

Automated Assessment Tools

Barton P. Miller

Computer Sciences Department
University of Wisconsin

bart@cs.wisc.edu

Elisa Heymann

Computer Sciences Department
University of Wisconsin
Universitat Autònoma de Barcelona

elisa@cs.wisc.edu

CTSC

CENTER FOR TRUSTWORTHY
SCIENTIFIC CYBERINFRASTRUCTURE
The NSF Cybersecurity Center of Excellence



THE UNIVERSITY
of
WISCONSIN
MADISON



Universitat
Autònoma
de Barcelona



1. What You Need to Know about How Tools Work

2. The Tools And Their Use

Source Code Analysis Tools



```
p = requesttable;
while (p != (struct table *)0)
{
    if (p->entrytype == PEER_MEET)
    {
        found = (!(strcmp (her, p->me)) &&
                !(strcmp (me, p->her)));
    }
    else if (p->entrytype == PUTSERVR)
    {
        found = !(strcmp (her, p->me));
    }
    if (found)
        return (p);
    else
        p = p->next;
}
return ((struct table *) 0);
```



ID	Tool	Rule	CVSS	Severity	Codebase Location	Status
27	Cppcheck	3D-Invalidscaf	20	Medium	example-tutorial.c:20	Gone
25	Clang	Return value is not checked in c...	252	Medium	example-tutorial.c:32	Gone
24	Clang	Return value is not checked in c...	252	Medium	example-tutorial.c:31	Gone
26	Cppcheck	3D-Invalidscaf	20	Low	example-tutorial.c:20	Gone

A Bit of History

Compiler warnings

Let the Compiler Help

- Turn on compiler warnings and fix problems
- Easy to do on new code
- Time consuming, but useful on old code
- Use lint, multiple compilers
- **-Wall** is not enough!

gcc: **-Wall, -W, -O2, -Werror, -Wshadow, -Wpointer-arith, -Wconversion, -Wcast-qual, -Wwrite-strings, -Wunreachable-code** and many more

- Many useful warning including security related warnings such as format strings and integers

A Bit of History

- **Lint (1979)**
 - C program checker.
 - Detects suspicious constructs:
 - Variables being used before being set.
 - Division by zero.
 - Conditions that are constant.
 - Calculations whose result is likely to overflow.
- **Current automated assessment tools are a sort of “super-Lint”.**

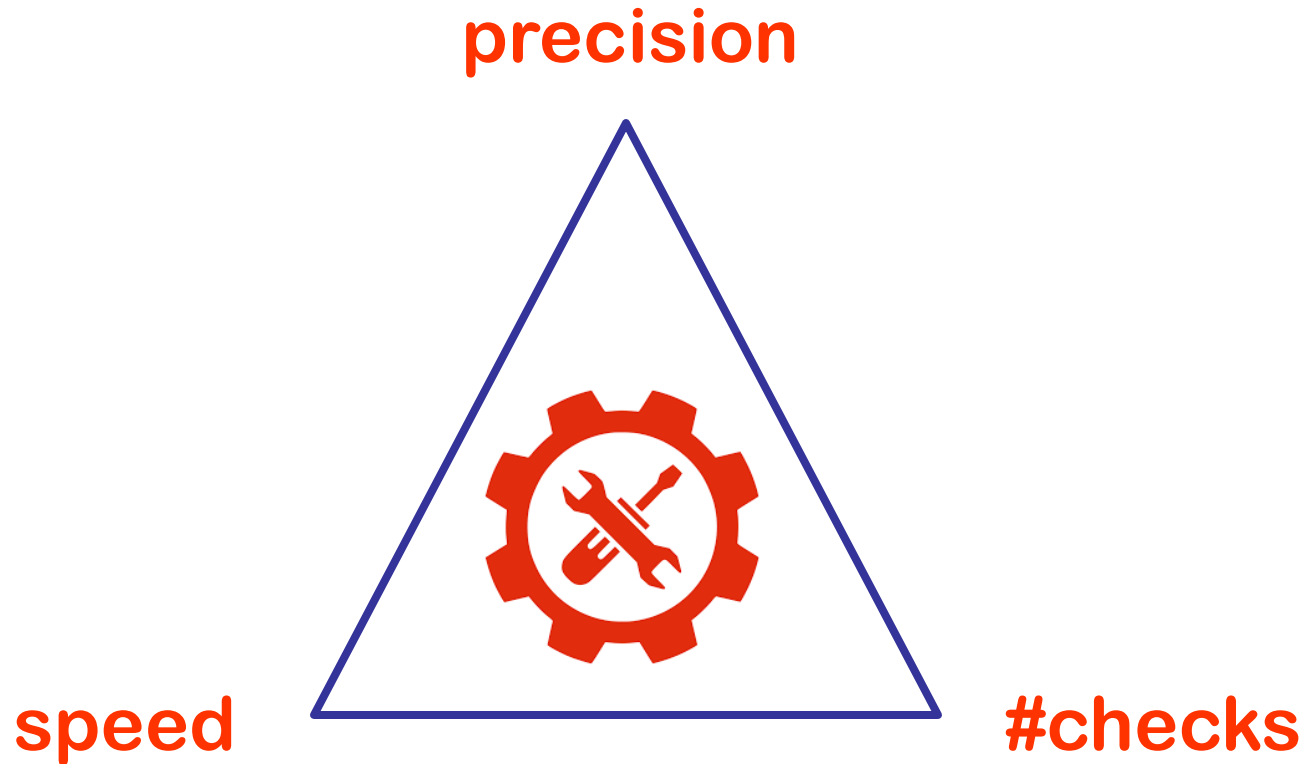
Source Code Analysis Tools

- Designed to analyze **source code** or **binaries** to help find **security flaws**.
- The source code may contain inadvertent or deliberate weaknesses that could lead to security vulnerabilities in the executable versions of the application program.
- Better to use them from the beginning of the software development life cycle.
 - Though commonly applied to legacy code.

Source Code Analysis Tools

- Program that parses and then analyses the source code.
- Doesn't know what the program is supposed to do.
- Looks for violations of good programming practices.
- Looks for specific programming errors.
- Works like a compiler
 - Instead of binaries, it produces an intermediate representation

Source Code Analysis Tools



You can get 2 out of 3

Source Code Analysis Tools

Different kind of tools:

Syntax vs. semantics

Interprocedural

Whole program analysis

Local vs. paths

Data flow analysis

Sound vs. approximate

Implications:

Scalability

Accuracy

Different kind of tools

```
cmd = “/bin/ls”;  
execl (cmd, NULL);
```

Pattern (syntax) matching

Will say “**always dangerous**”.

Semantic analysis

Sometimes definitely **no**.

Different kind of tools

```
fgets(cmd,MAX,stdin);  
execl (cmd, NULL);
```

Pattern (syntax) matching

Will say “**always dangerous**”.

Semantic analysis

Sometimes definitely **no**.

Sometimes definitely **yes**.

Different kind of tools

```
cmd=makecmd();  
exec1 (cmd, NULL);
```

Pattern (syntax) matching

Will say “**always dangerous**”.

Semantic analysis

Sometimes definitely **no**.

Sometimes definitely **yes**.

Sometimes **undetermined**.

Source Code Analysis Tools

How do they work

Identify the code to be analyzed.

- **Scripts or build systems that build the executable.**

The parser interprets the source code in the same way that a compiler does.

Source Code Analysis Tools

How do they work

Each invocation of the tool creates a model of the program:

- Abstract representations of the source
 - Control-flow graph
 - Call graph
 - Information about symbols (variables and type names)

Source Code Analysis Tools

How do they work

Symbolic execution on the model:

- Abstract values for variables.
- Explores paths.
- Based on abstract interpretation and model checking.
- The analysis is **path sensitive**.
 - The tool can tell the path for the flow to appear.
 - Points along that path where relevant transformations occur and conditions on the data values that must hold.

Source Code Analysis Tools

How do they work

The tool issue a set of warnings.

- List with priority levels.

The user goes through the warning list and labels each warning as:

- True positive.
- False Positive.
- Don't care.

Source Code Analysis Tools

The Output

A tool grades weaknesses according things such as

severity,

potential for exploit, or

certainty that they are vulnerabilities.

Problems:

- False positives.
- False negatives.

Source Code Analysis Tools

The Output

Ultimately people must analyze the tool's report and the code then decide:

- Which reported items are not true weaknesses.
- Which items are acceptable risks and will not be mitigated.
- Which items to mitigate, and how to mitigate them.

Source Code Analysis Tool Limitations

No single tool can find every possible weaknesses:

- A weakness may result in a vulnerability in one environment but not in another.
- No algorithm can correctly decide in every case whether or not a piece of code has a property, such as a weakness.
- Practical analysis algorithms have limits because of performance, approximations, and intellectual investment.
- **And new exploits are invented and new vulnerabilities discovered all the time!**



Source Code Analysis Tools

What can they find

- Stylistic programming rules.
- Type discrepancies.
- Null-pointer dereferences.
- Buffer overflows.
- Race conditions.
- Resource leaks.
- SQL Injection.

Source Code Analysis Tools

What is difficult to find

- **Authentication problems.**
 - Ex: Use of non-robust passwords.
- **Access control issues.**
 - Ex: ACL that does not implement the principle of least privilege.
- **Insecure use of cryptography.**
 - Ex: Use of a weak key.

Source Code Analysis Tools

What is not possible to find

- **Incorrect design.**
- **Code that incorrectly implements the design.**
- **Configuration issues, since they are not represented in the code.**
- **Complex weaknesses involving multiple software components.**

Code Analysis Basics

Control flow analysis

- Analyze code structure and build a graph representation.
- Basics blocks and branch/call edges.
- Pointers are difficult.

Data flow analysis

- Usage, calculation, and setting of variables.
- Extract symbolic expressions.
- Arrays are annoying.
- Pointers are difficult.

Control Flow Analysis

Control Flow Analysis

Detects control flow dependencies among different instructions.

Control Flow Graph (CFG)

- Abstract representation of the source code.
- Each node represents a basic block.
- Call or jump targets start a basic block.
- Jumps end a basic block.
- Directed edges represent the control flow.

```
void foo() {
    char buf[MAX] = "example";
    int i, j, k;
    char a, b;
    char *p = buf;

    i = 0;
    if (c)
        j = i;
    else
        j = MAX;
    a = buf[i];
    b = buf[j];
    k = 0;
    while (k < MAX) {
        if (buf[k]== 'x ')
            print(k);
        if (*p == 'z ')
            print(p);
        p++;
        k++;
    }
}
```

```

void foo() {
    char buf[MAX] = "example";
    int i, j, k;
    char a, b;
    char *p = buf;

    i = 0;
    if (c)
        j = i;
    else
        j = MAX;
    a = buf[i];
    b = buf[j];
    k = 0;
    while (k < MAX) {
        if (buf[k]== 'x ')
            print(k);
        if (*p == 'z ')
            print(p);
        p++;
        k++;
    }
}

```

```

p=buf
i=0
if (c)

```

```

void foo() {
    char buf[MAX] = "example";
    int i, j, k;
    char a, b;
    char *p = buf;

```

```

    i = 0;
    if (c)

```

```

        j = i;
    else
        j = MAX;

```

```

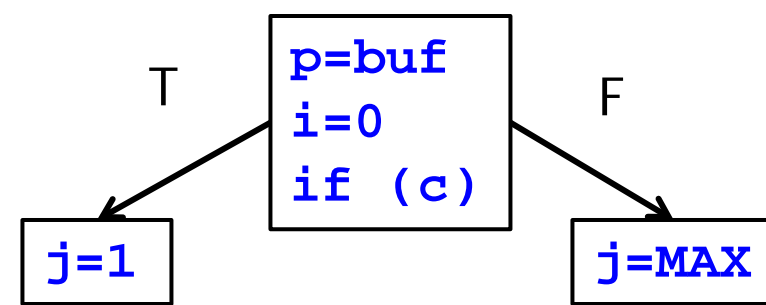
    a = buf[i];
    b = buf[j];
    k = 0;

```

```

    while (k < MAX) {
        if (buf[k] == 'x')
            print(k);
        if (*p == 'z')
            print(p);
        p++;
        k++;
    }

```

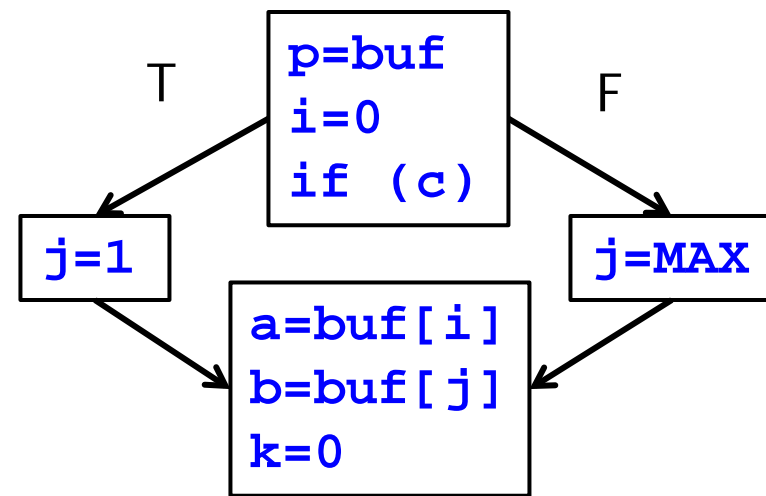


```

void foo() {
    char buf[MAX] = "example";
    int i, j, k;
    char a, b;
    char *p = buf;

    i = 0;
    if (c)
        j = i;
    else
        j = MAX;
    a = buf[i];
    b = buf[j];
    k = 0;
    while (k < MAX) {
        if (buf[k] == 'x')
            print(k);
        if (*p == 'z')
            print(p);
        p++;
        k++;
    }
}

