# Scaling Software Testing & Evaluation

SEI CERT Division Cyber Security Foundations Directorate

Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213



[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

Carnegie Mellon University Software Engineering Institute Copyright 2018 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.

Carnegie Mellon<sup>®</sup> is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM18-1194

### The Problem



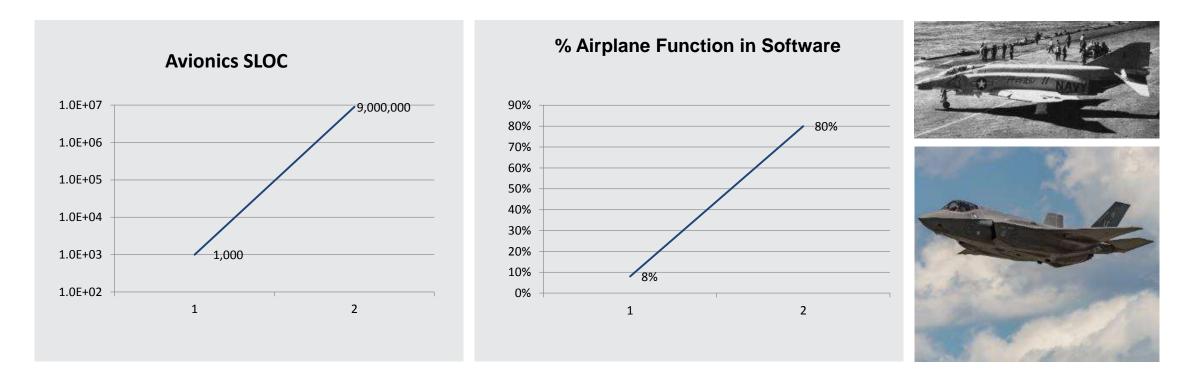
**Carnegie Mellon University** Software Engineering Institute

© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

# **Complex Software Is Business and Mission Critical**

#### Evolution of avionics size and function from F-4A (1960) to F-35 (2000):\*

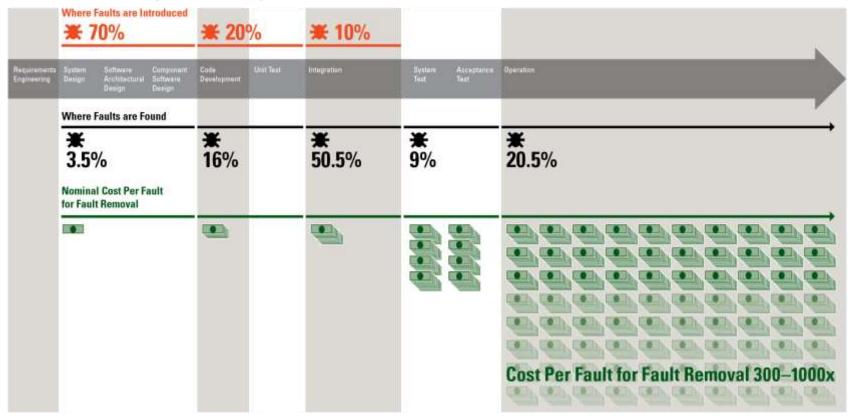


\*Final Report, NASA Study on Flight Software Complexity, Mar. 2009; Mel Conway, "Tower of Babel and the Fighter Plane," Oct. 9, 2013.

# Catching Software Faults Early Saves Money

#### Faults account for 30%–50% of total software project costs.\*

Software Development Lifecycle



\*Critical Code; NIST, NASA, INCOSE, and Aircraft Industry Studies.

**Carnegie Mellon University** Software Engineering Institute © 2018 Carnegie Mellon University

## Enduring Software Challenges: Scaling Software Testing and Evaluation

#### Affordable

Be Affordable such that the cost of acquisition and operations, despite increased capability, is reduced and predictable



#### Trustworthy

Be Trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties



#### Capable

Bring Capabilities that make new missions possible or improve the likelihood of success of existing ones

#### Timely

Be Timely so that the cadence of fielding is responsive to and anticipatory of the operational tempo of the warfighter



# Testing

**Carnegie Mellon University** Software Engineering Institute

© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

# **Testing Methods**

Method	Strengths	Weaknesses
Architectural/Design Analysis	Early identification of costly defects	Conceptual and early
Inspection & Reviews	Effective at identifying nuanced defects that require developer context	Manual (expensive, slow)
Static Analysis	More thorough coverage	High false-positive rates Generally requires buildable source
Dynamic Analysis	Very low false-positive rate	Difficult to get good coverage
Formal Methods	Proves software attributes	Requires significant time and space resources, plus model validation is challenging; significant manual effort
Simulation Carnegie Mellon University	Useful for gaining validation	Testbed setup can be costly
Software Engineering Institute	confidence University	unlimited distribution

