
Jie Ren, Richard Taylor, Paul Dourish, David Redmiles
Institute for Software Research
University of California, Irvine

Software Engineering for Secure Systems
May 15, 2005
Outline

❖ Background and Insight
  – Architecture and Security
❖ Approach
  – Connector-centric
❖ Case Study
  – A not-so-small example application
❖ Conclusion and future work
Main Goal

- Integrate security and software architecture
  - Integrate
  - Architecture level
  - Security: confidentiality, integrity, availability
  - Modern software: componentized, networked, heterogeneous
A Tale of IIS
### The history of IIS

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>OS</th>
<th># Security Bulletins</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1995</td>
<td>NT 3.51</td>
<td></td>
</tr>
<tr>
<td>2.0</td>
<td>1996</td>
<td>NT 4</td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>1997</td>
<td>NT 4</td>
<td>8</td>
</tr>
<tr>
<td>4.0</td>
<td>1997</td>
<td>NT 4</td>
<td>42</td>
</tr>
<tr>
<td>5.0</td>
<td>2000</td>
<td>2000</td>
<td>29</td>
</tr>
<tr>
<td>5.1</td>
<td>2001</td>
<td>XP</td>
<td>5</td>
</tr>
<tr>
<td>6.0</td>
<td>2003</td>
<td>2003</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 1. Secure by design.

<table>
<thead>
<tr>
<th>POTENTIAL PROBLEM</th>
<th>PROTECTION MECHANISM</th>
<th>DESIGN PRINCIPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>The underlying dll (ntdll.dll) was not vulnerable because...</td>
<td>Code was made more conservative during the Security Push.</td>
<td>Check precondition</td>
</tr>
<tr>
<td>Even if it were vulnerable...</td>
<td>Internet Information Services (IIS) 6.0 is not running by default on Windows Server 2003.</td>
<td>Secure by default</td>
</tr>
<tr>
<td>Even if it were running...</td>
<td>IIS 6.0 does not have WebDAV enabled by default.</td>
<td>Secure by default</td>
</tr>
<tr>
<td>Even if Web-based Distributed Authoring and Versioning (WebDAV) had been enabled...</td>
<td>The maximum URL length in IIS 6.0 is 16 Kbytes by default ( &gt; 64 Kbytes needed for the exploit)</td>
<td>Tighten precondition, secure by default</td>
</tr>
<tr>
<td>Even if the buffer were large enough...</td>
<td>The process halts rather than executes malicious code due to buffer-overflow detection code inserted by the compiler.</td>
<td>Tighten postcondition, check precondition</td>
</tr>
<tr>
<td>Even if there were an exploitable buffer overrun...</td>
<td>It would have occurred in w3wp.exe, which is running as a network service (rather than as administrator).</td>
<td>Least privilege</td>
</tr>
</tbody>
</table>

(Data courtesy of David Aucsmith.)


Traditional SA

- Component-based Software Engineering
- Software Architecture
  - Structure
  - Behavior
    - Process Algebra (Wright), Labeled Transition System (Darwin)
Connectors

- Should they be first class citizens?
  - Capture and reuse
- Existing work
  - Taxonomy: Mehta 2000
  - Assembly Language: Mehta 2004
  - Constructions: Lopes 2003
  - Transformation: Spitznagel 2001
- No rich security
  - Dependability: Spitznagel 2004
Our Approach

- Describe and Enforce Architectural Security
  - Extend xADL
  - Security Models, Users ( Principals ), Privileges, Trusts, Contexts
  - Component: supply security contract
  - Connector: regulate and enforce contract
Extensible xADL

- xADL provides structural infrastructure and enables extension
- Security Models Support
  - Extensible and Neutral
  - Classic Access Control, Role-based Access Control, Trust Management
Security Constructs

- Users (Principals)
  - Single principal, determinable at design time, no impact on architecture

- Privileges
  - Traditional access control privileges
  - Architectural privileges: change, inspect

- Contexts

- Trust
Component and Connector

* Component
  - Supply “security contract”
  - Requires and Provides

* Connector
  - Regulate and enforce “security contract”
  - Determine components’ principals
  - Decide compatibility
  - Adapt incompatibilities and impose security
  - Derivation, composition, replacement
Case Study: Impromptu

- An ad-hoc peer-to-peer file sharing application for a workgroup
  - User-centered context
  - Users make decisions
  - Visualization facilitates making decisions
  - Perception, decision, and action

- Security Goals:
  - Make security visible
  - Ease security configuration
Impromptu UI
Impromptu Architecture
Secure Connector

<connectorType
type="ConnectorType"
id="SecureWebDAVConnector">
  <signature id="WebDAVClient">
  </signature>
  <signature id="WebDAVServer">
  </signature>
  <description>
    IP-based authentication
    Method-based authorization
  </description>
</connectorType>
Connecting Components

<component type="ProxyType" id="Local">
  <principal>Me</principal>
</component>

<component type="ProxyType" id="Remote">
  <principal>Other</principal>
</component>

<connector type="SecureWebDAVConnector" id="Impromptu_Impromptu">
  <interface signature="WebDAVClient" id="Remote"/>
  <interface signature="WebDAVServer" id="Local"/>
</connector>
Enhanced Secure Connectors

<connectorType id="DigestAuthenticationConnector"/>
</connectorType>
<connectorType id="WebXMLAuthorizationConnector"/>
</connectorType>
<connectorType id="WebDAVACLConnector"/>
</connectorType>
<connectorType id="SecureWebDAVConnector">
  <subArchitecture>
    <sequence>
      <connector type="DigestAuthenticationConnector"/>
      <connector type="WebXMLAuthorizationConnector"/>
      <connector type="WebDAVACLConnector"/>
    </sequence>
  </subArchitecture>
</connectorType>
Conclusion

- Background and Insight
  - Combine security and software architecture
  - Connector-centric

- Approach
  - Extend xADL with security constructs
  - Users ( Principals), Privileges, Trusts, Contexts

- Case Study
  - Secure WebDAV Connector

- Future work
  - Language Semantics
  - Tool Support