```

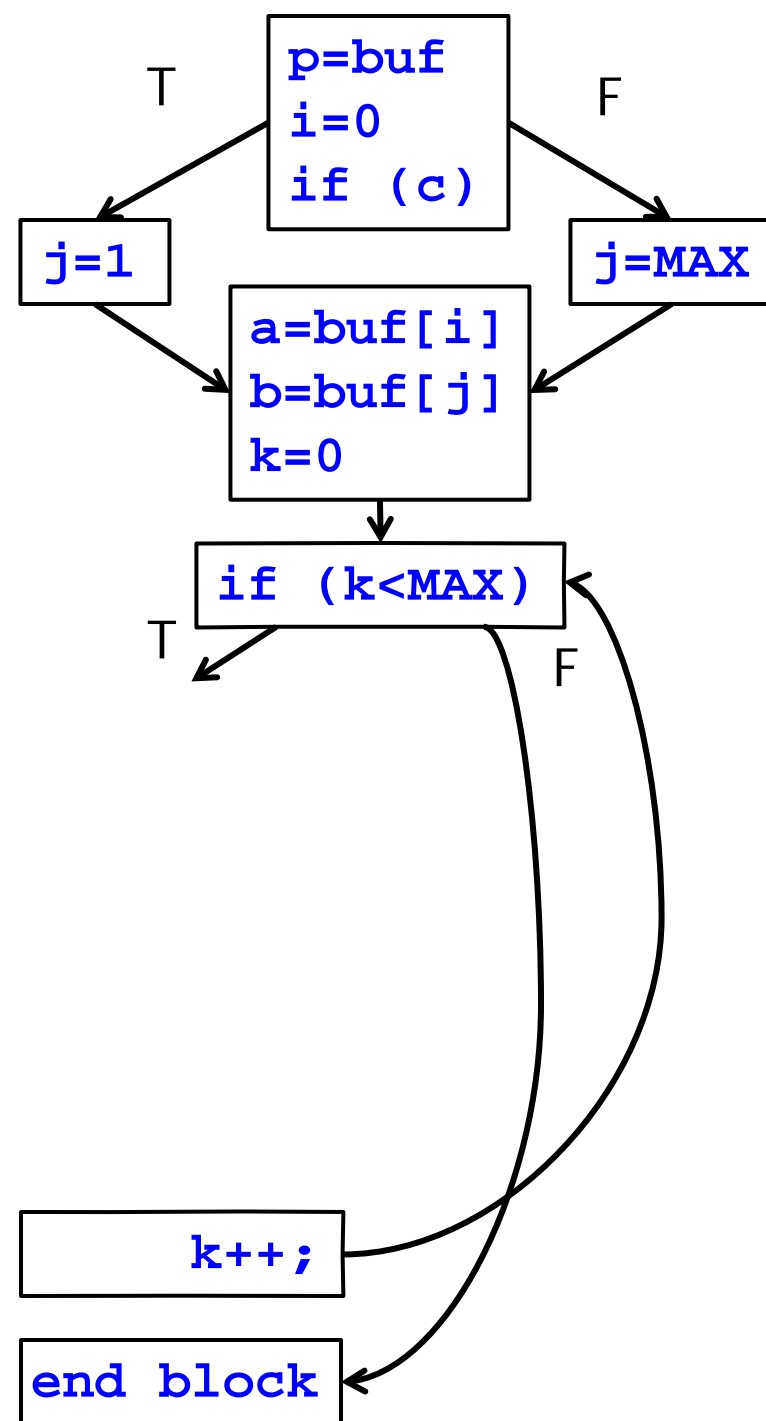


```

void foo() {
    char buf[MAX] = "example";
    int i, j, k;
    char a, b;
    char *p = buf;

    i = 0;
    if (c)
        j = i;
    else
        j = MAX;
    a = buf[i];
    b = buf[j];
    k = 0;
    while (k < MAX) {
        if (buf[k] == 'x')
            print(k);
        if (*p == 'z')
            print(p);
        p++;
        k++;
    }
}

```



Data Flow Analysis

Goal: Is this code safe?

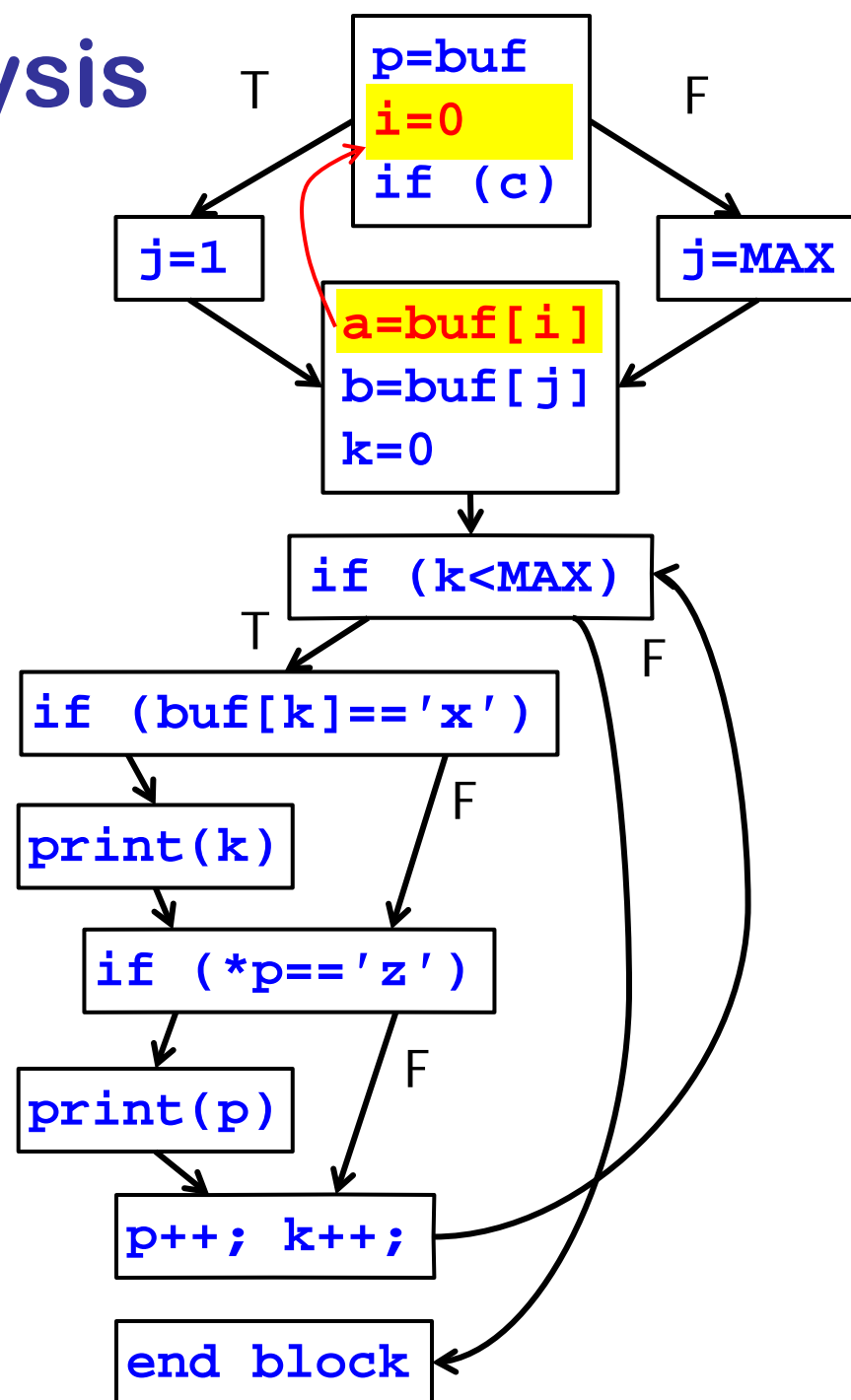
Subgoal:

Do we ever violate the borders of buf?

