Risk-based Approach to Cyber Vulnerability Assessment using Static Analysis

Sponsor: OUSD(R&E) | CCDC AC [ART-006]

By
Barry Horowitz, Tim Sherburne
Peter Beling, Cody Fleming, Stephen Adams, Davis Loose
11th Annual SERC Sponsor Research Review
November 19, 2019
FHI 360 CONFERENCE CENTER
1825 Connecticut Avenue NW, 8th Floor
Washington, DC 20009

www.sercuarc.org

Intellectual Property Notice

U.S. Provisional Patent Application Serial No. 62/935,835
Filed on November 15, 2019
Title: Method and System for Risk-based Approach to Cyber Vulnerability Assessment Using Static Analysis
UVA LVG Reference: HOROWITZ-RISK (02620-01)

Static Analysis for Detecting Cyber Attack Vulnerabilities

- Static Analysis (SA) is a method of computer program debugging that is done by examining the code without executing the program.
- The process provides an understanding of the code structure, and can help to ensure that the code adheres to industry standards.
- Automated tools can assist programmers and developers in carrying out static analysis.

As defined by Margaret Rouse of Whatis.com

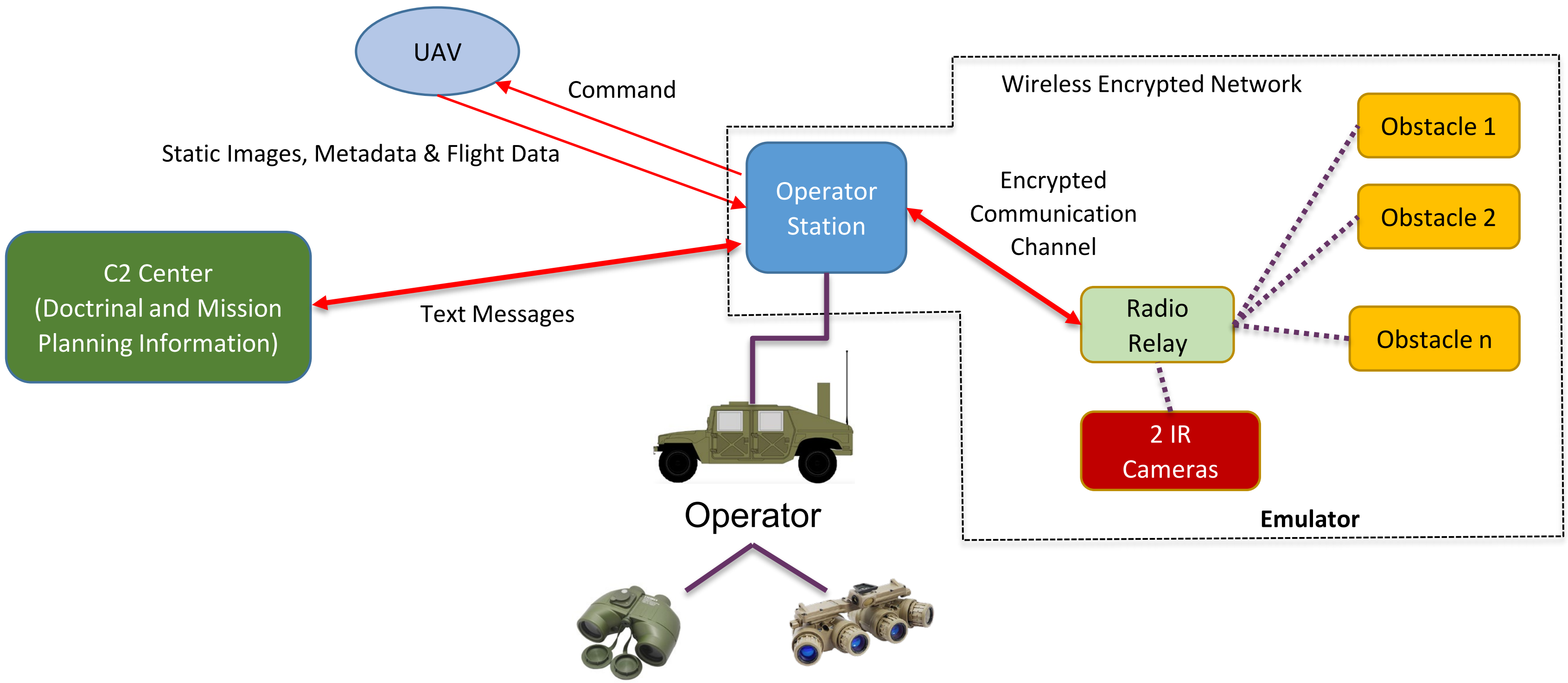
Static Analysis Tools Applied to Discovering Cyber Security Vulnerabilities

