Rapid Expansion of Classification Models to Prioritize Static Analysis Alerts for C

Lori Flynn, PhD
Software Security Researcher
Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.


DM17-0790
Overview

Problem: too many alerts
Solution: automate handling

Project Goal
Classification algorithm development using “pre-audited” and manually-audited data, that
accurately classifies most of the diagnostics as:
- Expected True Positive (e-TP)
- Expected False Positive (e-FP),
- and the rest as Indeterminate (I)

Scientific Approach

Build on novel (in FY16) combined use of:
1) multiple analyzers, 2) variety of features,
3) competing classification techniques!

Problem: too many alerts
Solution: automate handling

Competing Classifiers to Test
- Lasso Logistic Regression
- CART (Classification and Regression Trees)
- Random Forest
- Extreme Gradient Boosting (XGBoost)

Some of the features used (many more)
Analysis tools used
- Significant LOC
- Complexity
- Coupling
- Cohesion
- SEI coding rule

Per-rule alert classifiers
Classifiers for all alerts
Rapid Expansion of Alert Classification

Problem 2
Too few manually audited alerts to make classifiers (i.e., to automate!)

Problems 1 & 2: Security-related code flaws detected by static analysis require too much manual effort to triage, plus it takes too long to audit enough alerts to develop classifiers to automate the triage.

Extension of our FY16 alert classification work to address challenges:
1. Too few audited alerts for accurate classifiers
2. Manually auditing alerts is expensive

Solution 2
Automate auditing alerts, using test suites

Solution for 1 & 2: Rapid expansion of number of classification models by using “pre-audited” code, plus collaborator audits of DoD code.

Approach
1. Automated analysis of “pre-audited” (not by SEI) tests to gather sufficient code & alert feature info for classifiers
2. Collaboration with MITRE: Systematically map CERT rules to CWE IDs in subsets of “pre-audited” test code (known true or false for CWE)
3. Modify SCALe research tool to integrate CWE (MITRE’s Common Weakness Enumeration)
4. Test classifiers on alerts from real-world code: DoD data

Problem 1: too many alerts
Solution 1: automate handling
Overview: Method, Approach, Validity

Problem 2: too few manually audited alerts to make classifiers (i.e., to automate)
Solution 2: automate auditing alerts, using test suites

Rapidly create many coding-rule-level classifiers for static analysis alerts, then use DoD-audited data to validate the classifiers.

Technical methods:
- Use test suites’ CWE flaw metadata, to quickly and automatically generate many “audited” alerts.
  - Juliet (NSA CAS) 61,387 C/C++ tests
  - IARPA’s STONESOUPO: 4,582 C tests
  - Refine test sets for rules: use mappings, metadata, static analyses
- Metrics analyses of test suite code, to get feature data
- Use DoD-collaborator enhanced-SCALe audits of their own codebases, to validate classifiers. Real codebases with more complex structure than most pre-audited code.
Make Mappings Precise

Problem 3: Test suites in different taxonomies (most use CWEs)
Solution 3: Precisely map between taxonomies, then partition tests using precise mappings


Imprecise mappings ("some relationship") \rightarrow Precise mappings (set notation, often more)

<table>
<thead>
<tr>
<th>Mappings</th>
<th>Precise</th>
<th>Imprecise TODO</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 CWEs subset of CERT rule, AND partial overlap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CERT Rule c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWE Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWE Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CWE N</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now: all CERT C rules mappings to CWE precise

If a condition of a program violates a CERT rule R and also exhibits a CWE weakness W, that condition is in the overlap.
Test Suite Cross-Taxonomy Use

Partition sets of thousands of tests relatively quickly.
Examine together:
- Precise mapping
- Test suite metadata (structured filenames)
- Rarely examine small bit of code (variable type)

CWE test programs useful to test CERT rules
STONESOUP: 2,608 tests
Juliet: 80,158 tests
  • Test set partitioning incomplete (32% left)

Some types of CERT rule violations not tested, in partitioned test suites (“0”s).
  - Possible coverage in other suites

