Welcome to the STONESOUP Proposers' Day

Time	Title	Speaker
08:30 – 08:35	Administrative Remarks	
08:35 – 09:00	IARPA Overview	Dr. Lisa Porter IARPA Director
09:00 – 10:30	STONESOUP Overview and Q&A	Dr. Carl Landwehr STONESOUP Program Manager
10:30 – 10:45	Break	
10:45 – 12:15	Proposers' 5-minute Briefings	LGS Innovations Honeywell International Rether Networks, Inc. Symantec Assured Information Security, Inc. University of Illinois at Urbana-Champaign Kestrel Technology, LLC Palo Alto Research Center Telcordia Technologies GrammaTech, Inc. MIT CSAIL / University of Pennsylvania SEAS Penn State University / U. Wisconsin-Madison / CMU Kestrel Institute Southwest Research Institute / University of Texas at Austin Teknowledge IBM
12:15 – 1:15	Lunch Break (Gov't Representatives Depart)	
1:15 – 3:15	Posters, Proposers' Networking and Teaming Discussions	

IARPA

BE THE FUTURE

STONESOUP Proposers Day
IARPA-BAA-09-08

Securely Taking On New Executable Software of Uncertain Provenance

Carl Landwehr
Program Manager
Intelligence Advanced Research Projects Activity (IARPA)
301-226-9100
email: carl.e.landwehr@ugov.gov

Disclaimer

This presentation is provided solely for information and planning purposes.

The Proposers' Day Conference does not constitute a formal solicitation for proposals or proposal abstracts.

Nothing said at Proposers' Day changes the requirements set forth in a BAA.

Any conflict between what is said at Proposers' Day and what is in a BAA will be resolved in favor of the BAA.

No White Papers

- No White Papers will be requested for STONESOUP
- Proposals will be due 45 days after BAA is published
- Take advantage of this time to start developing your ideas

Outline

- Program Overview
- Program Phases, Metrics, and Milestones
- Award Information
- Eligibility Information
- Application Review Information

