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

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

36

Departments

Our Challenge

283

Applications

Our Challenge

18

Months

Our Challenge

140

Developers &
Business Managers

Our Challenge

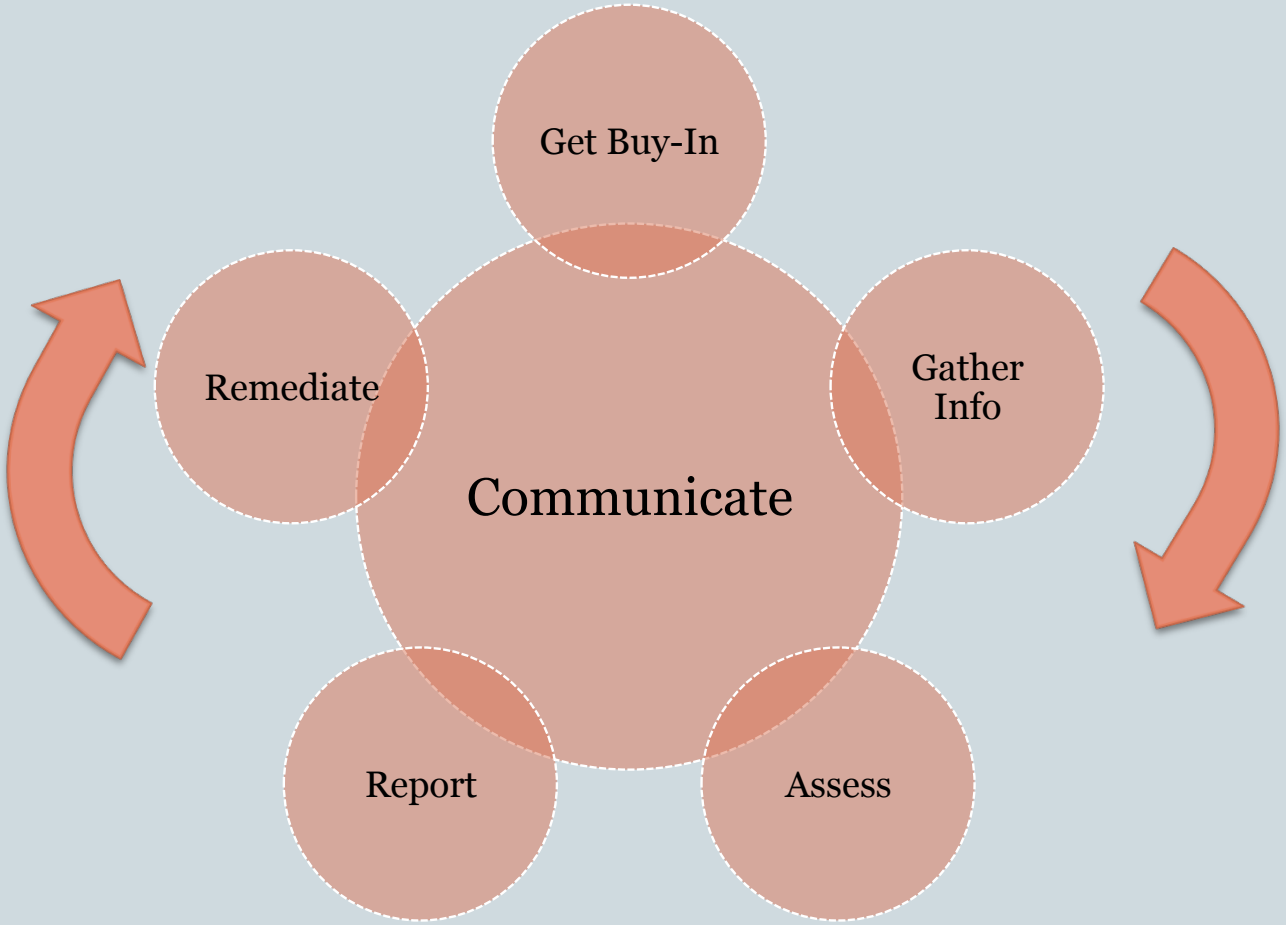
4

Analysts

Our Challenge

Total Departments:	36
Web Applications Assessed:	283
Timeframe:	18 months
Developers & Managers:	140
ISO Resources:	4 analysts

Our Methodology



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.

Remediation: Timeframe

- 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.
- What ever it took, short of coding ourselves, to get the message across.

Report: Changes Affected

- 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?