

Software Assurance: Crippling Coming Cyberassaults

Dr. Paul E. Black

National Institute of Standards and Technology

<http://samate.nist.gov/>

paul.black@nist.gov

Report Documentation Page

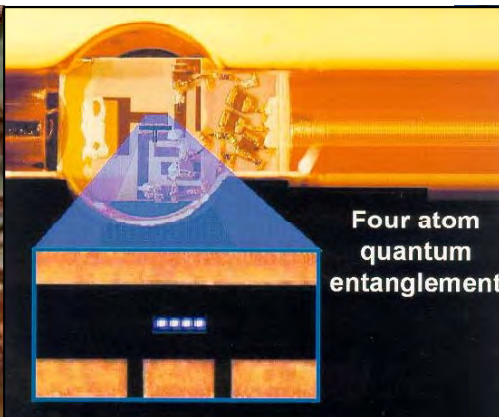
Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE APR 2010		2. REPORT TYPE		3. DATES COVERED 00-00-2010 to 00-00-2010	
4. TITLE AND SUBTITLE Software Assurance: Crippling Coming Cyberassaults				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Institute of Standards and Technology, 100 Bureau Dr # 1000, Gaithersburg, MD, 20899				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES Presented at the 22nd Systems and Software Technology Conference (SSTC), 26-29 April 2010, Salt Lake City, UT.					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 29	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

What is NIST?

- **U.S. National Institute of Standards and Technology**
- **A non-regulatory agency in Dept. of Commerce**
- **3,000 employees + adjuncts**
- **Gaithersburg, Maryland and Boulder, Colorado**
- **Primarily research, not funding**
- **Over 100 years in standards and measurements: from dental ceramics to microspheres, from quantum computers to fire codes, from body armor to DNA forensics, from biometrics to text retrieval.**



The NIST SAMATE Project

- **Software Assurance Metrics And Tool Evaluation (SAMATE) project is sponsored in part by DHS**
- **Current areas of concentration**
 - Web application scanners
 - Source code security analyzers
 - Static Analyzer Tool Exposition (SATE)
 - Software Reference Dataset
 - *Software labels*
 - *Malware research protocols*
- **Web site <http://samate.nist.gov/>**



Software Reference Dataset

The screenshot shows the SAMATE Software Reference Dataset website. At the top, there are logos for SAMATE, NIST (National Institute of Standards and Technology), and DHS National Cyber Security Division. Below the logos is a navigation bar with links: SRD Home, View / Download, Search / Download, More Downloads, Submit, and Test Suites. The main content area is divided into two tabs: 'Extended Search' and 'Source Code Search'. The 'Source Code Search' tab is active, showing a search form on the left and a list of weaknesses on the right. The search form includes fields for 'Number (Test case ID)', 'Description contains', 'Contributor/Author', 'Bad / Good' (dropdown), 'Language' (dropdown), 'Type of Artifact' (dropdown), 'Status' (Candidate and Approved checkboxes), 'Weakness' (dropdown), 'Code complexity' (dropdown), and 'Date' (radio buttons for Any, Before, After, and a date input field). A 'Search Test Cases' button is at the bottom of the form. The list of weaknesses on the right includes: Any..., CWE-485: Insufficient Encapsulation, CWE-388: Error Handling, CWE-389: Error Conditions, Return Values, Status Codes, CWE-254: Security Features, CWE-227: Failure to Fulfill API Contract (API Abuse), CWE-019: Data Handling, CWE-361: Time and State, CWE-398: Indicator of Poor Code Quality, CWE-470: Use of Externally-Controlled Input to Select Classes, CWE-465: Pointer Issues, CWE-411: Resource Locking Problems, CWE-401: Failure to Release Memory Before Removing Last, CWE-415: Double Free, CWE-416: Use After Free, and CWE-417: Channel and Path Errors.

- Public repository for software test cases
- Almost 1800 cases in C, C++, Java, and Python
- Search and compose custom Test Suites
- Contributions from Fortify, Defence R&D Canada, Klocwork, MIT Lincoln Laboratory, Praxis, Secure Software, etc.