Question periods sprinkled throughout

STONESOUP
Program Overview

STONESOUP Program Goal

Develop and demonstrate technology to <u>automatically</u>

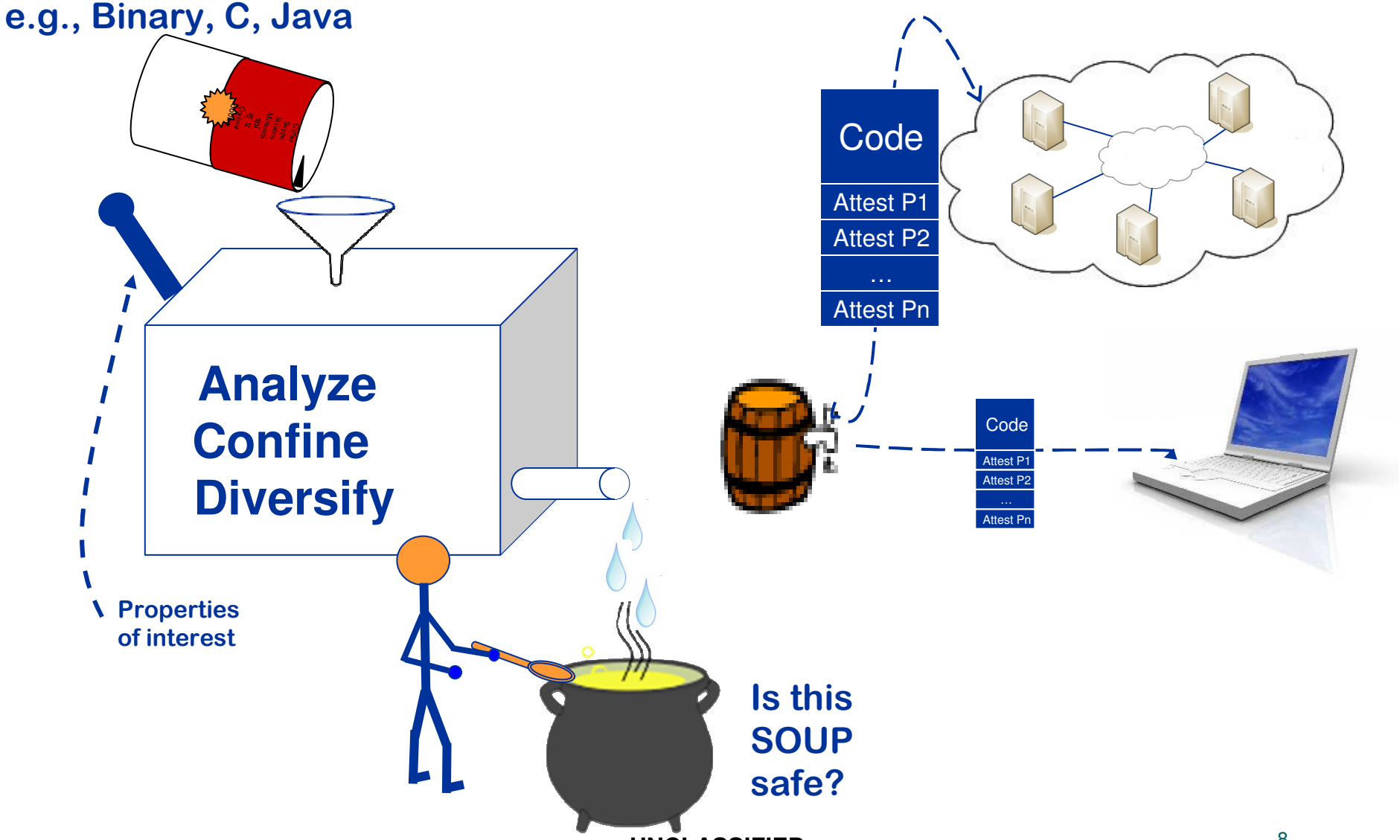
analyze
confine
diversify

software (source or binary)

so that the end users can safely execute software of uncertain provenance

STONESOUP Vision

e.g., Binary, C, Java



Out of Scope

What is the program NOT trying to do?

- **Develop new "secure" programming languages**
- **Prevent all possible attacks**
- **Detect all possible attacks**
- **Advance network security**

Code Analysis Today

Static analysis tools can identify many software weaknesses, for example:

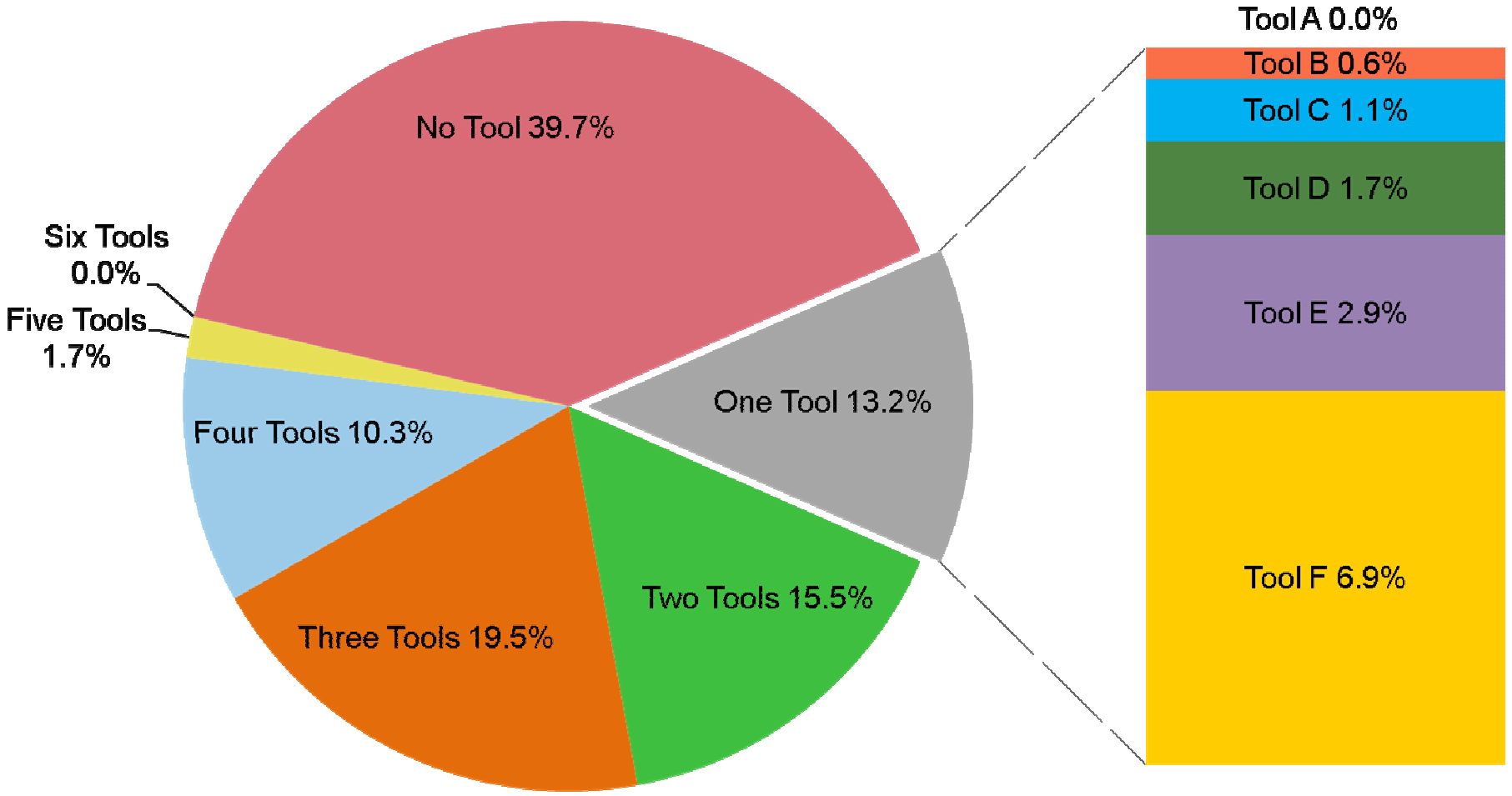
- Bug pattern detection based on abstract syntax trees
- Strong type checking (type casting vulnerabilities, uninitialized variable use)
- Memory allocation checking (memory leaks, deallocation of unallocated memory)
- Dead (unreachable) code detection
- Security checking
  - Buffer overflow/underflow
    - Stack overflows
    - Heap overflows
  - Integer overflow/underflow
  - Tainted data
  - Error path problems
  - Locking problems

Academic and commercial tools are available that can perform some or all of these functions for C, C++, Java, and binaries, but ….