- Simple dependence
- Flow insensitive
- Loops
- Pointers
- Aliasing

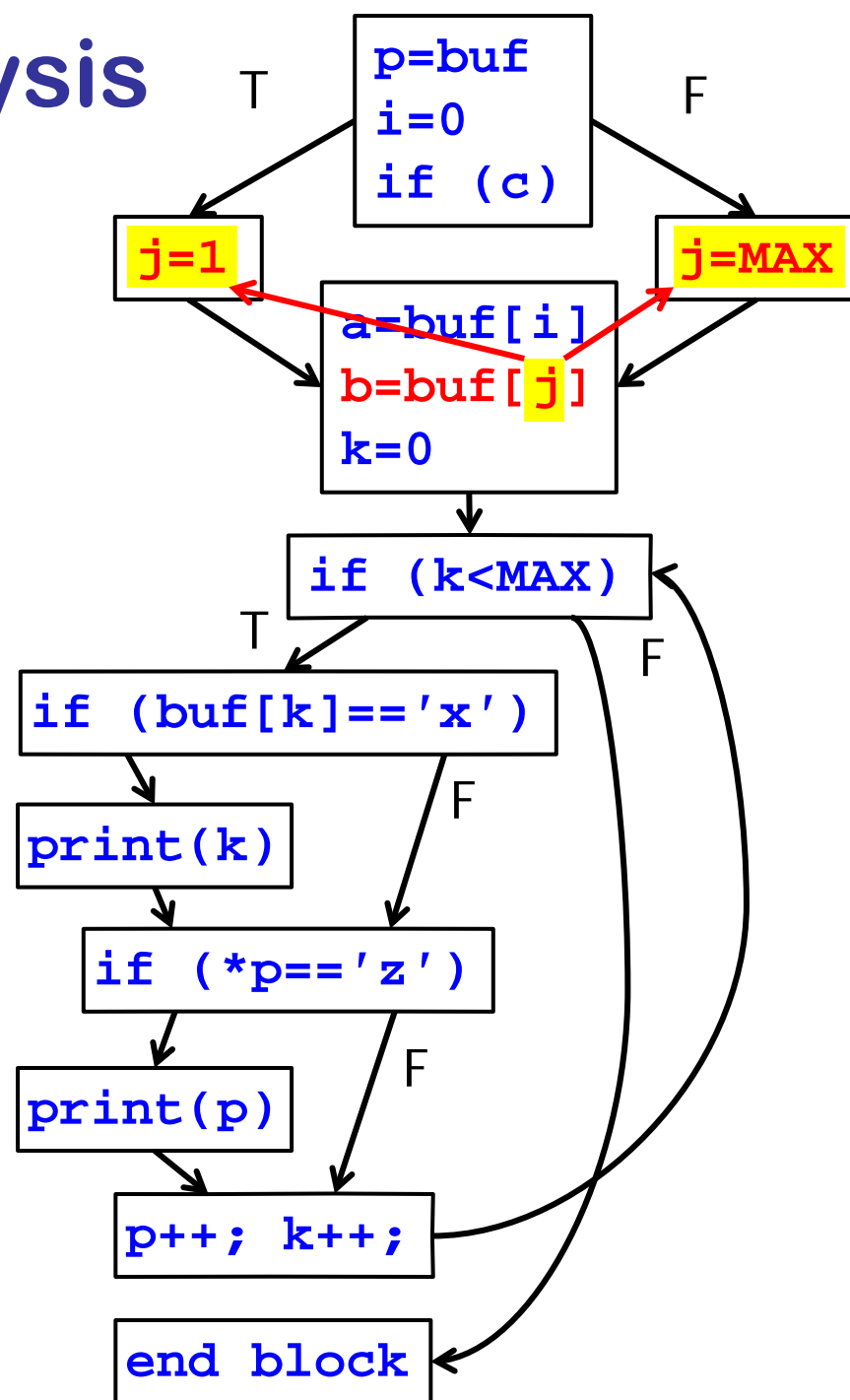
Data Flow Analysis

- Simple dependence



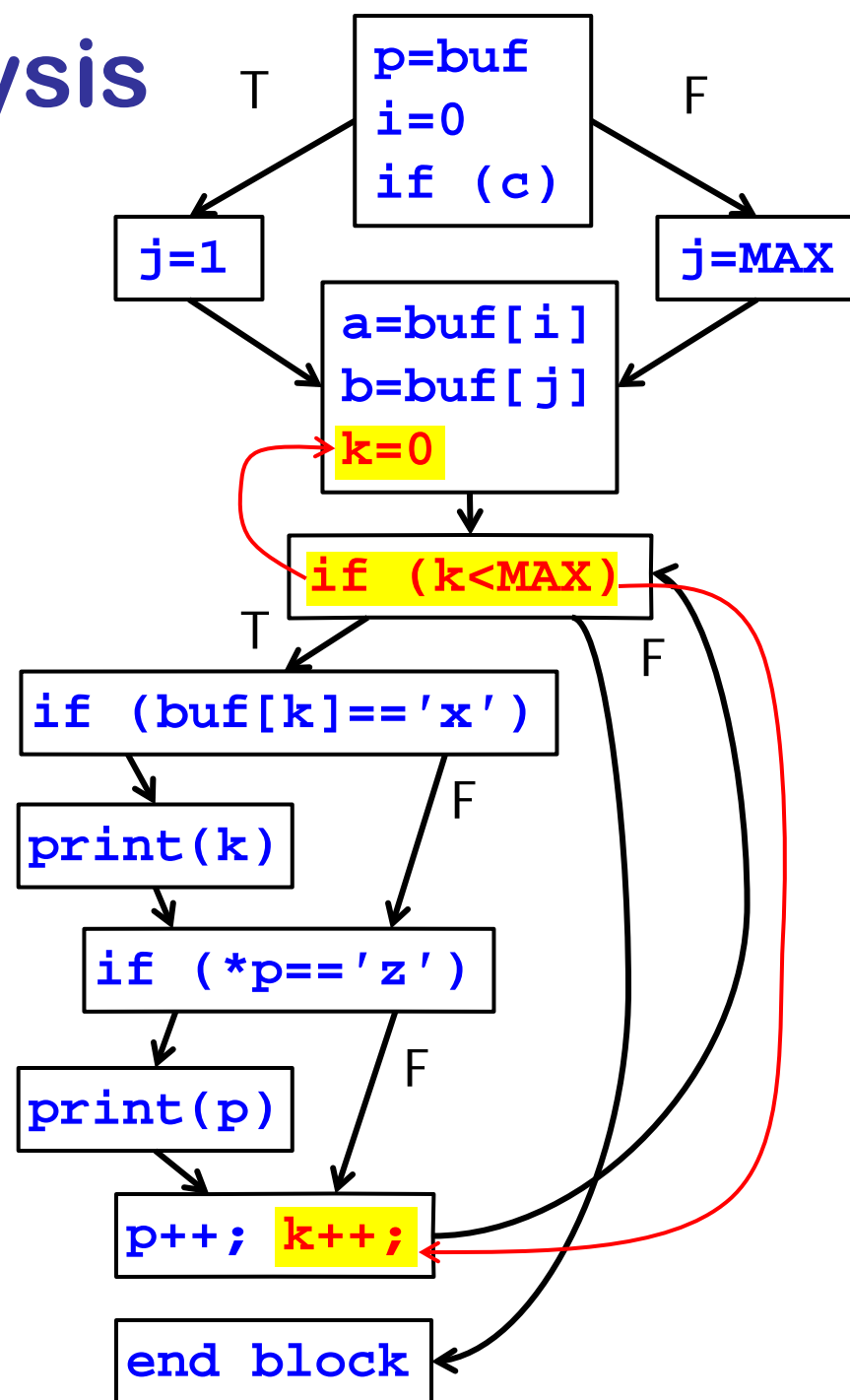
Data Flow Analysis

- Flow insensitive



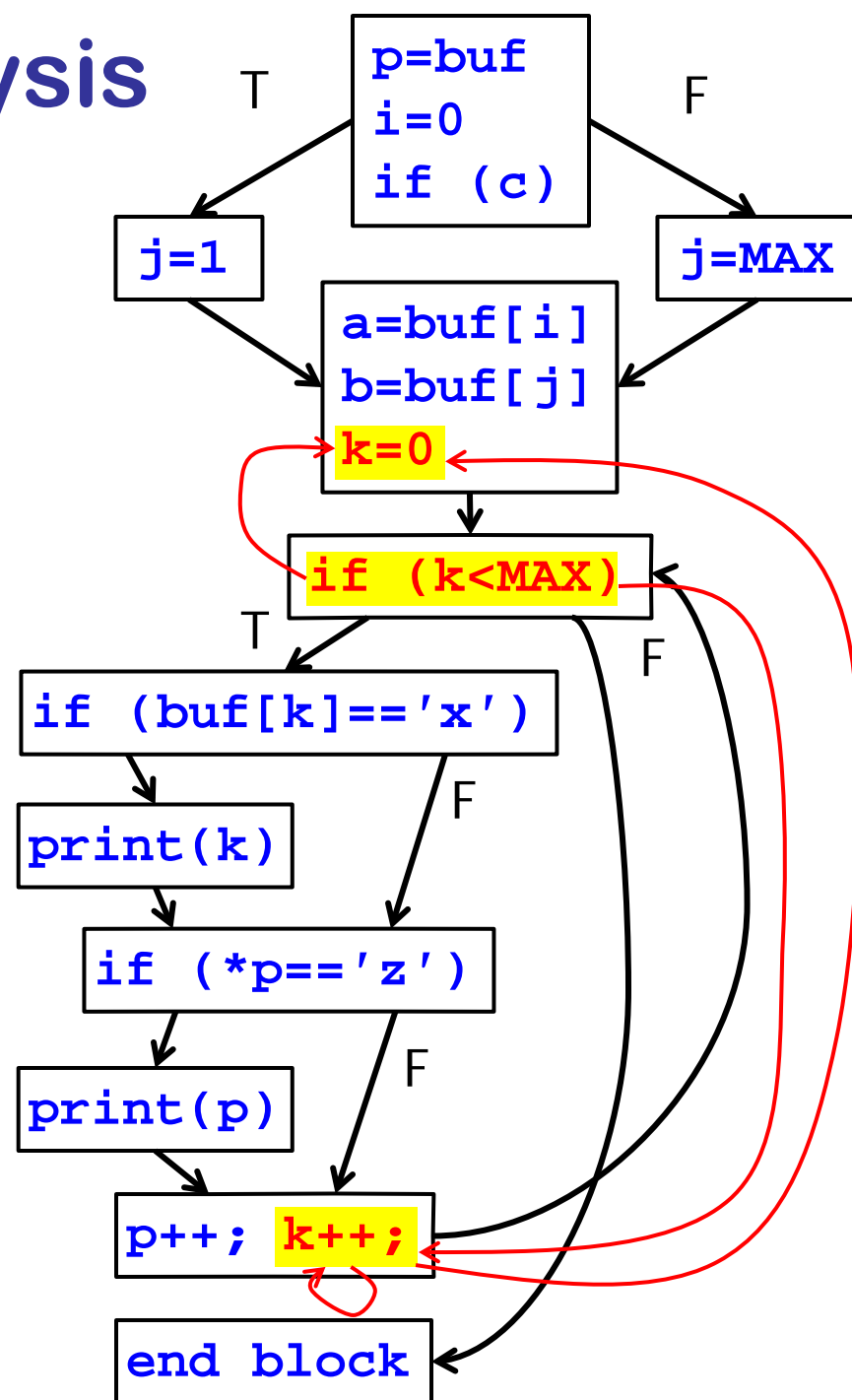
Data Flow Analysis

- Loops



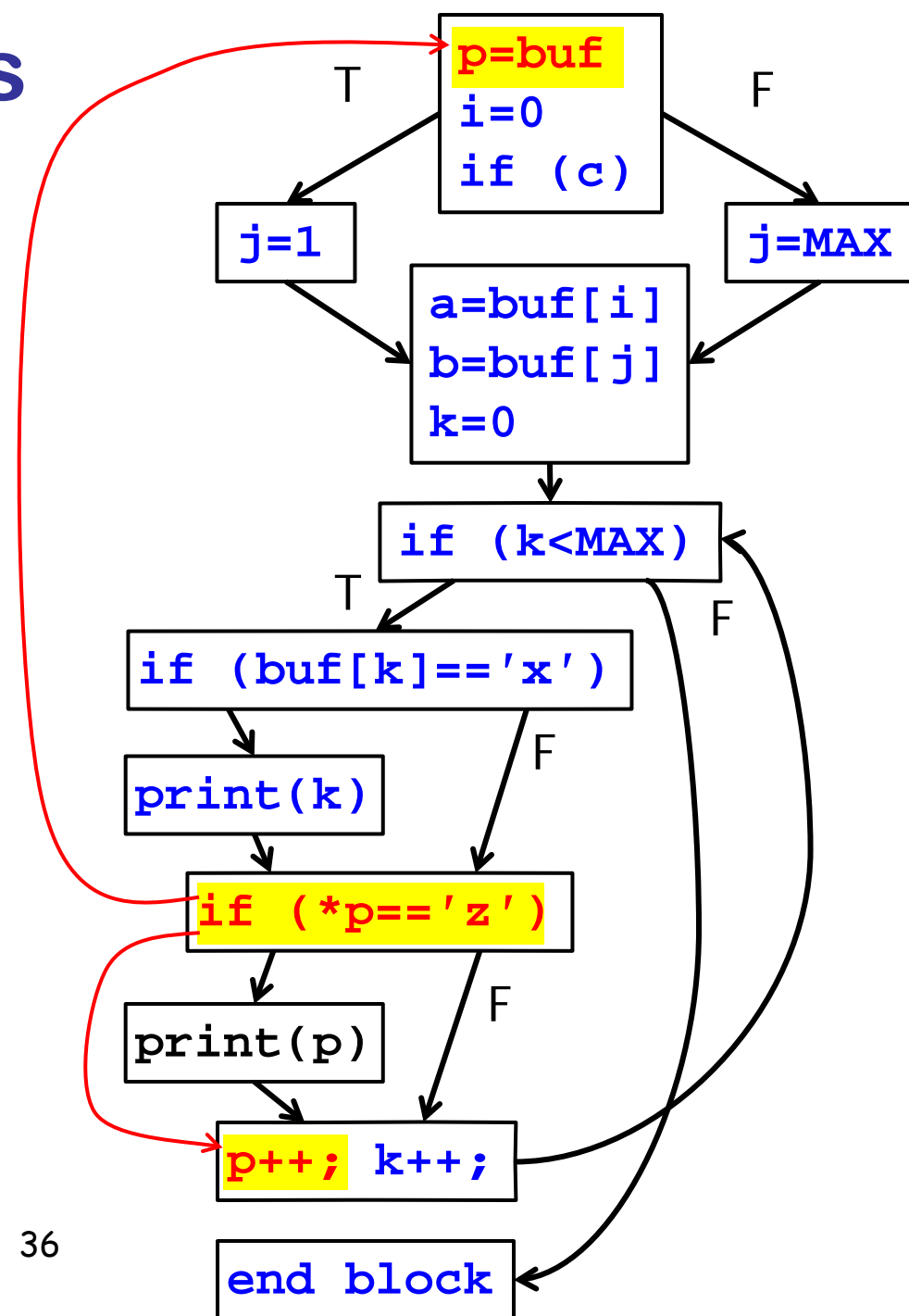
Data Flow Analysis

- Loops



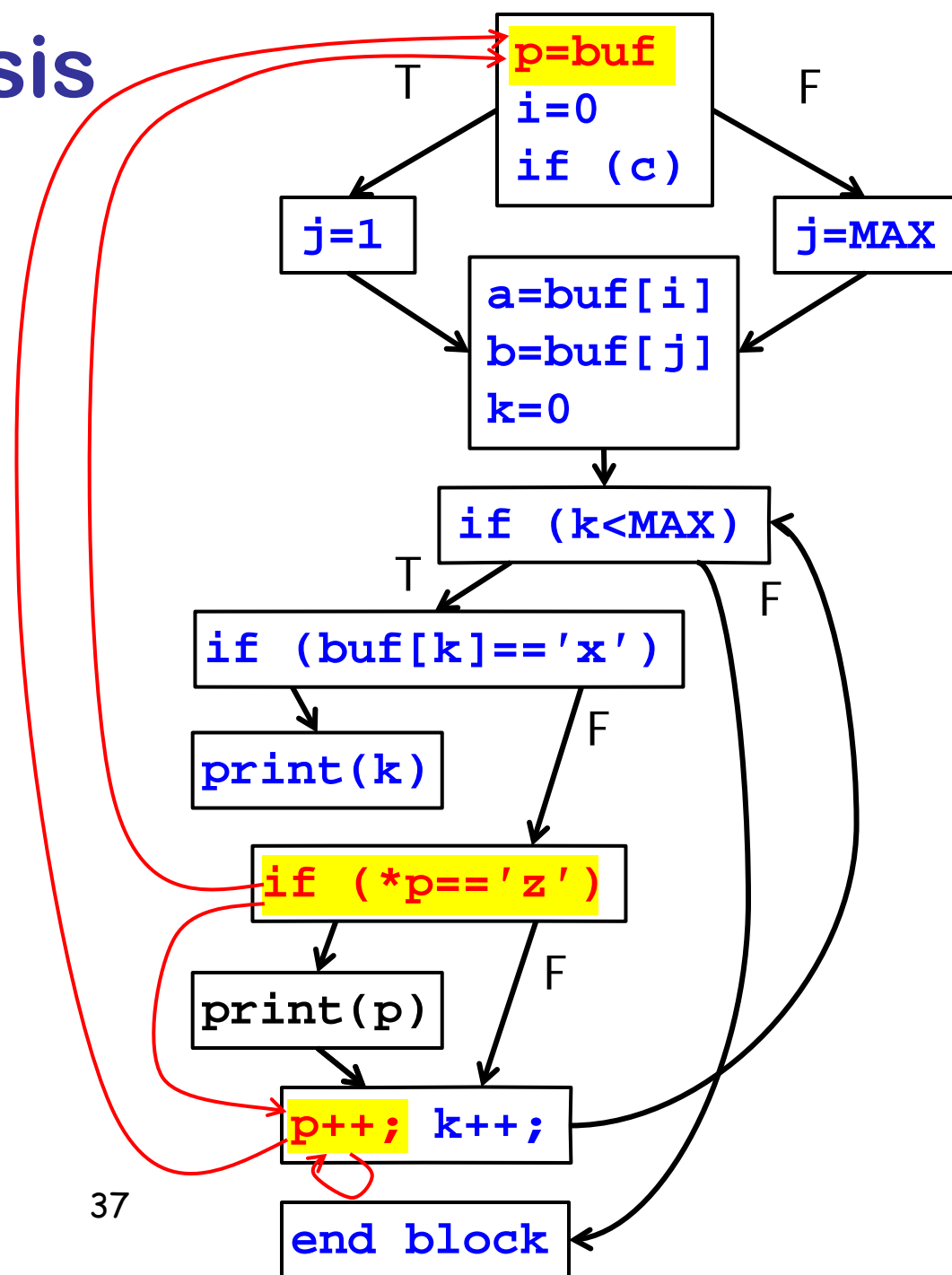
Data Flow Analysis

- Pointers



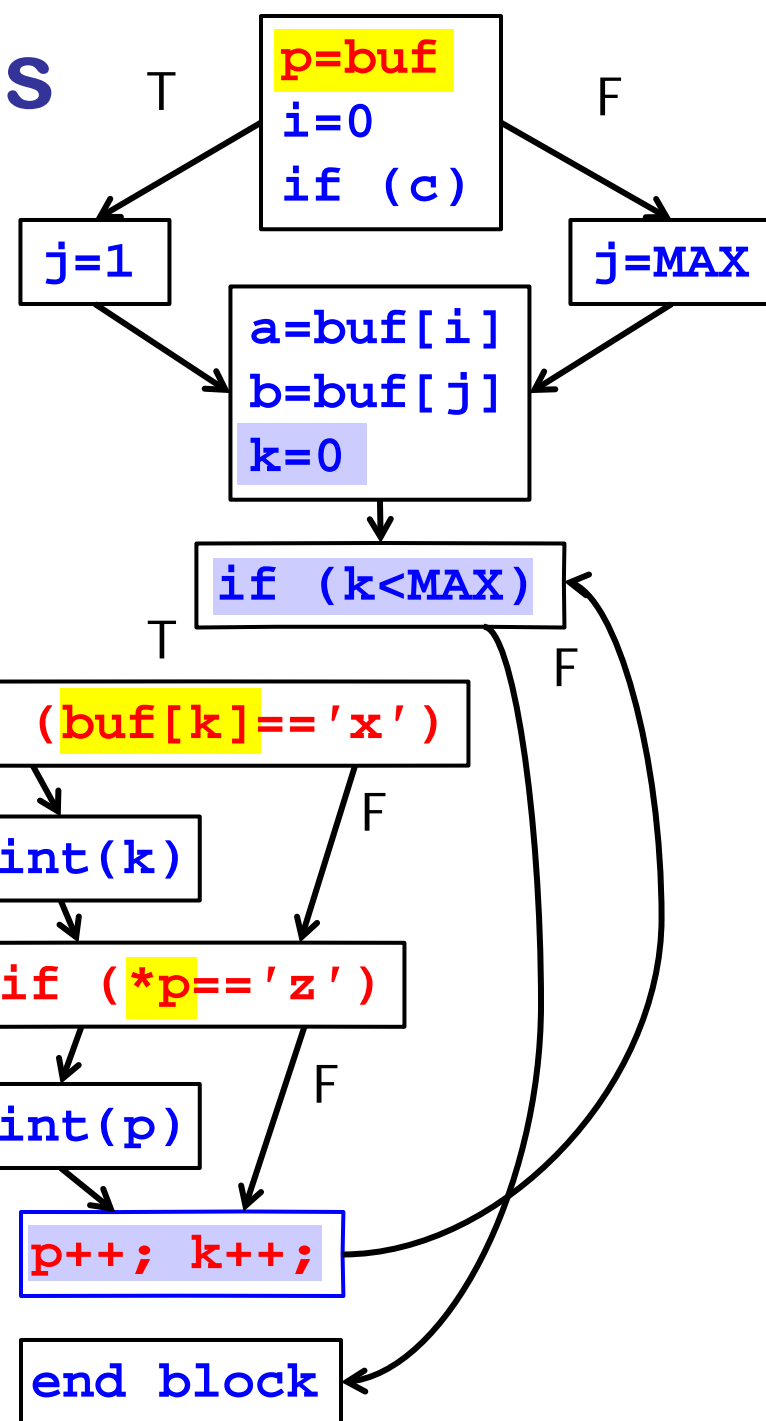
Data Flow Analysis

- Pointers



Data Flow Analysis

- Aliasing



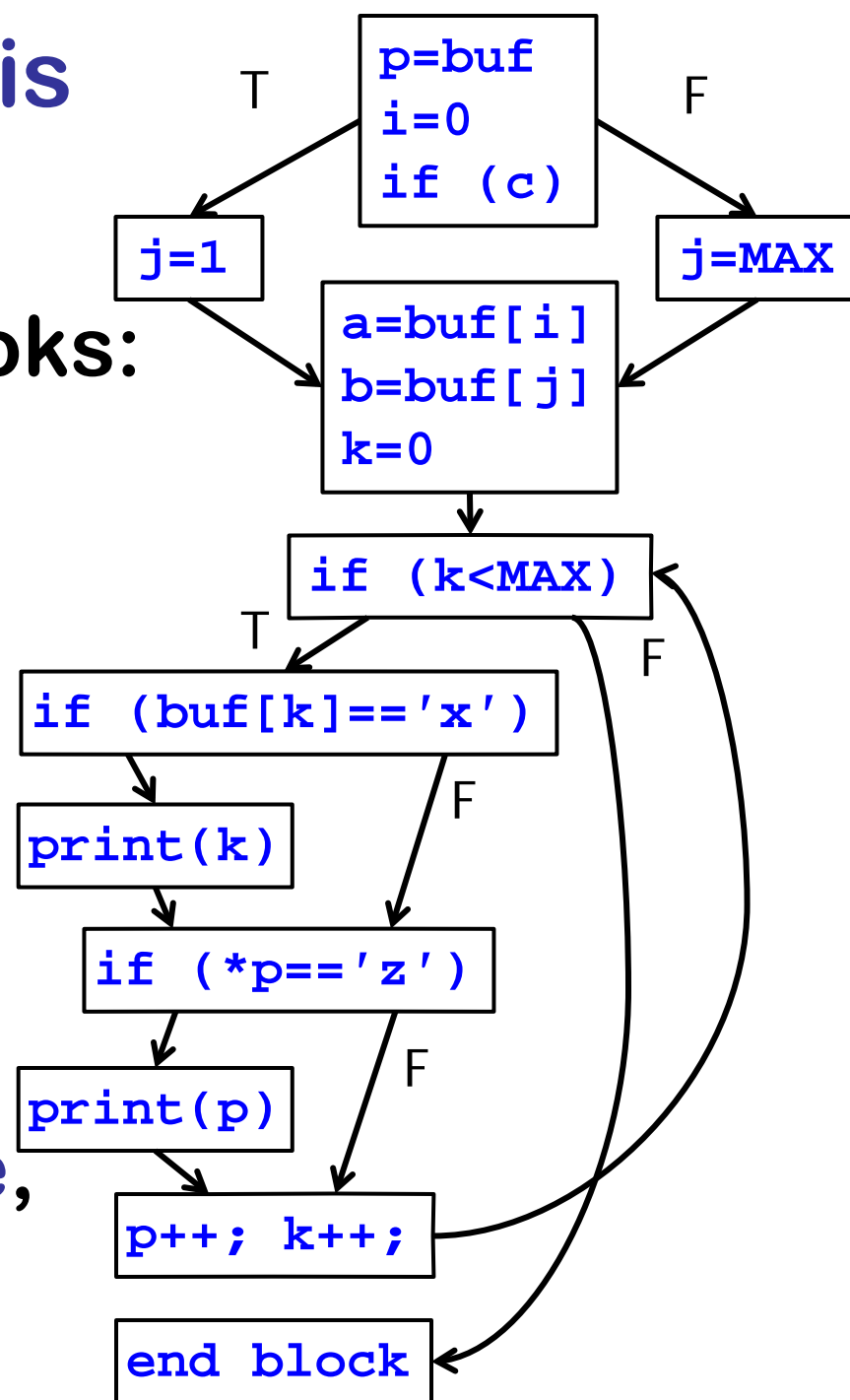
Are these
the same?

Semantic Analysis

But it's **harder** than it looks:

- Pointers to functions
- Virtual functions
- Interprocedural analysis
- Context sensitivity

These make program analysis **slow, imprecise,** or **both.**



Source Code Analysis Tools.

What is expensive to find

It's difficult for a tool to explore all the paths.

- Loops handled considering a small fixed number of iterations.
- Most tools ignore concurrency.
- Many tools ignore recursive calls.
- Many tools struggle with calls made through function pointers.

Common Weakness Enumeration (CWE)

- “CWE provides a unified, measurable set of software weaknesses”.
- “Allows a more effective use of software security tools”.
- 719 weaknesses in 244 categories.
- Id, description, consequences, examples, relationship, taxonomy mapping.

<http://cwe.mitre.org/>

Common Weakness Scoring System (CWSS)

- It “provides a mechanism for prioritizing software weaknesses in a consistent, flexible, open manner”.
- Based on three metric groups:
 - Base finding metric group.
 - Attack surface metric group.
 - Environmental metric group.

1. What You Need to Know about How Tools Work

2. The Tools And Their Use

Roadmap

- **Motivation**
- **Source code examples**
- **Tools for C/C++ applied to the source code**
- **Tools for Java applied to the source code**
- **The SWAMP**

What and Why

- Learn about different automated tools for vulnerability assessment.
- Start with small programs with weaknesses.
- Apply different tools to the programs.
- Understand the output, and strong and weak points of using specific tools.

CWE 78: OS Command Injection

```
1. void CWE78_OS_Command_Injection__char_console_execl_41_bad(){
2.     char *data; char dataBuffer[100] = "";
3.     data = dataBuffer;
4.     /* Read input from the console */
5.     size_t dataLen = strlen(data);
6.     /* If there is room in data, read into it from the cons */
7.     if (100-dataLen > 1) {
8.         /* POTENTIAL FLAW: Read data from the console */
9.         if (fgets(data+dataLen,(int)(100-dataLen),stdin)!= NULL)
10.        {
11.            /* Remove the carriage return from the string */
12.            dataLen = strlen(data);
13.            if (dataLen > 0 && data[dataLen-1] == '\n')
14.                data[dataLen-1] = '\0';
15.            else {
16.                printf("fgets() failed\n");
17.                data[dataLen] = '\0';
18.            }
19.            /* POTENTIAL FLAW: Execute command without
20.               validating */
21.            system (data);