# **Testing Purposes and Evolution**



#### **Testing Purposes**

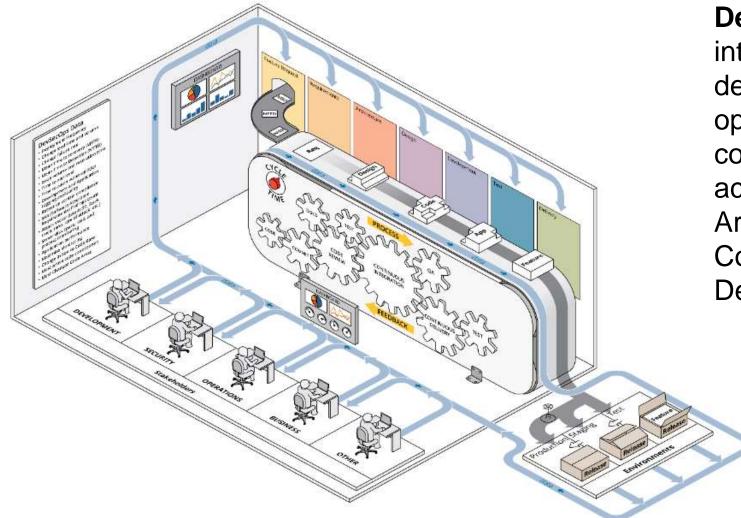
- Unit testing
- Integration/system testing
- Regression testing
- Acceptance testing



#### Evolution (as possible)

- Manual inspection
- Tool-supported
- Integrated
- Automated testing
- Automated repair

### Secure DevOps



**DevSecOps** is a model integrating the software development and operational process considering security activities: Requirements, Architecture, Design, Coding, Testing, and Delivering.

© 2018 Carnegie Mellon University

# Scaling Testing

SEI is researching how to make testing (where possible)

- less expensive
- more precise
- automatable

SEI also researches scalable automated repairs, following testing.

### Predicting Security Flaws through Architectural Flaws



**Carnegie Mellon University** Software Engineering Institute

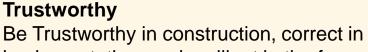
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

### Enduring Software Challenges: Predicting Security Flaws through Architectural Flaws

#### Affordable

Be Affordable such that the cost of acquisition and operations, despite increased capability, is reduced and predictable



implementation, and resilient in the face of operational uncertainties



### Problem

Software security defects

risk exposure and \$\$\$.

Existing analysis methods have limitations:

- Some security flaws influenced by code structure and module relationships.
- Not easily found or fixed locally.\*



\*"Analyzing Security Bugs from an Architectural Perspective," Kazman et al., 2017.

Research Review 2018

# Impact of Issues Involving >10 Files (Chromium)

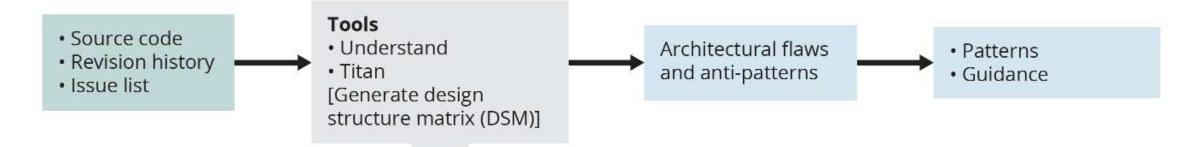
Potential Impact: ~50% of the total effort (LoC) to fix security issues came from fixing <10% of the security issues.\*



\* Analyzing Security Bugs from an Architectural Perspective," Kazman et al., 2017.