- A significant number of static analysis tools exist for discovering cyber security vulnerabilities supporting a variety of:
  - Operating Systems, including Linux and Windows
  - Programming and scripting languages, including C/C++, Java, Perl and Python
- Tools discover SW structures that can support the objectives of cyber attackers, and provide input to programmers and designers for consideration regarding modification of the identified vulnerable SW
- Tool outputs include identifying specific lines of code that are problematic and should be candidates for modification

DoD User Experience and the Research Problem

- DoD user experience has identified significant productivity issues with using Static Analysis tools for cyber security purposes:
  - Time and analyst skill level required to assess identified vulnerabilities as worthy of correction
  - The large number of identified vulnerabilities, most of which are not considered as worthy of change (false positives)
- UVA is involved with the Armament Division of the Army in exploring a system level risk based methodology for using automation for prioritizing detected vulnerabilities derived from static analysis tools
- Methodology includes a system user developed prioritized list of system failure (hard and soft) consequences to be avoided

System Diagram (pre-resilience) for a hypothetical Landmine-based Weapon System



Example Table of Prioritized System Failure Consequences to be Avoided

Attack Outcome	Related Attack Target(s)	User Priority
Inappropriate firings via manipulating operator commands	Operator control display, radio comm links	1
Delays in fire time (sufficient delay to cross field)	Obstacles, control station, radio comm links	1
Delays in deployment	Obstacles, deployment support equipment	1
Deactivation of a set of obstacles	Obstacles	1
Delays in situational awareness	Operator display, sensors	2
Prevent or corrupt transmission of situational awareness data	Radio comm links, operator display, sensors	2
Gain information to help adversary navigate through field	Obstacle, operator control station	2
Reduced operational lifespan	Obstacle	3
Prevent transmission/execution of non-firing commands	Operator display, obstacles	3
Delays in sending/receiving C2 information	Operator display, radio comm links	4
Delays in un-deployment	Obstacles	4

Problem: Which of the consequences to be avoided are dependent on which lines of code identified via static analysis

Combining System Consequence Analysis and Static Analysis Results to Support Risk Analysis (1 of 2)

