Source Code Analysis Tool Evaluation

By Jaime Merced

Center for Assured Software

National Security Agency

Outline

- Overview of the project
- Description of the test suite
- Evaluation results

About the project…

- Objective – Measure the accuracy and soundness of static analysis tools for C, C++, and Java source code

Challenges with “real” source

- Difficult to…
  - Determine correctness of individual findings
  - Identify errors not found by tools
  - Find real code that represents a very broad range of targeted code constructs

Artificial Test Cases

- Each test case consists of code that exhibits a coding flaw and one or more safe ways of doing the same thing
- Locations of all errors are documented

Test Suite

- Test case development was subject to constraints of time and money
  - Test cases only used functions available in the standard language libraries for the underlying platforms
  - Very few C++ object-oriented and STL features were used

Example Test Case

```
void CWE134_Uncontrolled_Format_String__scanf_to_printf_01_bad()
{
        char buf[SRC_NO_NTZ_SZ + 1];
        if (scanf(FMT_STR, buf) == 1)
        {
                /* FLAW: buf (obtained from scanf) is passed as the
                   format string to printf */
                printf(buf);
        }
}
```

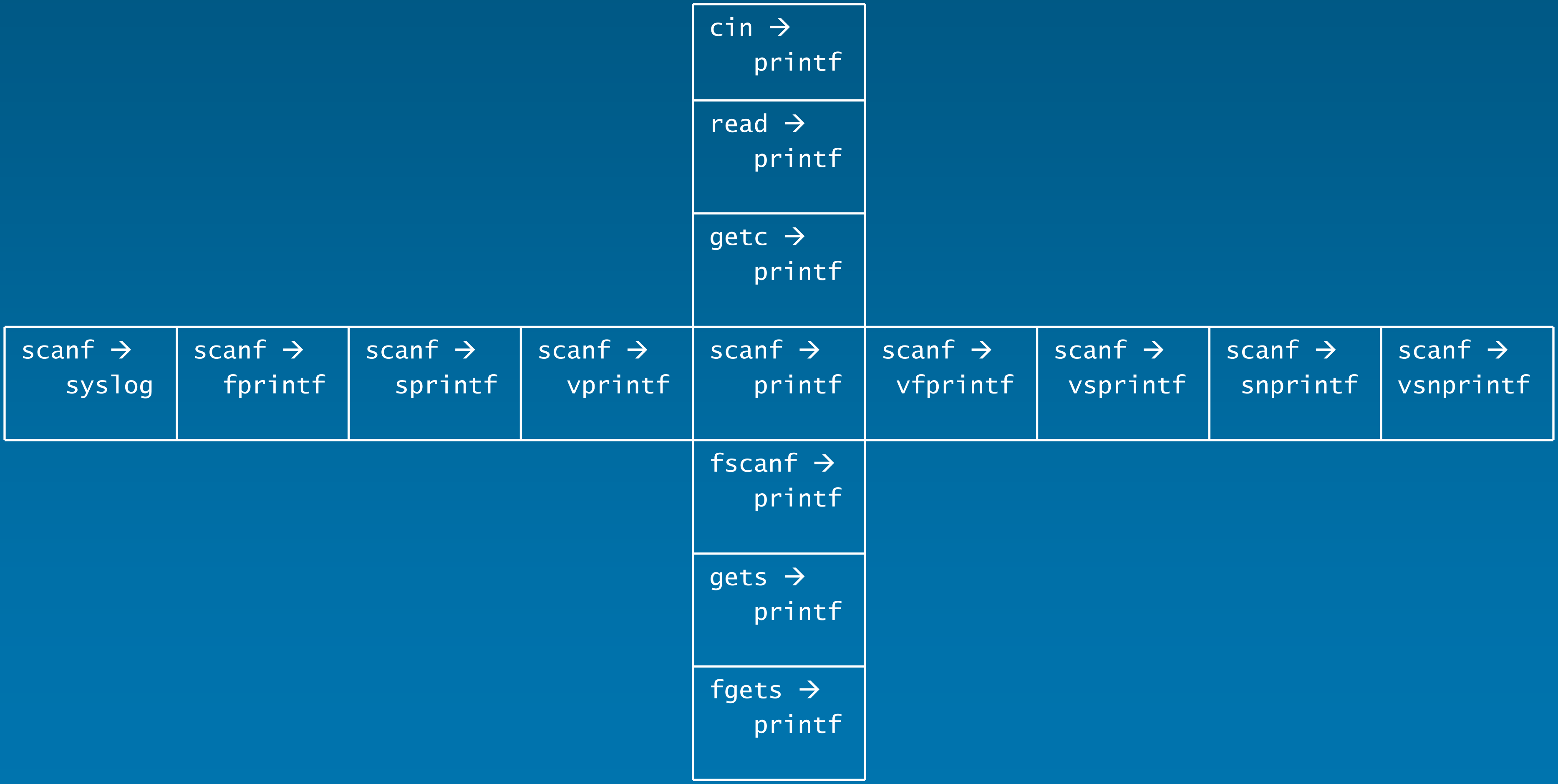
Example Test Case (cont’d)

```
static void good1() {
        /* FIX: Use a static string for a format string */
        printf(“good1\n”)
}
static void good2() {
        /* FIX: Use a variable derived from a static string
           for a format string */
        char * s = “good2”;
        printf(s);
}
static void good3() {
        char buf[SRC_NO_NTZ_SZ + 1];
        if (scanf(FMT_STR, buf) == 1)
        {
                /* FIX: Use %s as a format string and
                  pass buf as an argument */
                printf(“%s”, buf);
        }
}
```