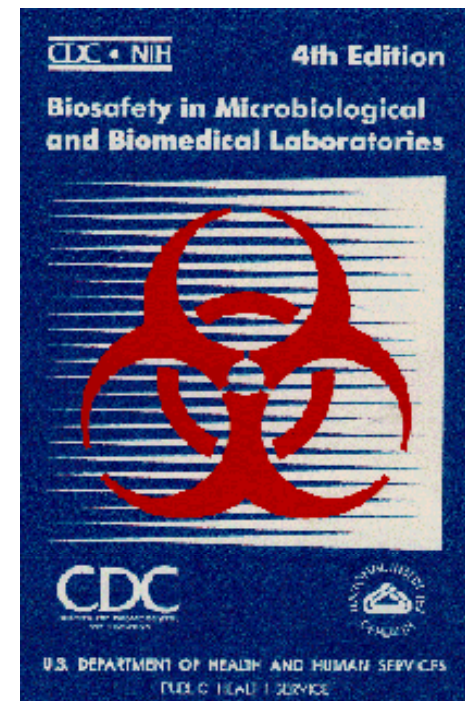
Software Facts Label

- **Software Facts should:**
 - Voluntary
 - Absolutely simple to produce
 - Have a standard format for other claims
- **What could be easily supplied?**
 - Source available? Yes/No/Escrowed
 - Default installation is secure?
 - Accessed: network, disk, ...
 - What configuration files? (registry, ...)
 - Certificates (eg, "No Severe weaknesses found by CodeChecker ver. 3.2")
- **Cautions**
 - A label can give false confidence.
 - A label shut out better software.
 - Labeling diverts effort from real improvements.

Software Facts	
Name	InvadingAlienOS
Version	1996.7.04
Expected number of users	15
<hr/>	
Modules	5 483 Modules from libraries 4 102
<hr/>	
% Vulnerability	
<hr/>	
Cross Site Scripting	22
Reflected	12
Stored	10
SQL Injection	2
Buffer overflow	5
Total Security Mechanisms	284
Authentication	15
Access control	3
Input validation	230
Encryption	3
AES 256 bits, Triple DES	
<hr/>	
Report security flaws to: ciwnmcyi@motherhip.milkyway	
<hr/>	
Total Code	3.1415×10 ⁹ function points
C	1.1×10 ⁹ function points
Ratfor	2.0415×10 ⁹ function points
Test Material	2.718×10 ⁶ bytes
Data	2.69×10 ⁶ bytes
Executables	27.18×10 ³ bytes
Documentation	12 058 pages
Tutorial	3 971 pages
Reference	6 233 pages
Design & Specification	1 854 pages
<hr/>	
Libraries: Sun Java 1.5 runtime, Sun J2EE 1.2.2, Jakarta log4j 1.5, Jakarta Commons 2.1, Jakarta Struts 2.0, Harold XOM 1.1rc4, Hunter JDOMv1	
<hr/>	
Compiled with gcc (GCC) 3.3.1	
<hr/>	
Stripped of all symbols and relocation information.	

Researching Risky Software

- Many people research malware, but there are no widely accepted protocols.
- Biological research has defined levels with associated practices, safety equipment, and facilities.
- Some approaches are
 - Weakened programs (auxotrophs)
 - Programs that **ALERT**
 - Outgoing firewalls
 - Isolated networks



- **Assurance that software is less vulnerable to coming cyberassaults**
- Static and dynamic analysis
- Static Analysis Tool Exposition - 2009 outcomes and 2010 progress

Assurance from three sources

$$A = f(p, s, e)$$

where A is functional assurance, p is process quality, s is assessed quality of software, and e is execution resilience.

p is process quality

- **High assurance software must be developed with care, for instance:**
 - **Validated requirements**
 - **Good system architecture**
 - **Security designed- and built in**
 - **Trained programmers**

s is assessed quality of software

- **Two general kinds of software assessment:**
 - **Static analysis**
 - e.g. code reviews and scanner tools
 - examines code
 - **Testing (dynamic analysis)**
 - e.g. penetration testing, fuzzing, and red teams
 - runs code