22.        }
```

How to Describe a Weakness

Descriptive name of weakness (CWE XX)

An intuitive summary of the weakness.

- **Attack point:** How does the attacker affect the program.
- **Impact point:** Where in the program does the bad thing actually happen.
- **Mitigation:** A version of the program that does not contain the weakness.

(CWEXX_Long_Detailed_File_Name_Containing_The_Code_yy.cpp)

OS Command Injection (CWE 78)

User supplied data is used to create a string that will be interpreted by a command shell.

- **Attack Point:** Input read from the console.
- **Impact Point:** Executing command with `system()`.
- **Mitigation:** Don't execute user provided input; instead use a fixed string.

CWE78_OS_Command_Injection__char_console_execl_41.c
(Highly modified to compensate for errors.)

Tools for C/C++

- Goanna (RedLizards)
- Coverity analyze

Goanna (RedLizards)



- Commercial tool by **Red Lizard Software** available at redlizards.com
- The Goanna suite of static analysis tools pinpoints defects and vulnerabilities in C/C++ programs.
 - Access violations
 - Memory leaks
 - Array and string overruns
 - Division by zero
 - Unspecified, non-portable, and/or dangerous constructs
 - Security vulnerabilities

Goanna

1. Download Goanna Central
2. Activate the license and install the software
`./install-goanna`
3. Include in **PATH** the location of **goanna/bin**.
4. Initialize goanna for the project
`goanna-init`
3. Enable the security package:
`goanna-package --enable-pkg security`
4. Goanna Dashboard is the web interface to navigate and interact with analysis results.
`$goreporter start-server &`

Goanna

Three-step process:

- Run a full build of your program using the Goanna build integration utility to capture settings of the build.
`$goanna-trace make`
- Use this information from full build to run analysis.
`$goanna-analyze`
- Produce an analysis report
`$goanna-report`
- Read and interact with the analysis results.
 - After `goreporter` is running, load the provided URL in a web browser.

Goanna. OS Command Injection

\$ goanna-trace make

\$ goanna-analyze

\$ goanna-report

- **0 false positive.**
- **0 false negative.**
- **1 true positive:** It detects the command injection.

Goanna. OS Command Injection

The screenshot shows the Goanna Reporter web interface in Mozilla Firefox. The browser address bar shows the URL: localhost:1197/index.html#/project_id=2&snapshot_id=4&metric=SEC-INJECTION-o. The page title is "Goanna Reporter - 2-build-spec: Report - Mozilla Firefox". The main content area displays a table of warnings. The table has columns for File Directory, File Name, Line, Warning, Severity, Rules, Warning Message, Note, and Status. A single entry is shown for a file named "CWE78_OS_Command_Injection_char_console_excl_41.c" at line 69. The warning is "SEC-INJECTION-os" with a severity of "High" (indicated by a red circle). The rules listed are "cert-str02-c", "cert-env04-c", "cwe-78", "cwe-77", "sans-25-2", and "owasp-a1". The warning message states: "'data' contains user input and is used to executed a system command". The status is "Unclassified".

