SATE V Workshop
CVE-selected Analysis Results

Bertrand STIVALET, NIST
stivalet [at] nist [dot] gov
March 14, 2014

The SAMATE Project
http://samate.nist.gov
Outline

• Test sets

• CVE*-based analysis
  – Procedure
  – Observations

• Subset warnings analysis
  – Procedure
  – Observations

*CVE: Common Vulnerabilities and Exposures
Test sets

- **PHP language:**

<table>
<thead>
<tr>
<th>Software</th>
<th>Vulnerable version</th>
<th>Fixed version</th>
<th>Lines of code</th>
<th>Participating tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordpress</td>
<td>Blogging platform</td>
<td>2</td>
<td>2.2.3</td>
<td>&lt; 25k</td>
</tr>
</tbody>
</table>

Certain instruments, software, materials, and organizations are identified in this paper to specify the exposition adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the instruments, software, or materials are necessarily the best available for the purpose.
## Test sets

- **C/C++ language:**

<table>
<thead>
<tr>
<th>Software</th>
<th>Vulnerable version</th>
<th>Fixed version</th>
<th>Lines of code</th>
<th>Participating tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asterisk PBX</td>
<td>10.2.0</td>
<td>10.12.2</td>
<td>&gt; 500k</td>
<td>8</td>
</tr>
<tr>
<td>Asterisk PBX</td>
<td>10.2.0</td>
<td>10.12.2</td>
<td>&gt; 500k</td>
<td>8</td>
</tr>
<tr>
<td>Wireshark Traffic</td>
<td>1.8.0</td>
<td>1.8.7</td>
<td>&gt; 2M</td>
<td>9</td>
</tr>
<tr>
<td>Wireshark Traffic</td>
<td>1.8.0</td>
<td>1.8.7</td>
<td>&gt; 2M</td>
<td>9</td>
</tr>
</tbody>
</table>

### Table Notes:
- **Asterisk PBX Platform**
- **Wireshark Traffic Analyzer**
## Test sets

- **JAVA language:**

<table>
<thead>
<tr>
<th>Software</th>
<th>Vulnerable version</th>
<th>Fixed version</th>
<th>Lines of code</th>
<th>Participating tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSPWiki Wiki</td>
<td>2.5.124</td>
<td>2.5.139</td>
<td>&gt; 60k</td>
<td>6</td>
</tr>
<tr>
<td>Openfire Groupchat server</td>
<td>3.6.0</td>
<td>3.6.4</td>
<td>&gt; 200k</td>
<td>6</td>
</tr>
</tbody>
</table>
Analysis procedure for CVE-selected test cases

Tool warnings ~1.3M → CVE based → Warning subset

Methods:

CVE based

Selected warnings → Analyze for correctness and associate

Analyze the data
Analysis procedure for CVE-selected test cases

Methods:

- Tool warnings ~1.3M
- CVE based
- Warning subset
- Selected warnings
- Analyze for correctness and associate
- Analyze the data
CVE-based analysis

• CVE: Common Vulnerabilities and Exposures

• Focused on real-life exploitable vulnerabilities

• 73 CVEs in the 5 test cases
  – Identify source, sink or path locations
  – Match to tool warnings
CVE-based observations

• CVEs found by tools:
  - Total CVEs: 73
  - CVEs found: 40 (45%)
  - CVEs not found: 33 (55%)

• CWEs* found by tools:
  - Total CWEs: 35
  - CWEs found: 16 (46%)
  - CWEs not found: 19 (54%)

*CWE: Common Weakness Enumeration
CVE-based observations

Top 5 CWEs present in CVEs
- Uncontrolled Memory: 4
- Improper Neutralization: 5
- Improper Restriction: 7
- Signed to Unsigned: 17
- Integer Overflow: 4

- Top 5 CWEs cover 37 of 73 CVEs

Top 5 CWEs found by tools
- Signed to Unsigned: 4
- Integer Overflow: 7
- Uncontrolled Memory: 8
- Improper Restriction: 9
- Improper Neutralization: 37

- “Infinite loop” weakness has only 1 hit.
Analysis procedure for CVE-selected test cases

Tool warnings ~1.3M

CVE based

Warning subset

Selected warnings

Methods:

Analyze for correctness and associate

Analyze the data
Warning subset analysis

• Helps understand what weaknesses are found by tools in real-world software

• 30 warnings from each tool report statistically selected => 900 warnings!

