CYBER SECURITY DIVISION

2014 R&D SHOWCASE AND TECHNICAL WORKSHOP

Qualitative & Quantitative Evaluation of Static Code Analysis Tools

Indiana University-Purdue University Indianapolis

Dr. James H. Hill (hillj@cs.iupui.edu)

December 16-18, 2014

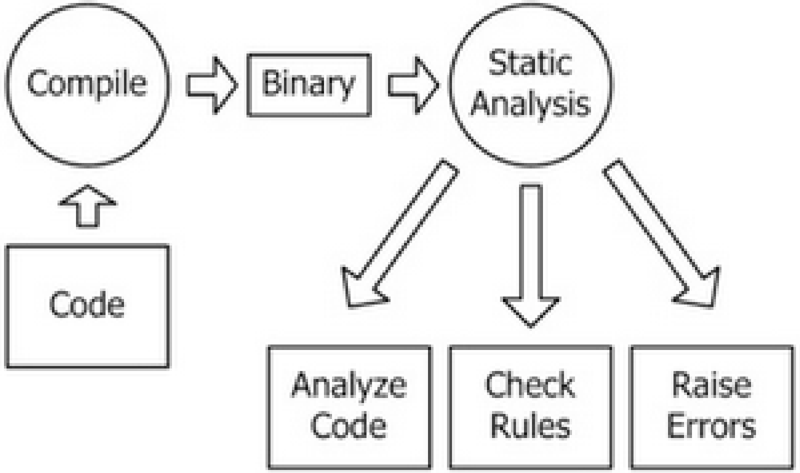
Homeland Security

Science and Technology

Project Overview

- Static code analysis (SCA) is a methodology of detecting errors in program based on the review of code marked by the analyzer in areas where potential errors may occur

- SCA tools aid developers in quickly identifying errors through automation
  - memory leaks
  - dead code
  - code conformance
  - etc.



Existing Static Code Analysis Tools

- □ Given the vast number of SCA tools, it can be hard identifying what SCA tools are best for the job!

HP Fortify Source Code Analyzer	AdaControl	Pylint
Axivion Bauhaus Suite	Astrée	Parasoft C/C++test
IBM Rational AppScan Source Edition	cpplint	Klocwork Insight
Imagix 4D	Clang	SofCheck Inspector
MALPAS	PVS-Studio	CodeRush
CodeSonar	Cppcheck	Visual Studio Team System
CodeIt.Right	Protecode	DMS Software Reengineering Toolkit
FxCop	FindBugs	Kalistick
Apparat	PMD	…

It is also a costly & time-consuming process evaluating the quality of each tool…

Our objective is to evaluate the quality of static code analysis tool, and understand how to best apply them to a given piece of source code

Current Status

- Acquired and deployed three commercial SCA tools into the System Integration Lab at IUPUI
- Developed an extensible framework for automating the evaluation of SCA tools (SCATE)
- Exploring methodology and reporting features
  - Granularity
  - Aggregating multiple tools
  - Permutation heat map

Granularity

- Controls the required accuracy for the tool

Granularity

- Controls the required accuracy for the tool

CWE835_Infinite_Loop__do_01.c

```c
void CWE835_Infinite_Loop__do_01_bad() {
  int i = 0;

  /* FLAW: Infinite Loop - do..while() with no break point */
  do
  {
    printIntLine(i);
    i = (i + 1) % 256;
  } while(i >= 0);
}
```

Granularity

- Controls the required accuracy for the tool

CWE835_Infinite_Loop__do_01.c

```c
void CWE835_Infinite_Loop__do_01_bad() {
  int i = 0;

  /* FLAW: Infinite Loop - do..while() with no break point */
  do
  {
    printIntLine(i);
    i = (i + 1) % 256;
  } while(i >= 0);
}
```

Flaw

Granularity

- Controls the required accuracy for the tool

CWE835_Infinite_Loop__do_01.c

```
10 void CWE835_Infinite_Loop__do_01_bad() {
11   int i = 0;
12
13   /* FLAW: Infinite Loop - do..while() with no break point */
14   do
15   {
16     printIntLine(i);
17     i = (i + 1) % 256;
18   } while(i >= 0);
19 }
```