**Carnegie Mellon University** Software Engineering Institute © 2018 Carnegie Mellon University

### Approach – Today

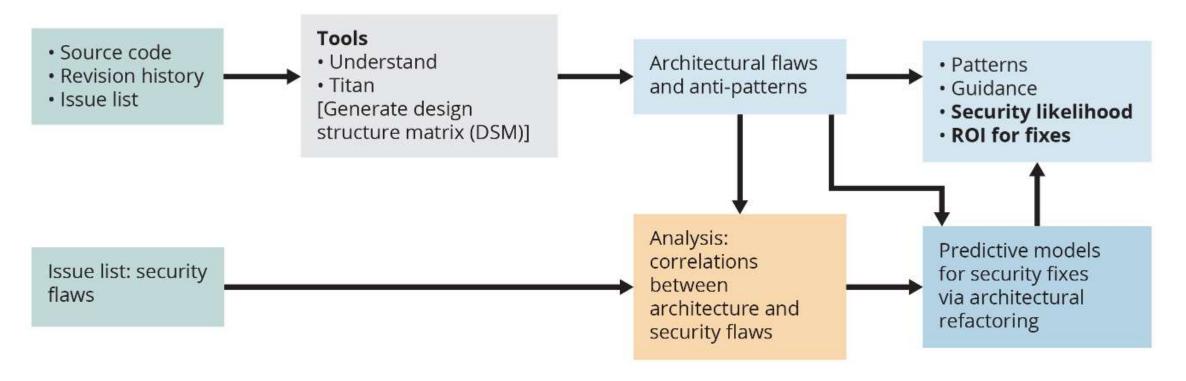


	Im	prop	er Int	erfa	ce						
		1	2	3	4	5	6	7	8	9	10
1	122654.content.browser.ssl.ssl_error_handler.h	(1)									
2	122654.content.browser.ssl.ssl_manager.h	1,6	(2)								
3	122654.content.browser.renderer_host.resource_dispatcher_host_impl.h	I,Pu,2	,2	(3)							
4	122654.content.browser.ssl.ssl_cert_error_handler.h	I,Pu,5	,12	,2	(4)						
5	122654.content.browser.ssl.ssl_error_handler.cc	U,I,6	,6	,2	C,1,4	(5)					
6	122654.content.browser.ssl.ssl_manager.cc	,4	C,I,11	1,2	C,I,10	C,4	(6)				
7	122654.content.browser.ssl.ssl_cert_error_handler.h	,4	C,U,I,23	1,2	I,10	,12	C,10	(7)			
8	122654.content.browser.rendererHost.socket_stream_dispatcher_host.h	I,Pu	,2		,2				(8)		
9	122654.content.browser.rendererHost.socket_stream_dispatcher_host.cc		1					С	U,I,9	(9)	
10	122654.content.browser.rendererHost.resource_dispatcher_host_impl.cc	,2	1,3	U,I,18	,2	,2	,2	C,3			(10)

C = Call; U = Use; I = Include; T = Type; S = Set; O = Override; Pu = Public Inherit; ,# = # concurrent check-ins

Research Review 2018

### Approach – Research and Vision





Used in build pipeline as part of SDLC, and potentially automated

- Regular reports on architecture flaws introduced
- Indicators of security risk
- Suggestions for refactoring

### **Progress and Results**

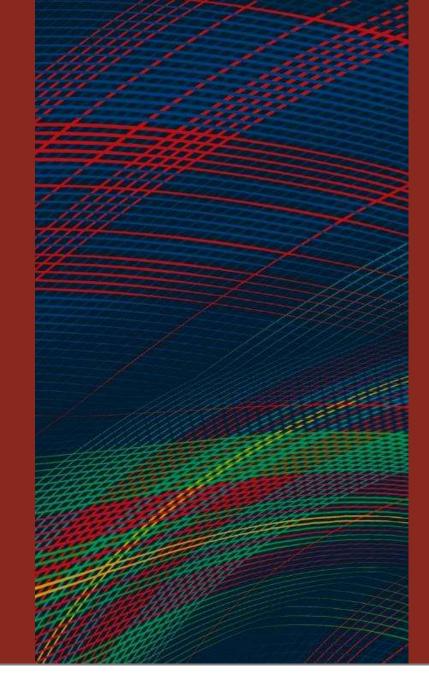
Chromium, OpenSSL, and Mozilla analyzed

- 6000 Chromium issues analyzed after commits (50% security / 50% non-security)
- 1600 Chromium issues analyzed before/after commits
- Analyzing entire Chromium project over entire per-file history
- Analyzing Chromium issue chains for common files

Tools

- Scripts for data extraction of code repository and issue logs
- Scripts for filtering, categorizing, and analyzing data
- Findings (anecdotal; still analyzing for statistical significance)
  - Architectural flaws have been strongly correlated with security flaws at a project-wide level
  - Still iterating on precision of defining architectural flaws for correlative analysis