• Analyzers:
  – Aure
  – Bertrand
  – Charles
  – Kamillia
  – Paul
  – Sean (volunteer)
  – Vadim
  – Yan
Decision process

Correctness categories:

• True security weakness: a weakness relevant to security

• True quality weakness: poor code quality

• True but insignificant claim.

• Not a weakness – false: invalid conclusion about the code

• Weakness status unknown
Wordpress precision analysis

\[
\text{Precision} = \frac{\text{#Useful Warnings}}{\text{#All Warnings}}
\]

\[
= \frac{q + s}{f + i + q + s}
\]

\[
= \frac{6 + 9}{10 + 5 + 6 + 9}
\]

\[
= 50\%
\]

false: Useless warnings
insignificant: Insignificant warnings
quality: Quality warnings
security: Security warnings
Asterisk + Wireshark precision analysis

Average precision: 31.8%
JSPWiki + Openfire precision analysis

Java Testcases
Average precision: **60.8%**
Top 5 CWE groups

- Top CWE groups reported in **security** and **quality** weaknesses

<table>
<thead>
<tr>
<th>CWE GROUP - C</th>
<th>OCCURENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer operation</td>
<td>82</td>
</tr>
<tr>
<td><strong>Code quality</strong></td>
<td>69</td>
</tr>
<tr>
<td>Denial of Service</td>
<td>46</td>
</tr>
<tr>
<td><strong>Design and implementation</strong></td>
<td>38</td>
</tr>
<tr>
<td>Memory leak</td>
<td>38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CWE GROUP - JAVA</th>
<th>OCCURENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input validation</strong></td>
<td>165</td>
</tr>
<tr>
<td><strong>Code quality</strong></td>
<td>146</td>
</tr>
<tr>
<td><strong>API</strong></td>
<td>109</td>
</tr>
<tr>
<td><strong>Design and implementation</strong></td>
<td>90</td>
</tr>
<tr>
<td>Concurrency</td>
<td>83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CWE GROUP - PHP</th>
<th>OCCURENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web</strong></td>
<td>13</td>
</tr>
<tr>
<td><strong>Input validation</strong></td>
<td>6</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>5</td>
</tr>
<tr>
<td>Dynamic code</td>
<td>3</td>
</tr>
<tr>
<td><strong>API</strong></td>
<td>2</td>
</tr>
</tbody>
</table>

Overlapping:
Summary

• Finalize our results analysis

• Precision varies a lot by tool and weakness category

• Some CWEs are still difficult to find
WEIRD — MY CODE'S CRASHING WHEN GIVEN PRE-1970 DATES.
Selection procedure

• We selected 30 warnings from each tool report (except one report, which had fewer than 30 warnings) using the following procedure. Here, a warning class is a (weakness name, severity), e.g., (Buffer Underrun, 1) pair. For one of the tools we could not use weakness names to identify warning classes, so we used a (CWE id, severity) pair instead.

• The procedure is similar to that used in previous SATEs. The main difference is that some tool reports had more than 30 warning classes with severities 1 through 4, so we chose a random subset of warning classes.

• While more warnings are needed, repeat for severity S (where S is from 1 through 4):
  -- Randomly select one warning from each warning class (or a randomly selected subset of warning classes if there are more warning classes left than warnings needed) identified by a warning name (or by CWE id) with severity S.
Selection procedure

• While more warnings are needed, repeat:
  -- Randomly select 3 of the remaining warnings (or all remaining warnings if there are less than 3 left) from each warning class with severity 1,
  -- Randomly select 2 of the remaining warnings (or all remaining warnings if there are less than 2 left) from each warning class with severity 2,
  -- Randomly select 1 of the remaining warnings from each warning class (if it still has any warnings left) with severity 3.

• If more warnings are still needed, select warnings from warning class with severity 4, then select warnings from warning class with severity 5.

If a tool did not assign severity, we assigned severity based on weakness names and our understanding of their relevance to security.

We excluded from the selection process the warnings that refer exclusively to test code, parser generator code, or external header files.