

- Every day has 8 hours (12 if pizza and Jolt are made available)
- A given defect is going to require $X$ hours to fix (+/- 50%)
- Tell me which defects you want me to fix and I will be done when I am done (+/- 50%)
Why is Vulnerability Management Hard for Application-Level Vulnerabilities

- Actual business risk is challenging to determine
- People who find the problems do not typically know how to fix them
  - Or at the very least they are not going to be the people who fix them
- People who have to fix the problems often do not understand them
- Infrastructure fixes are typically cookie-cutter, Application fixes are much more varied
  - Patches and configuration settings
  - Versus a full custom software development effort
- Software development teams are already overtaxed
- Applications no longer under active development may not have development environments, deployment procedures, etc
Making It Work

- Application security vulnerabilities must be treated as software defects
- Use risk and effort to prioritize
Application Vulnerabilities as Software Defects

- Track them in your defect management system (bug tracker)
- Select defects to address for each development cycle or release
Interesting Resource

- **DefectLogger**
  - *Extension to IBM Rational AppScan to send vulnerabilities to defect tracking systems*
  - *Available for Microsoft Team Foundation System (TFS), Quality Center and ClearQuest*
  - *I wrote the TFS version and won a Nintendo Wii*
  - *See: http://code.google.com/p/defectloggertfs/*
Risk and Effort

- Risk crossed with remediation effort
- Risk: STRIDE and DREAD (there are others)
- Effort: Development hours and other resources
Risk Calculation Exercise

- Quantitative risk can be hard to calculate

- Weighted Cost = Likelihood of occurrence x Cost of occurrence

- What is the chance (%) that Amazon.com will have a publicly-accessible SQL injection vulnerability exploited within the next year?
- What would the monetary damage be to Amazon.com if a publicly-accessible SQL injection vulnerability was exploited?
STRIDE

• Spoofing Identity
• Tampering with Data
• Repudiation
• Information Disclosure
• Denial of Service
• Elevation or Privilege
DREAD

- Damage Potential
- Reproducibility
- Exploitability
- Affected Users
- Discoverability

- Assign levels: 1, 2, 3 with 3 being the most severe
- Average the level of all 5 factors

- Define your DREAD levels up-front and apply consistently
  - Organization-wide DREAD baseline
  - Application-specific DREAD standards
Level of Effort Calculation

• Varies widely by type of vulnerability and number of vulnerabilities
• Logical Vulnerabilities versus Technical Vulnerabilities
  – *Technical Vulnerabilities tend to be based on coding issues*
    • Injection flaws, XSS, configuration issues
  – *Logical Vulnerabilities are specific to the application*
    • Depend on business logic and business context
    • Authentication, authorization, trust
Estimating Technical Vulnerabilities

- time per fix x number of issues
- Verification typically easier because the application should behave as it always did, except that it now handled problem inputs correctly
- Go back to “coding” phase of SDLC
Estimating Logical Vulnerabilities

- Fix strategies are more varied than technical vulnerabilities
- May have to go farther back in the SDLC
  - Coding
  - Architecture/Design
  - Even Requirements
- Change may require more broad change management initiatives
  - If other applications within your organization or other organizations are impacted
Application Vulnerability Management

• Policy
  – *Have specific, actionable policies*
  – *What are development standards?*
  – *What types and levels of risks are acceptable in production?*

• Baseline
  – *Know your application portfolio*
  – *Have a regular program of assessment in place*

• Prioritize
  – *Use a risk-based approach tuned to your organization*
  – *Involve development teams to understand levels of effort for remediation*
Application Vulnerability Management

- **Shield**
  - *Consider deploying technologies to provide short-term relief*
    - Web Application Firewalls (WAFs)
    - Web-relevant IDS/IPF
  - *Understand that these are temporary measures*

- **Mitigate**
  - *Work application remediation into development schedules*
  - *Confirm that fixes have been applied and were applied correctly*
  - *Be sure fixes are deployed into production*

- **Maintain**
  - *Vulnerability management is not a one-time activity*
Case Studies

- Authentication FUBAR
- Legacy Nightmares
- When Tools Fail
Authentication FUBAR

• Situation
  – Several public-facing flagship applications under moderate ongoing development

• Vulnerabilities
  – Various SQL injection and XSS
  – Authorization problems
  – Pervasive poor deployment practices (backup files, configuration issues)
  – Verbose HTML comments with sensitive information
  – Major, fundamental issue with Authentication
    • Along the line of using SSNs to authenticate users to a system
    • Connected to many partner organizations
Authentication FUBAR (continued)

• Approach
  – Fix the serious SQL injection and publicly-accessible XSS immediately in an out-of-cycle release
  – Address authorization problems and some other issues during next planned release
  – Major full lifecycle, change management initiative to address Authentication issue
  – Defer remaining issues as “nice to fix”
Legacy Nightmares

• Situation
  – 10 year old application with hundreds of pages
  – Has been on end-of-life status for 5 years
  – NO active development

• Vulnerabilities
  – Hundreds of SQL injection, XSS
  – Authorization issues

• Approach
  – Sit in the corner and cry softly for a few minutes
  – Identify most critical SQL injection and XSS issues for code-level fixes
  – Fix authorization issues
  – Rely on WAF to address remaining issues
When Tools Fail

• Situation
  – *Thick-client application with a local database*
  – *Connects to web services and ERP*

• Vulnerabilities
  – *Code scanner identified many SQL injection vulnerabilities affecting the local database*
  – *Code scanner identified some quality issues that could impact security*
  – *Manual code inspection identified some frightening design issues affecting attack surface*

• Approach
  – *Ignore local SQL injection issues for now*
  – *Ignore quality issues for now*
  – *Address design issues before the initial release*
Questions?

Dan Cornell

dan@denimgroup.com
(210) 572-4400

Web: www.denimgroup.com
Blog: denimgroup.typepad.com