e is execution resilience

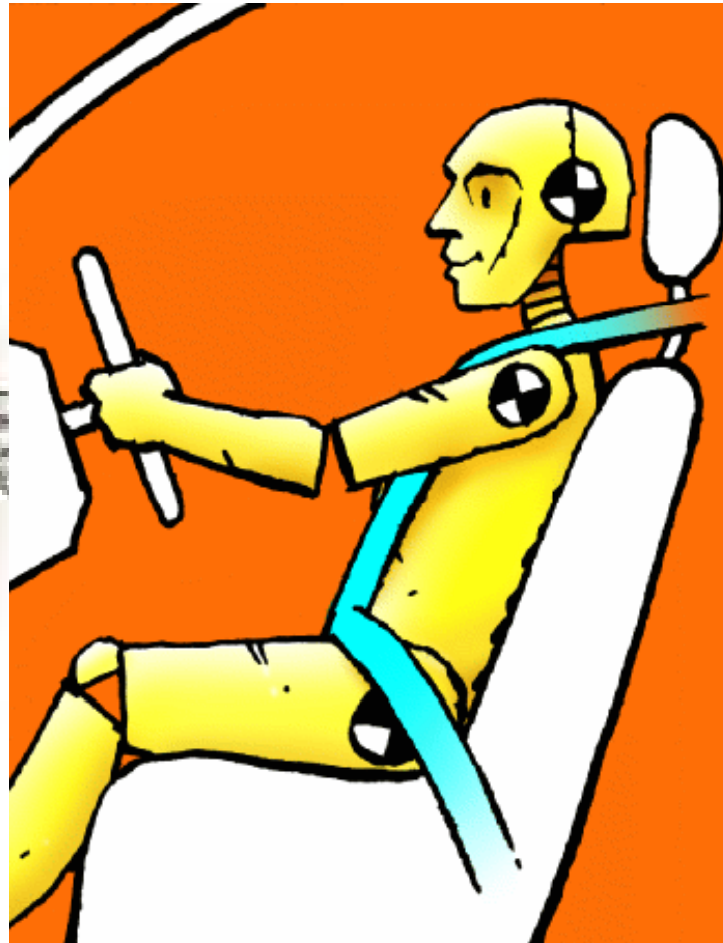
- **The execution platform can add assurance that the system will function as intended.**
- **Some techniques are:**
 - Randomize memory allocation
 - Execute in a “sandbox” or virtual machine
 - Monitor execution and react to intrusions
 - Replicate processes and vote on output

Software analysis is vital

- **Benefits are:**

- **Provide feedback to development process**
- **Build product assurance when process is less visible**
 - **contractors**
 - **open source**
 - **legacy software**
- **Confirm minimum quality for execution**

Analysis is like a seatbelt ...



- Assurance that software is less vulnerable to coming cyberassaults
- **Static and dynamic analysis**
- Static Analysis Tool Exposition - 2009 outcomes and 2010 progress

Comparing Static Analysis with Dynamic Analysis

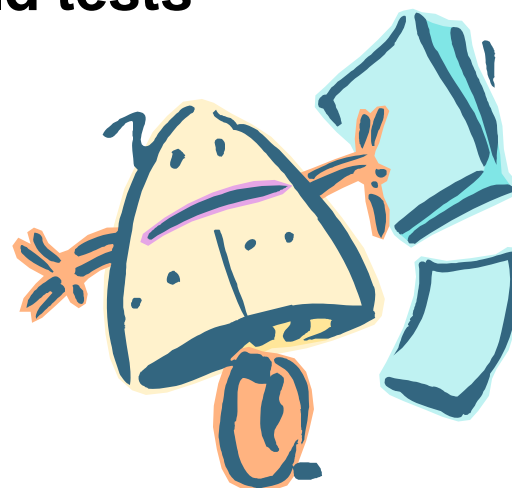
Static Analysis

- Code review
- Binary, byte, or source code scanners
- Model checkers & property proofs
- Assurance case



Dynamic Analysis

- Execute code
- Simulate design
- Fuzzing, coverage, MC/DC, use cases
- Penetration testing
- Field tests



Strengths of Static Analysis

- **Applies to many artifacts, not just code**
- **Independent of platform**
- **In theory, examines *all possible* executions, paths, states, etc.**
- **Can focus on a single specific property**

