Automated Assessment Tools Theory & Practice

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

August 2018

Overview

- Very dangerous: Injection Attacks.
- Introduction to automated assessment tools.
- The SWAMP.
- Hands-on exercise in Java and the SWAMP.

Injection Attacks

Objectives

- Understand the general problem of injections.
- Understand what are SQL injections, and how to mitigate them.
- Understand what are Command injections, and how to mitigate them.

The Basic Idea

The attacker's goal:

> ***Getting the system to execute commands that were not intended (or desired) by the programmer.***

In other words, can I put words into the system's mouth?

Let's look at an example based on a popular (and silly) game.

The Word Game

Ask for a list of words:

❶ A vehicle: chariot

❷ An outdoor location: rooftop

❸ A food: scrambled eggs

❹ Another food: pickles

❺ A sport: javelin throwing

❻ A relaxing activity: stand on our heads

The Word Game

Ask for a list of words:

❶ A vehicle: chariot

❷ An outdoor location: rooftop

❸ A food: scrambled eggs

❹ Another food: pickles

❺ A sport: javelin throwing

❻ A relaxing activity: stand on our heads

Then use them in story:

It was a lovely day for a picnic, so we packed the _ ❶ and headed to the ❷. The basket was loaded full of delicious ❸ and ❹. We spread out our blanket and first decided to play ❺ and then ❻ for a while.

The Word Game

Ask for a list of words:

❶ A vehicle: chariot

❷ An outdoor location: rooftop

❸ A food: scrambled eggs

❹ Another food: pickles

❺ A sport: javelin throwing

❻ A relaxing activity: stand on our heads

Then use them in story:

It was a lovely day for a picnic, so we packed the chariot and headed to the rooftop. The basket was loaded full of delicious scrambled eggs and pickles. We spread out our blanket and first decided to play javelin throwing and then stand on our heads for a while.

But it can take a darker turn …

The Word Game

Ask for a list of words:

❶ A vehicle: chariot

❷ An outdoor location: rooftop

❸ A food: scrambled eggs

❹ Another food: pickles

❺ A sport: javelin throwing

❻ A relaxing activity: stand on your heads

go to the bank and rob it, while we stay here

Then use them in story:

It was a lovely day for a picnic, so we packed the chariot and headed to the rooftop. The basket was loaded full of delicious scrambled eggs and pickles. We spread out our blanket and first decided to play javelin throwing and then relax. Hey kids, now go to the bank and rob it, while we stay here for a while.

So, What Went Wrong?

The creator of the game made assumptions about the words to be provided (the input).

And they trusted the person providing the words to be reasonable and not cause someone to do something illegal.

In other words, they did not check the input nor did they try to prevent any abuse.

Now, let's look at this in a more technical way …

Injection Attacks

- **Description:**
  - A string constructed with user input, that is then interpreted by another function, where the string is not parsed as expected
    - Command injection (in a shell)
    - SQL injection
    - Code injection (to a language interpreter)
    - XML injections
- **General causes:**
  - Allowing metacharacters in the user input
  - Not properly neutralizing user data if metacharacters are allowed.

The Attack Surface

Network Data

Database Entry

Input File

Environ Variable

Web Form Field

Program Option

```
void execSafeProgram(String programName) throws Exception
{
  // only allow execution of programs found in this directory
  String safeDir = "cmd /c C:\\safe_programs\\";

  // check for separator character to prevent path traversal
  if (programName.contains(File.separator)) {
      throw new SecurityException("Path Traversal Detected");
  }

  // start specified program from the safe directory
  Process p = Runtime.getRuntime().exec(safeDir +
programName);
  p.waitFor();
}
```

Input from the user, the *attack surface:*

Web Form

User: `bart`

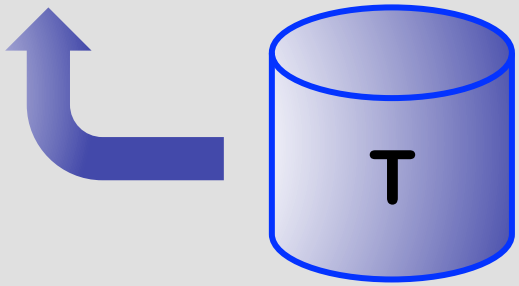
Network Data

`xxxx bart xxxxx`

Effecting the attack, *impact surface:*

Database Server

select * from T where u = $input



Command Shell

```
%
% mail $input < message
%
```

Interpreter

prog = begin + input + end
eval prog

SQL Injection Attacks

SQL Injections

- User supplied values used in SQL command must be validated, quoted, escaped, or prepared statements must be used.
- Signs of vulnerability:
  - Uses a database mgmt system (DBMS).
  - Creates SQL statements at run-time.
  - Inserts user supplied data directly into statement without validation.

SQL Injections:
attacks and mitigations

PERL

- **Dynamically generated SQL without validation or quoting is vulnerable**

```
$u = " '; drop table t --";
$sth = $dbh->do("select * from t where u = '$u'");
```

Database sees <u>two</u> statements:

```
select * from t where u = ' '; drop table t --'
```

- **Use *prepared statements* to mitigate**

```
$sth = $dbh->do("select * from t where u = ?", $u);
```

  - SQL statement template and value sent to database
  - No mismatch between intention and use

Successful SQL Injection Attack

2. DB Queried

```
SELECT * FROM members
WHERE u='admin' AND p='' OR 'x'='x'
```

3. Returns all row of table members

1. User sends malicious data

```
user="admin"; pwd="'OR 'x'='x"
```

```
boolean Login(String user, String pwd)  {
    boolean loggedIn = false;
    conn = pool.getConnection( );
    stmt = conn.createStatement();
    rs = stmt.executeQuery("SELECT * FROM members"
                           + "WHERE u='" + user
                           + "' AND p='" + pwd + "'");
    if (rs.next())
        loggedIn = true;
}
```

4. System grants access

```
Login() returns true
```

Successful SQL Injection Attack

2. DB Queried

```
SELECT * FROM members
WHERE u='admin' AND p='' OR 'x'='x'
```

3. Returns all row of table members

1. User sends malicious data

```
user="admin"; pwd="'OR 'x'='x"
```

```
boolean Login(String user, String pwd)  {
    boolean loggedIn = false;
    conn = pool.getConnection( );
    stmt = conn.createStatement();
    rs = stmt.executeQuery("SELECT * FROM members"
                           + "WHERE u='" + user
                           + "' AND p='" + pwd + "'");
    if (rs.next())
        loggedIn = true;
}
```

4. System grants access

```
Login() returns true
```

Mitigated SQL Injection Attack

```
SELECT * FROM members WHERE u = ?1 AND p = ?2
        ?1 = "admin"    ?2 = "' OR 'x'='x"
```