File Directory	File Name	Line	Warning	Severity	Rules	Warning Message	Note	Status
	CWE78_OS_Command_Injection_char_console_excl_41.c	69	SEC-INJECTION-os	High	cert-str02-c cert-env04-c cwe-78 cwe-77 sans-25-2 owasp-a1	'data' contains user input and is used to executed a system command		Unclassified

Goanna. OS Command Injection

The screenshot displays the Goanna Reporter web interface in Mozilla Firefox. The browser's address bar shows the URL `localhost:1197/index.html#/?project_id=2&snapshot_id=4&location=2&warning_id=`. The interface includes a navigation bar with tabs for 'Projects', '2-build-spec', and 'Code Browser'. On the left, a 'Warnings for file: CWE78_OS_Command_Injection_char_console' section is visible, containing a warning message: '69: [SEC-INJECTION-os] `data` contains user input and is used to executed a system command'. Below this, a 'Trace' section shows the execution flow for the function `CWE78_OS_Command_Injection_char_console`, highlighting the execution of `CWE78_OS_Command_Injection_char_console_execl_41_badSink(data)` at line 69. The main area is a code editor showing the source code for `CWE78_OS_Command_Injection_char_console_execl_41_badSink`. The code includes a buffer `dataBuffer` and a loop that reads user input from the console into `data`. A comment indicates a 'POTENTIAL FLAW: Read data from the console'. The code uses `fgets` to read input and `printf` to output an error message if `fgets` fails. The warning in the left pane points to line 69, which is highlighted in red in the code editor.

```
41 char * data;
42 char dataBuffer[100] = "";
43 data = dataBuffer;
44 {
45     /* Read input from the console */
46     size_t dataLen = strlen(data);
47     /* if there is room in data, read into it from the console */
48     if (100-dataLen > 1)
49     {
50         /* POTENTIAL FLAW: Read data from the console */
51         if (fgets(data+dataLen, (int)(100-dataLen), stdin) != NULL)
52         {
53             /* The next few lines remove the carriage return from the string that is
54              * inserted by fgets() */
55             dataLen = strlen(data);
56             if (dataLen > 0 && data[dataLen-1] == '\n')
57             {
58                 data[dataLen-1] = '\0';
59             }
60         }
61         else
62         {
63             printf("fgets() failed\n");
64             /* Restore NUL terminator if fgets fails */
65             data[dataLen] = '\0';
66         }
67     }
68 }
69 CWE78_OS_Command_Injection_char_console_execl_41_badSink(data);
70 }
71
72
73
```

Coverity Analyze

Coverity

- **Commercial tool. Available at <http://www.coverity.com/>**
- **Starting Point: Accurate Compilation.**
- **Depth and Accuracy of Analysis**
 - **Interprocedural Dataflow Analysis.**
 - **False Path Pruning.**
 - **Design Pattern Intelligence.**
 - **Enterprise Framework Analyzer.**
 - **White Box Fuzzer.**
- **Scalable.**

Coverity

1. **Download the license and the software:**
<https://coverity.secure.force.com/apex/LicenseManagement2>
2. **Run the installation script:** `cov-analysis-linux64-7.6.0.sh`
3. **Include in `PATH` the location of**
`~elisa/cov-analysis-linux64-7.6.0/bin`
4. **Command line and graphic interface.**

Coverity

Steps:

- **Generate a configuration for the compiler:**
`cov-configure --gcc`
- **Build the intermediate representation of the source code:**
`cov-build --dir <intermediate-dir> make`
- `cov-analyze --dir <intermediate-dir>`
- **Check the checkers included by `cov-analyze`:**
`cov-analyze --list-checkers`
- **Read and interact with the analysis results.**
- **Graphic mode: `cov-wizard`**

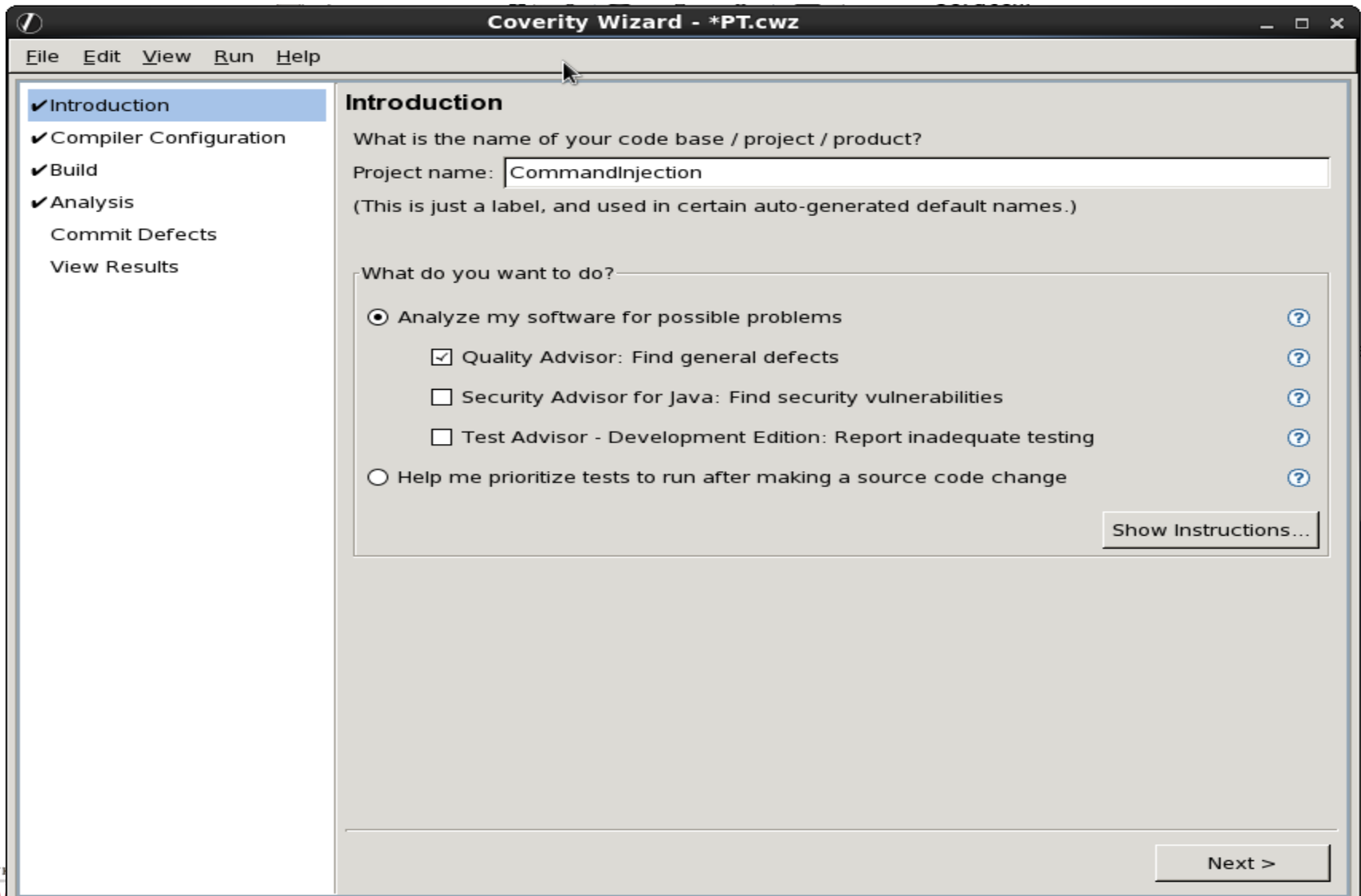
Coverity. OS Command Injection

```
$ cov-build --dir cov-comm-injection make
```

```
$ cov-analyze --dir cov-comm-injection  
--security
```

- **1 defect found.**
- **1 true positive:** It detects the command injection.

Coverity. OS Command Injection



Coverity. OS Command Injection

The screenshot shows the Coverity Wizard interface. The title bar reads "Coverity Wizard - *PT.cwz". The menu bar includes "File", "Edit", "View", "Run", and "Help". On the left, a sidebar contains a list of steps: "Introduction", "Compiler Configuration" (highlighted), "Build", "Analysis", "Commit Defects", and "View Results". The main area is titled "Compiler Configuration" and contains the following text: "The Coverity plugin will monitor builds performed with the compilers that are registered on this page. Please ensure that all compilers that are used to build the code in your workspace are registered here." Below this text, there is a section for "Compiler configuration" with a text field for the "Configuration file" containing the path "sa/cov-analysis-linux64-7.6.0/config/coverity_config.xml" and a "Browse..." button. A "Configured compilers:" table is shown below, listing two compilers. At the bottom of the main area, there are buttons for "Autoconfigure Compilers...", "Add...", "Edit...", "Duplicate...", and "Delete". The bottom of the window features a "Help" icon, a "< Previous" button, and a "Next >" button.

Compiler Configuration

The Coverity plugin will monitor builds performed with the compilers that are registered on this page. Please ensure that all compilers that are used to build the code in your workspace are registered here.

Compiler configuration

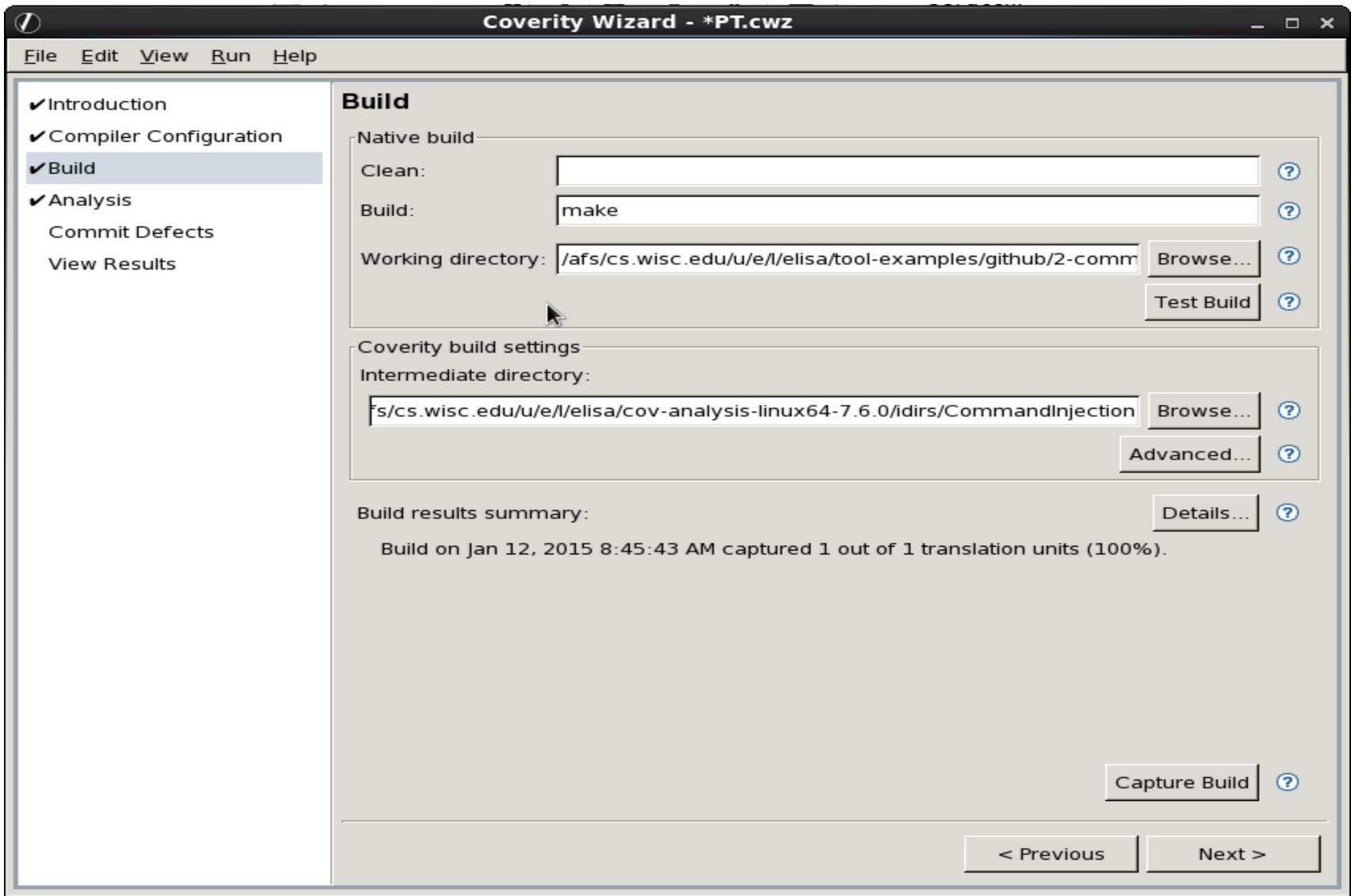
Configuration file:

Configured compilers:

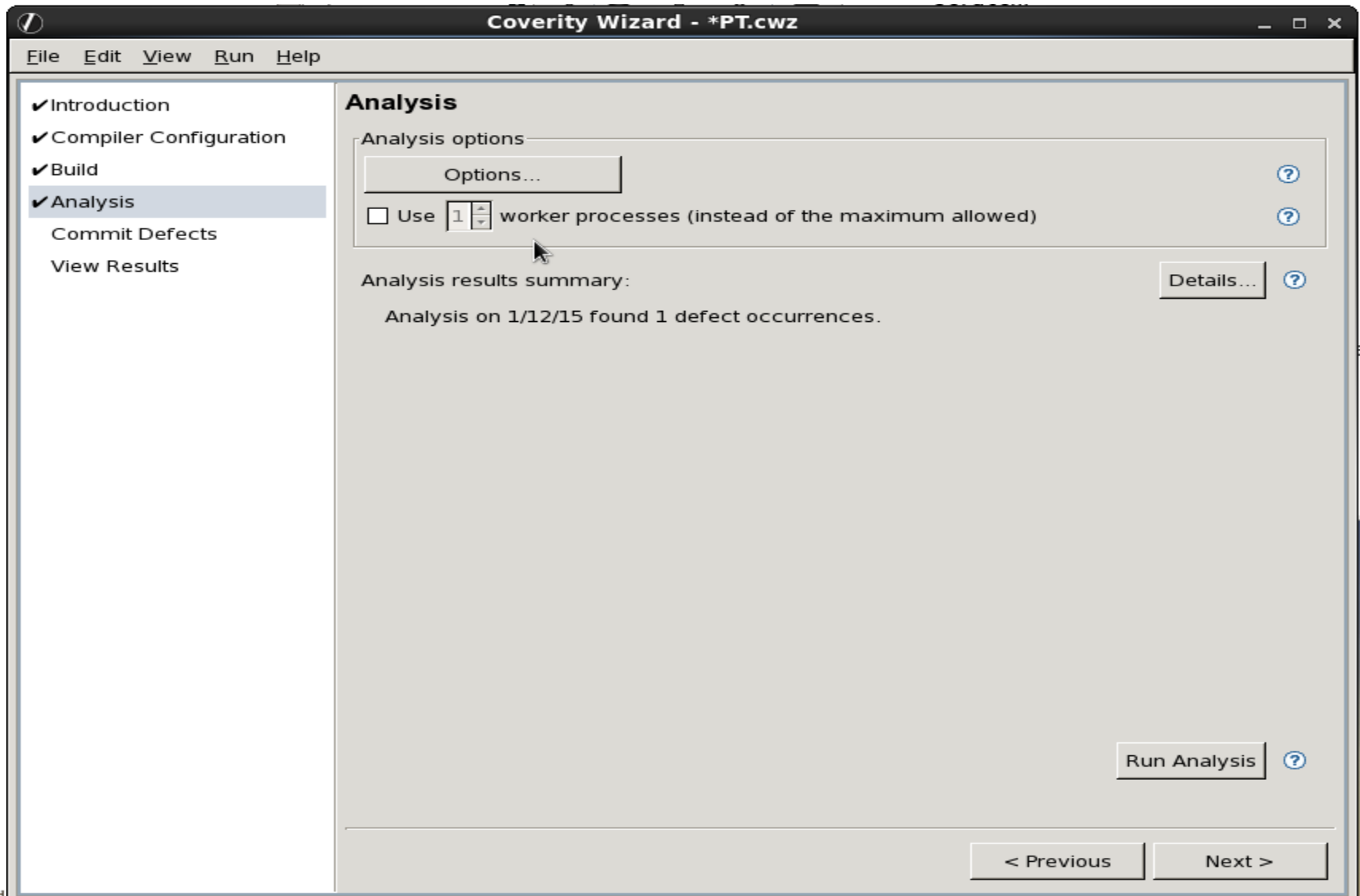
Name	Type	Executable	Template
template-gcc-config-0	GNU C compiler	gcc	✓
template-gcc-config-1	GNU C compiler	g++	✓



Coverity. OS Command Injection



Coverity. OS Command Injection



Coverity. OS Command Injection

```
Coverity Console
> /afs/cs.wisc.edu/u/e/l/elisa/cov-analysis-linux64-7.6.0/bin/cov-analyze --dir /afs/cs.

Coverity Static Analysis version 7.6.0 on Linux 2.6.32-431.3.1.el6.x86_64 x86_64
Internal version numbers: 9b77a50df0 p-harmony-push-21098.563

Using 4 workers as limited by CPU(s)
Looking for translation units
|0-----25-----50-----75-----100|
*****
[STATUS] Loading topological sort from disk (6 functions)
|0-----25-----50-----75-----100|
*****
[STATUS] Computing node costs
|0-----25-----50-----75-----100|
*****
[STATUS] Starting analysis run
|0-----25-----50-----75-----100|
*****
[STATUS] Calculating 46 cross-reference bundles...
|0-----25-----50-----75-----100|
*****
Analysis summary report:
-----
Files analyzed           : 1
Total LoC input to cov-analyze : 13485
Functions analyzed      : 6
Paths analyzed         : 25
Time taken by analysis  : 00:00:01
Defect occurrences found : 1 TAINTED_STRING

 Show commands only

Clear Close
```

Running Analysis

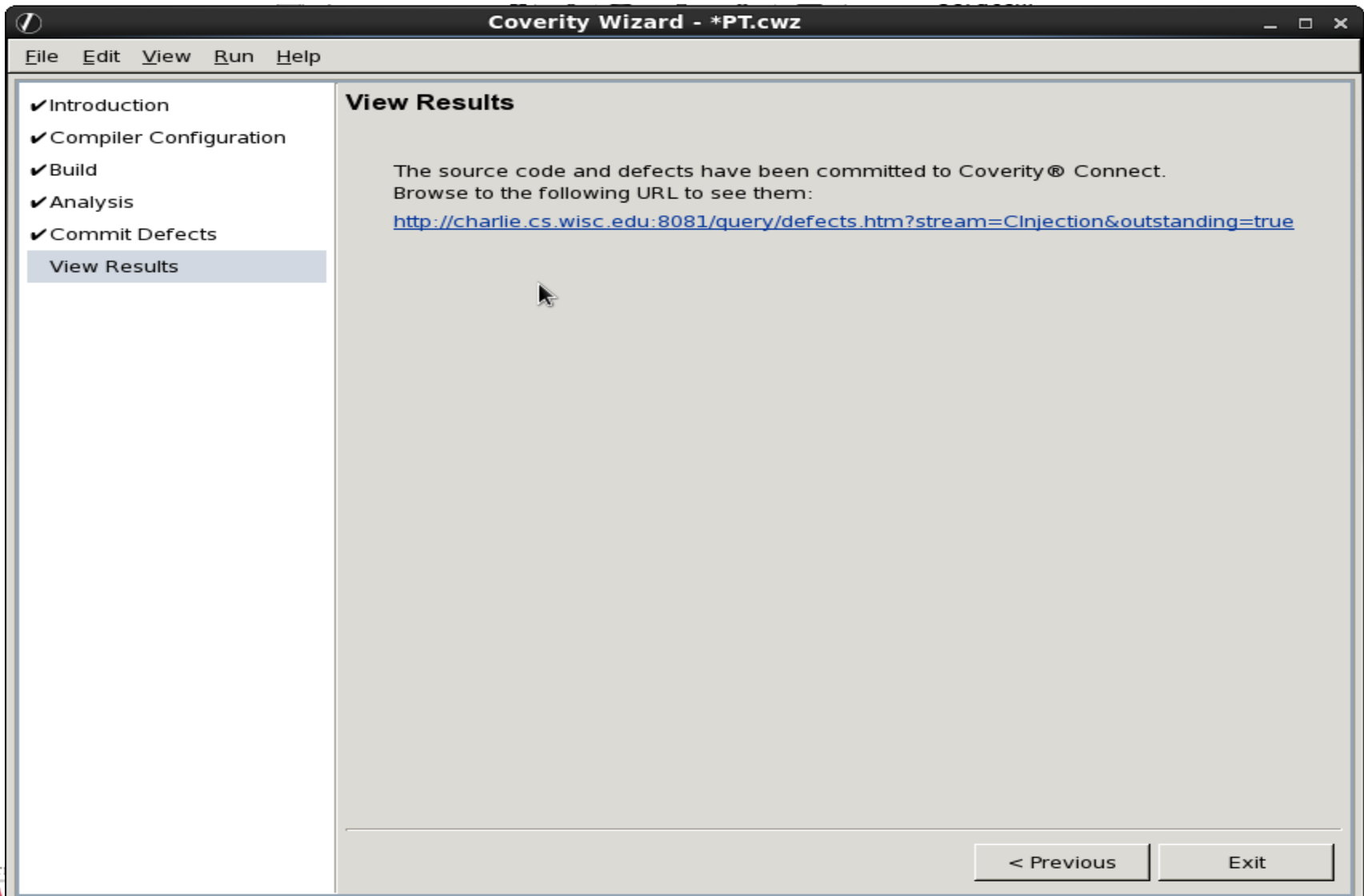
Running analysis

Analysis finished successfully.

Console... Close



Coverity. OS Command Injection



Coverity. OS Command Injection

The screenshot displays the Coverity Connect web interface in Mozilla Firefox. The browser address bar shows the URL: `charlie.cs.wisc.edu:8081/reports.htm#v10025/p10003/fileInstanceId:`. The interface includes a navigation bar with 'CInjection', 'Configuration', 'Help', and 'Admin User' options. Below this is a table of issues, with one issue selected: CID 10001, Type 'Use of untrusted string', Impact 'Medium', Status 'New', Count '1', First Detected '01/12/15', Owner 'Unassigned', and Classification 'Unclassified'. The main area shows a code editor for the file `CWE78_OS_Command_Injection_char_console_execl_41.c`. The code snippet is as follows:

```
42 char databuffer[100] = "";  
43 data = databuffer;  
44 {  
45     /* Read input from the console */  
46     size_t dataLen = strlen(data);  
47     /* if there is room in data, read into it from the console */  
48     1. Condition 100 - dataLen > 1, taking true branch  
49     if (100 - dataLen > 1)  
50     {  
51         /* POTENTIAL FLAW: Read data from the console */  
52         2. tainted_string_argument: fgets taints variable data.  
53         3. Condition fgets(data + dataLen, (int)(100 - dataLen), stdin) != NULL, taking false branch  
54         if (fgets(data+dataLen, (int)(100 - dataLen), stdin) != NULL)  
55         {  
56             /* The next few lines remove the carriage return from the string that is  
57             * inserted by fgets() */  
58             dataLen = strlen(data);  
59             if (dataLen > 0 && data[dataLen-1] == '\n')  
60             {
```

The right sidebar contains a triage panel for issue 10001, titled '10001 Use of untrusted string'. It provides a description: 'The string may be incorrectly assumed to not contain certain metacharacters or element names in later operations.' Below this are triage controls: Classification (Unclassified), Severity (Unspecified), Action (Undetermined), Ext. Reference (Type attribute), and Owner (Unassigned). There is also a text area for comments and buttons for 'Apply + Next' and 'Apply'.

Coverity. OS Command Injection

Coverity® Connect :: CInjection :: Unsaved view :: Issue 10001 - Mozilla Firefox

charlie.cs.wisc.edu:8081/reports.htm#v10025/p10003/fileInstanceId=

CInjection Configuration Help Admin User Enter CID(s)

Issues: By Snapshot | Unsaved view Filters: Status, Streams

CID	Type	Impact	Status	Count	First Detected	Owner	Classification	Severity
10001	Use of untrusted string	Medium	New	1	01/12/15	Unassigned	Unclassified	Unassigned

All 1 issue selected Page 1 of 1

```
CWE78_OS_Command_Injection__char_console_execl_41.c
56     if (dataLen > 0 && data[dataLen-1] == '\\')
57     {
58         data[dataLen-1] = '\\0';
59     }
60 }
61 else
62 {
63     printf("fgets() failed\n");
64     /* Restore NUL terminator if fgets fails */
65     data[dataLen] = '\\0';
66 }
67 }
68 }
```

CID 10001 (#1 of 1): Use of untrusted string value (TAINTED_STRING)
4. tainted_string: Passing tainted string data to CWE78_OS_Command_Injection__char_console_execl_41_badSink to accept tainted data. [show details]

```
69     CWE78_OS_Command_Injection__char_console_execl_41_badSink(data);
70 }
71 }
72 }
```

10001 Use of untrusted string
The string may be incorrectly assumed to not contain certain metacharacters or element names in later operations.
In CWE78_OS_Command_Injection__char_console_execl_41_badSink More

Triage

Classification: Uncl
Severity: Unsp
Action: Unde
Ext. Reference: Type attrib
Owner: Unassigne

Enter comments (See the Triage History section below for previous comments)

Apply + Next
Apply

Projects & Streams
Detection History

Java

CWE 601: Open Redirect

```
public void doGet(HttpServletRequest request,
1.           HttpServletResponse response)
2.           throws ServletException, IOException {
3.     response.setContentType("text/html");
4.     PrintWriter returnHTML = response.getWriter();
5.     returnHTML.println("<html><head><title>");
6.     returnHTML.println("Open Redirect");
7.     returnHTML.println("</title></head><body>");
8.
9.     String data;
10.    data = ""; // initialize data in case there are no cookies.
11.    // Read data from cookies.
12.    Cookie cookieSources[] = request.getCookies();
13.    if (cookieSources != null)
14.    // POTENTIAL FLAW: Read data from the first cookie value.
15.        data = cookieSources[0].getValue();
16.    if (data != null) {
17.        URI uri;
18.        uri = new URI(data);
19.        // POTENTIAL FLAW: redirect is sent verbatim.
20.        response.sendRedirect(data);
21.        return;
22.    }
```

Open Redirect (CWE 601)

Web app redirects user to malicious site chosen by an attacker.

- **Attack Point:** Reading data from the first cookie using `getCookies()`.
- **Impact Point:** `SendRedirect()` uses user supplied data.
- **GoodSource:** Use a hard-coded string.

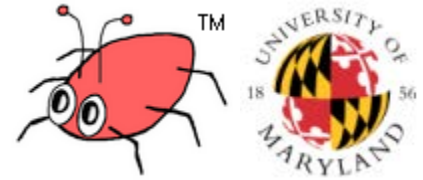
CWE601_Open_Redirect__Servlet_getCookies_Servlet_01.java
It's a Servlet

Tools for Java

- FindBugs
- Parasoft Jtest

FindBugs

FindBugs



- Open source tool available at findbugs.sourceforge.net/downloads.html
- Uses static analysis to look for bugs in Java code.
- Need to be used with the **FindSecurityBugs** plugin.
- Installation: Easy and fast.

FindBugs

1. Define **FINDBUGS_HOME** in the environment.
2. Install the **Find Security Bugs** plugin.
3. Learn the command line instructions and also use the graphical interface.

