Rapid Expansion of Classification Models

For prioritizing static analysis alerts for C

Problem: 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.

Solution: Rapid expansion of number of classification models by using “pre-audited” code, plus collaborator-audited code.

Approach:

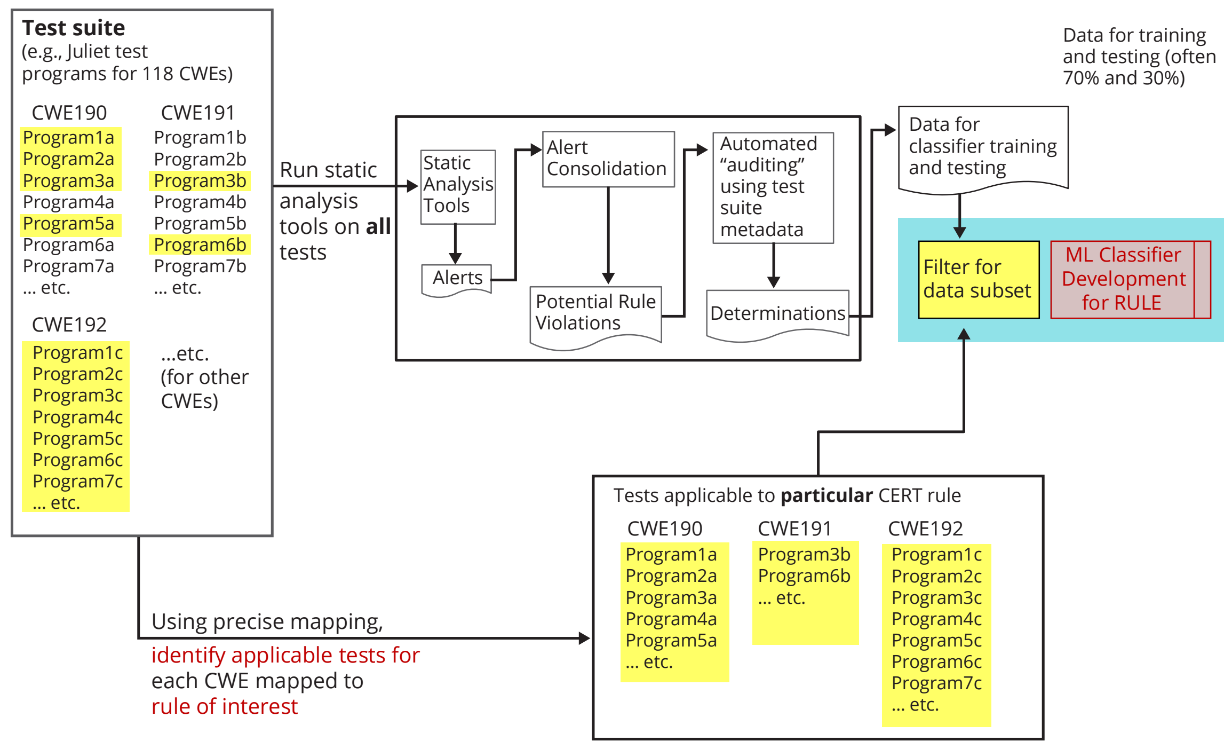
1. Modify SCALe research tool to map alerts to CWE
2. Systematically map CERT rules to named flaws in subsets of pre-audited code (published as true or false for flaw Automated analysis of pre-audited (not by SEI) codebases to gather sufficient code & alert feature info for classifiers)
3. Test classifiers on alerts from real-world code: DoD data

Process:

1. Generate data for Juliet: Proprietary and open-source static analysis tools and metrics tools
2. Generate data for STONESOUP: similar/ same tools
3. Generate scripts for classifier development
4. Build classifiers: directly for CWEs, partitioned test suite for CERT rules
5. Test classifiers

Using CWE Test Suites for CERT Rule 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 → Automated “auditing” using test suite metadata → Determinations

Data for classifier training and testing

Data for training and testing (often 70% and 30%)

Filter for data subset → ML Classifier Development for RULE

Using precise mapping, identify applicable tests for each CWE mapped to rule of interest

Tests applicable to **particular** CERT rule

CWE190: Program1a, Program2a, Program3a, Program4a, Program5a ... etc.

CWE191: Program3b, Program6b ... etc.

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

Novel method developed that successfully and quickly partitioned sets of thousands of tests.