### Rapid Construction of Accurate Automatic Alert Handling System



**Carnegie Mellon University** Software Engineering Institute

© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

# Enduring Software Challenges: Rapid Construction of Accurate Automatic Alert Handling System

#### Affordable

Be Affordable such that the cost of acquisition and operations, despite increased capability, is reduced and predictable

#### **Trustworthy**

Be Trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties





#### Capable

Bring Capabilities that make new missions possible or improve the likelihood of success of existing ones

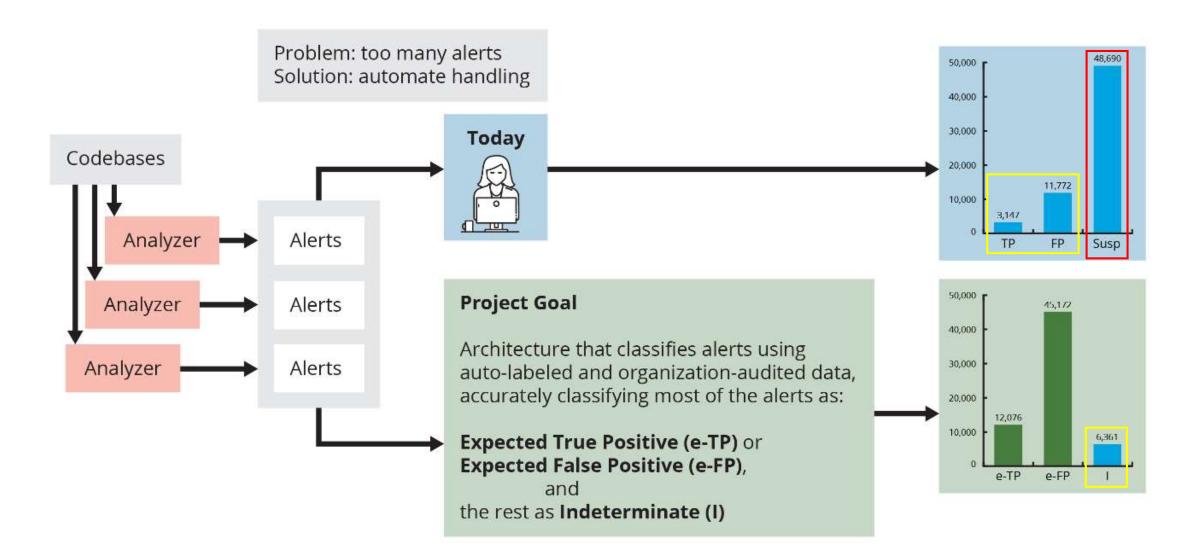
#### Timely

Be Timely so that the cadence of fielding is responsive to and anticipatory of the operational tempo of the warfighter



- Affects state-of-the-art and stateof-the- practice for static analysis
- Novel use of test suites for classification
- Effect: more secure code at same cost

### Overview



# FY16-18 Static Analysis Alert Classification Research

#### FY16

- Issue addressed: classifier accuracy
- Novel approach: multiple static analysis tools as features
- Result: increased accuracy

#### FY17

- Issue addressed: too little labeled data for accurate classifiers for some conditions (CWEs, coding rules)
- Novel approach: use test suites to automate production of labeled (True/False) alert archives for many conditions
- Result: high accuracy for more conditions

#### FY18

 Issue addressed: little use of automated alert classifier technology

(requires \$\$, data, experts)

- Novel approach: develop extensible architecture with novel test-suite data method
- Result: extensible architecture, API definition, software to instantiate architecture, adaptive heuristic research

### Code

API definition (swagger) and code development

SCALe v2.1.3.0 static analysis alert auditing tool

- New features for prioritization and classification
  - Fused alerts, CWEs, new determinations (etc.) for collaborators to generate data
- Released to collaborators Dec. 2017–Feb. 2018
- GitHub publication Aug. 2018 (2.1.4)

SCALe v3.0.0.0 released Aug. 2018 to collaborators

Develop and test classifiers. Novel work includes

- enabling cross-taxonomy test suite classifiers (using precise mappings)
- enabling "speculative mappings" for tools (e.g., GCC)

### Non-code Publications & Papers FY18

#### Architecture API definition and new SCALe features

- Special Report: "Integration of Automated Static Analysis Alert Classification of A
  - Technical Report: public version (Sep. or Oct. 2018)
- SEI blog post: "SCALe: A Tool for Managing Output from Static Code Analyzers" (Sep. 2018)