4. Command line interface:

```
$FINDBUGS_HOME/bin/findbugs -textui  
-javahome $JAVA_HOME  
RelativePathTRaversal.java
```

5. Graphic Interface: `java -jar`

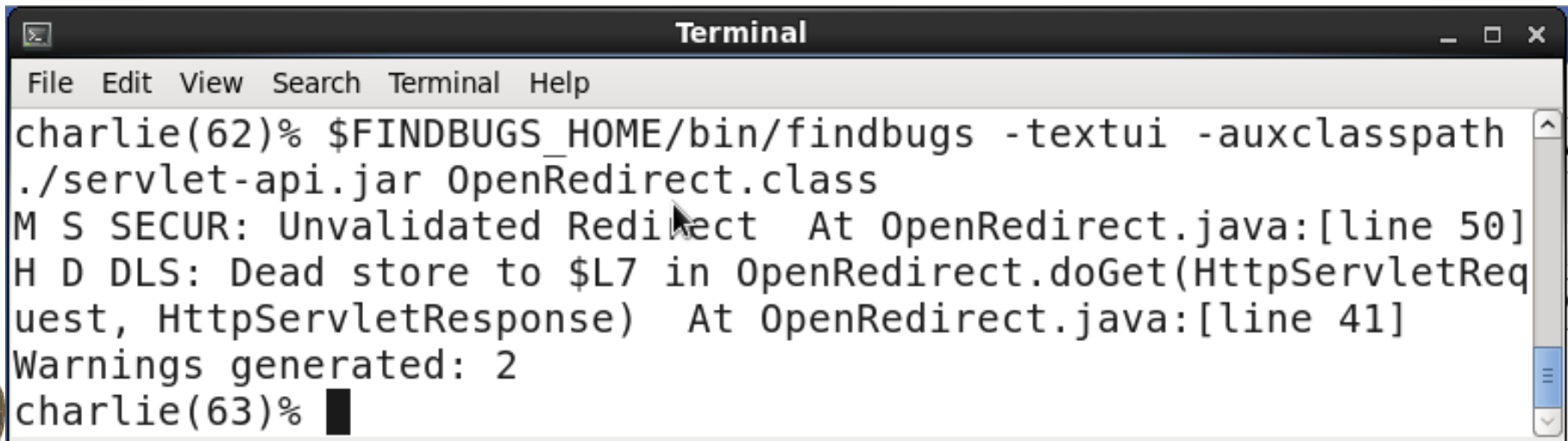
```
$FINDBUGS_HOME/lib/findbugs.jar -gui
```

FindBugs. Open Redirect

- FindBugs

- `$FINDBUGS_HOME/bin/findbugs -textui -auxclasspath ./servlet-api.jar OpenRedirect.class`

- **1 irrelevant warning.**
 - **1 true positive:** It detects the Open Redirect vulnerability.



```
Terminal
File Edit View Search Terminal Help
charlie(62)% $FINDBUGS_HOME/bin/findbugs -textui -auxclasspath
./servlet-api.jar OpenRedirect.class
M S SECUR: Unvalidated Redirect At OpenRedirect.java:[line 50]
H D DLS: Dead store to $L7 in OpenRedirect.doGet(HttpServletRequestReq
uest, HttpServletResponse) At OpenRedirect.java:[line 41]
Warnings generated: 2
charlie(63)%
```

FindBugs. Open Redirect

The screenshot displays the FindBugs application interface. The main window is titled "FindBugs" and contains several panels:

- Class name filter:** A text input field with a "Filter" button.
- Group bugs by:** A dropdown menu with options "Category", "Bug Kind", and "Bug Pattern".
- Bugs (2):** A tree view showing the hierarchy of bugs. The "Security (1)" folder is expanded, showing "Unvalidated Redirect (1)". The "Unvalidated Redirect (1)" folder is also expanded, showing a single bug instance with a yellow circle icon.
- OpenRedirect.java in:** A code editor showing the source code of the file. The code is as follows:

```
30     data = cookieSources[0].getValue();
31   }
32
33   if (data != null)
34   {
35     /* This prevents \r\n (and other chars) and should prevent incidentals such
36      * as HTTP Response Splitting and HTTP Header Injection.
37      */
38     URI uri;
39     try
40     {
41       uri = new URI(data);
42     }
43     catch (URISyntaxException exceptURISyntaxException)
44     {
45       response.getWriter().write("Invalid redirect URL");
46       return;
47     }
48     /* POTENTIAL FLAW: redirect is sent verbatim; escape the string to prevent ancillary
49      // IMPORTANT: Comment the 2 following lines to see the good case working!
50     response.sendRedirect(data);
51     return;
52   }
53
```
- Find, Next, Previous:** Buttons for navigating through the bugs.
- Unvalidated Redirect:** A detailed view of the selected bug. It shows the bug type "Unvalidated Redirect" and its location "At OpenRedirect.java:[line 50] In method OpenRedirect.doGet(HttpServletRequest, HttpServletR".
- Unvalidated Redirect:** A detailed description of the bug. It states: "Unvalidated redirects occur when an application redirects a user to a destination URL specified by a user supplied parameter that is not validated. Such vulnerabilities can be used to facilitate phishing attacks." It includes a "Scenario" section with a list of steps: 1. A user is tricked into visiting the malicious URL: `http://website.com/login?redirect=http://evil.vwebsite.com/fake/login`; 2. The user is redirected to a fake login page that looks like a site they trust. (`http://evil.vwebsite.com/fake/login`); 3. The user enters his credentials; 4. The evil site steals the user's credentials and redirects him to the original website. It concludes with: "This attack is plausible because most users don't double check the URL after the redirection. Also, redirection to an authentication page is very common."

Parasoft Jtest

Jtest



- Commercial tool available at <http://www.parasoft.com/product/jtest/>
- Automates a broad range of practices proven to improve development team productivity and software quality.
- Standalone Linux 9.5 version used.
 - gui mode and command line mode.
- Installation process: Slow download & easy installation.

Jtest

1. Include `/u/e/l/elisa/Jtest/9.5` in path.
2. Include the license.
3. Learn the command line instructions and also use the graphical interface.

Jtest

1. **Command line interface:** `$jtestcli`
`<options>`
2. **Graphic Interface:** `jtest&`
3. **Create a project and copy the .java files to the `project/src` directory.**
4. **Different tests available. We chose Security->CWE Top 25.**

Jtest. Open Redirect

Create the OpenRedir project.

Include servlet-api.jar in the OpenRedir project.

```
cp OpenRedirect.java
```

```
~elisa/parasoft/workspace1/OpenRedir/src
```

- **4 issues detected:**

- `getCookies()` returns tainted data.
- `cookieSources[0].getValue()` should be validated.
- 2 Open Redirect detected.

- **It detects the Open Redirect for both the good and bad cases.**

Jtest. Open Redirect

The screenshot displays an IDE window titled "Java - OpenRedir/src/OpenRedirect.java - Parasoft Jtest". The main editor shows the following Java code:

```
data = ""; /* initialize data in case there are no cookies */  
/* Read data from cookies */  
Cookie cookieSources[] = request.getCookies();  
if (cookieSources != null) {  
    /* POTENTIAL FLAW: Read data from the first cookie value */  
    data = cookieSources[0].getValue();  
}  
  
if (data != null)  
{  
    /* This prevents \r\n (and other chars) and should prevent incidentals such  
    * as HTTP Response Splitting and HTTP Header Injection.  
    */  
    URI uri;  
    try  
    {  
        uri = new URI(data);  
    }  
}
```

The IDE interface includes a menu bar (File, Edit, Source, Refactor, Navigate, Search, Project, Parasoft, Run, Window, Help), a toolbar with various icons, and a sidebar with the following panels:

- Task List:** Contains a search field and buttons for "All" and "Activate...".
- Outline:** Shows a tree view with "import declarations" and "OpenRedirect" (expanded to show "doGet(HttpServletRequest, Ht").
- Problems:** Shows "0 errors, 2 warnings, 0 others".
- Quality Tasks:** Shows a list of warnings.

The "Quality Tasks" panel displays the following warnings:

- SECURITY.IBA.VPPD: 'getCookies()' is a tainted data-returning method and should be encapsulated by a validation
- SECURITY.IBA.VPPD: 'getValue()' is a dangerous data-returning method and should be encapsulated by a validation

The status bar at the bottom of the IDE shows the warning: "SECURITY.IBA.VPPD: 'getCookies()' is a ta...nd should be encapsulated by a validation".

Jtest. Open Redirect

The screenshot shows an IDE window titled "Java - OpenRedir/src/OpenRedirect.java - Parasoft Jtest". The main editor displays the following Java code:

```
data = ""; /* initialize data in case there are no cookies */
/* Read data from cookies */
Cookie cookieSources[] = request.getCookies();
if (cookieSources != null) {
    /* POTENTIAL FLAW: Read data from the first cookie value */
    data = cookieSources[0].getValue();
}

if (data != null)
{
    /* This prevents \r\n (and other chars) and should prevent incidentals such
    * as HTTP Response Splitting and HTTP Header Injection.
    */
    URI uri;
    try
    {
        uri = new URI(data);
    }
}
```

The IDE interface includes a menu bar (File, Edit, Source, Refactor, Navigate, Search, Project, Parasoft, Run, Window, Help), a toolbar with various icons, and several panels on the right:

- Task List:** Contains a search field and buttons for "All" and "Activate...".
- Outline:** Shows a tree view with "import declarations" and "OpenRedirect" expanded, listing "doGet(HttpServletRequest, Http)".

At the bottom, the **Problems** panel shows "0 errors, 2 warnings, 0 others". The description of the warnings is:

- Warning 1: SECURITY.IBA.VPPD: 'getCookies()' is a tainted data-returning method and should be encapsulated by a validation
- Warning 2: SECURITY.IBA.VPPD: 'getValue()' is a dangerous data-returning method and should be encapsulated by a validation

The status bar at the bottom of the IDE displays the message: "SECURITY.IBA.VPPD: 'getValue()' is a dang...nd should be encapsulated by a validation".

Jtest. Open Redirect

The screenshot shows an IDE window titled "Java - OpenRedir/src/OpenRedirect.java - Parasoft Jtest". The main editor displays the following Java code:

```
OpenRedirect.java
{
/* This prevents \r\n (and other chars) and should prevent incidentals such
 * as HTTP Response Splitting and HTTP Header Injection.
 */
URI uri;
try
{
    uri = new URI(data);
}
catch (URISyntaxException exceptURISyntax)
{
    response.getWriter().write("Invalid redirect URL");
    return;
}
/* POTENTIAL FLAW: redirect is sent verbatim; escape the string to prevent a
// IMPORTANT: Comment the 2 following lines to see the good case working!
response.sendRedirect(data);
return;
```

On the right side, the Task List panel shows a search bar and a list of tasks. The Outline panel shows the project structure with "OpenRedirect" expanded, showing a task for "doGet(HttpServletRequest, Http)".

At the bottom, the Problems panel shows "0 errors, 4 warnings, 0 others". The description of the warnings is as follows:

Description
⚠ SECURITY.IBA.VPPD: 'getCookies()' is a tainted data-returning method and should be encapsulated by a validation
⚠ SECURITY.IBA.VPPD: 'getValue()' is a dangerous data-returning method and should be encapsulated by a validation
⚠ SECURITY.IBA.VRD: No validation check in redirect URL
⚠ SECURITY.IBA.VRD: No validation check in redirect URL

The status bar at the bottom of the IDE displays the warning: "SECURITY.IBA.VRD: No validation check in redirect URL".

Roadmap

- **What is the SWAMP?**
- **Using the SWAMP**
 - Register
 - Create a project
 - Upload your software package
 - Run your assessment
 - View the results
 - Java
 - C/C++

<https://continuousassurance.org/swamp/SWAMP-User-Manual.pdf>

Getting Started with the SWAMP

- **Software Assurance Market Place.**
- **Objective:** Automate and simplify the use of (multiple) tools.
- A national, no-cost resource for software assurance (SwA) technologies used across research institutions, non-governmental organizations, and civilian agencies and their communities as both a research platform and a core component of the software development life cycle.

Register to use the SWAMP

https://www.mir-swamp.org/#

MozBackup.Ink Más visitados Programmes & Initiati... Cerca Comenzar con Firefox Iniciar sesión Últimas noticias Tanca la Sessió

SWAMP About Contact Resources Policies Help Sign In



SWAMP

SOFTWARE ASSURANCE MARKETPLACE

Welcome to the SWAMP

The Software Assurance Marketplace (SWAMP) is a service that provides continuous software assurance capabilities to developers and researchers.

This no-cost code analysis service is open to the public. Let the SWAMP help you to build better, safer, and more secure code today!

[Sign Up!](#) [Sign Up with GitHub!](#)

Get results in just three steps:

Rather than spending time installing, licensing and configuring software assessment tools on your own machine, let the SWAMP do the work for you.

- 1) Upload your package**
First, upload your software code. Rest assured that it will remain private and secure.
- 2) Create / run assessment**
Next, create and run an assessment by choosing a package, tool, and platform.
- 3) View your results**
Last, view your results using a native viewer or Code Dx™ for full featured analysis.



What can I do in the SWAMP?

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

ort Strategy ... IEEE http...e.jsp Mission and ... 10Kstudents: ... Software ... Software ... x Microsoft Wo... Software ... Software Ass...

https://www.mir-swamp.org/#home

ntinuos assurance swamp

MozBackup.Ink Más visitados Programmes & Initiati... Cerca Comenzar con Firefox Iniciar sesión Últimas noticias Tanca la Sessió

SWAMP About Contact Resources Policies Help Welcome, elisa Sign Out

CONTINUOUS ASSURANCE

SWAMP

SOFTWARE ASSURANCE MARKETPLACE

Do It Early. Do It Often.

Feature	Count	Description
Packages	14	Create and manage your software packages and upload your code for assessment.
Assessments	32	Perform assessments on a software package using our software analysis tools.
Results	34	View assessments' status and the results of completed assessments.
Runs	0	View assessments that are scheduled to run periodically at regular intervals.
Projects	2	Create and manage projects to share assessment results with other SWAMP users.
Events	11	View the events associated with your user account and projects.

MADISON de Barcelona

U100 MELAND SECUR OLAN

Create a Project

MozBackup.Ink Más visitados Programmes & Initiati... Cerca Comenzar con Firefox Iniciar sesión Últimas noticias Tanca la Sessió

SWAMP About Contact Resources Policies Help Welcome, elisa Sign 0

Add New Project

Home / Projects / + Add New Project

Please enter the details of your new project below.