Code Analysis Today

Java "Breadth" Case Coverage

Fraction of 177 test cases caught by zero of more of 6 Java code analysis tools

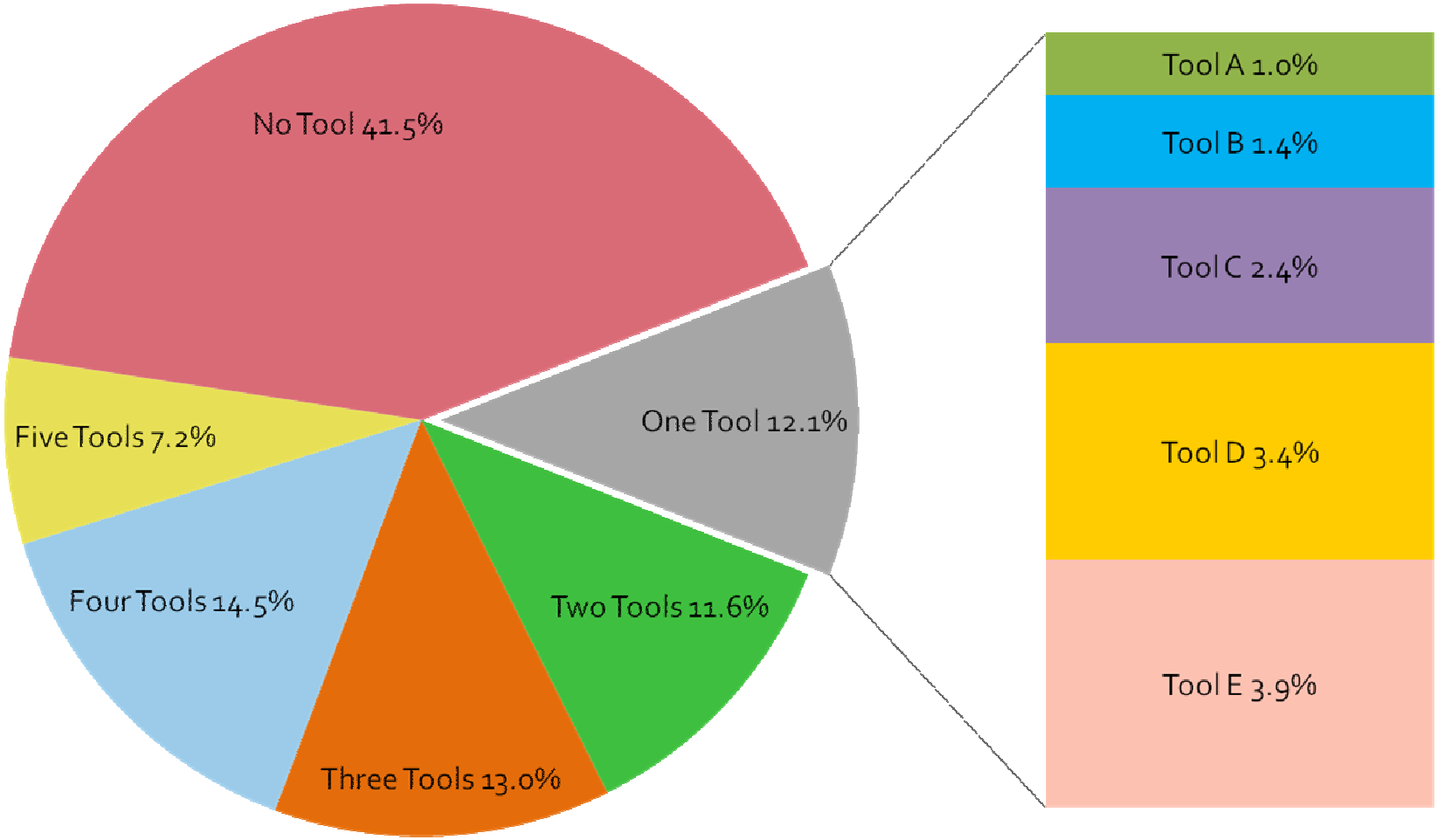


Test cases derived from 112 Common Weakness Enumeration (CWE) classes

Code Analysis Today

C / C++ "Breadth" Case Coverage

Fraction of 210 test cases caught by zero of more of 5 C / C++ code analysis tools



Test cases derived from 103 Common Weakness Enumeration CWE classes

Confinement Today

Mechanisms for confining the extent of damage caused by programming errors/weaknesses include:

- "Core constants" for uninitialized storage
- Bounds checks on array accesses
- Stack "canaries"
- Inline reference monitors
- Non-executable memory
- Process bounds
- Virtual machine bounds
- Processor modes

Limitations: these mechanisms can provide protection/assurance only for the current run of the program.

They operate either fail-stop or fail-oblivious, leaving program either vulnerable to DoS attack or open to unpredictable behavior

Diversification Today

Mechanisms to diversify programs today

Primarily at the memory layer:

- Unix versions:
  - Linux: PaX, grsecurity: project to add Address Space Layout Randomization (ASLR), non-executable pages, and other security features to Linux; started ~2000 claims influence on several OS's
  - RedHat: Execshield includes randomization of load addresses
  - FreeBSD: option to randomize load address
  - MacOS 10.5 – randomizes library offsets
- Windows Vista: includes ASLR Stack randomization, heap randomization

Limitations: hard to provide guarantees -- diversified program may have the same vulnerabilities as the original, just in different places.