Flaw

FILE

The tool can find the flaw anywhere in the file

FUNCTION

The tool can find the flaw anywhere in the function

LINE

The tool must find the flaw on line 14

Granularity

- Controls the required accuracy for the tool

Granularity	Detected Flaws
File	25,511
Function	3,565
Line	2,215

- Increasing granularity reduces the quality of a Tool

Using Multiple Tools

- Organizations will often run multiple tools to reduce risk

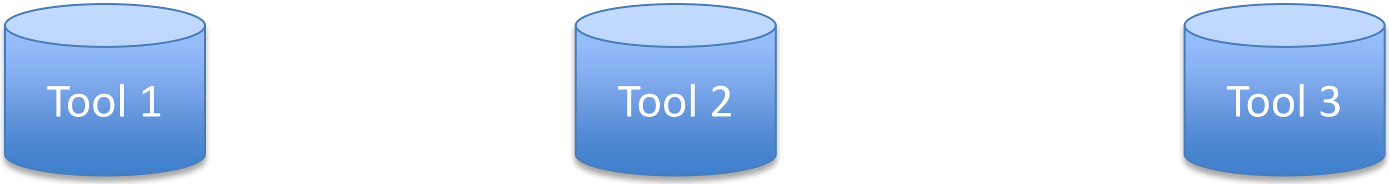
Using Multiple Tools

- Organizations will often run multiple tools to reduce risk

TP	FP
5,155	206,433

TP	FP
12,235	9,174

TP	FP
25,511	111,397



Tool 1

Tool 2

Tool 3

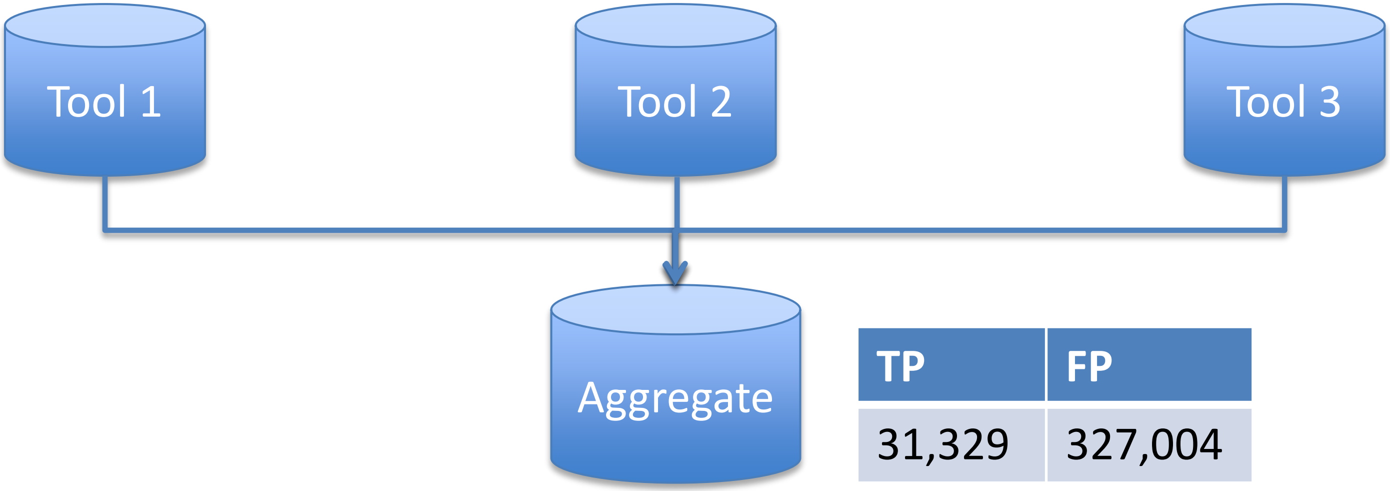
Using Multiple Tools