2. DB Queried

3. Returns null set

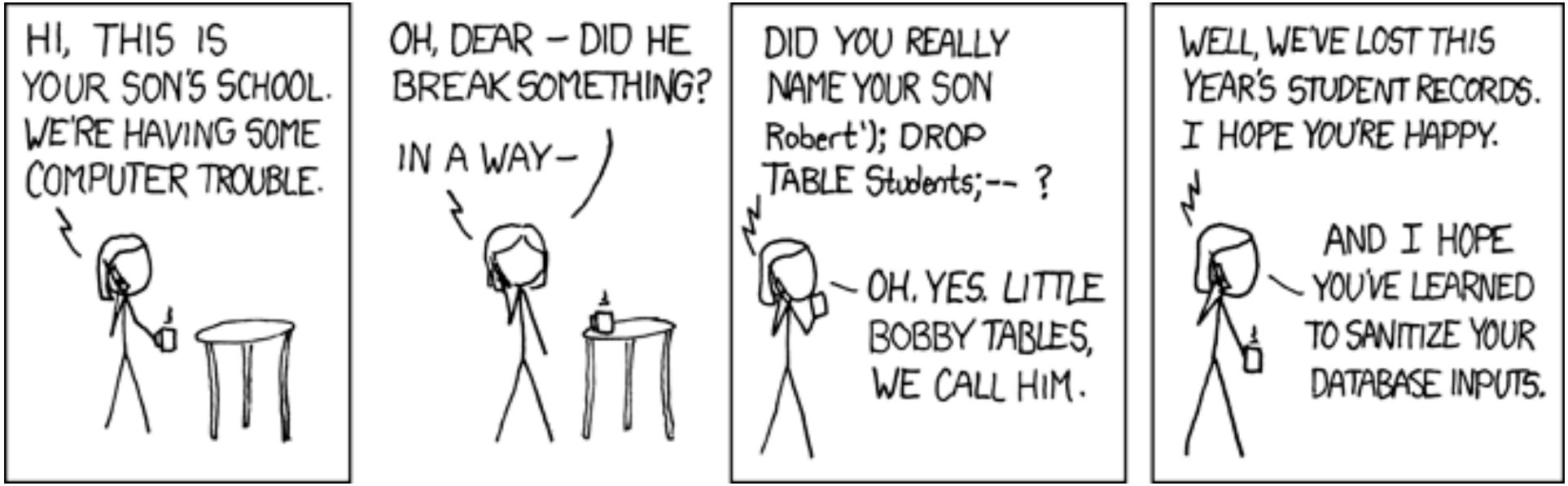
1. User sends malicious data

```
user="admin"; pwd="' OR 'x'='x"
```

```
boolean Login(String user, String pwd)  {
    boolean loggedIn = false;
    conn = pool.getConnection( );
    PreparedStatement pstmt = conn.prepareStatement(
        "SELECT * FROM members WHERE u = ? AND p = ?");
    pstmt.setString( 1, user);
    pstmt.setString( 2, pwd);
    ResultSet rs = pstmt.executeQuery( );
    if (rs.next())
        loggedIn = true;
}
```

4. System does not grant access

```
Login() returns false
```



http://xkcd.com/327

Command Injections

Input from the User

Web Form

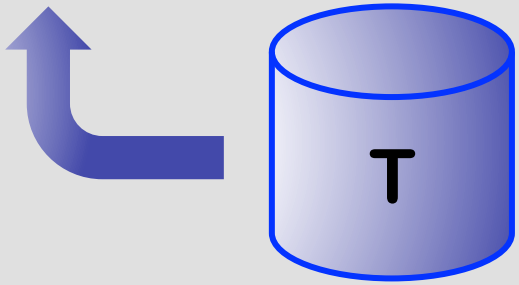
User: bart

Network Packet

xxxx bart xxxxx

Database Server

select * from T where u = $input



Command Shell

```
%
% mailx $input < message
%
```

Interpreter

prog = begin + input + end
eval prog

Command Injections

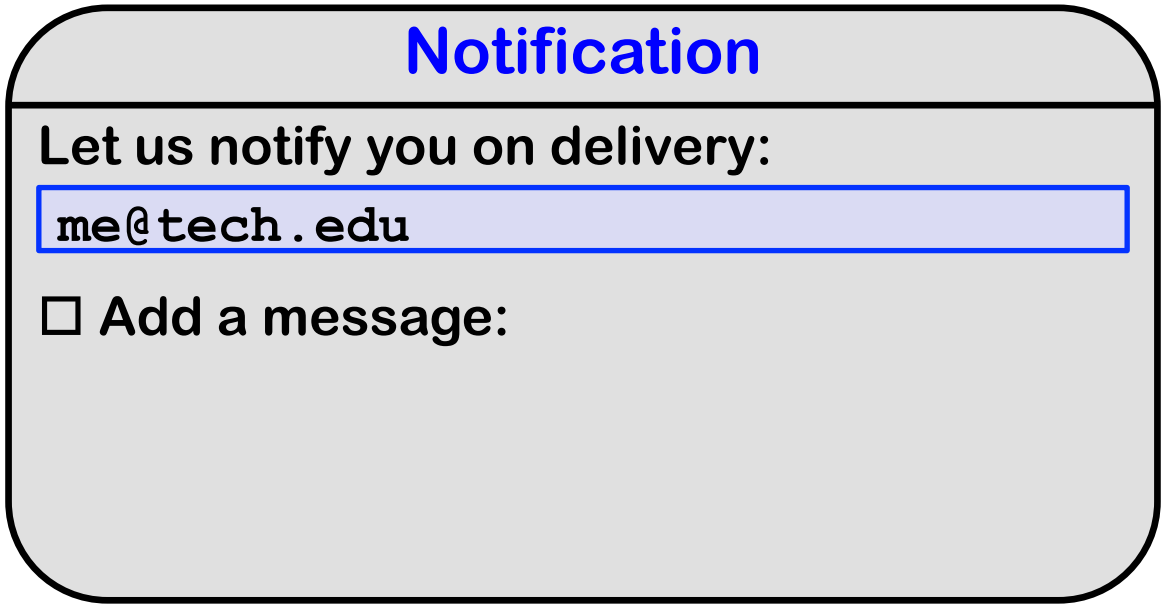
- We're looking for a path from the attack surface to the variables used in constructing a shell command.
- User supplied values must be validated, quoted, escaped or avoided.
- Does not attack shell itself. Modifies the command line of program started by shell.
- Need to fully understand command line interface.

An Example: A Server Sending Email

Servers often want to send email to users:

- Your package arrived. ☺
- Your flight is canceled. ☹
- You're over your credit limit. ☹

The email address comes from input that is provided by the user.



An Example: A Server Sending Email

A common (and risky) way for a program to send email is to generate a command-line."