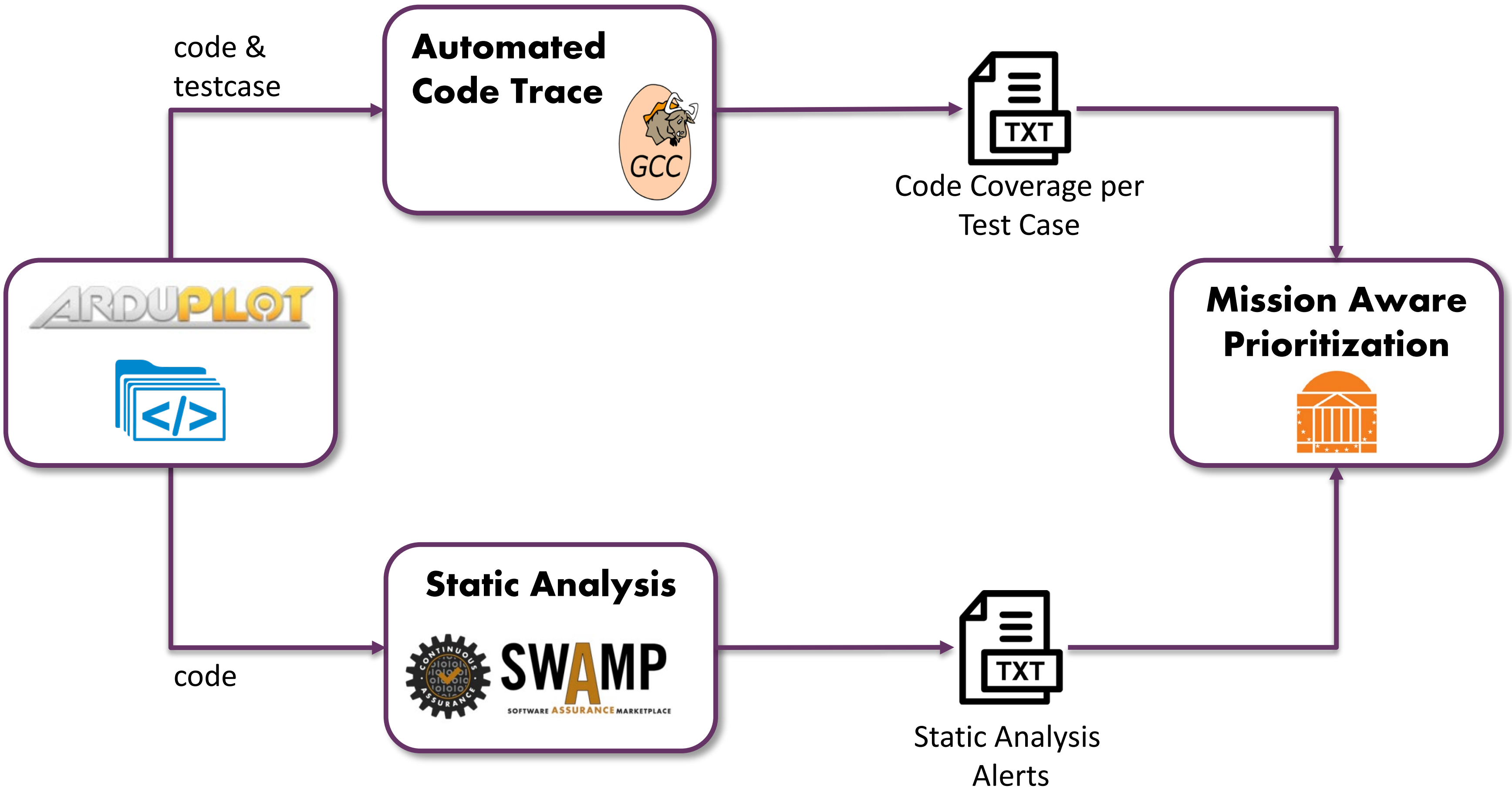
- Dynamic system testing currently includes tests that relate to other than cyber attack stimulants for system fault tolerance and resilience features:
  - —Safety
  - —Operator errors
  - —Out of Spec situations (e.g., overloads, potential anomalous circumstances)
  - —System countermeasures (electronic warfare, tampering)
  - —Technology component failures (hard and soft failures)
  - —Etc.
- These focused dynamic system tests provide a basis to use already available compiler-based SW tracing results as a means for identifying the specific SW modules and files used in the process of evaluating system design related to specific faults

Combining System Consequence Analysis and Static Analysis Results to Support Risk Analysis (2 of 2)

- Hypothesis regarding application of static analysis results to prioritization of cyber attack risks:
  - System risks which are currently tested for will include significant consequence overlaps with those derived from cyber attacks, thereby providing a basis for using SW tracing based upon already existing system tests as a mechanism for identifying which of the static analysis results relate to which of the identified cyber attack consequences
  - As cyber attack resilience emerges as an additional area of system design, dynamic system test results for cyber resilience can be utilized to enable more effective use of new static analysis results that emerge over the life cycle due to:
    - system design changes,
    - changing cyber attack techniques,
    - new findings that result from modifications in static analysis tool designs