Breadth of Analysis

- Goal: Identify the variety of flaw types and code features that a tool targets
  - Useful in selecting complementary tools
  - Supplements product documentation which may be written for a different purpose
- Method: Use very simple code constructions that vary the data sources, data sinks, and/or the library functions that implement a feature

Breadth of Analysis (cont’d)



cin → printf

read → printf

getc → printf

scanf → syslog

scanf → fprintf

scanf → sprintf

scanf → vprintf

scanf → printf

scanf → vfprintf

scanf → vsprintf

scanf → snprintf

scanf → vsnprintf

fscanf → printf

gets → printf

fgets → printf

Depth of Analysis

- Goal: Identify the extent to which a tool explores more complex data and control flows
- Method: Generate test cases from templates that represent different degrees of complexity

Size of Test Case Suite

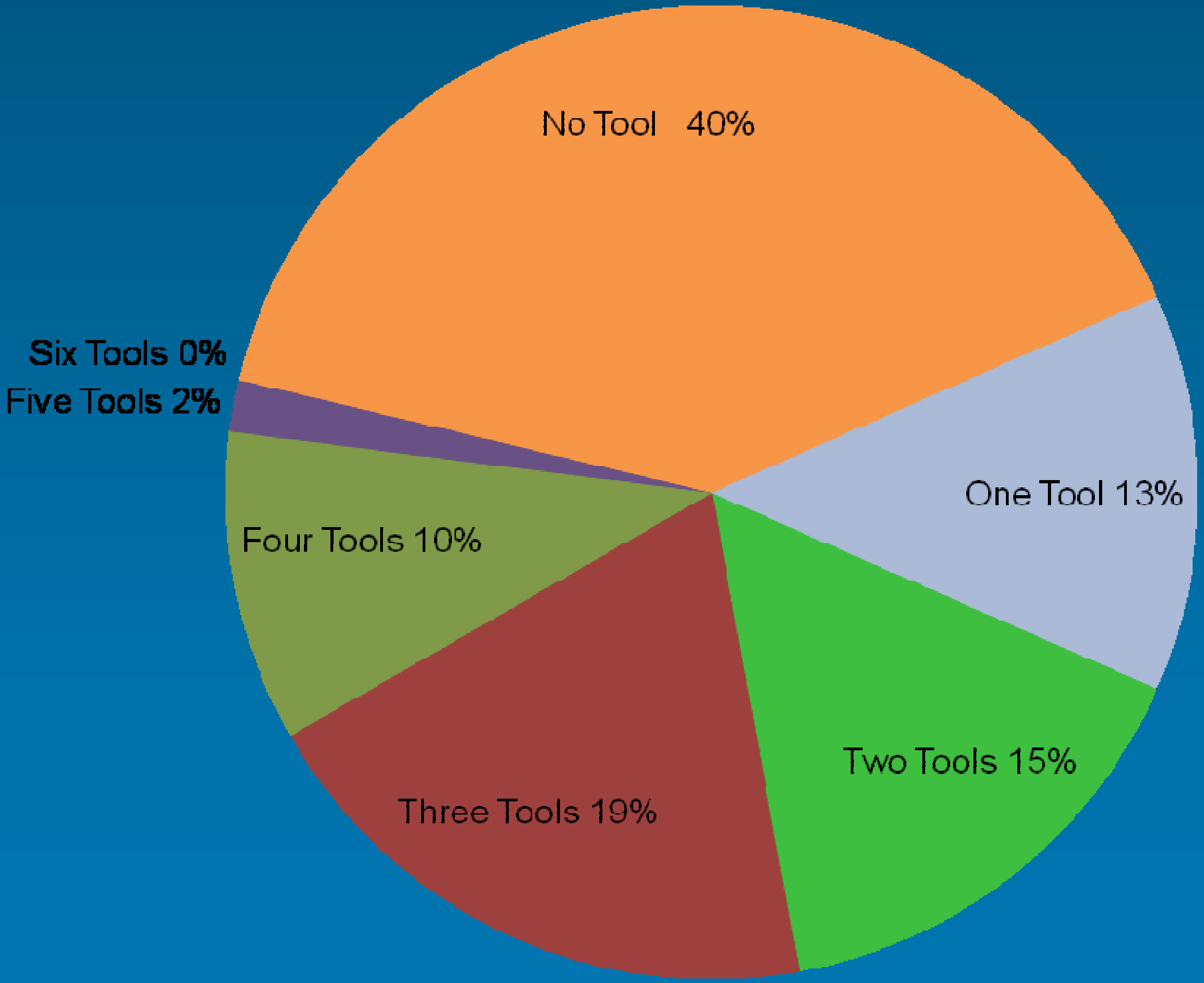
		# Test Cases	# CWEs Covered
C/C++	"Breadth"	210	103
	"Depth"	201	10
	All C/C++	411	103
Java	"Breadth"	177	112
	"Depth"	183	11
	All Java	360	112
	All	771	175

Tools Evaluated

Tool	C/C++	Java
Coverity Prevent 4.3	√	√
FindBugs 1.3.7		√
Fortify SCA 5.2	√	√
GrammaTech Code Sonar 3.2	√	
Klocwork Insight 8.1	√	√
Ounce Labs Ounce 6	√	√
PMD 4.2.5		√

