Motivation

• We started by trying to do something simple: Increase our confidence in the security of some critical grid middleware
• We ended up developing a new manual methodology: First Principles Vulnerability Assessment (FPVA)
• We found some serious vulnerabilities … and more vulnerabilities … and more.
First Principles Vulnerability Assessment

• Manual assessment process – analyst centric
• Insider – have access to
  – Developers
  – Source code
  – Documentation
• Independent from development team
  – No agenda
  – No blinders
• First Principles – let the process guide the search

FPVA: 4 Step Process

1. Architectural Analysis
2. Resource Analysis
3. Trust and Privilege Analysis
4. Component Analysis

Post-FPVA Activities:
• Disseminate vulnerability reports to developers with suggested remediation
• Council developers about fix, disclosure and security release process
**FPVA: Steps 1 - 3**

*Understanding the System*

1. **Architectural Analysis** – functionality and structure of the system, major components, communication channels
2. **Resource Analysis** – Objects in the system and allowed operations
3. **Trust and Privilege Analysis** – trust boundaries of components, privilege model presented to users, and external privilege systems used

---

**Condor Job Submission**

[Diagram of Condor Job Submission process]
FPVA: Step 4
Searching for Vulnerabilities

• Connect user supplied data to security violation of a resource
• Audit the source code
• Guide search using
  – Previous analyses and diagrams
  – Knowledge of how vulnerabilities arise
    • Dangerous functions
    • Dangerous idioms

Systems Assessed

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Vulnerabilities</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Condor</strong>, University of Wisconsin</td>
<td>Batch queuing workload management system</td>
<td>15</td>
<td>600 KLOC of C and C++</td>
</tr>
<tr>
<td><strong>SRB</strong>, SDSC</td>
<td>Storage Resource Broker - data grid</td>
<td>5</td>
<td>280 KLOC of C</td>
</tr>
<tr>
<td><strong>MyProxy</strong>, NCSA</td>
<td>Credential Management System</td>
<td>5</td>
<td>25 KLOC of C</td>
</tr>
<tr>
<td><strong>gLExec</strong>, Nikhef</td>
<td>Identity mapping service</td>
<td>5</td>
<td>48 KLOC of C</td>
</tr>
<tr>
<td><strong>Gratia Condor Probe</strong>, FNAL and Open Science Grid</td>
<td>Feeds Condor Usage into Gratia Accounting System</td>
<td>3</td>
<td>1.7 KLOC of Perl and Bash</td>
</tr>
</tbody>
</table>
Systems Assessed (cont.)

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Vulnerabilities</th>
<th>KLOC of Language(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Condor Quill</strong>, University of Wisconsin</td>
<td>DBMS Storage of Condor Operational and Historical Data</td>
<td>6</td>
<td>7.9</td>
</tr>
<tr>
<td><strong>Condor Privilege Separation</strong>, University of Wisconsin</td>
<td>Restricted Identity Switching Module</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td><strong>VOMS Admin</strong>, INFN</td>
<td>Web management interface to VOMS data (role mgmt)</td>
<td>4</td>
<td>35</td>
</tr>
<tr>
<td><strong>CrossBroker</strong>, Universitat Autònoma de Barcelona</td>
<td>Resource Manager for Parallel and Interactive Applications</td>
<td>2</td>
<td>97</td>
</tr>
</tbody>
</table>

Improving FPVA

FPVA requires a costly asset: a skilled security assessor

Can existing tools reduce the cost of manual assessment?

What can the tools find?
Case Study
Goal is to study the best tools out there. Apply them to a system we studied. Use our results as a ground truth.

• Talked to academics, military, and industry people about what they thought were the best tools:
  – Coverity Prevent
  – Fortify SCA
• Review tool output
  – Defect with matching location of known vulnerability is a positive result
  – Sample tool output to understand results

Ground Truth:
FPVA Condor Results
15 significant vulnerabilities discovered
http://www.cs.wisc.edu/condor/security/vulnerabilities
– 7 implementation bugs
  • easy to discover - localized in code
  • use of troublesome functions:
    exec, popen, system, strcpy, tmpnam
– 8 design flaws
  • hard to discover in code - higher order problems
  • defects include:
    – injections, directory traversals, file permissions, authorization & authentication, and a vulnerability in third party library
**Results of Static Analysis Tool Study of FPVA Vulnerabilities in Condor**

<table>
<thead>
<tr>
<th></th>
<th>Coverity</th>
<th>Fortify SCA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Defect Reports:</strong></td>
<td>2,986</td>
<td>15,466</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total</td>
</tr>
<tr>
<td>critical</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>hot</td>
<td>2,301</td>
<td></td>
</tr>
<tr>
<td>warm</td>
<td>8,101</td>
<td></td>
</tr>
<tr>
<td>info</td>
<td>5,061</td>
<td></td>
</tr>
<tr>
<td><strong>Defect Categories:</strong></td>
<td>70</td>
<td>45</td>
</tr>
<tr>
<td><strong>FPVA Vulnerabilities Found:</strong></td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total</td>
</tr>
<tr>
<td>impl. bug</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>design flaw</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Tools: The Good and the Bad**

**Good:**
- Easy to use
- Finds some simple implementation security problems
- Finds many minor security problems such as resource leaks
- Finds questionable programming practices

**Bad:**
- Reports false defects - False Positive problem (large number is overwhelming)
- Misses real vulnerabilities - False Negative problem
- Requires skilled operator to understand output
Research Directions

• Can we automate the discovery of some of the vulnerabilities not found by current tools?
• Can we automate some of the architecture, resource, and trust and privilege analyses using static or run-time analysis of the system?

Questions

For more information see:

http://www.cs.wisc.edu/mist