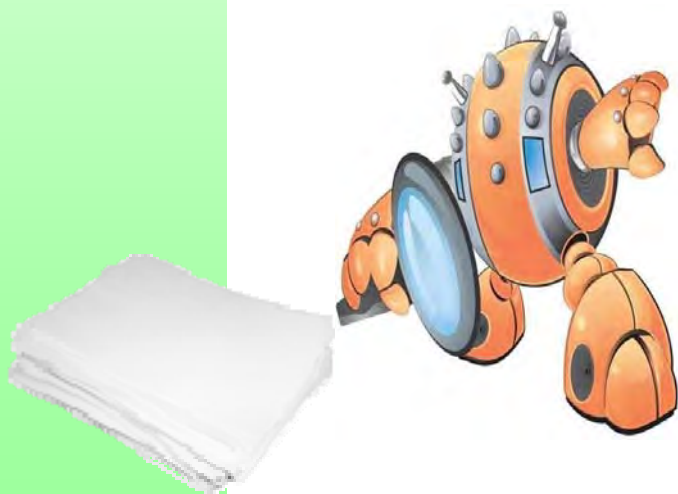
Strengths of Dynamic Analysis

- **No need for code**
- **Conceptually easier - “if you can run the system, you can run the test”.**
- **No (less) need to build or validate models or make assumptions.**
- **Checks installation and operation, along with end-to-end or whole-system.**

Static and Dynamic Analysis Complement Each Other

Static Analysis

- Handles unfinished code
- Can find backdoors, eg, full access for user name “JoshuaCaleb”
- Potentially complete



Dynamic Analysis

- Code not needed, eg, embedded systems
- Has few(er) assumptions
- Covers end-to-end or system tests
- Assess as-installed



- Assurance that software is less vulnerable to coming cyberassaults
- Static and dynamic analysis
- **Static Analysis Tool Exposition - 2009 outcomes and 2010 progress**

Static Analysis Tool Exposition (SATE) Overview

- **Goal: advance research in, and improvement of, static analysis tools for security-relevant defects and speed tool adoption by demonstrating use on real software.**
- **Checkpoints**
 - Participants run tools on Java and C programs we choose
 - NIST-led researchers analyze reports
 - Everyone shares results and observations at a workshop
 - Later release final report and all data
- **<http://samate.nist.gov/SATE.html>**
- **Co-funded by NIST and DHS/NCSD**

SATE Participants

- **2008:**

- Aspect Security ASC
- Checkmarx CxSuite
- Flawfinder
- Fortify SCA
- Grammatech CodeSonar