- Organizations will often run multiple tools to reduce risk

TP	FP
5,155	206,433

TP	FP
12,235	9,174

TP	FP
25,511	111,397



TP	FP
31,329	327,004

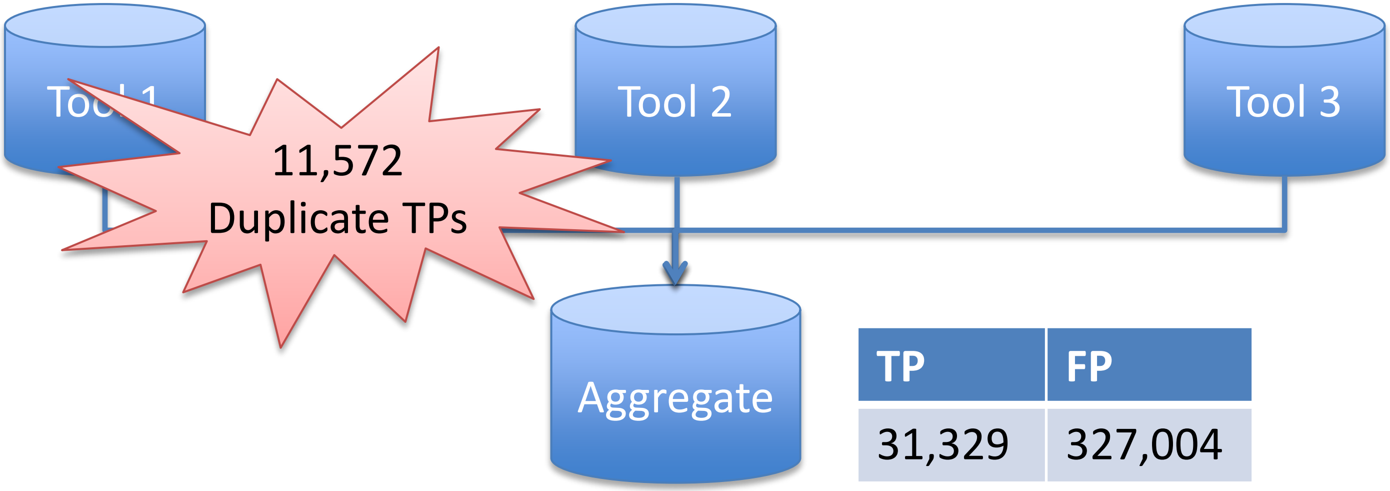
Using Multiple Tools

- Organizations will often run multiple tools to reduce risk

TP	FP
5,155	206,433

TP	FP
12,235	9,174

TP	FP
25,511	111,397



TP	FP
31,329	327,004

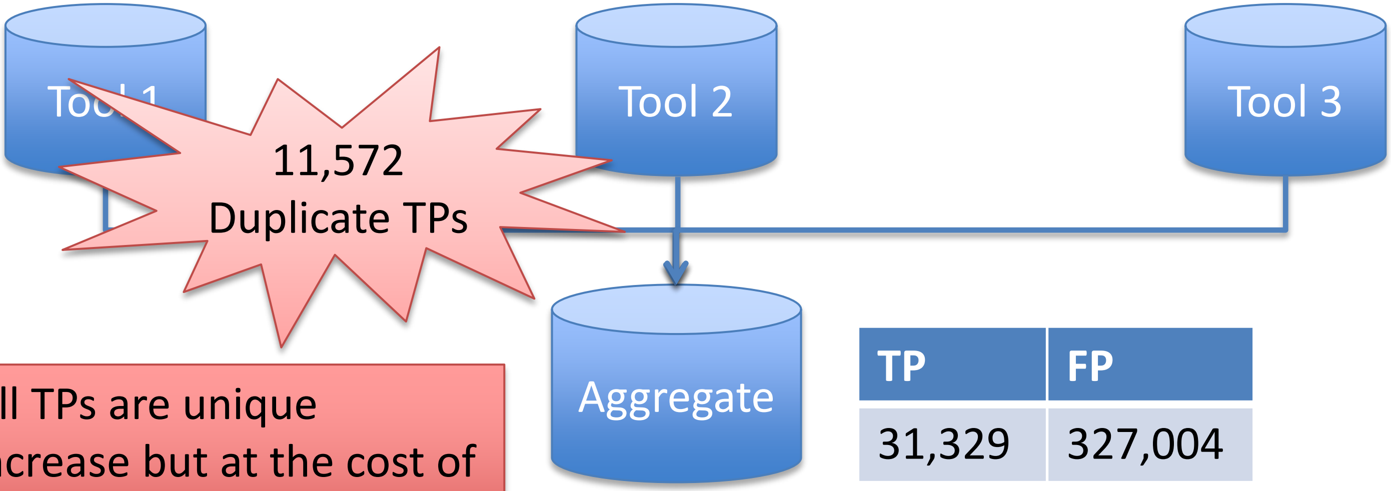
Using Multiple Tools

- Organizations will often run multiple tools to reduce risk

TP	FP
5,155	206,433

TP	FP
12,235	9,174

TP	FP
25,511	111,397



Tool 1

Tool 2

Tool 3

11,572 Duplicate TPs

Aggregate

TP	FP
31,329	327,004

- Not all TPs are unique
- TPs increase but at the cost of many more FPs (reduced precision)