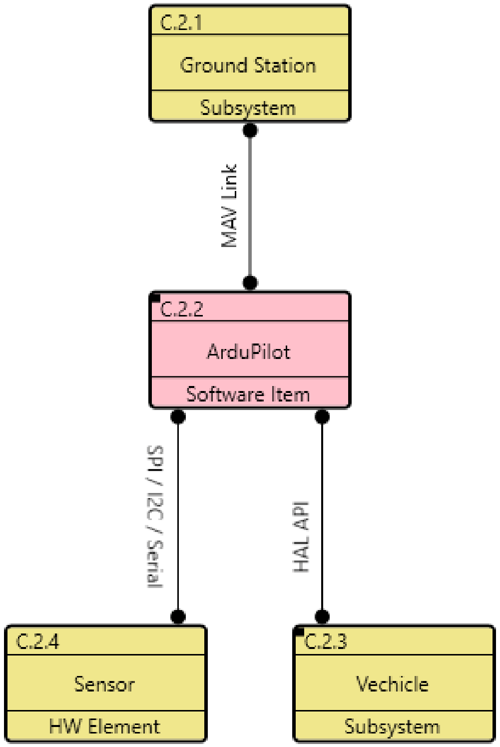
Initial Research Findings

Initial Research Context



ArduPilot Overview

ArduPilot is an open source autopilot system supporting multiple autonomous vehicle types.



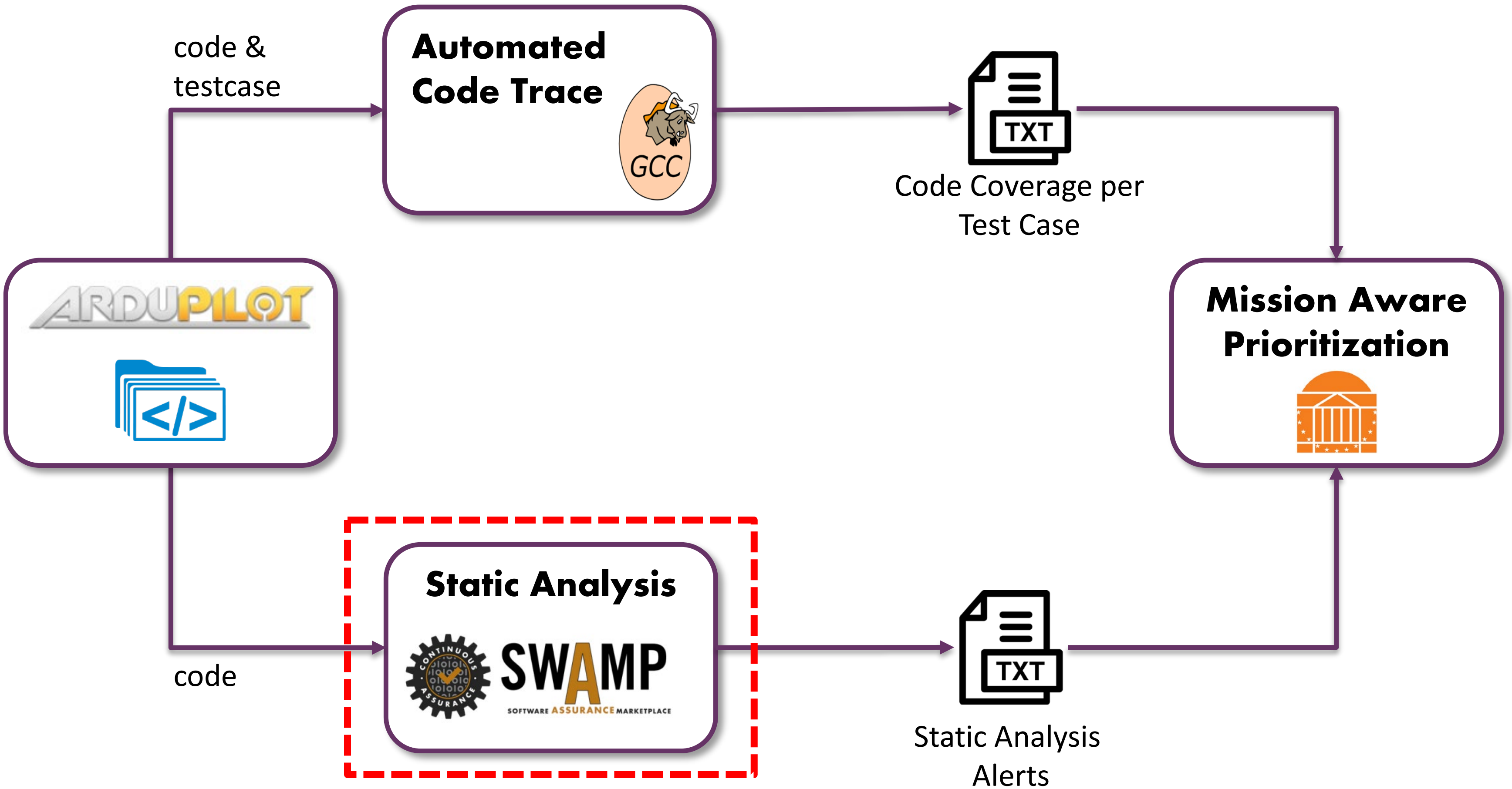
Component Name	KLOC (C/C++)
Helicopter	15
Fixed Wing Plane	13
Land Rover	6
Submarine	6
Shared Libraries	177
Total	217