HP DevInspect
SofCheck Inspector for Java
UMD FindBugs
Veracode SecurityReview

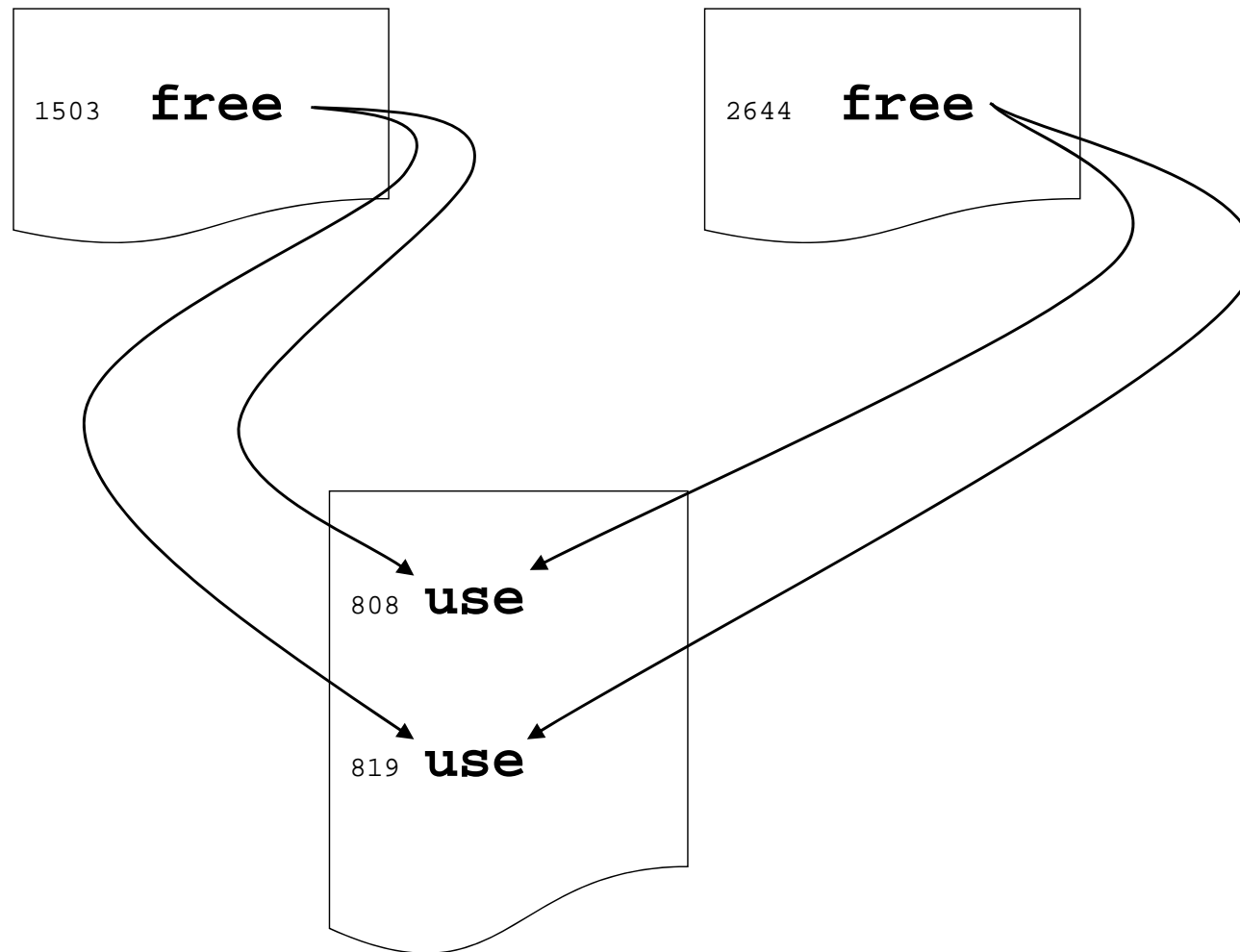
- **2009:**

- Armorize CodeSecure
- Checkmarx CxSuite
- Coverity Prevent
- Grammatech CodeSonar

Klocwork Insight
LDRA Testbed
SofCheck Inspector for Java
Veracode SecurityReview