Permutation Heat Map

- Permutations use Data and/or Control flows to obscurify a Flaw to test SCA tools

Permutation Heat Map

- Permutations use Data and/or Control flows to obscurify a Flaw to test SCA tools

Permutation 01:

```
void bad (void)
{
  // FLAW: …
}
```

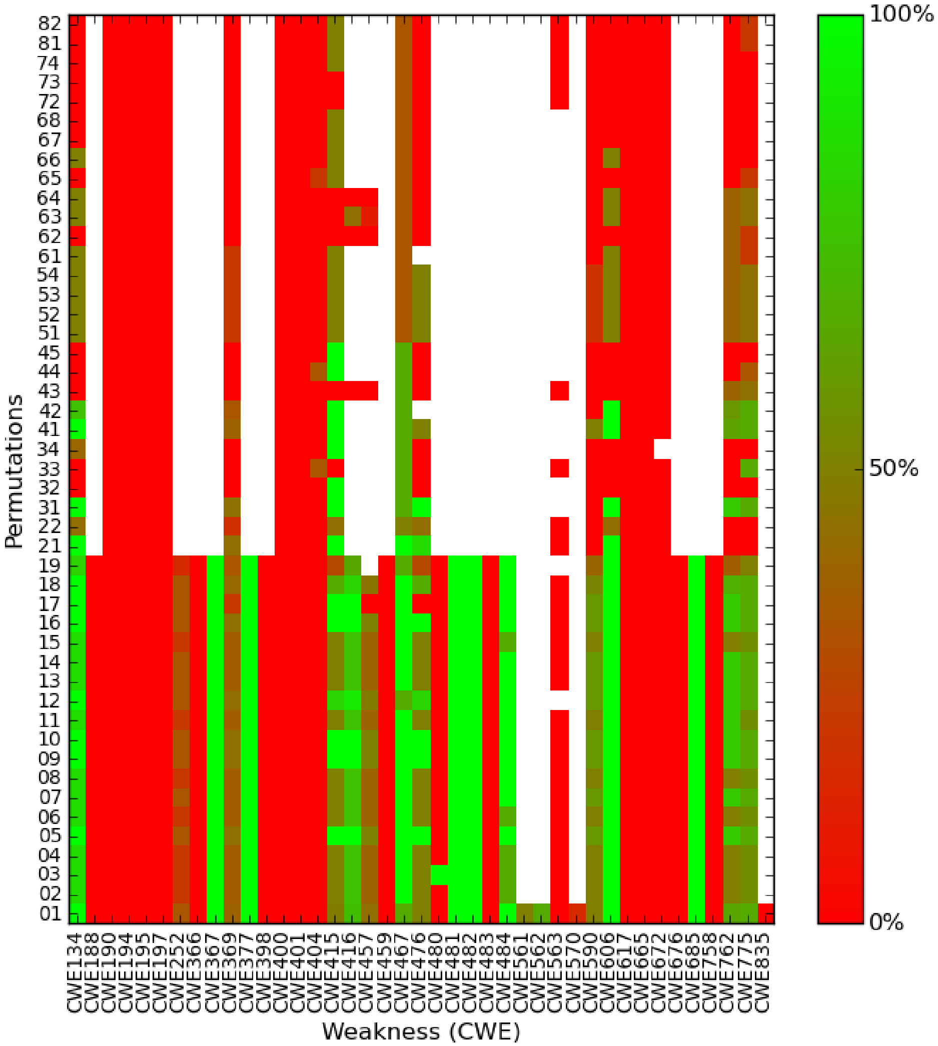
Permutation 02:

```
void bad (void)
{
  if (1)
  {
    // FLAW: …
  }
}
```

Permutation 03:

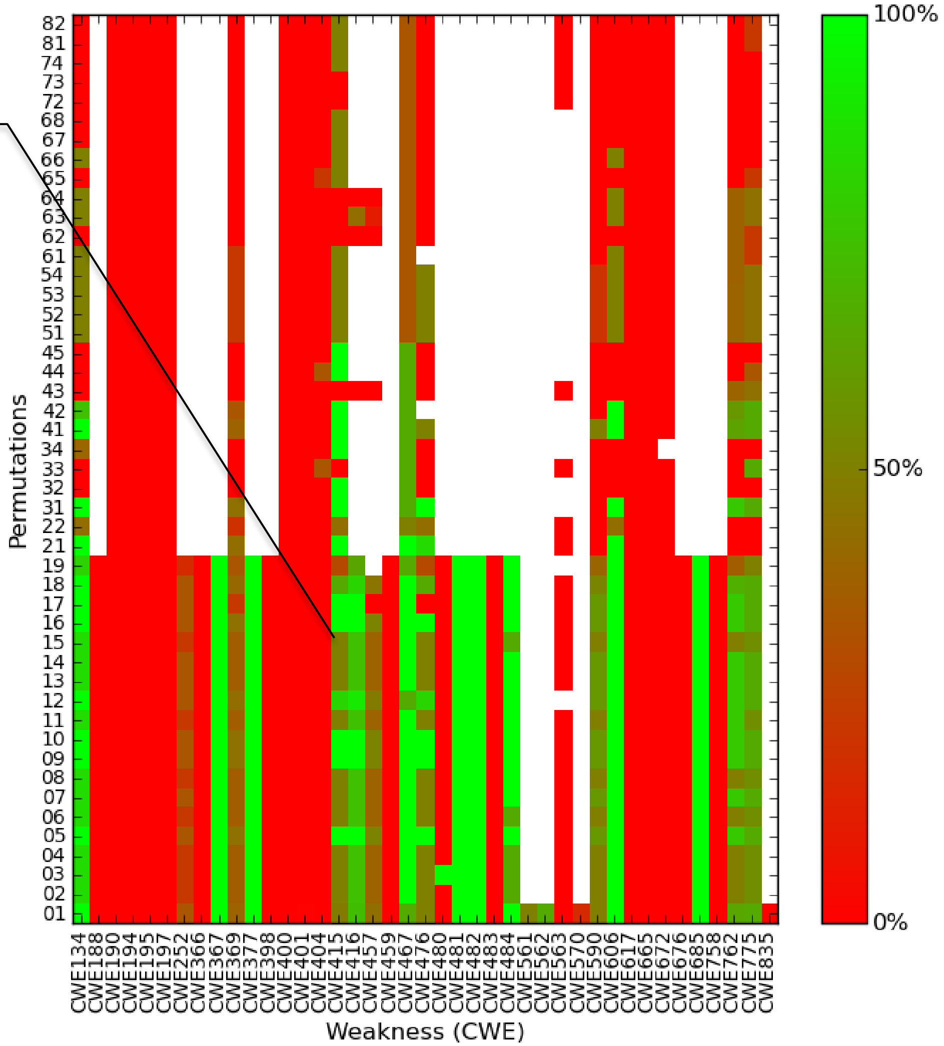
```
void bad (void)
{
  if (5==5)
  {
    // FLAW: …
  }
}
```