Integrating Analysis, Confinement, Diversification Today

Analysis, confinement, and diversification are rarely integrated today

Compilers have often included “debug” modes that integrate confinement (e.g. array bounds checking) with generated code

Some offer options to generate code with some added confinement / randomizaton
Microsoft –/GS, /SafeSEH compiler options – add stack canaries, randomize library offsets
GCC SSP (stack smashing protector – canaries)

STONESOUP Research Directions

Develop/exploit new methods within each area and integrate into a single comprehensive solution

Analysis:

- Improve vulnerability coverage
- Reduce false positives, false negatives

Confinement:

- Exploit newly available mechanisms (e.g. virtualization)
- Reduce performance penalties
- Improve breadth of coverage

Diversification:

- Account for / leverage techniques in use by malware writers
- Seeking approaches that do not depend on secrecy of method, only on secrecy of key
- Seeking quantification of effectiveness

STONESOUP Research Directions

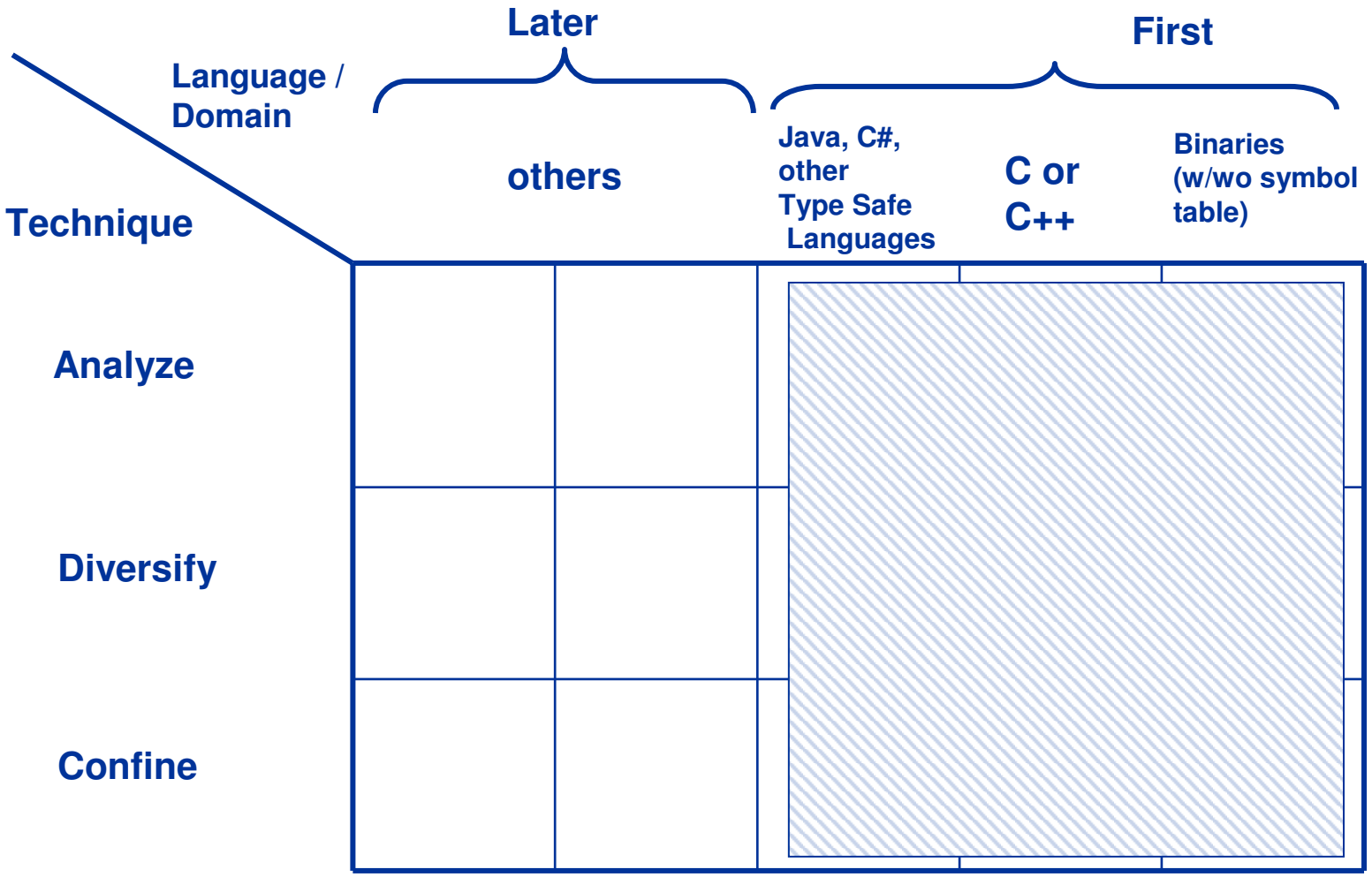
Exploit the synergy of integrating all three approaches:

- Tools for analysis (only) necessarily generate false positives and false negatives
- Confinement approaches are necessarily limited to the current run of the program
- Diversification is inherently “best effort”

Each of these approaches has advanced significantly in recent years, independently of the others

STONESOUP seeks innovations that exploit the complementary aspects of these approaches to yield a major reduction in software vulnerability

Languages of Interest



Questions?

STONESOUP
Program Phases, Milestones, and Metrics

Program Structure: Phase I

Phase I: Technical Feasibility (18 months)

Separate teams, each focused on at least one language domain (e.g., binary; C or C++; Java or C#)

- Development of initial toolsets for code analysis, confinement, diversification
- Demonstrate successful processing of a few significant weakness types indexed to the Common Weakness Enumeration (CWE) to be assured absent / confined / diversified (feasibility)
- Evaluation on Phase I dataset: many small programs / code fragments, several moderately sized programs: 10K – 100K SLOC

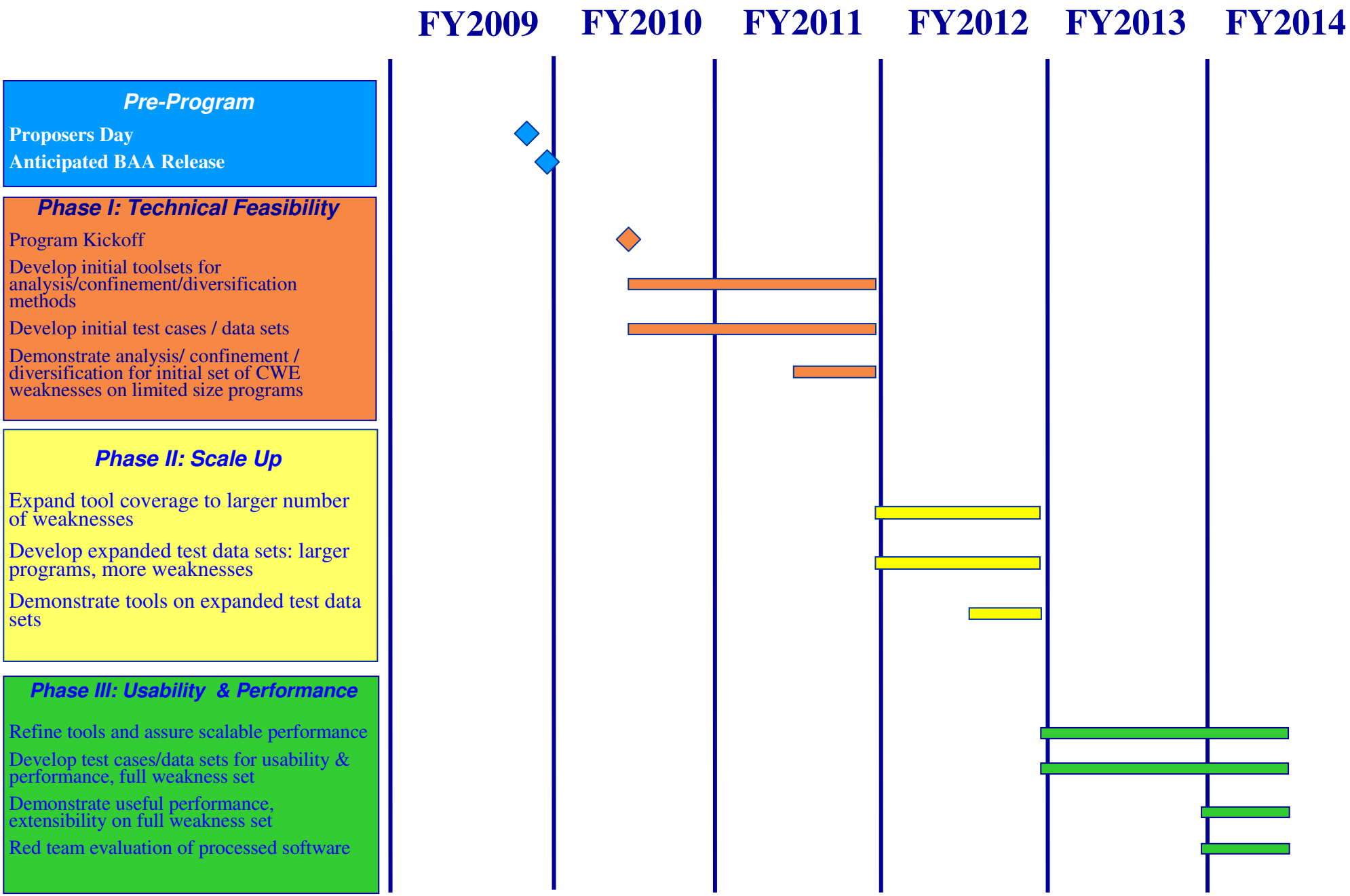
Program Structure: Phases II and III

Phase II Scale up (12 months)

- Evaluation on Phase II dataset
  - Expanded # weaknesses
  - Larger program sizes: 100K – 1M SLOC

Phase III Usability and Performance (18 months)

- Significantly expanded weakness set
- Performance: processed software runs with no more than 10% average longer execution time than unmodified version
- Red team evaluation of processed software


FY2009
FY2010
FY2011
FY2012
FY2013
FY2014
Pre-Program
Proposers Day
Anticipated BAA Release
Phase I: Technical Feasibility
Program Kickoff