If you enter `you@bad.com; evil-cmd`, the command line would be:

```
/bin/mailx -s "Your package" you@bad.com; evil-cmd
```

the server might execute any command you want on the server.

```
/bin/mailx -s "Your package" you@bad.com; evil-cmd
```

Notification

Let us notify you on delivery:

`you@bad.com; evil-cmd`

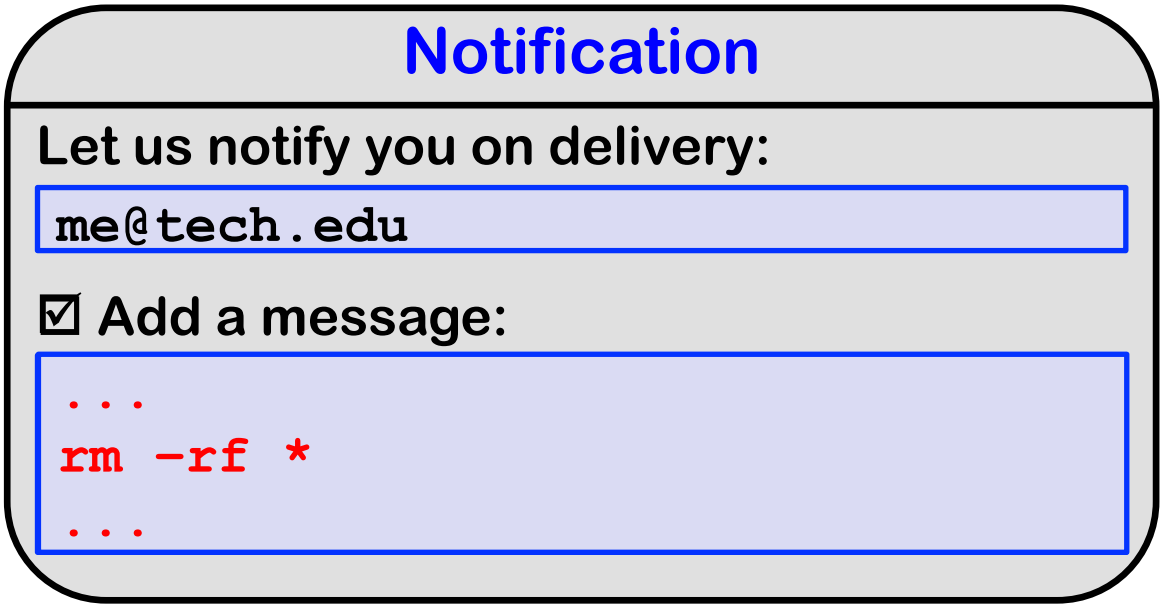
☐ Add a message:

A More Arcane Example

Now, suppose that you've prevented an injection attack from the email address by eliminating quotes and ";" from appearing in the email address....

... is there any more attack surface?

... could an attacker somehow use the message text to inject a command?



A More Arcane Example

To give away the ending: Yes!

Let's see how this could be done, using the Unix (Posix) standard mailx command-line mailer …

A More Arcane Example

mailx allows you to control some options from within the mail text.

For example:

```
~s Your package was delivered
~b you@bad.com
```

And, more interestingly:

```
~! ls -lt
```

You have to enable this feature with the mailx command-line option: `-~`

A More Arcane Example

Attack strategy is to enter email address:

```
-~ you@bad.com
```

And somewhere in the message text:

```
~! rm -rf *
```

Notification

Let us notify you on delivery:

```
-~ you@bad.com
```

☑ Add a message:

```
...
~! rm -rf *
...
```

Command Injection Mitigations

Avoid creating commands at runtime, if possible.

Use libraries or classes when available, e.g.:

- Java: Many choices, such as the standard JavaMail API. Includes simple methods for constructing and sending messages.
- Python: Also choices, such as the standard email package.
- Perl: Also choices, such as the popular MIME::Lite or Email::Stuffer packages.
- Web mail services: So so many of them, including mailgun, MailChimp, Drip, and SendGrid

Command Injection Mitigations

Input hygiene:

Check user input for metacharacters such as ";" and quotes.

Neutralize those metacharacters that can't be eliminated or rejected.

Isolate the program in a new process:

- On Linux, use `fork` to create process, drop privileges and then use `exec` for more control.

Command Injections

General signs of a vulnerability:

- Use of the `exec`, `popen` or `system` kernel calls.
- Program starting a shell such as `sh`, or `tcsh`, or `bash`.
- Not neutralizing command line arguments

  It is dangerous to let user input begin with "`-`" (Unix) or slash (Windows).

Perl Command Injections

You'll find commands in the most unexpected places:

- `open(F, $filename)`
  - Filename is a tiny language besides opening
    - Open files in various modes
    - Start programs
    - `dup` file descriptors
  - If `$filename` is `"rm -rf /|"`, you probably won't like the result

Perl Command Injections

Vulnerable to shell interpretation:

```
open(C, "$cmd|")          open(C, "-|", $cmd)
open(C, "|$cmd")          open(C, "|-", $cmd)
`$cmd`                    qx/$cmd/
system($cmd)
```

The string `$cmd` forms a complete shell command line, so is subject to injection.

Safer from shell interpretation:

```
open(C, "-|", @argList)
open(C, "|-", @argList)
system(@argList)
```

The program name and each argument are in a different location of array `@argList`. Can't change what program runs by modifying an argument.

Perl Command Injections


```
open(CMD, "|/bin/mailx -s $sub $to");
```

Bad if `$to` is `"badguy@evil.com; rm -rf /"`

```
open(CMD, “|/bin/mailx -s '$sub' '$to'");
```

Bad if `$to` is `"badguy@evil.com'; rm -rf /'"`

```
open(cmd, "|-", "/bin/mailx", "-s", $sub, $to);
```

Safe and simpler: use this whenever possible.

Ruby Command Injections

Functions prone to injection attacks:

- `Kernel.system(os command)`
- `Kernel.exec(os command)`
- `` `os command` `` (back tick operator)
- `%x[os command]`

Python Command Injections

Functions prone to injection attacks:

- `os.system()` **# execute a command in a subshell**
- `os.popen()` **# open a pipe to/from a command**