Problem 3: Test suites in different taxonomies (most use CWEs)
Solution 3: Precisely map between taxonomies, then partition tests with precise mappings

<table>
<thead>
<tr>
<th>CERT rule</th>
<th>CWE</th>
<th>Count files that match</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARR38-C</td>
<td>CWE-119</td>
<td>0</td>
</tr>
<tr>
<td>ARR38-C</td>
<td>CWE-121</td>
<td>6,258</td>
</tr>
<tr>
<td>ARR38-C</td>
<td>CWE-122</td>
<td>2,624</td>
</tr>
<tr>
<td>ARR38-C</td>
<td>CWE-123</td>
<td>0</td>
</tr>
<tr>
<td>ARR38-C</td>
<td>CWE-125</td>
<td>0</td>
</tr>
<tr>
<td>ARR38-C</td>
<td>CWE-805</td>
<td>2,624</td>
</tr>
<tr>
<td>INT30-C</td>
<td>CWE-190</td>
<td>1,548</td>
</tr>
<tr>
<td>INT30-C</td>
<td>CWE-191</td>
<td>1,548</td>
</tr>
<tr>
<td>INT30-C</td>
<td>CWE-680</td>
<td>984</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-119</td>
<td>0</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-125</td>
<td>0</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-129</td>
<td>0</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-131</td>
<td>0</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-190</td>
<td>3,875</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-191</td>
<td>3,875</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-20</td>
<td>0</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-606</td>
<td>0</td>
</tr>
<tr>
<td>INT32-C</td>
<td>CWE-680</td>
<td>984</td>
</tr>
</tbody>
</table>
Generate data for Juliet

Generate data for STONESOUP

Write classifier development and testing scripts

Build classifiers
  • Directly for CWEs
  • Using partitioned test suite data for CERT rules

Test classifiers

Problem 1: too many alerts
Solution 1: automate handling

Problem 2: too few manually audited alerts to make classifiers
Solution 2: automate auditing alerts, using test suites

Problem 3: Test suites in different taxonomies (most use CWEs)
Solution 3: Precisely map between taxonomies, then partition tests using precise mappings
Using CWE Test Suites for Multi-Taxonomy Classifiers

One time, develop data for classifiers. Per rule or CWE classifier, filter data.

Test suite
(e.g., Juliet test programs for 118 CWEs)

CWE190
Program1a
Program2a
Program3a
Program4a
Program5a
Program6a
Program7a
... etc.

CWE191
Program1b
Program2b
Program3b
Program4b
Program5b
Program6b
Program7b
... etc.

CWE192
Program1c
Program2c
Program3c
Program4c
Program5c
Program6c
Program7c
... etc.

... etc.
(for other CWEs)

Run static analysis tools on all tests

Static Analysis Tools

Alerts

Alert Consolidation

Potential Rule Violations

Determinations

Automated “auditing” using test suite metadata

Filter for data subset

ML Classifier Development for RULE

Data for classifier training and testing

Identify applicable tests for each CWE mapped to rule of interest

Using precise mapping,

Tests applicable to particular CERT rule

CWE190
Program1a
Program2a
Program3a
Program5a
... etc.

CWE191
Program3b
Program6b
... etc.

CWE192
Program1c
Program2c
Program3c
Program4c
Program5c
Program6c
Program7c
... etc.

Using test suites for multi-taxonomy classifiers
Analysis of Juliet Test Suite: Initial CWE Results

- We automated defect identification of Juliet flaws with location 2 ways
  - A Juliet program tells about only one type of CWE
  - Bad functions definitely have that flaw
  - Good functions definitely don’t have that flaw
  - Function line spans, for FPs
  - Exact line defect metadata, for TPs
- Used static analysis tools on Juliet programs
- We automated alert-to-defect matching
  - Ignore unrelated alerts (other CWEs) for program
  - Alerts give line number
- We automated alert-to-alert matching (alerts fused: same line & CWE)

Lots of new data for creating classifiers!