#### **Classifier development research methods and results**

- Paper "Prioritizing Alerts from Multiple Static Analysis Tools, using Classification Models," SQUADE (ICSE workshop)
- SEI blog post: "Test Suites as a Source of Training Data for Static Analysis Alert Classifiers" (Apr. 2018)
- SEI Podcast (video): "Static Analysis Alert Classification with Test Suites" (Sep. 2018)
- In-progress conference papers (4): precise mapping, architecture for rapid alert classification, test suites for classifier training data, API development
   Static analysis tool developers

#### Precise mappings on CERT C Standard wiki

- CERT manifest for Juliet (created to test CWEs) to test CERT rule coverage
- Per-rule precise CWE mapping

For code flaws you care about, understand your tool coverage

For collaborators, others to

implement API calls or use new

Explain research methods and results

can automatically test for CERT

rule coverage (some rules)

# Analysis of Juliet Test Suite: Initial CWE Results

Alert Type	Labeled Fused Alerts (counts a fused alert once)
TRUE	13,330
FALSE	24,523
	Lots of new data for

Lots of new data for creating classifiers (37,853 labeled alerts) Big savings: manual audit of 37,853 alerts from non-testsuite programs would take an unrealistic minimum of 1,230 hours (117 seconds per alert audit\*).

- First 37,853 alert audits wouldn't cover many conditions (and sub-conditions) covered by the Juliet test suite!
- Need true and false labels for classifiers.
- **Realistically:** enormous amount of manual auditing time to develop that much data.

These are initial metrics (more data as we use more tools and test suites).

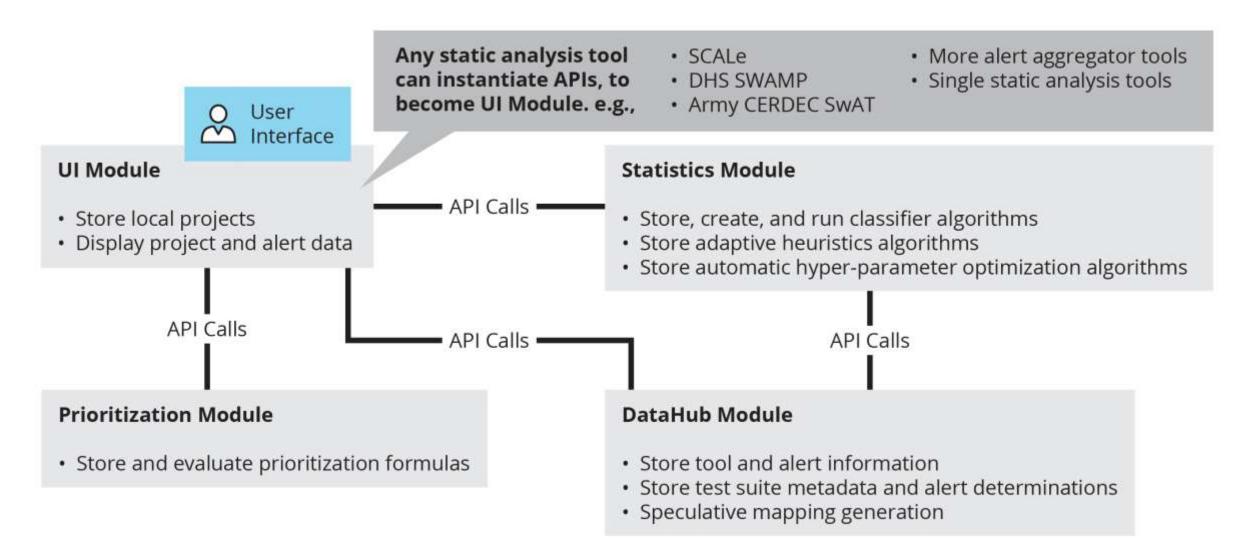
\*Nathaniel Ayewah and William Pugh, "The Google FindBugs Fixit," Proceedings of the 19th International Symposium on Software Testing and Analysis, ACM, 2010.