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

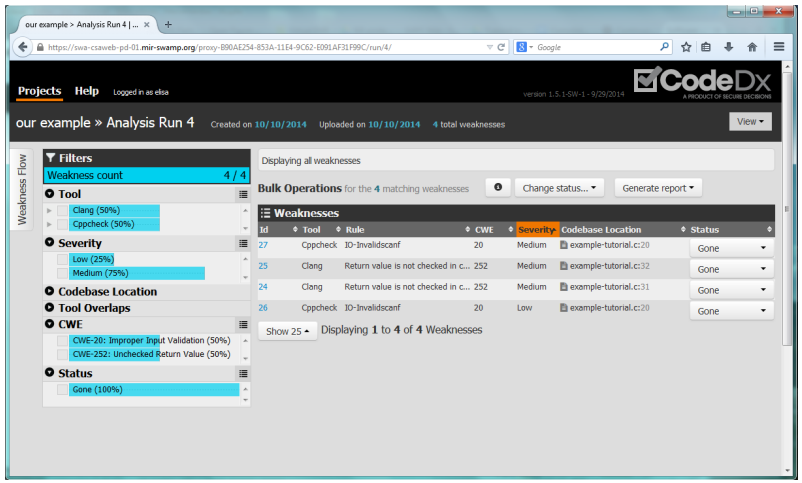
August 2018

1. What You Need to Know about How Tools Work

2. The SWAMP

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);
```



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

197

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".

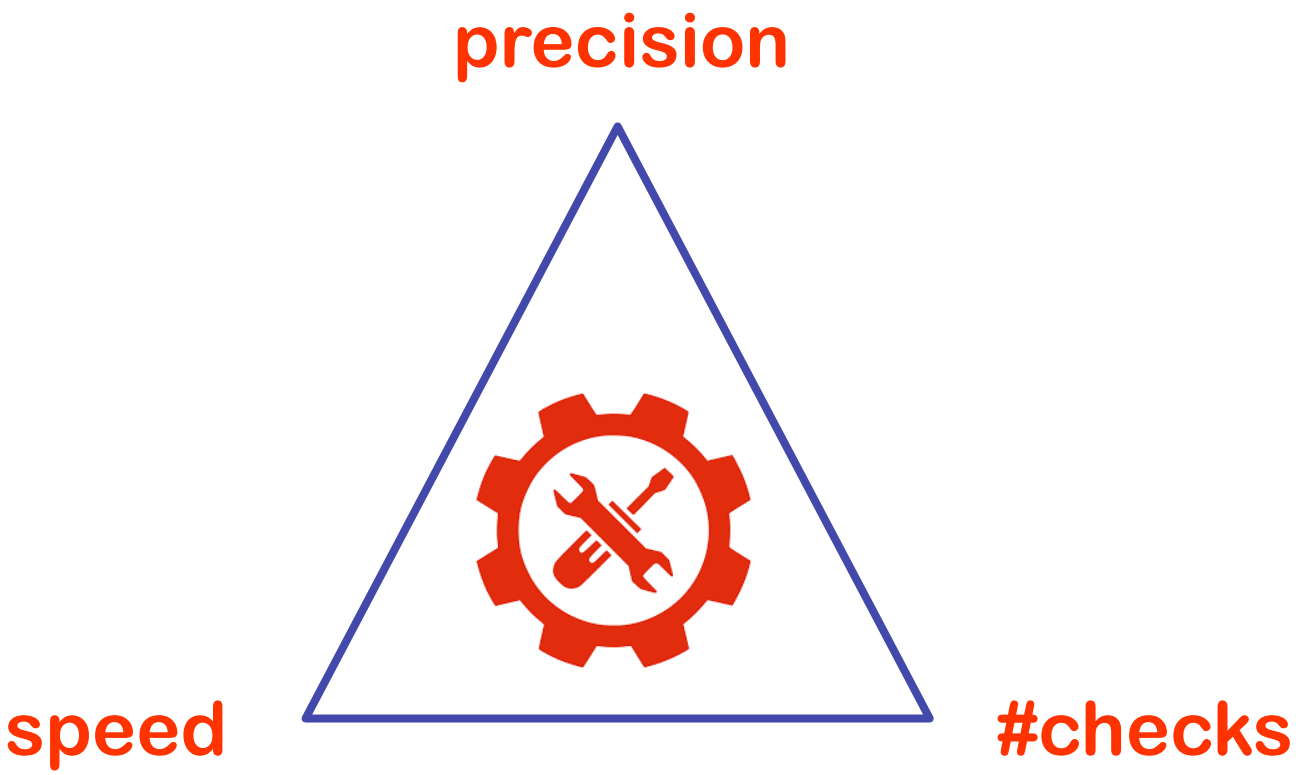
Source Code Analysis Tools

- 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

Courtesy of RedLizards

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();
execl (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)

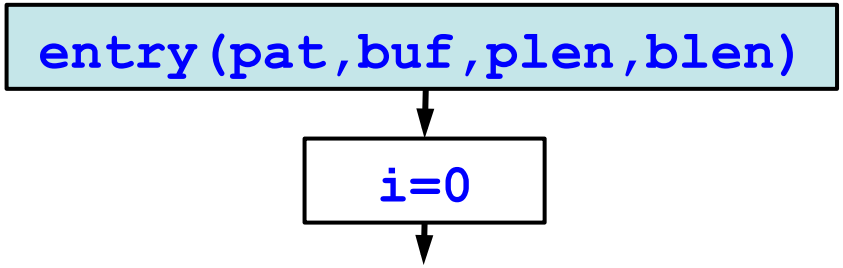
- Abstract representation of the source code.
- Each node represents a basic block.
- Call or jump targets start a basic block.
- Jumps end a basic block.
- Directed edges represent the control flow.

```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
        if (*p != pat[j]) break;
        p++;
        j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```

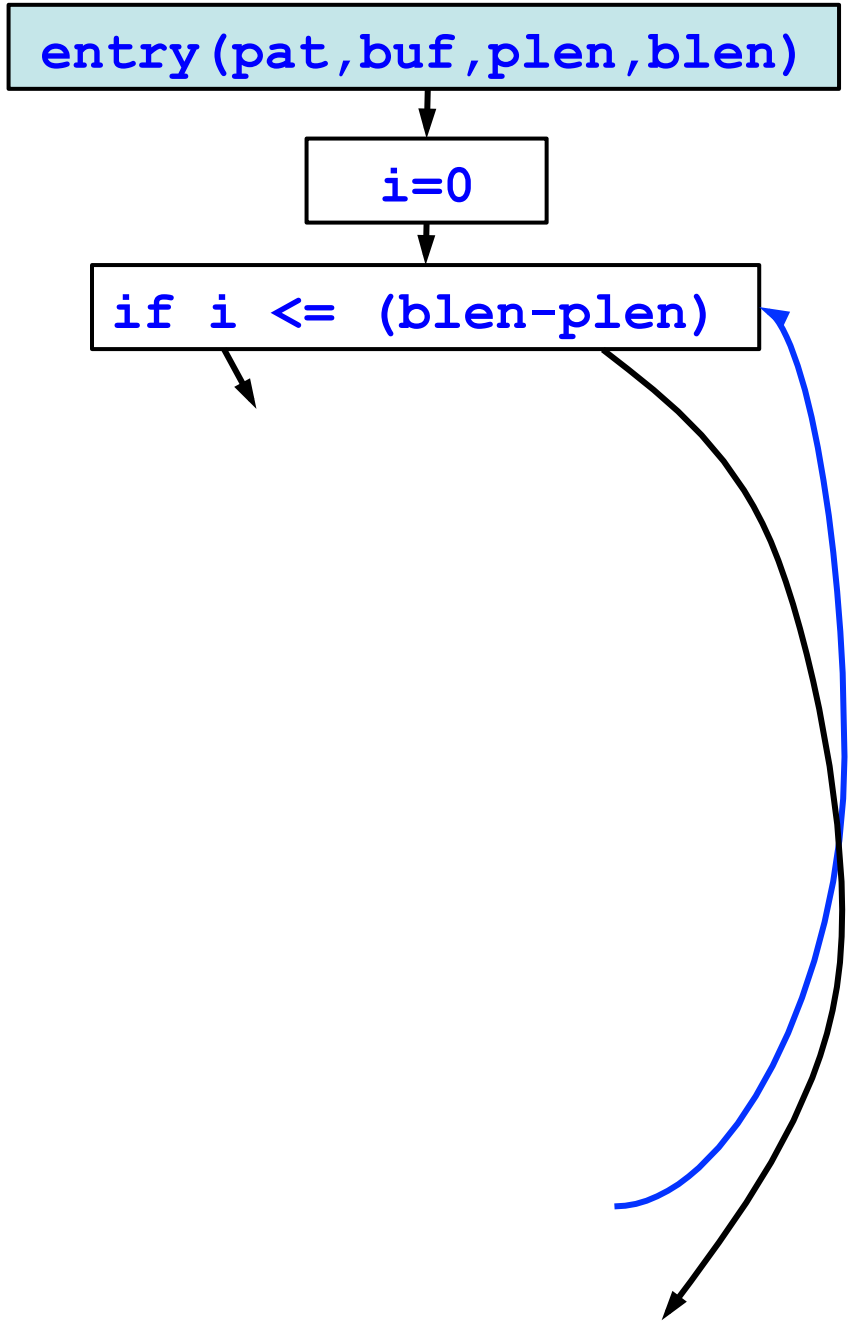
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