- These are initial metrics (more EC as use more tools, STONESOUP)

<table>
<thead>
<tr>
<th>Alert Type</th>
<th>Equivalence Classes: (EC counts a fused alert once)</th>
<th>Number of Alerts Fused (from different tools)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td></td>
<td>22,885</td>
</tr>
<tr>
<td>FALSE</td>
<td></td>
<td>29,507</td>
</tr>
</tbody>
</table>

Number of “Bad” Functions | 103,376
Number of “Good” Functions | 231,476

<table>
<thead>
<tr>
<th>“Pre-audited”</th>
<th>Tool A</th>
<th>Tool B</th>
<th>Tool C</th>
<th>Tool D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>1,655</td>
<td>162</td>
<td>7,225</td>
<td>16,958</td>
<td>26,000</td>
</tr>
<tr>
<td>FALSE</td>
<td>8,539</td>
<td>3,279</td>
<td>2,394</td>
<td>23,475</td>
<td>37,687</td>
</tr>
</tbody>
</table>
Juliet: Data from 4 Tools, per CWE

35 CWEs with **at least** 5 HCFPs and 45 HCTPs

More data to be added
- Tools
- STONESOUP

Classifier development requires **True** and **False**

Successfully generated lots of data for classifiers

**The 35 CWEs**
- 457 • 680 • 252 • 843 • 483
- 195 • 404 • 369 • 377 • 126
- 197 • 415 • 606 • 398 • 835
- 134 • 665 • 122 • 196
- 758 • 191 • 121 • 468
- 194 • 761 • 681 • 469
- 190 • 127 • 476 • 688
- 401 • 563 • 775 • 587
### Classifiers: Accuracy, #Alerts, AUROC

#### Lasso per-CERT-rule classifiers (36)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Accuracy</th>
<th># Alerts</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARR30-C</td>
<td>96.9%</td>
<td>483</td>
<td>99.8%</td>
</tr>
<tr>
<td>ARR32-C</td>
<td>100.0%</td>
<td>947</td>
<td>100.0%</td>
</tr>
<tr>
<td>ARR36-C</td>
<td>63.3%</td>
<td>30</td>
<td>50.0%</td>
</tr>
<tr>
<td>ARR37-C</td>
<td>74.0%</td>
<td>77</td>
<td>83.6%</td>
</tr>
<tr>
<td>ARR38-C</td>
<td>94.0%</td>
<td>397</td>
<td>98.0%</td>
</tr>
<tr>
<td>ARR39-C</td>
<td>67.7%</td>
<td>31</td>
<td>50.0%</td>
</tr>
<tr>
<td>CON33-C</td>
<td>100.0%</td>
<td>88</td>
<td>100.0%</td>
</tr>
<tr>
<td>ERR33-C</td>
<td>91.2%</td>
<td>376</td>
<td>94.9%</td>
</tr>
<tr>
<td>ERR34-C</td>
<td>100.0%</td>
<td>947</td>
<td>100.0%</td>
</tr>
<tr>
<td>EXP30-C</td>
<td>100.0%</td>
<td>947</td>
<td>100.0%</td>
</tr>
<tr>
<td>EXP33-C</td>
<td>89.5%</td>
<td>5214</td>
<td>96.3%</td>
</tr>
<tr>
<td>EXP34-C</td>
<td>91.8%</td>
<td>546</td>
<td>95.4%</td>
</tr>
<tr>
<td>EXP39-C</td>
<td>70.7%</td>
<td>116</td>
<td>83.1%</td>
</tr>
<tr>
<td>EXP46-C</td>
<td>82.5%</td>
<td>143</td>
<td>87.8%</td>
</tr>
<tr>
<td>FIO30-C</td>
<td>86.5%</td>
<td>1065</td>
<td>95.1%</td>
</tr>
<tr>
<td>FIO34-C</td>
<td>72.5%</td>
<td>1132</td>
<td>78.5%</td>
</tr>
<tr>
<td>FIO42-C</td>
<td>83.9%</td>
<td>933</td>
<td>93.2%</td>
</tr>
<tr>
<td>FIO46-C</td>
<td>100.0%</td>
<td>947</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