SITL (software in the loop) simulator allows ArduPilot execution without vehicle hardware.

ArduPilot's **Auto Test** suite allows for the creation of repeatable tests of autopilot behavior based on SITL simulator.

https://github.com/ArduPilot/ardupilot

Initial Research Context



SWAMP Overview

Led by the Morgridge Institute for Research in Madison WI, the Software Assurance Marketplace (SWAMP) is a no-cost, cloud service that provides Static Code Analysis to developers and researchers.

Available SA Tools (C/C++)	Type
Clang	Open Source
Cppcheck	Open Source
CodeSonar	Commercial
Coverity	Commercial
Code DX (Consolidated Results Viewer)	Commercial

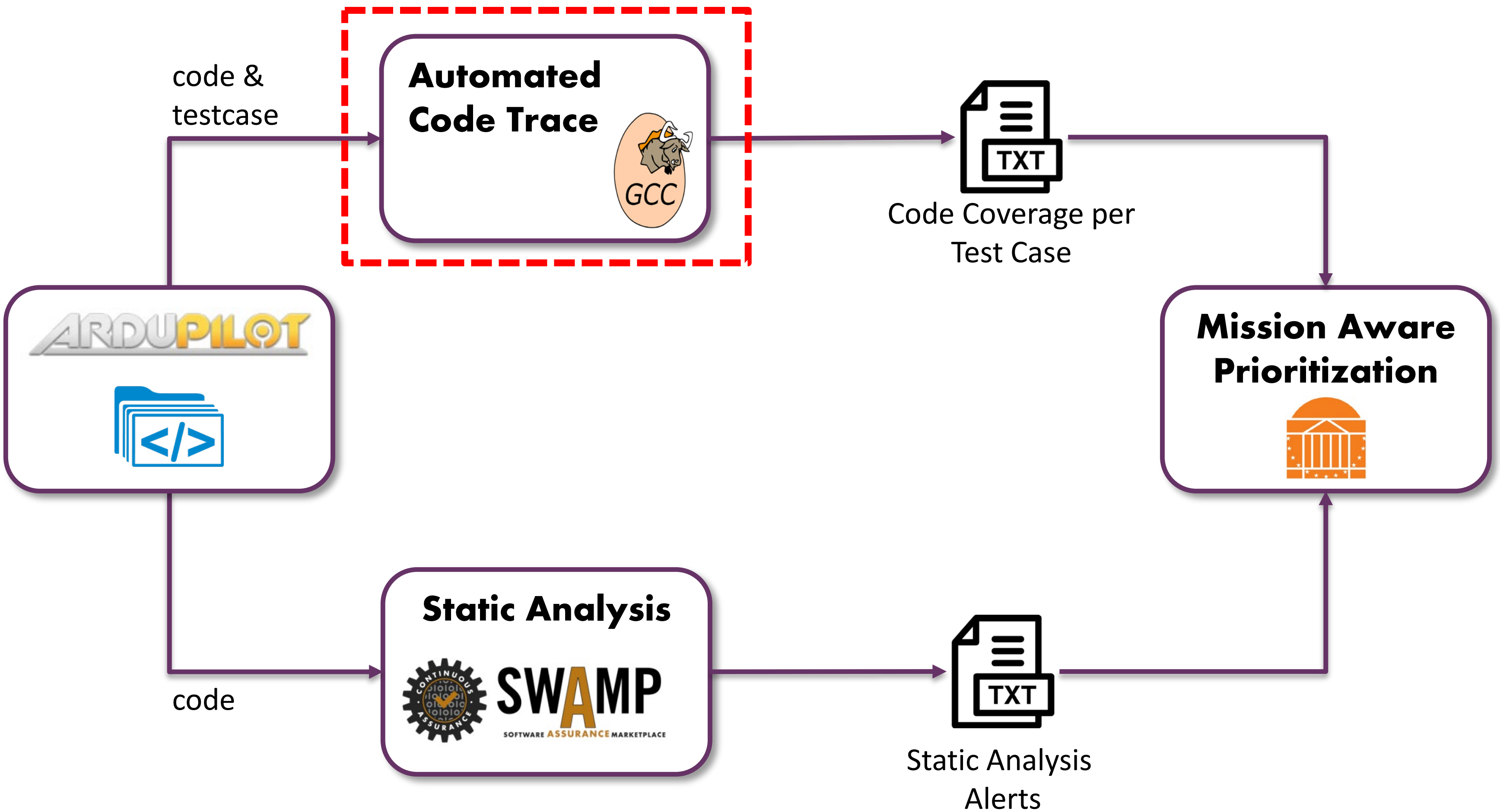
- Variation in SA tool results are similar to findings of NIST Static Analysis Tool Exposition (SATE).
- Alert Density of 2.0 equates to 2,000 alerts for a 1 million LOC project.