    int i, j;
    char *p;

    i = 0;

    while (i <= (blen - plen)) {
        p = &buf[i];
        j = 0;
        while (j < plen) {
          if (*p != pat[j]) break;
          p++;
          j++;
        }
        if (j >= plen) return i;
        i++;
    }

    return -1;
}
```

entry(pat,buf,plen,blen)

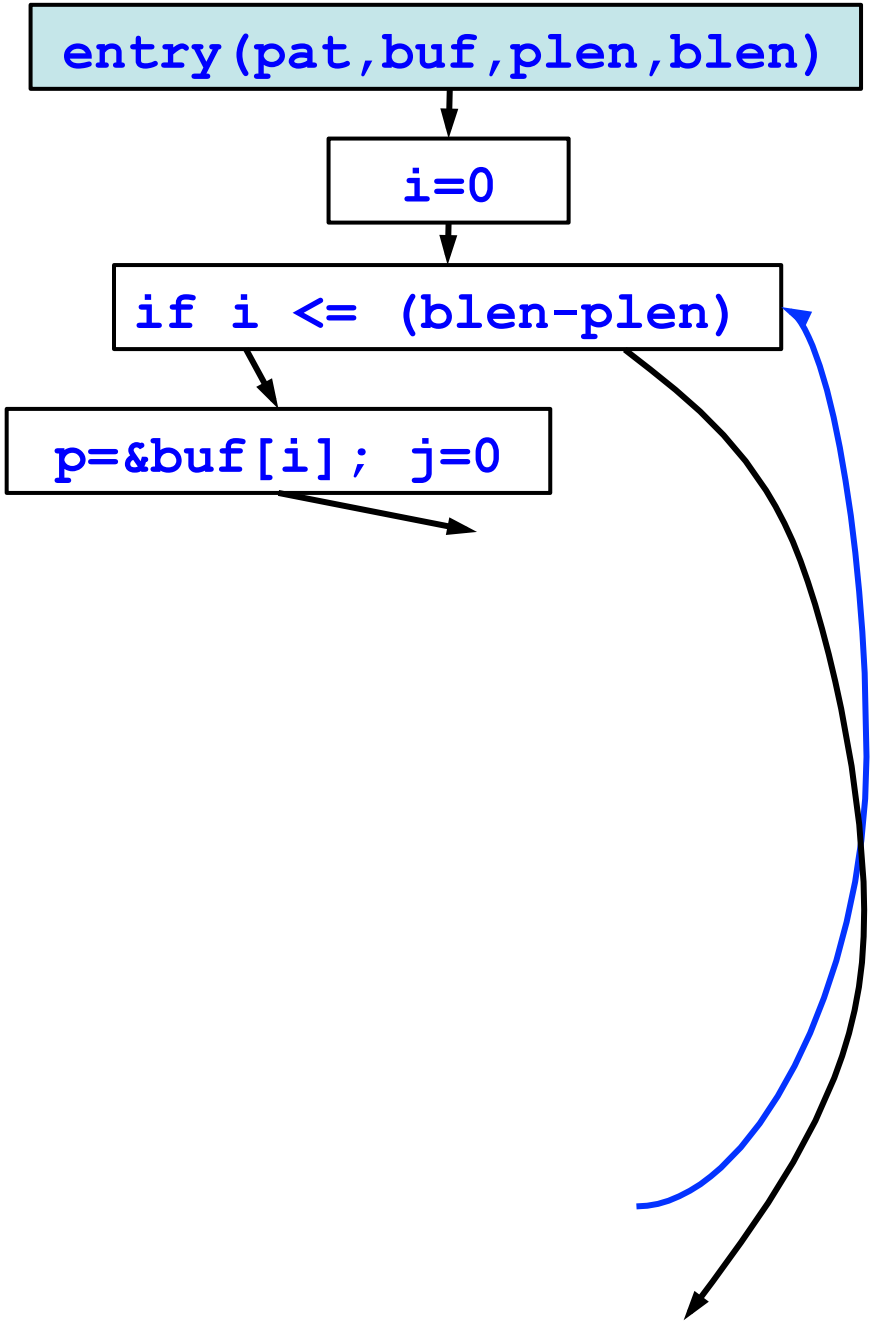
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
        if (*p != pat[j]) break;