Develop initial toolsets for analysis/confinement/diversification methods
Develop initial test cases / data sets
Demonstrate analysis/ confinement / diversification for initial set of CWE weaknesses on limited size programs
Phase II: Scale Up
Expand tool coverage to larger number of weaknesses
Develop expanded test data sets: larger programs, more weaknesses
Demonstrate tools on expanded test data sets
Phase III: Usability & Performance
Refine tools and assure scalable performance
Develop test cases/data sets for usability & performance, full weakness set
Demonstrate useful performance, extensibility on full weakness set
Red team evaluation of processed software

Metrics and Milestones

Three classes of programming languages to be addressed:

A. Type-safe source (e.g., Java, C#)

B. Non type-safe source (e.g., C, C++)

C. Binary

STONESOUP program will develop data sets (program fragments, programs, or systems of programs) for each phase and each program class with graduated sets of seeded vulnerabilities drawn from different CWE classes

In successive phases, data sets will increase in

- program size
- program complexity

For benign inputs, for all programs, no change in program behavior

Other Government Agencies, Federally Funded Research and Development Centers (FFRDCs) or University Affiliated Research Centers (UARCs) may be used to assist the STONESOUP PM in developing test scenarios and data sets and conducting T&E

Metrics and Milestones

Phase I Targets

Language Class	Percent of submitted programs successfully processed	Percent of seeded vulnerabilities rendered unexploitable*
A: Type-safe source	100	75
B: Non type-safe source	90	75
C: Binary	75	75