Code DX Results Summary[1]	Alert Count[2]	Alert Density[3]
Tool A	439	2.02
Tool B	778	3.59
Tool C	66	.30
Tool D	43	.20
(ArduPilot) Total Alerts[4]	**1,326**	

[1]Licensing terms prevent publication of tool specific results (list is reordered)
[2]Little to no overlap between tools
[3]Alerts / 1,000 Lines of Code (LOC)
[4]Over full code base
(all vehicle modes, test drivers, link protocol libs, etc.)

https://samate.nist.gov/SATE5/SATE5%20Report.pdf
https://www.mir-swamp.org/
https://scan.coverity.com/

Initial Research Context



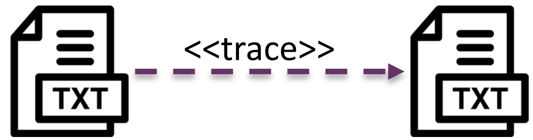
Code Trace Overview

SYSTEMS ENGINEERING RESEARCH CENTER

UNIVERSITY of VIRGINIA ENGINEERING

Code is instrumented at *Compile-time* to output function / line execution counts per code file at *Run-time*.

Each Test Case captures a Coverage Report

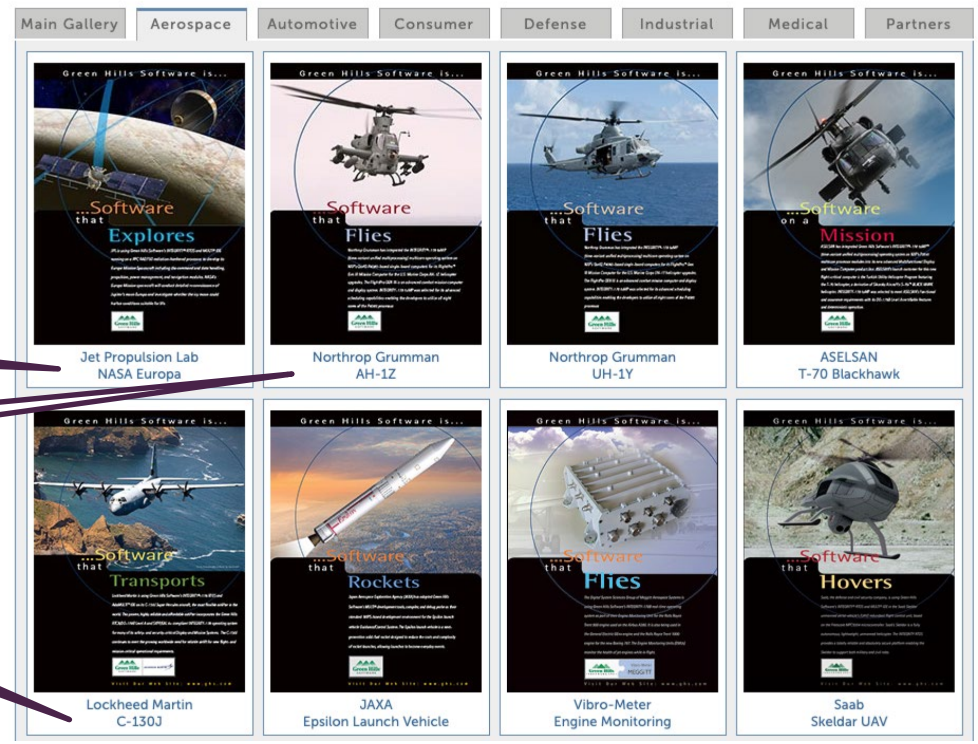


<<test #n>> <<coverage report #n>>

Example Code Coverage Tools

- GNU C Compiler (GCC) – GCOV
- LLVM – X-ray
- Greenhills – Time Machine : Coverage Trace