        p++;
        j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



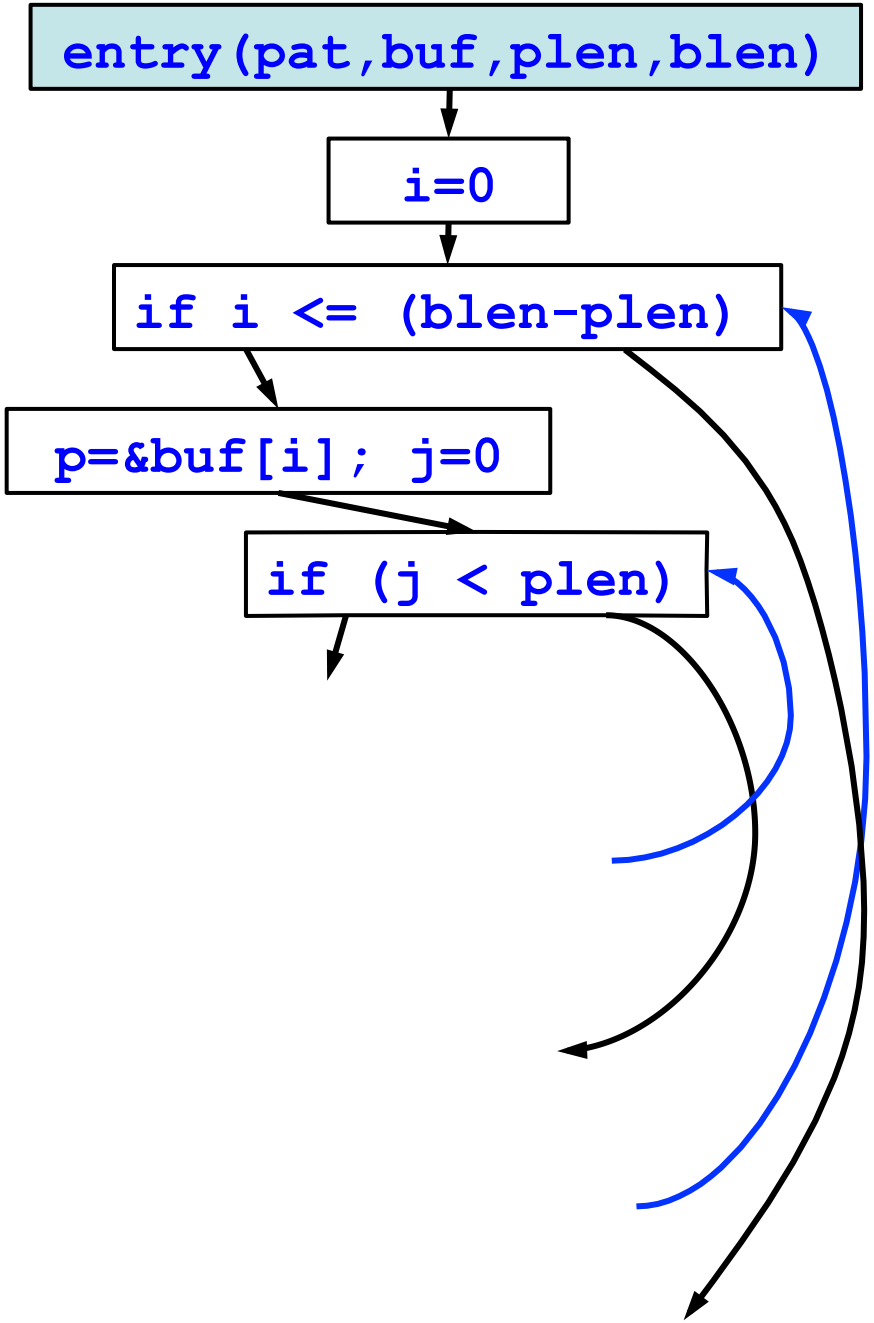
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
        if (*p != pat[j]) break;
        p++;
        j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



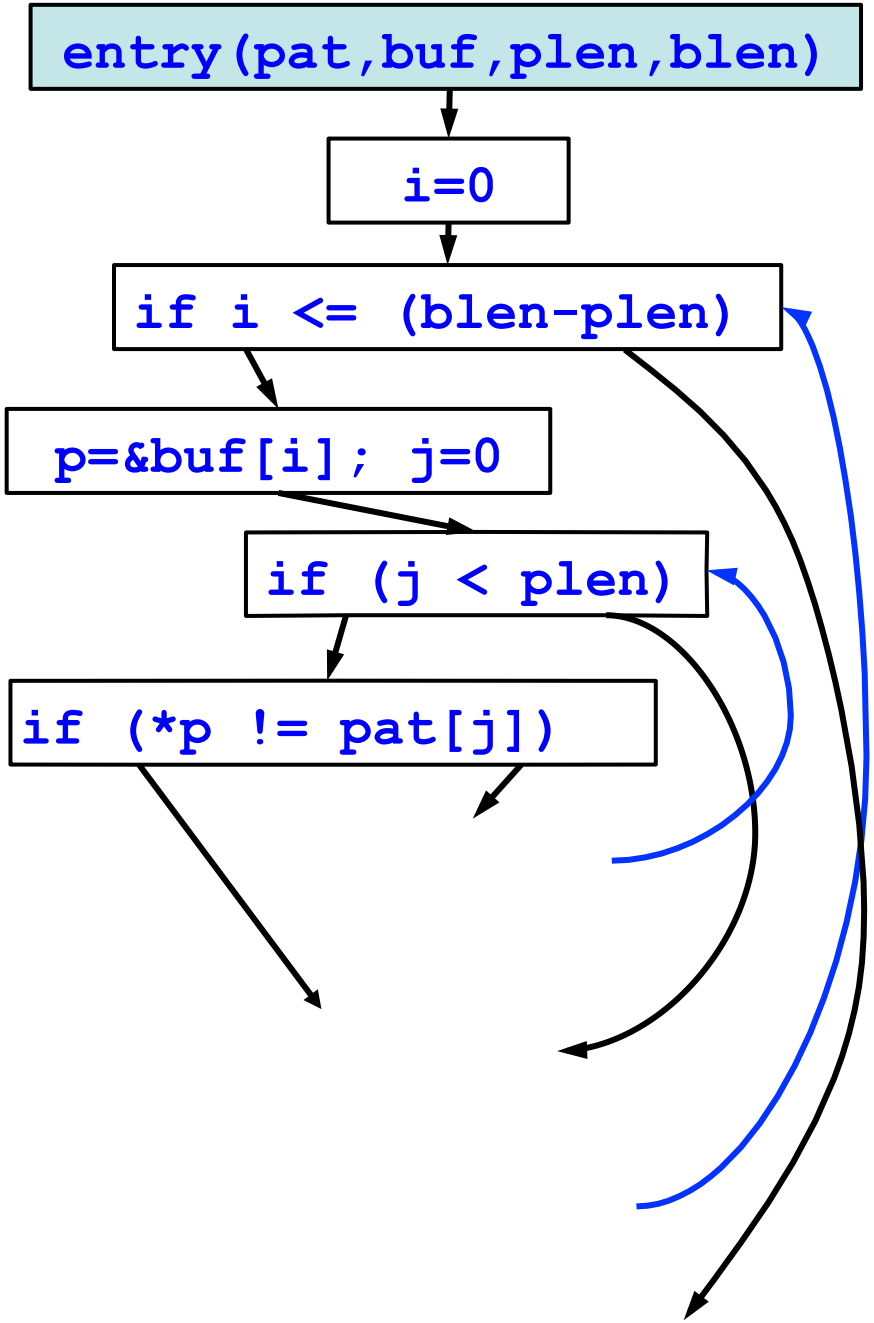
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
         if (*p != pat[j]) break;
         p++;