### Juliet Test Suite Classifiers: Initial Results (Hold-out Data)

Classifier	Accuracy	Precision	Recall
rf	0.938	0.893	0.875
lightgbm	0.942	0.902	0.882
xgboost	0.932	0.941	0.798
lasso	0.925	0.886	0.831

		Actual cond			
	Total population	Condition true	Condition false	Accuracy =	$\Sigma$ True positive + Σ True negative Σ Total population
	Predicted condition true	True positive	False positive	Precision =	<u>Σ True positive</u> Σ Predicted condition true
Predicted	Predicted				
condition	condition false	False negative	True negative		
		True positive rate, $Σ$ True positive recall, sensitivity = $Σ$ (Condition true)	False positive $\frac{\Sigma}{\Gamma}$ False positive rate = $\Sigma$ (Condition false)		

### Architecture



# Architecture Development

Representational State Transfer (REST)

- Architectural style that defines a set of constraints and properties based on HTTP
- RESTful web services provide interoperability between systems
- Client-server

We chose to develop a RESTful API.

- Swagger/OpenAPI open-source development toolset
  - Develop APIs
  - Auto-generate code for server stubs and clients
  - Test server controllers with GUI
  - Wide use (10,000 downloads/day)

# SCALe Development for Architecture Integration

SCALe will make UI Module API calls in prototype system.

• Other alert auditing tools (e.g., DHS SWAMP) also can instantiate UI Module API.

Research Review 2018

### Continue FY19: Classifier Research and Development

#### Using test suite data for classifiers, research:

- Adaptive heuristics:
  - How classifiers incorporate new data
  - Test suite vs. non-test-suite data
  - Weighting recent data
- Semantic features for cross-project prediction
  - Test suites as different projects

#### **Collaborator API implementation**

More collaborator audit archive data sharing

Metrics of success:

- Compare classifier precision on DoD datasets (cross-validation on test set):
  - Test with semantic features
  - Variations of adaptive heuristics
- Test fault detection rates by tracking true positives detected versus number of manual alert inspections
- Goal: minimum 60% classified e-TP or e-FP with 95% accuracy against collaborator data
- Test architecture generality using varied plug-ins to API

### Can Deep Learning Predict Security Defects in Synthetic Code?



**Carnegie Mellon University** Software Engineering Institute

© 2018 Carnegie Mellon University

### Enduring Software Challenges: Can Deep Learning Predict Security Defects in Synthetic Code?



#### Capable

Bring Capabilities that make new missions possible or improve the likelihood of success of existing ones

Trustworthy

Be Trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties





- Developed, sa-bAbI, a new software assurance benchmark and training set to be included in NIST Software Assurance Reference Dataset (SARD)
- Identified next steps for AI in software assurance: better representations of code and different learning strategies

### Can Deep Learning Predict Security Defects in Synthetic Code?

**Problem:** Predicting security defects in source code is of significant national security interest (e.g., NIST SAMATE), but existing static analysis tools have unacceptable performance.\*

- Artificial intelligence approaches may improve performance, but
- Existing software assurance datasets have limited variability in examples of defects (e.g., Juliet, SARD, IARPA Stone Soup, LAVA)

#### Our Approach:

- Develop a new software assurance dataset: sa-bAbl
- Benchmark state-of-the-art artificial intelligence system and existing static analysis tools on sa-bAbl

\*Oliveira et al., 2017.

# sa-bAbl: "Baby AI" Software Assurance Tasks

#### Conditional Reasoning Example

```
char entity_7[27];
entity_1 = 45; entity_8 = 74;
if(entity 8 > entity 1){
    entity_8 = 64;
} else {
    entity_8 = 17;
}
entity 7[entity 8] = 'i';
```

Is the last access safe?

No.

Modeled after bAbl\*

Code generator for detecting buffer overflow errors

Intentionally very simple

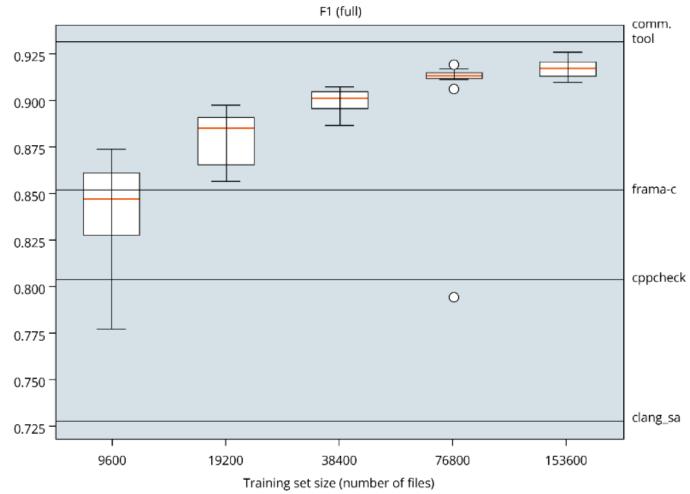
- Valid C code
- Conditionals
- Loops
- Unknown values such as rand()

Complements existing software assurance datasets for training AI

Will be included in NIST SARD

\*Weston et al., 2015

# Results: Deep Learning Cannot Do This Yet



The state-of-the-art AI system can be **competitive** with existing static analysis engines, but it **fails to generalize**.