* either (a) detected, (b) prevented through confinement, or (c) rendered unexploitable through diversification

Metrics and Milestones

Phase II Targets

Language Class	Percent of submitted programs successfully processed	Percent of seeded vulnerabilities rendered unexploitable
A: Type-safe source	100	90 (Phase I CWE classes) 80 (Phase II CWE classes)
B: Non type-safe source	90	90 (Phase I CWE classes) 80 (Phase II CWE classes)
C: Binary	75	90 (Phase I CWE classes) 80 (Phase II CWE classes)

Metrics and Milestones

Phase III Targets

Language Class	Percent of submitted programs successfully processed	Percent of seeded vulnerabilities rendered unexploitable
A: Type-safe source	100	95 (Phase I&II CWE classes) 90 (Phase III CWE classes)
B: Non type-safe source	90	95 (Phase I&II CWE classes) 90 (Phase III CWE classes)
C: Binary	75	95 (Phase I&II CWE classes) 90 (Phase III CWE classes)

Performance of modified programs not more than 10% slower than unmodified

Red team evaluation of processed software, including usability

Questions?

STONESOUP
Award Information

Award Plan

4-year Program starting 2Q FY2010

- Phase I – Base Period -- 18 months
- Phase II – Option 1 Period -- 12 months
- Phase III – Option 2 Period -- 18 months

Criteria for advancing to next phase: sufficient progress in achieving prior phase metrics

Multiple awards anticipated in each language category (binary, non-type-safe, type-safe), depending upon

- quality of the proposals received
- availability of funds

STONESOUP
Eligibility Information

STONESOUP Eligibility Information

All responsible sources capable of satisfying the STONESOUP program goals may submit a proposal or join with others in submitting proposals.

Proposals must address a comprehensive solution integrating methods for analysis, confinement and diversification.

Collaborative efforts/teaming among potential performers is encouraged where it enables program success.

Foreign organizations and/or individuals may participate
Must comply with Non-Disclosure Agreements, Security Regulations, Export Control Laws, etc, as appropriate.

Ineligible Organizations

Other Government Agencies,

Federally Funded Research and Development Centers (FFRDCs),

University Affiliated Research Centers (UARCs), and

Any other similar type of organization that has a special relationship with the Government, that gives them access to privileged and/or proprietary information or access to Government equipment or real property,

are NOT eligible to submit proposals or participate as team members under proposals submitted by eligible entities.

STONESOUP
Evaluation Criteria

STONESOUP Evaluation Criteria

Evaluation criteria in descending order of importance:

- **Overall Scientific and Technical Merit**
- **Effectiveness of Proposed Work Plan**
- **Relevance to IARPA Mission and STONESOUP Program Goals**
- **Relevant Experience and Expertise**
- **Cost Realism**

STONESOUP Can Make a Difference!

Automate software evaluation and make it more effective

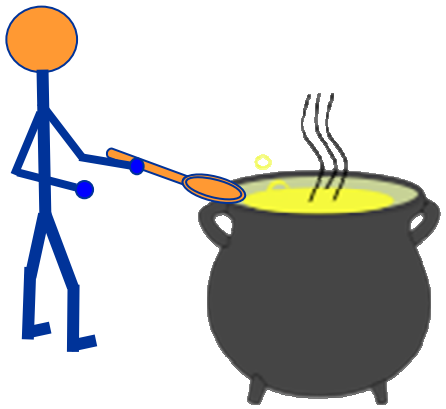
Provide integrated tools to developers so that

- Software is harder to exploit from the beginning
- Customers can know attributes of the software they receive other than its provenance

STONESOUP

Thank you!

Final questions?



Point of Contact

Dr. Carl Landwehr
Program Manager
IARPA, Safe and Secure Operations Office
Office of the Director of National Intelligence
Intelligence Advanced Research Projects Activity
Washington, DC 20511

Phone: 301-226-9108
Fax: 301-226-9137

Electronic mail: dni-iarpa-baa-09-08@ugov.gov
(include IARPA-BAA-09-08 in the Subject Line)
Website: www.iarpa.gov