“Number of bugs” is undefined

Tangled Flow: 2 sources, 2 sinks, 4 paths



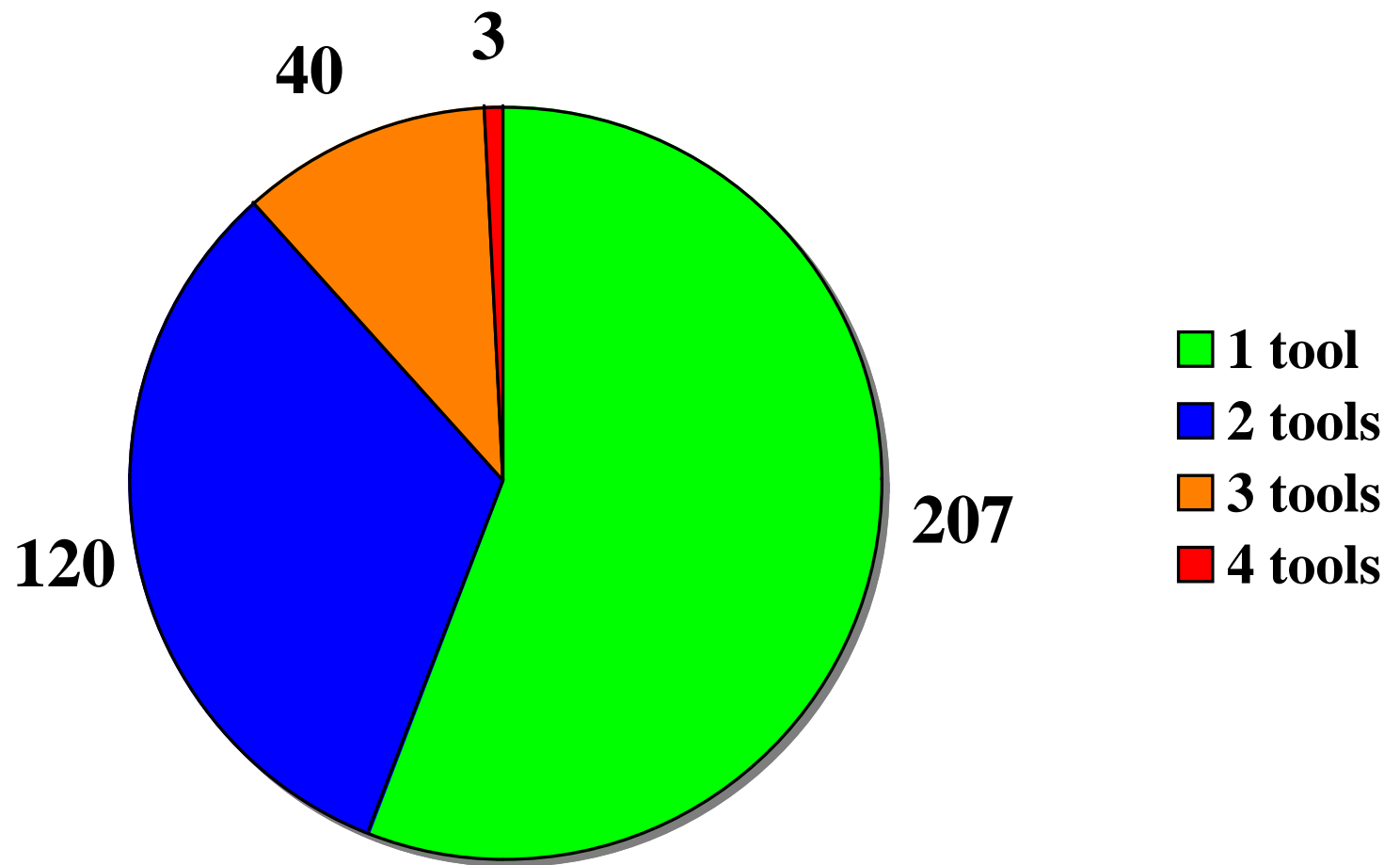
Summary of 2009 tool reports

- **Reports from 18 tool runs**
- **About 20,000 total warnings**
 - but tools prioritize by severity, likelihood
- **Reviewed 521 warnings - 370 were not false**

- **Number of warnings varies a lot by tool and case**
- **83 CWE ids/221 weakness names**