Permutation Heat Map



Permutation Heat Map

CWE 415 (Double Free)
15: 50 % Flaws Found
16: 100% Flaws Found



Permutation Heat Map

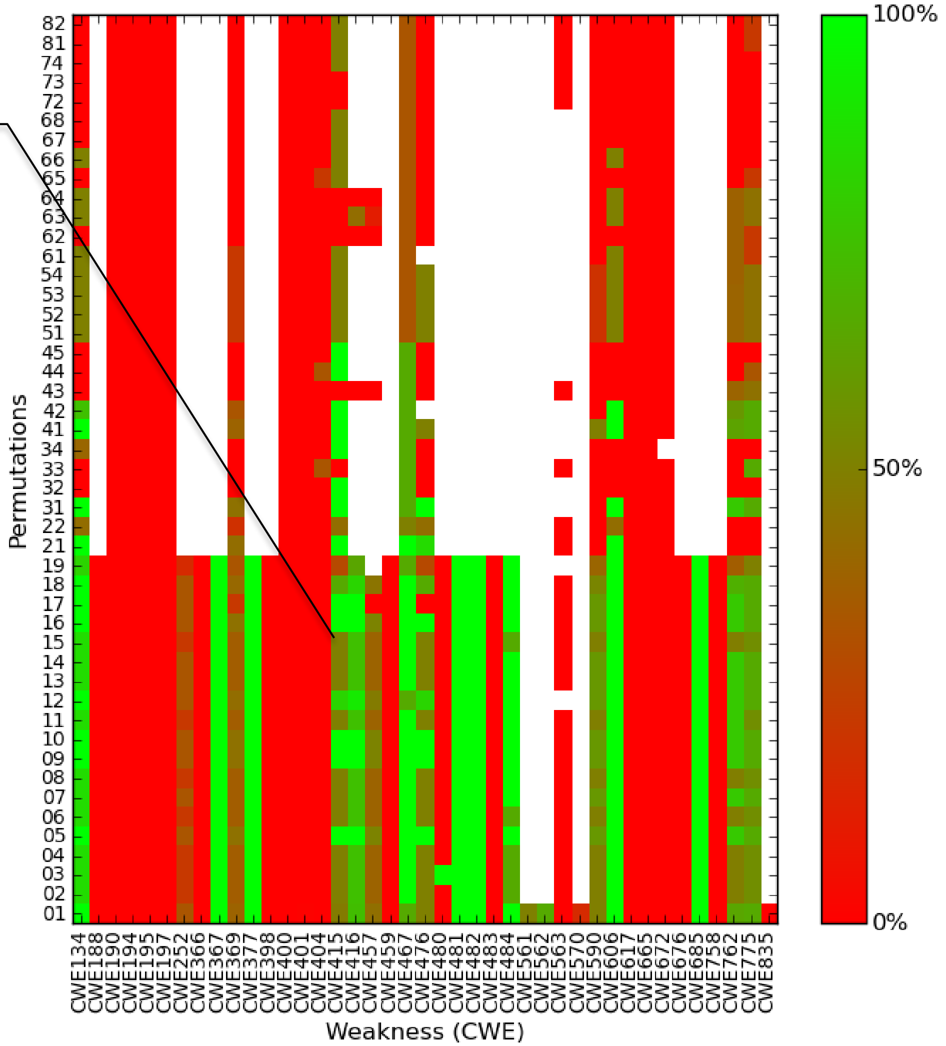
CWE 415 (Double Free)
15: 50 % Flaws Found
16: 100% Flaws Found

Permutation 15:

```
void bad (void)
{
  switch(6)
  {
  case (6):
    // FLAW: …
    break
  }
}
```

Permutation 16:

```
void bad (void)
{
  while (1)
  {
    // FLAW: …
    break
  }
}
```