Example Greenhills Users



Jet Propulsion Lab

Northrop Grumman

Lockheed Martin

https://gcc.gnu.org/onlinedocs/gcc/Gcov.html
https://llvm.org/docs/XRay.html
https://www.ghs.com/products/timemachine.html

ArduPilot Code Trace Example

ArduPilot Helicopter SITL build includes:

- **7,546 Functions**
- **69,743 LOC[1]**

Merged Coverage Results for "Common" Code Executed Across Test Cases.

Test Case	Test Type	Functions Executed	LOC Executed
Loiter to Altitude	Safety	533	4291
Battery Failsafe	Component Failure	329	2927
Camera Control	Component Failure	391	3377
GPS Glitch	Out of Spec	373	3287

[1]Does not include #define, #if <def> compiler directive, test code

Common Across:	Functions Executed	LOC Executed
2 (or more) Test Cases	376	3319
3 (or more) Test Cases	343	2914
4 Test Cases	245	2260

ArduPilot Code Trace – HTML Report

GCC includes tooling for an HTML view - to assist visualization of results.

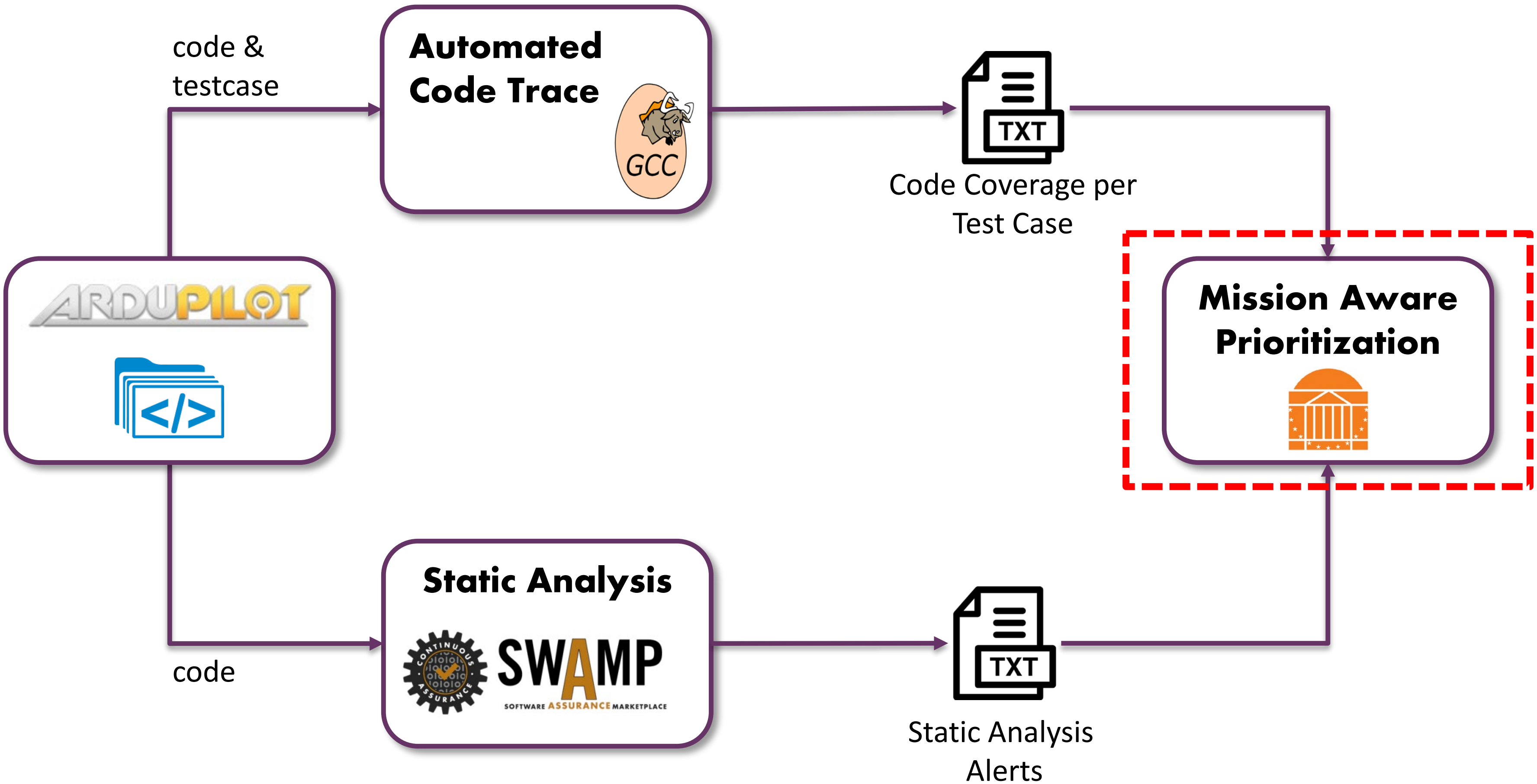
<u>“Common” (across 4 test cases) libraries include:</u>