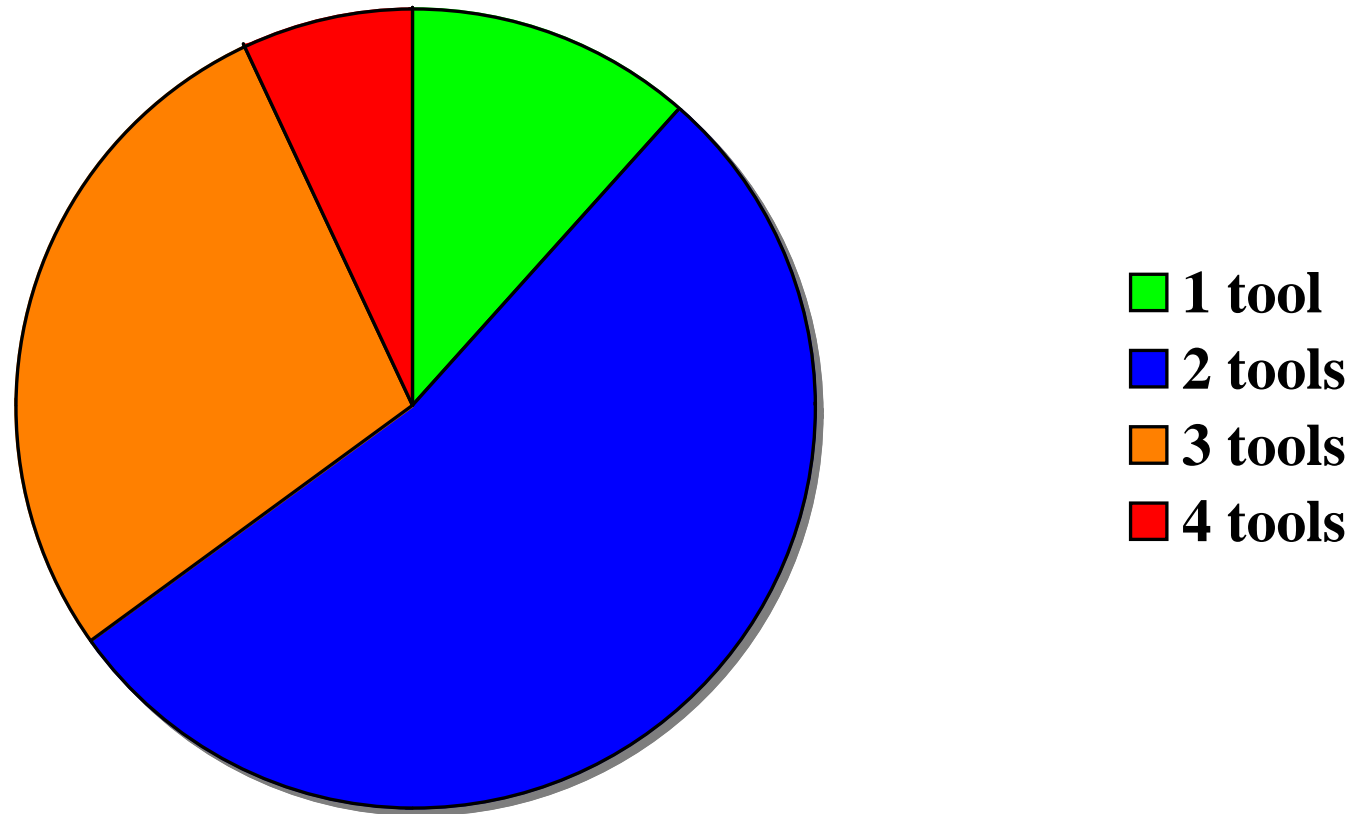
Tools don't report same warnings

Overlap in Not-False Warnings



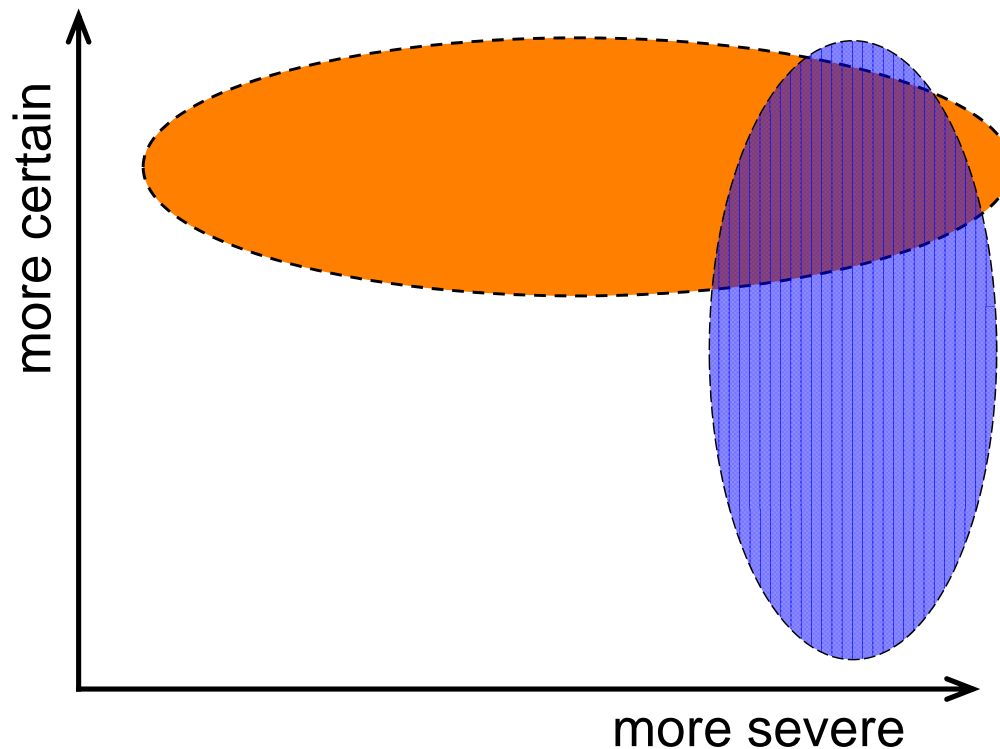
Some types have more overlap

Overlap in Not-False Buffer Errors

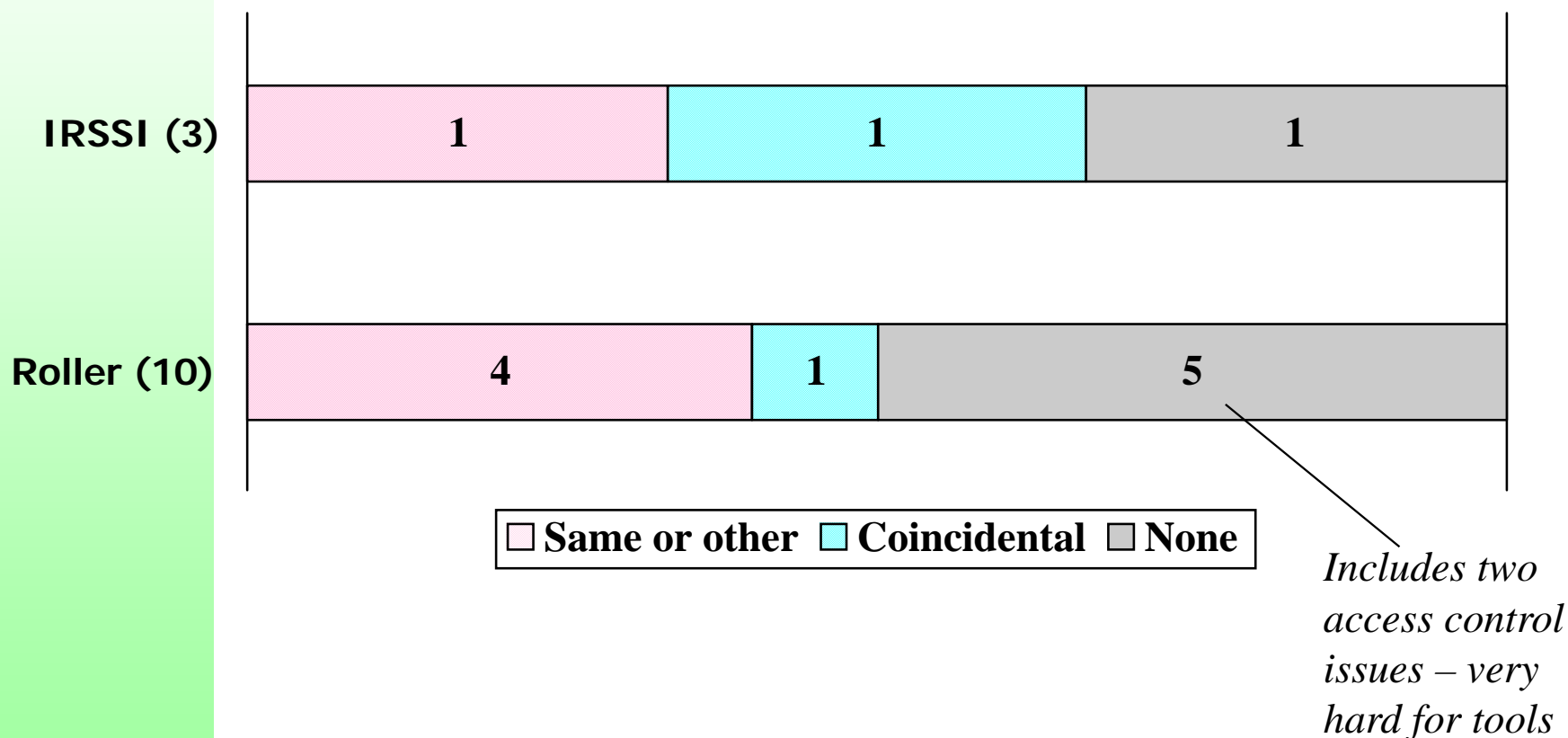


Why don't tools find same things?

- Tools look for different weakness classes
- Tools are optimized differently



Tools find things that people find



SATE 2010 tentative timeline

- ✓ **Hold organizing workshop (12 Mar 2010)**
- ✓ **Recruit planning committee.**
- **Revise protocol.**
- **Choose test sets. Provide them to participants (17 May)**
- **Participants run their tools. Return reports (25 June)**
- **Analyze tool reports (27 Aug)**
- **Share results at workshop (October)**
- **Publish data (after Jan 2011)**

Acronyms

- **CWE - Common Weakness Enumeration**
<http://cwe.mitre.com/>
- **DHS/NCSD - Department of Homeland Security/National Cyber Security Division**
- **MC/DC - Modified Condition/Decision Coverage**
- **SAMATE - Software Assurance Metrics And Tool Evaluation (project at NIST)**
- **SATE - Static Analysis Tool Exposition (annual event)**
- **NIST - National Institute of Standards and Technology**