**sa-bAbl illuminated why**: We need better

• representations of code and

• neural integer computation

See arXiv.org for more details.



**Carnegie Mellon University** Software Engineering Institute

### Status of Available Technology



**Carnegie Mellon University** Software Engineering Institute

© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

### Enduring Software Challenges: Status of Available Technology

#### Affordable

Be Affordable such that the cost of acquisition and operations, despite increased capability, is reduced and predictable

#### Trustworthy

Be Trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties



#### Capable

Bring Capabilities that make new missions possible or improve the likelihood of success of existing ones

#### Timely

Be Timely so that the cadence of fielding is responsive to and anticipatory of the operational tempo of the warfighter

- Secure coding standards: provide coverage target for static analysis tools, training for developers
- SCALe (static analysis alert auditing tool): provide implemented research features for others to use or adapt into their own tools

SOTA SOTP

• Automated Code Repair to Ensure Memory Safety: inexpensive process results in more-secure code

# Available Technology: Secure Coding Standards



SEI CERT C Coding Standard Rutes to Developing Sate, Relation, and Secure Syste





SEI CERT C++ Coding Standard Rates to Developing Soft, Reliable, and Social Sprates is C++ annuality

Aaron Daliman

amegie Melko Comenty officiale Engineering Politic Curated wisdom from thousands of contributors on community wiki since 2006

• Use the standards to develop analysis tools and to train developers.

#### SEI CERT C Coding Standard

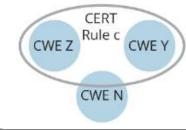
- Free PDF download: <u>cert.org/secure-coding/products-services/secure-coding-download.cfm</u>
- Basis for ISO TS 17961 C Secure Coding Rules

#### SEI CERT C++ Coding Standard

- Free PDF download (Released March 2017): <u>cert.org/secure-coding/products-services/secure-coding-cpp-download-</u> <u>2016.cfm</u>
- **CERT** Oracle Secure Coding Standard for Java
  - Latest guidelines available on CERT Secure Coding wiki: <u>securecoding.cert.org</u>

# Available Technology: Secure Coding Standards (cont'd)





**Precise mappings:** Defines *what kind* of relationship, and if overlapping, *how.* Also *when* mapped and *which versions*.

Imprecise mappings (*"some* relationship")

Precise mappings

Precise mappings on CERT C Standard wiki

- 1. Per-rule CWE precise mapping
  - "CERT-CWE Mapping Notes" (set notation)
  - Table with taxonomy and relationship detail
- 2. Metadata for using Juliet Test Suite to test CERT rule coverage
  - Plan: create similar metadata for STONESOUP and other test suites

For code flaws you care about, understand your tool coverage

Static analysis tool developers can automatically test for CERT rule coverage (some rules)

If a **condition** of a program violates a CERT rule *R* and also exhibits a CWE weakness *W*, that **condition** is in the overlap.