Evaluation Results

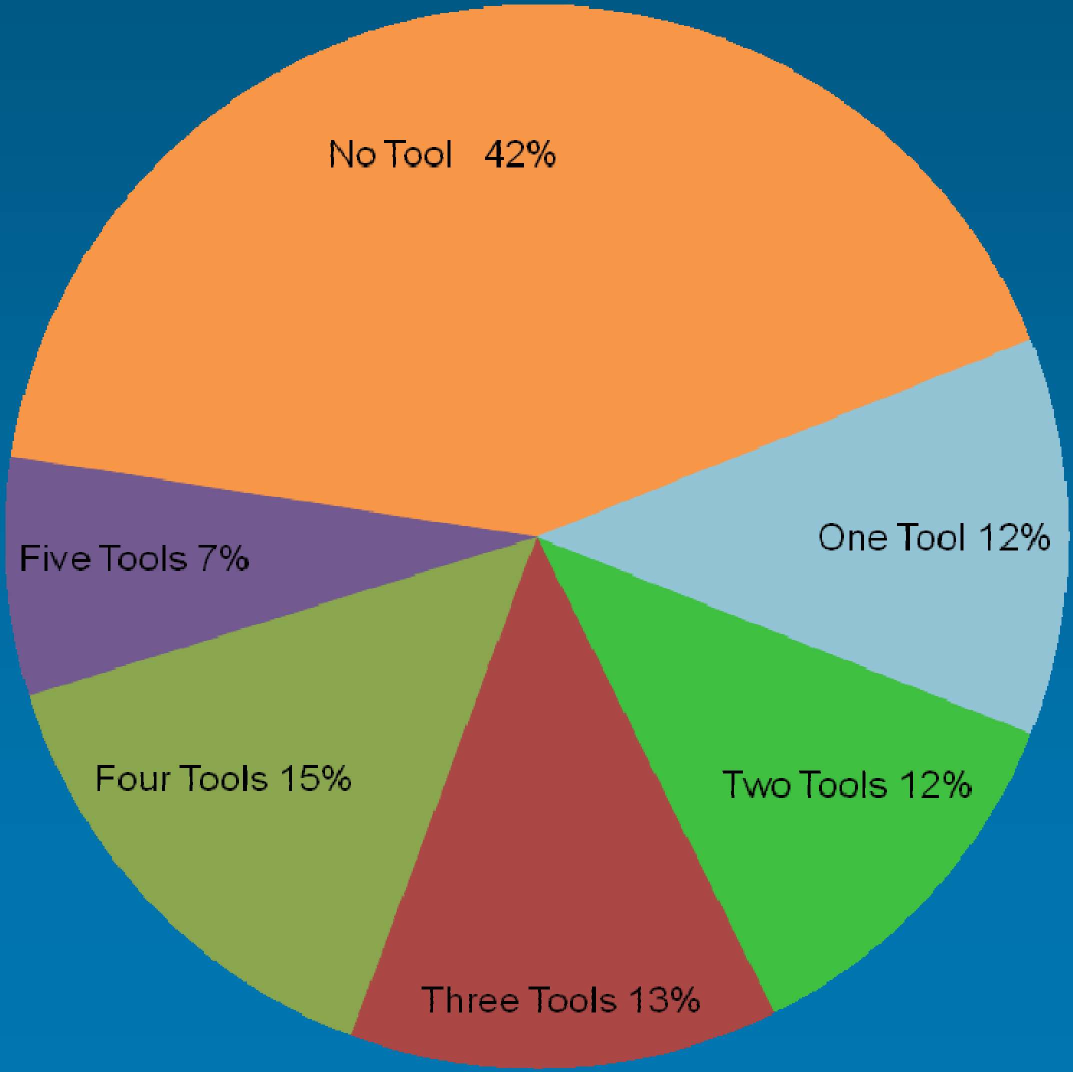
Java “Breadth” Test Case Coverage



Examples of Missed Test Cases (Java)

- CWE 369-Divide by zero
- CWE 482-Comparing instead of assigning
- CWE 484-Omitted break statement in switch
- CWE 606-Unchecked input for loop condition
- CWE 674-Uncontrolled recursion

C/C++ “Breadth” Test Case Coverage



Examples of Missed Test Cases (C/C++)

- CWE 190-Integer overflow or wraparound
- CWE 248-Uncaught exception
- CWE 374-Mutable objects passed by reference
- CWE 397-Declaration of throws for generic exception
- CWE 588-Attempt to access child of a non-structure pointer
- CWE 674-Uncontrolled recursion

Missed Test Case

- CWE 190-Integer overflow or wraparound (in C)

```c
void CWE190_Integer_Overflow__multiply_int_01_bad()
{
      int a, b, c;
      a = INT_MAX / 2;
      b = rand();
      /* FLAW: a * b may exceed INT_MAX and overflow */
      c = a * b;
      printIntLine(c);
}
```

CWE 190 in real code: CVE-2009-0583

- Original release date: March 23, 2009
- Overview
  - Multiple integer overflows in the International Color Consortium (ICC) Format library, allow attackers to cause a denial of service or possibly execute arbitrary code…

Source: National Vulnerability Database, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0583

CWE 190 in real code: CVE-2009-0583 – The Flaw

```
…
icmFileMem_read (…, size_t size, size_t count)
{
        …
        size_t len;
        len = size * count;
        …
}
```

CWE 190 in real code: CVE-2009-0583 – The Fix

```
…
icmFileMem_read (…, size_t size, size_t count)
{
    …
    if (count > 0 && size > SIZE_MAX / count)
        return 0;
    size_t len;
    len = size * count;
    …
}
```

Source Code Analysis Tool Evaluation

Questions?

Jaime Merced
Center for Assured Software
National Security Agency