         j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



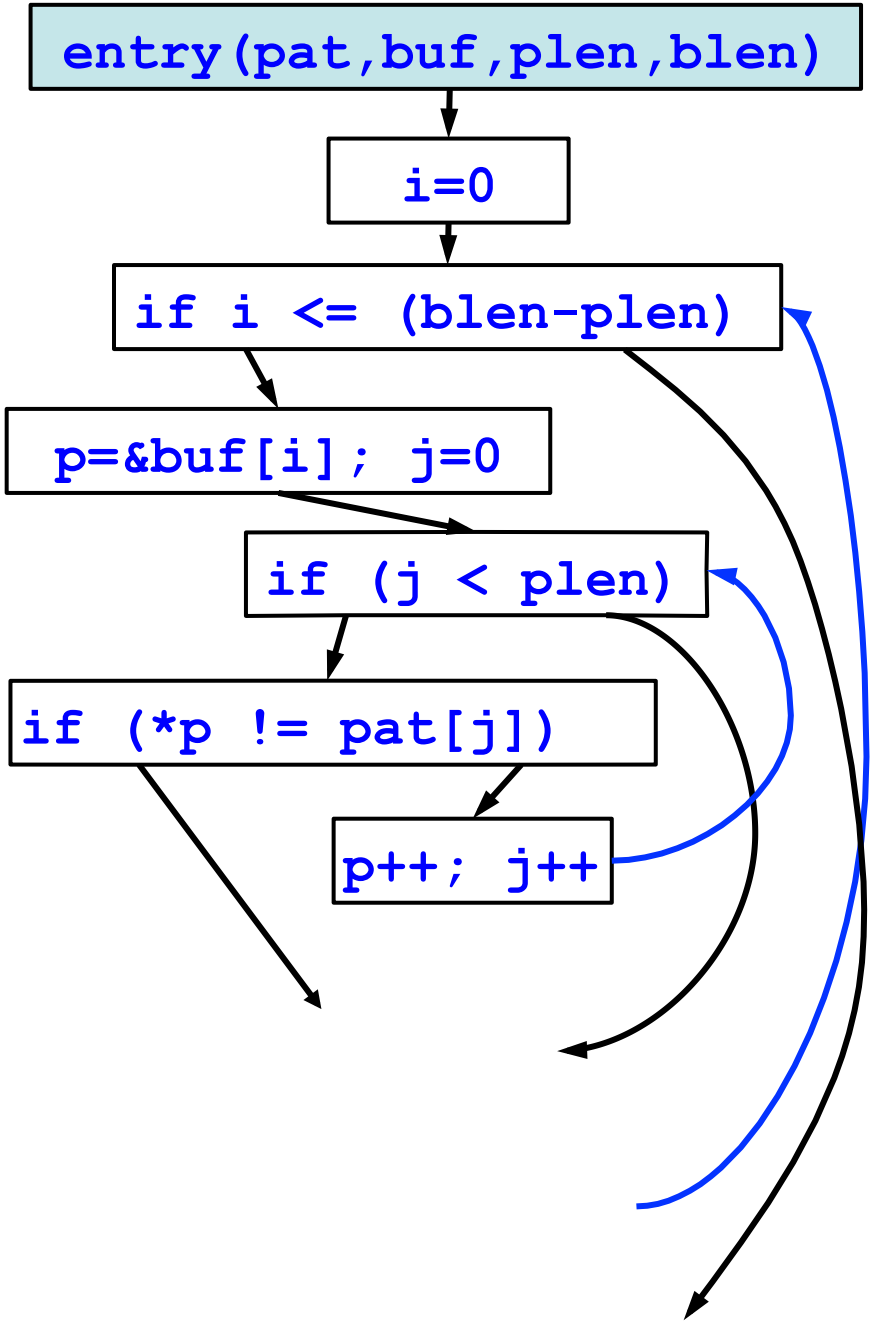
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
         if (*p != pat[j]) break;
         p++;
         j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



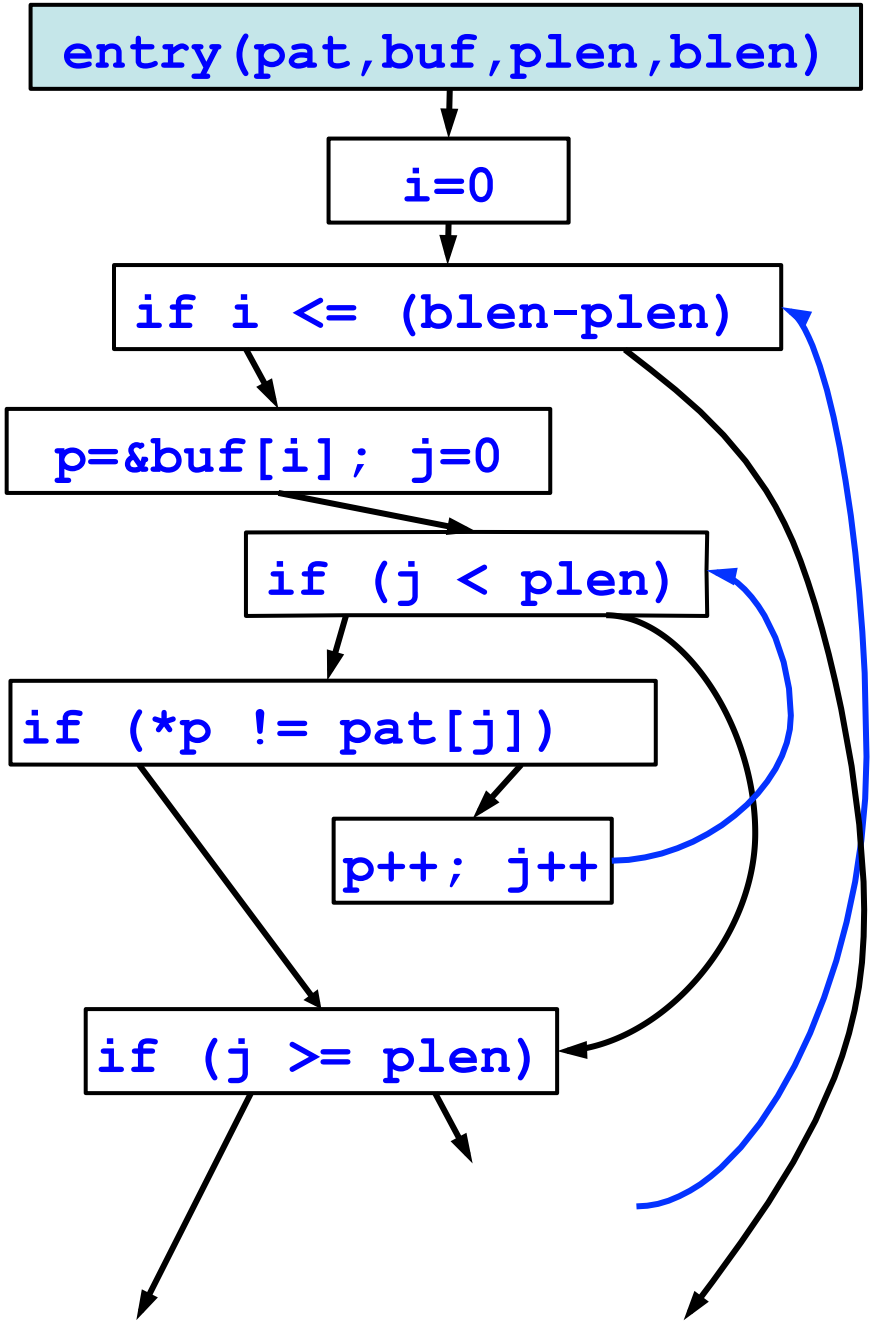
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
         if (*p != pat[j]) break;
         p++;
         j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