Full name * ✓

Short name * ✓

Description * ✓

*Fields are required

+ Save Project **Cancel**

My Projects

Browser address bar: <https://www.mir-swamp.org/#projects>

Browser tabs: MozBackup.Ink, Más visitados, Programmes & Initiati..., Cerca, Comenzar con Firefox, Iniciar sesión, Últimas noticias, Tanca la Sessió

Navigation bar: SWAMP | About | Contact | Resources | Policies | Help | Welcome, elisa | Sign Out

- +** (highlighted)
- ✓
- 🗑️
- 📄
- 📁
- 🔊
- 🔍
- 📺
- 📷

Projects

Home / Projects

Projects are used to share assessment results with other SWAMP users. You can invite other users to join a project and then all members of the project can add assessments to that project and view assessment results belonging to that project.

[+ Add New Project](#)

Projects I Own

Name	Description	Date Added	
Tutorial Java	Tutorial Java	2014-11-13 14:59	✕
Tools tutorial	Tools tutorial	2014-10-09 15:33	✕

Projects I Joined

No projects.

Show numbering

Upload your Software Package

Browser address bar: <https://www.mir-swamp.org/#packages/add>

Browser tabs: MozBackup.Ink, Más visitados, Programmes & Initiati..., Cerca, Comenzar con Firefox, Iniciar sesión, Últimas noticias, Tanca la Sessió

Navigation bar: SWAMP | About | Contact | Resources | Policies | Help | Welcome, elisa | Sign Out

Add New Package

Home / Packages / Add New Package

Details | Source | Build | Sharing

Name * ✓

Description ✓

External URL

File 10-open-redirect.tar ✓
formats supported

Version *

Version notes

PACKAGE INFO

PACKAGE VERSION INFO

My software Packages

The screenshot shows the SWAMP website interface. The browser address bar displays <https://www.mir-swamp.org/#packages>. The navigation bar includes links for About, Contact, Resources, Policies, and Help, along with a user profile for 'elisa' and a 'Sign Out' button. The main content area is titled 'Packages' and features a sidebar with a checkmark icon circled in red. Below the title, there is a breadcrumb trail 'Home / Packages' and a descriptive paragraph. A filter section shows 'any project', '</> any type', 'any date', and 'all items'. A '+ Add New Package' button is located on the right. A table lists several packages, with the first row circled in red.

SWAMP About Contact Resources Policies Help Welcome, elisa Sign Out

Packages

Home / Packages

Packages are collections of files containing code to be assessed along with information about how to build the software package, if necessary. Packages may be written in a variety of programming languages and may have multiple versions.

Filters: any project </> any type any date all items

+ Add New Package

Name	Description	Type	Date Added	
J Open Redir		Java Source Code	2015-01-07 18:26	✕
Java Command Injectio		Java Source Code	2014-11-13 21:31	✕
Java Path Traversal	Java Path Traversal	Java Source Code	2014-11-13 21:20	✕
our example		C/C++	2014-10-09 20:51	✕

Run your Assessments

Browser address bar: <https://www.mir-swamp.org/#assessments/run>

Browser tabs: MozBackup.Ink, Más visitados, Programmes & Initiati..., Cerca, Comenzar con Firefox, Iniciar sesión, Últimas noticias, Tanca la Sessió

Navigation: **SWAMP** | About | Contact | Resources | Policies | Help | Welcome, elisa | Sign Out

Run New Assessment

Home / Assessments / Run New Assessment

To create a new assessment, please specify the following information:

Package

Select a package to assess:

Select a version:

Tool

Select a tool to perform the assessment:

Select a version:

Buttons: **Save and Run** | Save | Cancel

My Assessments

Browser address bar: <https://www.mir-swamp.org/#assessments>

Browser tabs: MozBackup.Ink, Más visitados, Programmes & Initiati..., Cerca, Comenzar con Firefox, Iniciar sesión, Últimas noticias, Tanca la Sessió

Navigation: About, Contact, Resources, Policies, Help, Welcome, elisa, Sign Out

ID	Assessment Name	Tool	OS	Status	Actions
16	integer overflow latest	Clang Static Analyzer latest	Red Hat Enterprise Linux 64-bit latest	Running	Stop
17	J Open Redir latest	Parasoft Jtest latest	Red Hat Enterprise Linux 64-bit latest	Running	Stop
18	Java Command Injectio latest	Parasoft Jtest latest	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Running	Stop
19	Java Command Injectio latest	error-prone latest	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Running	Stop
20	Java Command Injectio latest	Findbugs latest	Red Hat Enterprise Linux 64-bit latest	Running	Stop
21	Java Path Traversal latest	Parasoft Jtest latest	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Running	Stop
22	Java Path Traversal latest	error-prone latest	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Running	Stop
23	Java Path Traversal latest	Findbugs latest	Red Hat Enterprise Linux 64-bit latest	Running	Stop
24	NIST Juliet C CWE023_01 1.2 latest	cppcheck latest	Red Hat Enterprise Linux 64-bit latest	Running	Stop
25	NIST Juliet C CWE023_01 1.2 latest	Clang Static Analyzer latest	Red Hat Enterprise Linux 64-bit latest	Running	Stop

View your Results

Browser address bar: <https://www.mir-swamp.org/#results>

Browser tabs: MozBackup.Ink, Más visitados, Programmes & Initiati..., Cerca, Comenzar con Firefox, Iniciar sesión, Últimas noticias, Tanca la Sessió

Navigation: About, Contact, Resources, Policies, Help, Welcome, elisa, Sign Out

Date / Time	Package	Tool	Platform	Status	Results
2015-03-12 17:44	J Open Redir 1.0	Parasoft Jtest 9.5.13	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Finished	<input type="checkbox"/> <input type="checkbox"/>
2014-12-15 21:55	Java Path Traversal 1.0	Parasoft Jtest 9.5.13	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Finished	<input type="checkbox"/> <input type="checkbox"/>
2014-12-15 21:55	Java Command Injectio 1.0	Parasoft Jtest 9.5.13	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Finished	<input type="checkbox"/> <input type="checkbox"/>
2014-12-15 19:52	hard coded password 1.0	Parasoft C/C++test 9.5.4.103	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Finished	<input type="checkbox"/> <input type="checkbox"/>
2014-12-15 19:52	info exposure 1.0	Parasoft C/C++test 9.5.4.103	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Finished	<input type="checkbox"/> <input type="checkbox"/>
2014-12-15 19:52	our example 1.0	Parasoft C/C++test 9.5.4.103	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Finished	<input type="checkbox"/> <input type="checkbox"/>
2014-12-09 21:18	Java Command Injectio 1.0	error-prone 1.1.1	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Finished	<input type="checkbox"/> <input type="checkbox"/>
2014-12-09 21:18	Java Path Traversal 1.0	error-prone 1.1.1	Red Hat Enterprise Linux 64-bit RHEL6.4 64-bit	Finished	<input type="checkbox"/> <input type="checkbox"/>

Browser address bar: <https://www.mir-swamp.org/#tools/6197a593-6440-11e4-a282-001a4a81450b>

My Results for Java: Jtest – J Open Redir - Native

ps-jtest v9.5 Report

Summary

Total
5

Group	File	Line	Message
SECURITY.IBA	/TempProject/10-b-OpenRedirect /OpenRedirect.java	27	'getCookies()' is a tainted data-returning method and should be encapsulated by a validation
SECURITY.IBA	/TempProject/10-b-OpenRedirect /OpenRedirect.java	30	'getValue()' is a dangerous data-returning method and should be encapsulated by a validation
	/TempProject/10-b-OpenRedirect /OpenRedirect.java	41	Injection of data received from servlet request ("data") to method accepting network resource properties
	/TempProject/10-b-OpenRedirect /OpenRedirect.java	50	Injection of data received from servlet request ("data") to http response
	/TempProject/10-b-OpenRedirect /OpenRedirect.java	59	Condition "data != null"; always evaluates to true



My Results for Java: Jtest – J Open Redir - CodeDx

The screenshot shows the CodeDx web interface. The browser address bar displays the URL: <https://swa-csaweb-pd-01.mir-swamp.org/proxy-EDDB9342-C8E3-11E4-AE7E>. The page title is "J Open Redir » Analysis Run 11". The interface includes a navigation menu with "Projects" and "Help", and a user login status "Logged in as elisa". The version information is "version 1.5.1-SW-1 - 9/29/2014".

The main content area shows the analysis results for "J Open Redir". It includes a "Filters" sidebar on the left and a "Weaknesses" table on the right. The "Filters" sidebar shows the following settings:

- Weakness count: 5 / 5
- Tool: ps-jtest (100%)
 - SECURITY.IBA (40%)
 - Unknown (60%)
- Severity: Unspecified (100%)
- Codebase Location
- Tool Overlaps
- CWE
- Status: Unresolved (100%)

The "Weaknesses" table displays the following data:

Id	Tool	Rule	CWE	Sev.	Codebase Locat.	Status
50	ps-jt...	BD.PB.CC	-	Uns...	OpenRedirect.ja...	Unresolved
49	ps-jt...	BD.SECURITY.TDR...	-	Uns...	OpenRedirect.ja...	Unresolved
48	ps-jt...	BD.SECURITY.TDN...	-	Uns...	OpenRedirect.ja...	Unresolved
47	ps-jt...	SECURITY.IBA.VPPD	-	Uns...	OpenRedirect.ja...	Unresolved
46	ps-jt...	SECURITY.IBA.VPPD	-	Uns...	OpenRedirect.ja...	Unresolved

The table indicates that 5 weaknesses are displayed, and the current view shows 1 to 5 of 5 weaknesses. The ID 48 is circled in red.

My Results for Java: Jtest – J Open Redir - CodeDx

The screenshot shows a web browser window displaying the CodeDx application. The address bar shows a proxy URL. The page header includes navigation links for 'Projects' and 'Help', a user login 'Logged in as elisa', the version '1.5.1-SW-1 - 9/29/2014', and the CodeDx logo with the tagline 'A PRODUCT OF SECURE DECISIONS'. The main content area is titled 'J Open Redir > Analysis Run 11 > Weakness 48'. Below this, it states 'BD.SECURITY.TDNET detected by ps-jtest with Unspecified severity', 'First seen on 3/12/2015', '5 weaknesses in this file', and '1 similar weakness in this analysis run'. A 'jump to weakness' link is visible. The page is divided into three main sections: 'Status', 'Description', and 'Source Code'. The 'Status' section shows a dropdown menu set to 'Unresolved'. The 'Description' section contains the text 'Injection of data received from servlet request ("data"); to method accepting network resource properties', which is circled in red. The 'Source Code' section shows the weakness occurs in '10-b-OpenRedirect/OpenRedirect.java' on line 41, followed by a code snippet starting with 'import javax.servlet.*;' and 'public class OpenRedirect extends HttpServlet {'.

https://swa-csaweb-pd-01.mir-swamp.org/proxy-EDDB9342-C8E3-11E4-AE7E

Buscar

Projects Help Logged in as elisa version 1.5.1-SW-1 - 9/29/2014

CodeDx
A PRODUCT OF SECURE DECISIONS

J Open Redir > Analysis Run 11 > Weakness 48

BD.SECURITY.TDNET detected by ps-jtest with Unspecified severity

First seen on 3/12/2015 5 weaknesses in this file 1 similar weakness in this analysis run

No Common Weakness Enumeration information available [jump to weakness](#)

Status

Unresolved

Description

Injection of data received from servlet request ("data"); to method accepting network resource properties

Source Code

The weakness occurs in **10-b-OpenRedirect/OpenRedirect.java** on line **41**

```
1 import javax.servlet.*;
2 import javax.servlet.http.*;
3 import java.io.*;
4 import java.net.URI;
5 import java.net.URISyntaxException;
6
7 public class OpenRedirect extends HttpServlet {
8
9     public void doGet(HttpServletRequest request,
```

Activity Stream

Post Clear

Write comments with Markdown

admin changed status to **Unresolved**

20 minutes ago

admin changed status to **New**

about an hour ago

My Results for Java: Jtest – J Open Redir - CodeDx

https://swa-csaweb-pd-01.mir-swamp.org/proxy-EDDB9342-C8E3-11E4-AE7E

J Open Redir > Analysis Run 11 > Weakness 48

BD.SECURITY.TDNET detected by **ps-jtest** with **Unspecified** severity
First seen on **3/12/2015** 5 weaknesses in this file 1 similar weakness in this analysis run
No Common Weakness Enumeration information available

Status
Unresolved

Activity Stream

admin changed status to **Unresolved** 16 minutes ago
admin changed status to **New** about an hour ago

The weakness occurs in **10-b-OpenRedirect/OpenRedirect.java** on line **41**

```
30     data = cookieSources[0].getValue();
31 }
32
33     if (data != null)
34     {
35         /* This prevents \r\n (and other chars) and should prevent
36         incidentals such
37         * as HTTP Response Splitting and HTTP Header Injection.
38         */
39         URI uri;
40         try
41         { uri = new URI(data);
42         }
43         catch (URISyntaxException exceptURISyntax)
44         {
45             response.getWriter().write("Invalid redirect URL");
46             return;
47         }
48         /* POTENTIAL FLAW: redirect is sent verbatim; escape the string to
49         prevent ancillary issues like XSS, Response splitting etc */
50         // IMPORTANT: Comment the 2 following lines to see the good case
51         working
52         response.sendRedirect(data);
```

Questions?

Elisa Heymann

Barton P. Miller

Elisa.Heymann@uab.es

bart@cs.wisc.edu

<http://www.cs.wisc.edu/mist/>