CONDUCTING AN INSTITUTION-WIDE, MULTI-DEPARTMENT APPLICATION SECURITY ASSESSMENT

Charlie Scott & Jay Paz
Information Security Office
The University of Texas at Austin
security@utexas.edu

LASCON 2010
Our Office

Information Security Office (ISO)

- Incident Handling
- Risk Management
- Internal Tools Development

App Assessment Team
Our Objectives

✓ To provide a thorough security assessment of the existing administrative applications.

✓ An education and awareness exercise for application developers so that future administrative applications will be more secure.
OUR CHALLENGE

283 Applications
OUR CHALLENGE

18 Months
OUR CHALLENGE

Developers & Business Managers
OUR CHALLENGE

4

Analysts
<table>
<thead>
<tr>
<th>Our Challenge</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Departments:</td>
<td>36</td>
</tr>
<tr>
<td>Web Applications Assessed:</td>
<td>283</td>
</tr>
<tr>
<td>Timeframe:</td>
<td>18 months</td>
</tr>
<tr>
<td>Developers &amp; Managers:</td>
<td>140</td>
</tr>
<tr>
<td>ISO Resources:</td>
<td>4 analysts</td>
</tr>
</tbody>
</table>
Our Methodology

- Communicate
  - Get Buy-In
  - Gather Info
  - Assess
  - Report
- Remediate
**Stakeholder Buy-In**

- Met with:
  - Technology Governance Groups (i.e. management).
  - Application registry stewards.
  - Developer community.
- Also announced initiative on university technical e-mail lists.
INFRASTRUCTURE COORDINATION

- Coordinated assessments with:
  - Infrastructure Stewards
  - Framework Owners
  - Authentication/Identity Management Team
- Setup Shared Resources
  - Exchange Calendar
  - Email Lists
GATHERING INFO & ASSIGNING WORK

- Used registry info to prioritize applications.
- Generated a risk rating based on data sensitivity and criticality.
- Split load for analysts based on number of apps per department.
- 1 primary, 1 backup analyst per team.
- Each department only had to work with one analyst team:
  - provided continuity
  - prevented contact exhaustion
Assessment: Initial Meeting

- Met with each development team for a group of apps.
- Agenda for each meeting:
  - Explained why (in case they didn't get the memo)
  - Explained the process.
- Asked for authorizations:
  - Role with full access.
  - Role with limited access.
ASSESSMENT: THE ENVIRONMENT

- Chose to scan in our QA environment.

- QA typically had:
  - Full set of QA data (no production data modified).
  - Code similar to what was in production.

- Found some developers were not using QA or test environments. This became a finding.
ASSessment: Tools

- Primarily used an automated web application vulnerability scanner (IBM AppScan).
- Followed-up with manual testing using tools such as:
  - Proxies (i.e. WebScarab, Ratproxy, etc..)
  - Attack frameworks (i.e. w3af)
  - Request Intercepts (i.e. Tamper Data, etc.)
- These tools are available in the Samurai WTF and OWASP LiveCDs
- Manual testing weeded out false positives and found flaws the scanner missed.
Remediation: What We Found

The usual culprits on the OWASP Top 10.

(Can't give you more details than that.)

Same vulnerabilities cropped up in most applications.
Remediation: Reports

- Provided developers two reports:
  - Basic: Brief overview of findings and URLs.
  - Detailed: Findings, URLs, fields, and remediation.
- They appreciated having basic for ease-of-reading and detailed for reference.
- Manual findings were included in separate reports and sometimes screen casts.
We were lenient unless the findings were critical.

Critical = Very Sensitive Data + Severe Flaw

Required immediate remediation of critical vulnerabilities.

Developers generally good at estimating remediation time and not taking advantage of leniency.
REPORT: DELIVERABLES

• Individual reports given to units.

• High-Level Application Security Posture Report given to management and governance groups.

• General overview presentation given to developers and technical managers.
Report: The Extra Mile

- Additional resources, explanations, and presentations where needed.
- Code walkthroughs of remediated code, follow-up scans to verify fixes.
- Whatever it took, short of coding ourselves, to get the message across.
Found ways common vulnerabilities could be addressed by infrastructure solutions, saving developer time.

Policy changes:

- Required use of existing infrastructure solutions.
- Required apps to be entered into a central application registry.
- Moving away from older development frameworks.
Report: Other Benefits

- Trust between the ISO and the developers.
  They're now coming to us more before deploying apps.

- Trust between ISO and systems staff.
  We were willing to work with them to minimize impact on the servers and network.
Report: Lessons Learned

- Analysts experienced assessment fatigue; especially with so many similar findings.
  ✓ Next time: Less broad, go deeper.

- Application registry was not always up-to-date, causing further delays.
  ✓ Providing new and improved Application Registry.
  ✓ Application Registry will provide better tools to ensure compliance.

- These lessons can be applied to future large-scale assessments.
Questions?