- Attitude Control
- Battery Monitor
- Etc.

			Hit	Total	Coverage
Current view:	top level				
Test:	lcov-COMMON-4.info	**Lines:**	2260	69743	3.2 %
Date:	2019-09-24 13:59:26	**Functions:**	245	7546	3.2 %

Directory	Line Coverage			Functions	
	....	% Lines Executes	Executed / Total	% Functions Executes	Executed / Total
AP_Motors		7.9 %	108 / 1374	7.4 %	9 / 122
AP_Arming		10.1 %	40 / 397	19.4 %	7 / 36
AC_PID		12.1 %	44 / 365	16.2 %	13 / 80
AP_NavEKF2		12.9 %	617 / 4778	2.9 %	8 / 272
AP_SmartRTL		17.7 %	66 / 372	21.9 %	7 / 32
AP_InertialNav		20.7 %	6 / 29	22.2 %	2 / 9
AP_BattMonitor		22.6 %	140 / 619	23.2 %	22 / 95
AC_AttitudeControl		25.3 %	287 / 1136	23.5 %	46 / 196
AP_Stats		31.2 %	20 / 64	16.7 %	2 / 12

Initial Research Context



ArduPilot Test Case Correlation

Static Analysis Alerts are Attenuated by Correlation to High Priority Mission Test Case Code Files (4 Static Analysis Tools with 4 Test Cases)

SA Tool	Total Alerts	ArduPilot SA Alerts Correlated to Code Coverage Test Files			
		Alerts found in Files Executed in at least 1 Test Case	Attenuation	Alerts found in Files Executed by all 4 Test Cases	Attenuation
Tool A	439	91	79%	75	83%
Tool B	778	204	74%	181	77%
Tool C	66	3	95%	0	100%
Tool D	43	12	72%	12	72%
Total	**1,326**	**310**	**77%**	**268**	**80%**

Initial Findings:

- For ArduPilot, filtering alerts based on Mission Priority test cases, provides attenuation of ~ 75%
- A small number of test cases can provide reasonable alert filtering results

Additional Research Efforts

- Conduct experiments with DoD application SW, engaging static analysis personnel to determine productivity advances
- Engage with DT and OT testing organizations to determine how one might address this opportunity into test programs
- Use of Natural Language AI technology to prioritize the cyber risk-related test cases through integration of documentation derived from Static Analysis, Hazard Analysis and prior test cases results
- Consider application to non-weapon related physical systems and IT systems
- Army interest in possible integration with SEI SCALe Tool