Secure By Design: Security in the Software Development Lifecycle

Twin Cities Rational User’s Group
Security Briefing by Arctec Group
(www.arctecgroup.net)
Integrating Security into Software Development

• Focus Areas
  – Collaborative Approach
  – Security Goals
  – Leveraging Existing Artifacts and Processes
  – Security-centric artifacts
Software Development Realities

• Blame game: why is software insecure?
Software Development: The Way Forward

- Build a shared understanding of risk & risk management
Security Goals

• Confidentiality
  – “Concealment Information and resources”

• Integrity
  – “Trustworthiness of data and resources”

• Availability
  – “Availability of information or resources”
  – *definitions from Computer Security Art and Science, By Matt Bishop, Addison Wesley*
Security & Security Mechanisms

• Protection - prevent the attack from succeeding
• Detection - detect abuse or malicious use
• Response - recover from attack
• Understand role of policy
• Understand trust assumptions
…but why do we need application security?

- Firewall/DMZ security model does not reflect current threat model or deal with threats emerging from web application, thin client, and web services application
- Current paradigm of front of front door protection does not protect against malicious insider
- 93% of vulnerabilities occur at the application level (source: ICAT)
...but why do we need application security?

<table>
<thead>
<tr>
<th></th>
<th>Highest Reported ($US)</th>
<th>Average Losses ($US)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of Service</td>
<td>60,000,000</td>
<td>1,427,000</td>
</tr>
<tr>
<td>Theft of Proprietary Information</td>
<td>35,000,000</td>
<td>2,700,000</td>
</tr>
<tr>
<td>Insider abuse</td>
<td>6,000,000</td>
<td>135,000</td>
</tr>
<tr>
<td>Viruses and Worms</td>
<td>6,000,000</td>
<td>200,000</td>
</tr>
<tr>
<td>Financial fraud</td>
<td>4,000,000</td>
<td>329,000</td>
</tr>
<tr>
<td>Sabotage</td>
<td>2,000,000</td>
<td>215,000</td>
</tr>
</tbody>
</table>

* Excerpted from CSI/FBI Report 2003 - poll of 530 US based corporations, government, and educational institutions
Security in Inception

"A problem, properly stated, is a problem on its way to being solved," Buckminster Fuller
Security in Inception

• Requirements
  – Functional versus non-functional requirements
  – Usability & Security & related ilities

• Risk Analysis
  – Asset classification

• Data Classification
  – Classify entities to drive out security model for access control and related controls

• Domain Glossary of Security Terms
# Data Classification Example

<table>
<thead>
<tr>
<th></th>
<th>Public</th>
<th>Private</th>
<th>Confdl</th>
<th>Site Config</th>
</tr>
</thead>
<tbody>
<tr>
<td>Everyone</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current Customers</td>
<td>R</td>
<td>C, R, U</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Administrators</td>
<td></td>
<td></td>
<td></td>
<td>R, U, D</td>
</tr>
</tbody>
</table>
Security in Inception

• Security Use Case Modeling
  – Show behavioral flows
  – Understand/analyze/vet security implications of pre and post conditions
  – Understand/analyze/vet exceptional flows
  – Useful to synthesize authorization structure
Security Use Case Modeling

• Modeling Identity
  – Identity forms the basis of many security mechanisms, including access control (authentication, authorization)
  – However identity is not an entity, but the result of a process
  – Use Use Case modeling to drive out the relationships and dependencies to establish identity in a system
  – SSO Federated Identity example: User logs on to local system in manufacturer policy domain, message is sent to supplier domain, message includes client user information & manufacturer domain security information; supplier system authenticates request
Security in Elaboration

• Elaborating Use Cases
  – Using relationships to model security
    • Includes relationships can illustrate many protection mechanisms, e.g. logon process
    • Extends relationships can illustrate many detection mechanisms, e.g. audit logging
Security in Elaboration

• Abuse Cases
  – Look at the system from an attacker point of view
  – Useful to glean/verify security requirements, create threat models
  – Discussed in detail paper by Guttorm Sindre and Andreas Opdahl.

Misuse Case Format

• “A misuse case is the inverse of a use case, i.e. A function that the system should not allow” -Sindre & Opdahl

• “A mis-actor is the inverse of an actor, i.e., an actor that one does not want the system to support, an actor who initiates misuse cases.” - Sindre & Opdahl

• Additional elements
  – Worst Case Threat: end system state if Misuse succeeds
  – Prevention and Detection Guarantees: these guarantees closely resemble a Use Case Post-condition, but encapsulate security-specific concepts of prevention and detection.
  – Stakeholders and Risks: this field gives the security team a place to address what the business risk that is generated by the application.
Misuse Case Example
Security in Elaboration

- Implement Saltzer and Schroeder’s Principles*
  - Principle of Least Privilege
  - Principle of Fail Safe Defaults
  - Principle of Economy of Mechanism
  - Principle of Complete Mediation
  - Principle of Open Design
  - Principle of Separation of Privilege
  - Principle of Least Common Mechanism
  - Principle of Psychological Acceptability

Security in Elaboration

• Threat Modeling
  – Howard and Leblanc’s STRIDE and DREAD
  – Identification
    • STRIDE
      – Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege
  – Prioritization
    • DREAD
      – Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability
Using Threat Models

• Threat Models consist of:
  – Entry points
  – Assets
  – Threats
  – Dependencies
Using Threat Models

• Order Books Web Service example Use Case
  – Entry point: web service is available via network, e.g. firewall does not block access
  – Assets: customer number, credit card, bookstore brand and trust
  – Threats
    • Information disclosure: attacker could gain access to Web Service
    • Denial of Service: attacker could effect accessibility of site
    • Tampering & Elevation of privilege: attacker could use vulnerability in service tamper with data or elevate privilege
Security in Construction

• Test cases
  – Use threat models and abuse cases to drive test cases - the same as use cases drive testing for functional requirements

• Logging & monitoring
  – Assess quality of logging and reporting mechanisms during development

• Code Review
  – Peer review
  – Source Code Analysis Tools
  – Scanning tools
Security in Transition

• Security baseline configuration
  – Utilize baseline configurations from vendors and best practice guides
• Secure builds - separation of privileges
• Incident response planning
• Security Metrics
The Way Forward

- Security in SDLC is still relatively new
- Aim for incremental improvement
- Aim for reuse
- Educate/mentor/co-evolve
- Manage risk:
  “We have no future because our present is too volatile. We have only risk management. The spinning of the given moment's scenarios. Pattern recognition…”
  -William Gibson “Pattern Recognition”