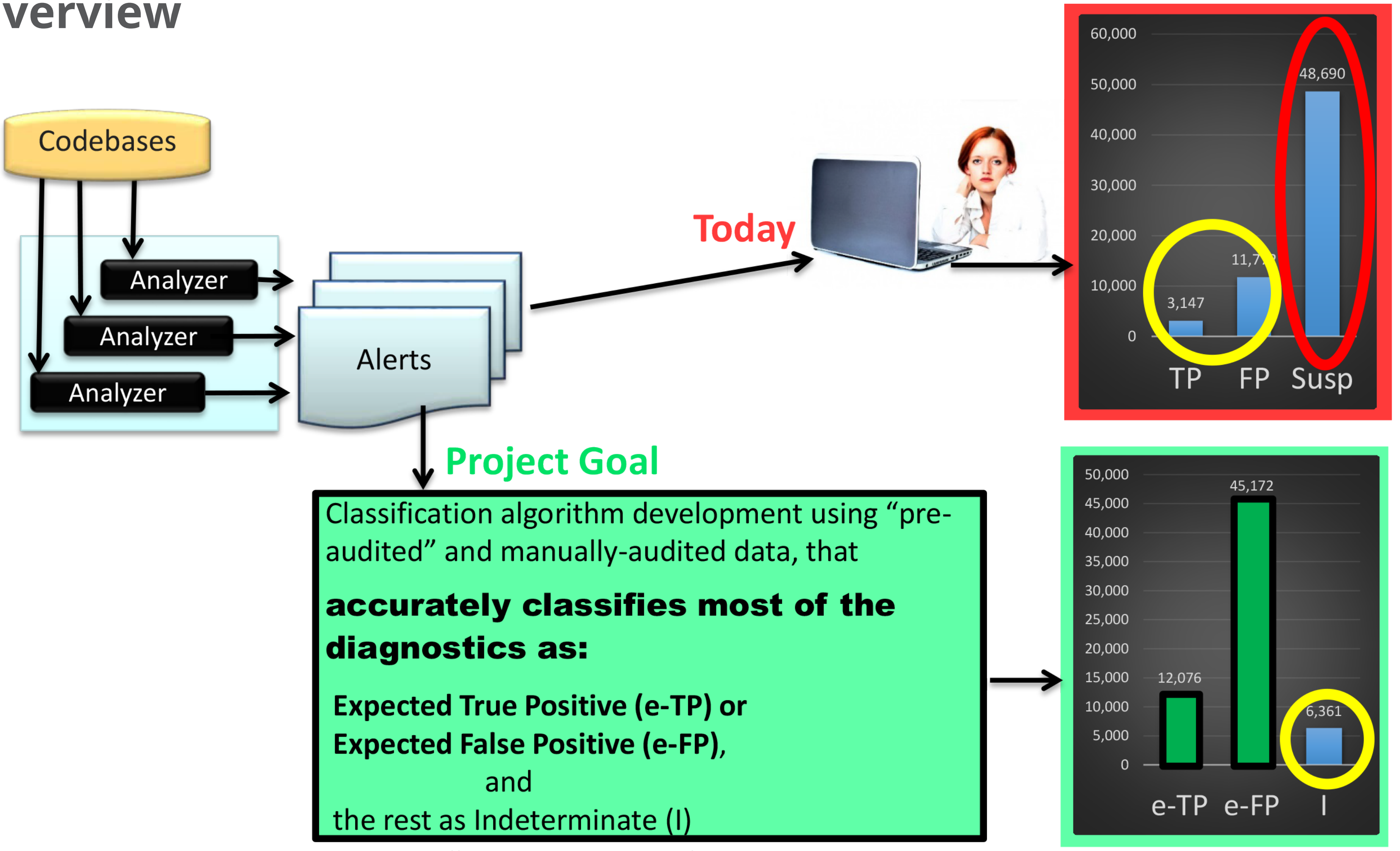
Examine together:

- Precise mapping
- Test suite metadata (structured filenames)
- Rarely examine small bit of code (variable type)

CERT rule	CWE	Count files that match
ARR38-C	CWE-119	0
ARR38-C	CWE-121	6,258
ARR38-C	CWE-122	2,624
ARR38-C	CWE-123	0
ARR38-C	CWE-125	0
ARR38-C	CWE-805	2,624
INT30-C	CWE-190	1,548
INT30-C	CWE-191	1,548
INT30-C	CWE-680	984
INT32-C	CWE-119	0
INT32-C	CWE-125	0
INT32-C	CWE-129	0
INT32-C	CWE-131	0
INT32-C	CWE-190	3,875
INT32-C	CWE-191	3,875
INT32-C	CWE-20	0
INT32-C	CWE-606	0
INT32-C	CWE-680	984

Rows with same color are for different CWEs mapped to same CERT rule

Overview



Codebases → Analyzer, Analyzer, Analyzer → Alerts

Today

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) or
Expected False Positive (e-FP),
and
the rest as Indeterminate (I)

Image of woman and laptop from http://www.publicdomainpictures.net/view-image.php?image=47526&picture=woman-and-laptop “Woman And Laptop”

CWE test programs useful to test CERT rules

STONESOUP: **2,608** tests

Juliet: **80,158** tests

- Test set partitioning incomplete (32% left (Static analysis tools might not alert, still!)