## Available Technology: SCALe Static Analysis Alert Auditing Tool

Г.					-		1	Western		2			*) Pres		ie (1			3		rain:								
	Line					Chi	eller.		Al Orecen			- 41	Test	10	Trella	4 6	ndeter.		140	1	Texastery		1	lev Ni				
	Set and	Der:		850		- 54	1995		19640																			
	-											10																
_	Showing 1 k	10 et 25	Digton	lits per peg					100		Ciel I									140	Paratasi 🚺	2.01	40.4	141	1 1		-13	( et ) et
	that series los	the:						-				·	Optim	Class	her Maleclad	Naine (lariected	Cunty											
Ē		IG Flag Weaks Supplemental Noise Province Path Line Noisasge									Checker		Tool	Condition	124			Costelen	ce Alert	Pri Sev		han Pr	Lev	CWF_LA				
	C 1812	0 11	(interest)	the state	1	0	heutenmen	2 10	Anigmentation	r patarrai	er has no effectio	date for	laiden Dityschep	distantion of the	-		oppotents	CWE-DIR	144.			+						NA.
L	0 1010	e 11	[Jrikesent	. née	10	6	Sevial men	4 722	Assignment of function parameter than no offset subside five function. Did you heppet developmenting 17 users						et lineada	in the	opportwolk	0WE-288	NA.			i <del>t</del>	-					NO.
	() 1000	n 11	Different	. Kant		0	(stotesting)	è .m	Contract Turker is a		• C				NHI	internation	e apotece	GWC-611	144.				-					1411
	0 1010	0 11	(2 triang	. mare		0	Seutosemen	é. 789	Condition VietWilliam	durings have	()				khirer Co	ndie (Traffie)	e opystwał	OWEIGHT	NA.			÷	-					NR.
L	0.000	6 0	(interest	Set 1		0	septement	× 18.8	Vocable Petitor's and	iprei av	alle Batil never	used.			HE WARD (V)	-	opperation	OWE SITE	-			T.						14.8
	() 1992	0 11	presser.	see		p	(accentration	e 141	The scope of the varia	alie teran	tan be reduced.				vatand	olahir	oppernets	DOL THE	Minimus the scope of vo	-	and kindow			+	1 1	2 2	8	
	0 1005	0 11	(University)	Sat.		0	anconno	a 188	The scope of the verta	-	san be extend				touted	-	opictera.	001/0-C	Minimize the access of yo		and bestore	-	-		1.1	1 2	2	
L	() 1008.	a 11	(these)	Ref.	8	0	Artboliner	2 548	The scope of the varia	die toor	contine rectaured.				valutions	1	oppotence	bound-	Alterinize the scope of va		and keeping	÷	14	4	1.1	2 2	1	
L	(1) 1001	0.0	[Jrikowet	dit .		(a))	seconter	e 111	Assignment of function	i pisteriei	w haa no effect a	enter tre	Unistiani. Distry au Yorg	n besteresong	er hinnan	-	opportwate.	VISICIA C	Detect and romovy color	-	interesting.	H.	-	- (ii	1.14	2 2	1	
	0 1002	e U	(Armone)	100		0	ter proprietant	8 M	Assignment of Tuncker	h posacive	er has no effect a	400.00 \$10	victor, Det jou terp	n mintere orig	er : 00000005		g operada	VEG 12 C	Detect part romove code	104114	ni rio alfred	+		9	1.12	1	1	
F	src																											
	SIC																						Cont total	dated 1	Maril As	an 22	72.09	30 HDT :

Used as a research platform

- Extend with new features
- Collaborators give us feedback
- Collaborators generate data required for our classifier research

Over last 3 years, new SCALe features are for classification and prioritization research.

- GitHub public release (SCALe v2), Aug. 2018
- SCALe v3 for research project collaborators

# Available Technology: SCALe Static Analysis Alert Auditing Tool

ame: myPrioritiz	ationScl	neme1									
Instructions	CWES	CERT	RULES								
cert_severity	2	3			Fo	ormula	or CEF	RT_RU	LES		
cert_likelihood	1 8	3)		()		+ /	-	cert_	severity	•	
cert_remediation	1										
cert_priority	0	3)	(cert_se	verity*2+c	cert_rem	ediation)'	confiden	ce*2			
cert_level	0	3)									
confidence	2	Ð									
			(	Generate	e The Fo	ormula					
rioritization Fo	rmula:										
CWES((confider	nce*2)+cv	/e_likeliho	od)+IF_C	ERT_RUL	ES((cert	_severity'	2+cert_r	emediati	on)*confid	ence*2)	

#### Recent features include

- Alert fusion for {filepath, line, condition} reduces auditor effort
- Determinations history
- Automatically cascaded determinations from previous audits
- Classification schemes
- Prioritization schemes with mathematical formulas user can create and/or use
- User field uploads



#### **Carnegie Mellon University** Software Engineering Institute

### Available Technology: Automated Code Repair to Ensure Memory Safety

Goal: Take a C codebase and repair potential bugs to enable a proof of memory safety.

What about distinguishing false alarms from true vulnerabilities?

• We simply apply a repair to all potential memory-safety vulnerabilities, at a cost of an often small runtime overhead. (Manual tuning might be needed for performance-critical parts.)

Available technology: Repair of integer overflows that lead to buffer overflows.

- Inferred specification: **inequality comparisons** involving array indices or bounds should behave as if normal (non-overflowing) arithmetic were used.
  - Includes malloc.
  - Excludes hash functions and crypto, where modular arithmetic is desired.
- We repair the code to satisfy this spec where possible.
- Tested on older versions OpenSSL and Jasper. Found and repaired known vuls with CVEs.

**Carnegie Mellon University** Software Engineering Institute