Permutation Heat Map

CWE 415 (Double Free)
15: 50 % Flaws Found
16: 100% Flaws Found

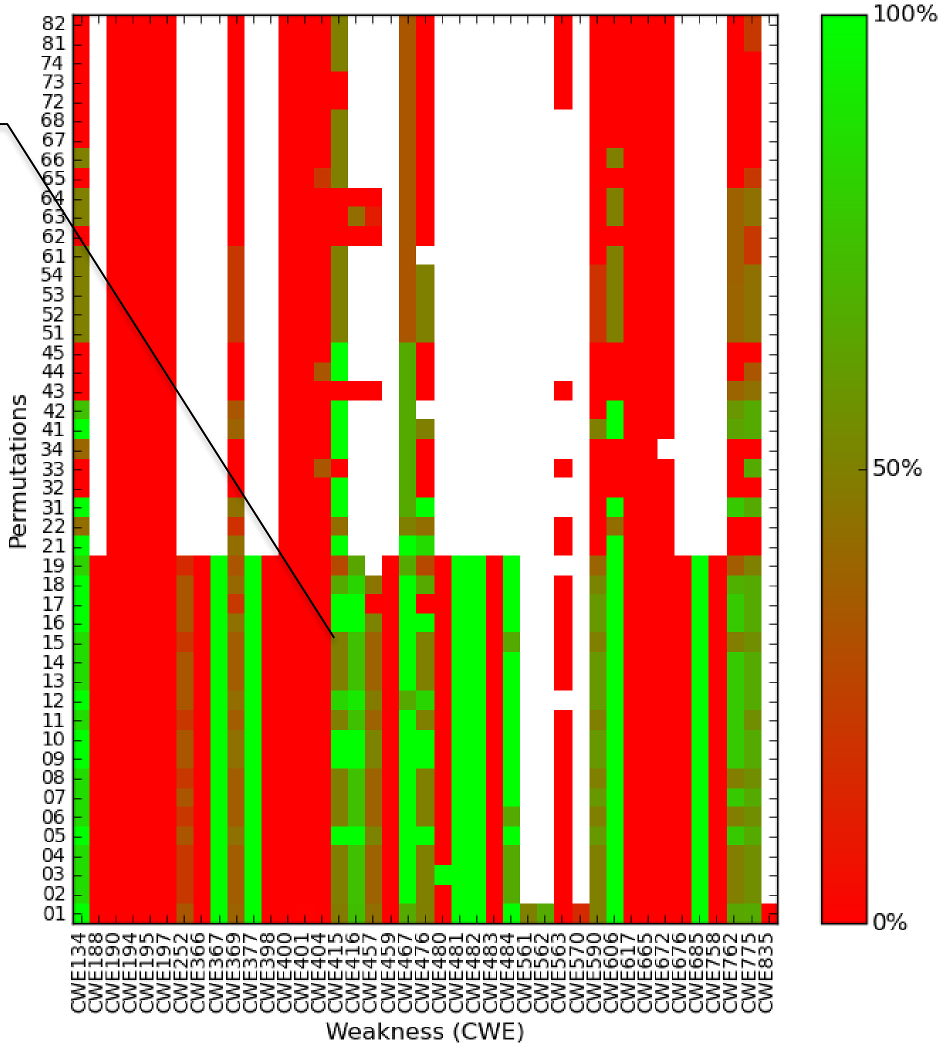
Permutation 15:

```
void bad (void)
{
  switch(6)
  {
  case (6):
    // FLAW: …
    break
  }
}
```

Permutation 16:

```
void bad (void)
{
  while (1)
  {
    // FLAW: …
    break
  }
}
```

- The type of permutation can affect a tool's quality

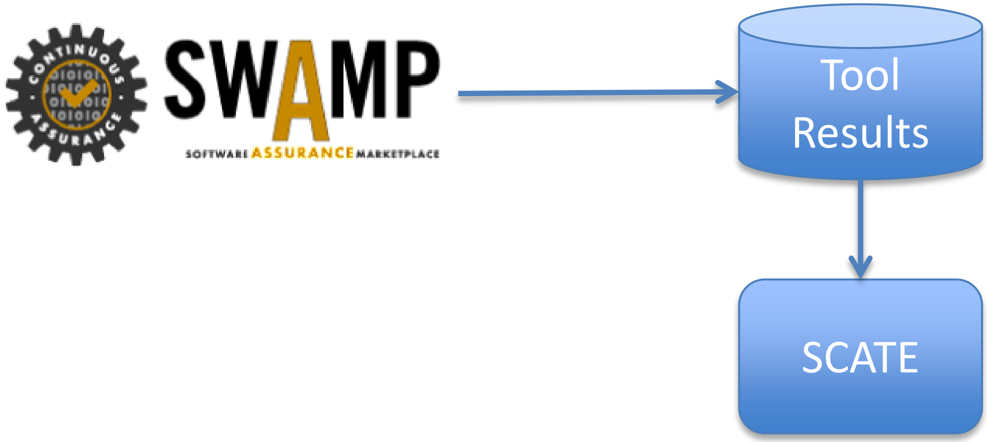


Future Work

- SWAMP Integration
- Tool behavioral model
- Predict tool quality against source code
- Streamline analysis into a cloud-based testing as a service product