Some types of CERT rule violations not tested, in partitioned test suites.

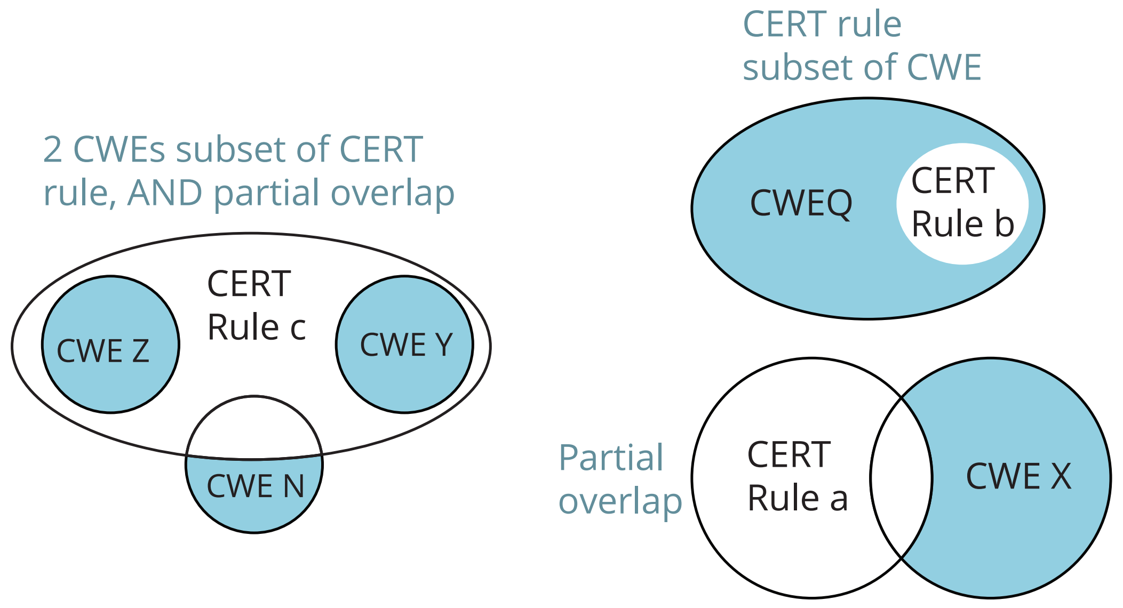
- Possible coverage in other suites

Precise mappings: Defines *what kind* of non-null relationship, and if overlapping, *how*. Enhanced-precision added to “imprecise” mappings.

Imprecise mappings (“*some* relationship”) → Precise mappings (set notation, often more)

Mappings	
Precise	248
Imprecise TODO	364
Total	612

Now: all CERT C rules mappings to CWE precise



Achievements:

- Preliminary classifier development and testing results (in progress):
  - *Such high accuracies may be artifact of test metadata, currently investigating cause. Expect reduced performance against native files.*
- Xgboost: classifier tested on **56 CWEs** (97.2% avg. accuracy)
- Lasso: classifier tested on **31 CWEs** (98.7% avg. accuracy)
- Xgboost: classifier tested on **44 CERT rules** (95% have **at least 95% accuracy**, and lowest accuracy was **83%**)
- Widely useful general method for using test suites across taxonomies
  - New mappings published on CERT and MITRE websites
- Large archive of “pre-audited” alerts, useful for both CWEs and CERT rules
- Improved tooling that can be transitioned to DoD organizations
- Code infrastructure for classifier development (extensible!)
- Classifier development and testing results (in progress)
- Research paper submission to ICSE 2018 workshop (in progress)
- IEEE SecDev 2017 Tutorial “Hands-on Tutorial: Alert Auditing with Lexicon & Rules”
- 2 SEI blogposts on classifier development
- **Novel** speculative mapping method, for mapping checkers from tools with no public mappings to both CWEs and CERT rules.
  - **16,305** speculatively-mapped-to-CWEs alerts, from 3 tools run on Juliet.

Juliet initial analysis:

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

This is a lot of new data for creating classifiers!

Alert Type	Equivalence Classes: (EC counts a fused alert once)	Number of Alerts Fused (from Different Tools)
HCTP	16,664	2,111
HCFP	32,684	2,699

We automated alert-to-alert matching (alerts fused: same line & CWE), combined with test suite metadata.

Above metrics after only used 3 tools on Juliet.

This project developed a large archive of “pre-audited” alerts useful for building accurate CWE and CERT rule classifiers. It developed reusable code and a method for using test suites across taxonomies.

Carnegie Mellon University
Software Engineering Institute

Lori Flynn | lflynn@sei.cmu.edu

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.

DM17-0742
Rapid Expansion of Classification Models