#### Lasso per-CWE-ID classifiers (31)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Accuracy</th>
<th># Alerts</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIO47-C</td>
<td>86.4%</td>
<td>1070</td>
<td>95.4%</td>
</tr>
<tr>
<td>FLP32-C</td>
<td>100.0%</td>
<td>947</td>
<td>100.0%</td>
</tr>
<tr>
<td>FLP34-C</td>
<td>70.5%</td>
<td>3619</td>
<td>78.0%</td>
</tr>
<tr>
<td>INT30-C</td>
<td>63.7%</td>
<td>1365</td>
<td>66.4%</td>
</tr>
<tr>
<td>INT31-C</td>
<td>68.7%</td>
<td>5139</td>
<td>77.5%</td>
</tr>
<tr>
<td>INT32-C</td>
<td>69.9%</td>
<td>1599</td>
<td>75.7%</td>
</tr>
<tr>
<td>INT33-C</td>
<td>79.8%</td>
<td>228</td>
<td>86.3%</td>
</tr>
<tr>
<td>INT34-C</td>
<td>100.0%</td>
<td>947</td>
<td>100.0%</td>
</tr>
<tr>
<td>INT35-C</td>
<td>64.3%</td>
<td>622</td>
<td>72.2%</td>
</tr>
<tr>
<td>INT36-C</td>
<td>100.0%</td>
<td>967</td>
<td>100.0%</td>
</tr>
<tr>
<td>MEM30-C</td>
<td>94.5%</td>
<td>1461</td>
<td>99.3%</td>
</tr>
<tr>
<td>MEM31-C</td>
<td>83.9%</td>
<td>933</td>
<td>93.2%</td>
</tr>
<tr>
<td>MEM35-C</td>
<td>66.7%</td>
<td>2514</td>
<td>76.0%</td>
</tr>
<tr>
<td>MSC37-C</td>
<td>100.0%</td>
<td>947</td>
<td>100.0%</td>
</tr>
<tr>
<td>POS54-C</td>
<td>90.0%</td>
<td>239</td>
<td>94.5%</td>
</tr>
<tr>
<td>PRE31-C</td>
<td>97.8%</td>
<td>46</td>
<td>99.1%</td>
</tr>
<tr>
<td>STR31-C</td>
<td>94.0%</td>
<td>397</td>
<td>98.0%</td>
</tr>
<tr>
<td>WIN30-C</td>
<td>95.6%</td>
<td>1465</td>
<td>97.8%</td>
</tr>
</tbody>
</table>

#### Similar for other classifier methods

<table>
<thead>
<tr>
<th>Avg. accuracy</th>
<th>Count accuracy 95+%</th>
<th>Count accuracy 85-94.9%</th>
<th>Count accuracy 0-84.9%</th>
</tr>
</thead>
<tbody>
<tr>
<td>85.8%</td>
<td>12</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>99.2%</td>
<td>90.9%</td>
<td>72.1%</td>
<td></td>
</tr>
</tbody>
</table>

#### Major improvement: 67 per-rule classifiers (and more coming!) vs. only 3 in FY16!
Summary and Future

FY17 Line “Rapid Classifiers” built on the FY16 LENS “Prioritizing vulnerabilities”.

- Developed widely useful general method to use test suites across taxonomies
- Developed large archive of “pre-audited” alerts
  - Overcame major challenge to classifier development
  - For CWEs and CERT rules
- Developed code infrastructure (extensible!)
- In-progress:
  - Classifier development and testing in process
  - Continue to gather data
  - Enhanced SCALe audit tool for collaborator testing: distribute to collaborators soon

FY18-19 plan: architecture for rapid deployment of classifiers in varied systems

- Goal: optimal automation of static alert auditing (and other code analysis and repair)

Publications:

- New mappings (CWE/CERT rule): MITRE and CERT websites
- 2 SEI blogposts on classifier development
- Research paper in progress