Future Work

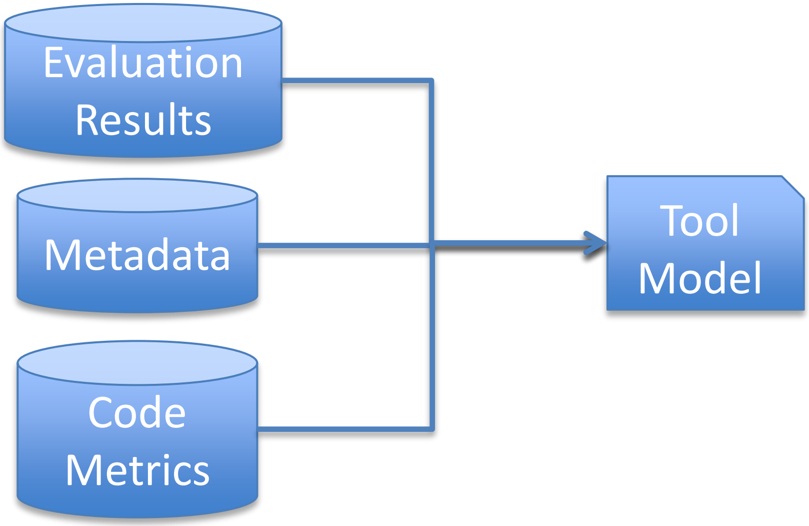
- SWAMP Integration
- Tool behavioral model
- Predict tool quality against source code
- Streamline analysis into a cloud-based testing as a service product



The SWAMP has multiple SCA tools integrated into their environment and can provide tool results

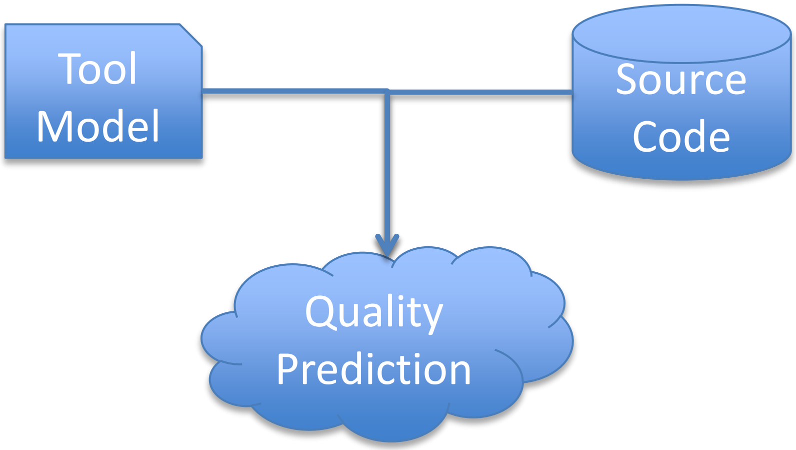
Future Work

- SWAMP Integration
- Tool behavioral model
- Predict tool quality against source code
- Streamline analysis into a cloud-based testing as a service product



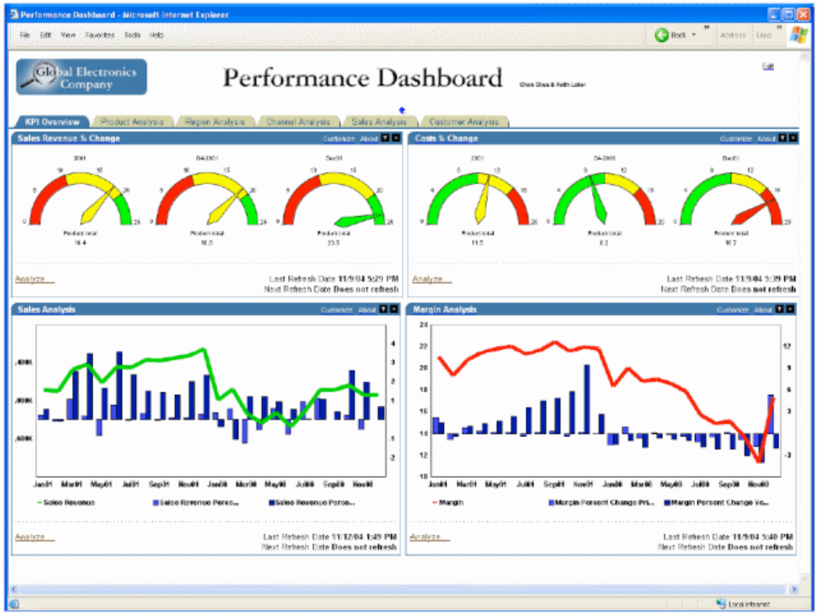
Future Work

- SWAMP Integration
- Tool behavioral model
- Predict tool quality against source code
- Streamline analysis into a cloud-based testing as a service product



Future Work

- SWAMP Integration
- Tool behavioral model
- Predict tool quality against source code
- Streamline analysis into a cloud-based testing as a service product



Questions