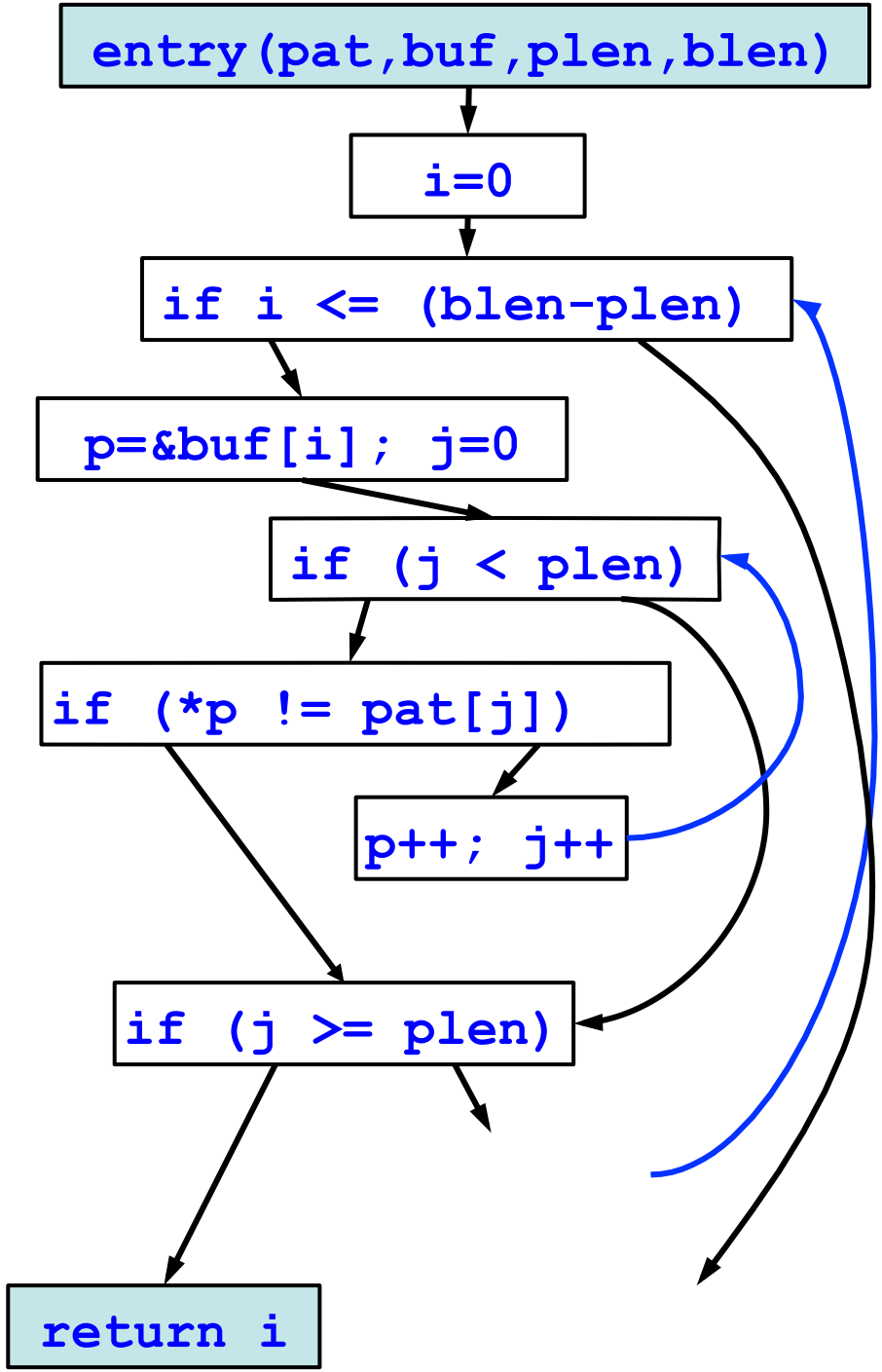
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
        if (*p != pat[j]) break;
        p++;
        j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



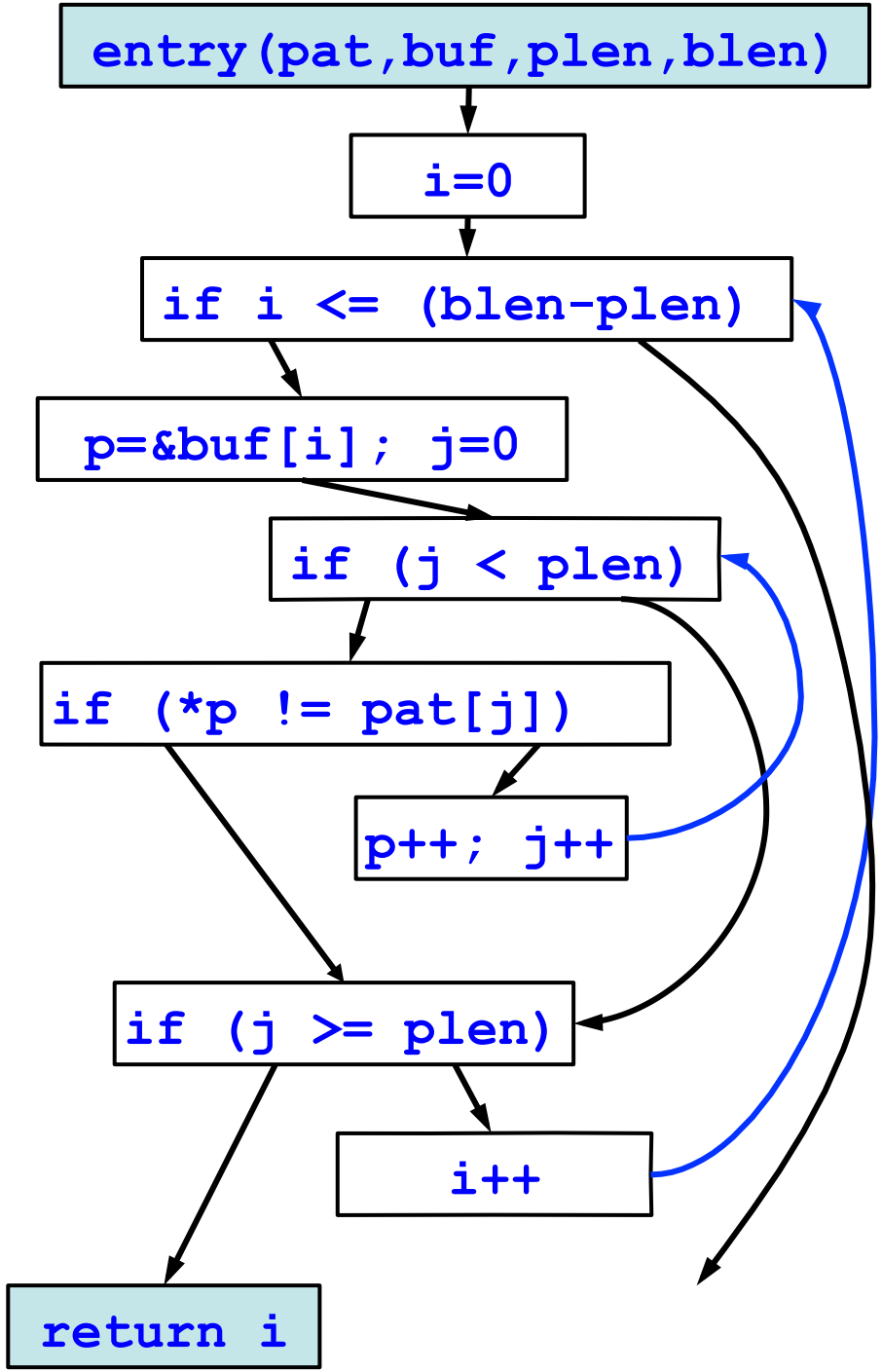
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
         if (*p != pat[j]) break;
         p++;
         j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



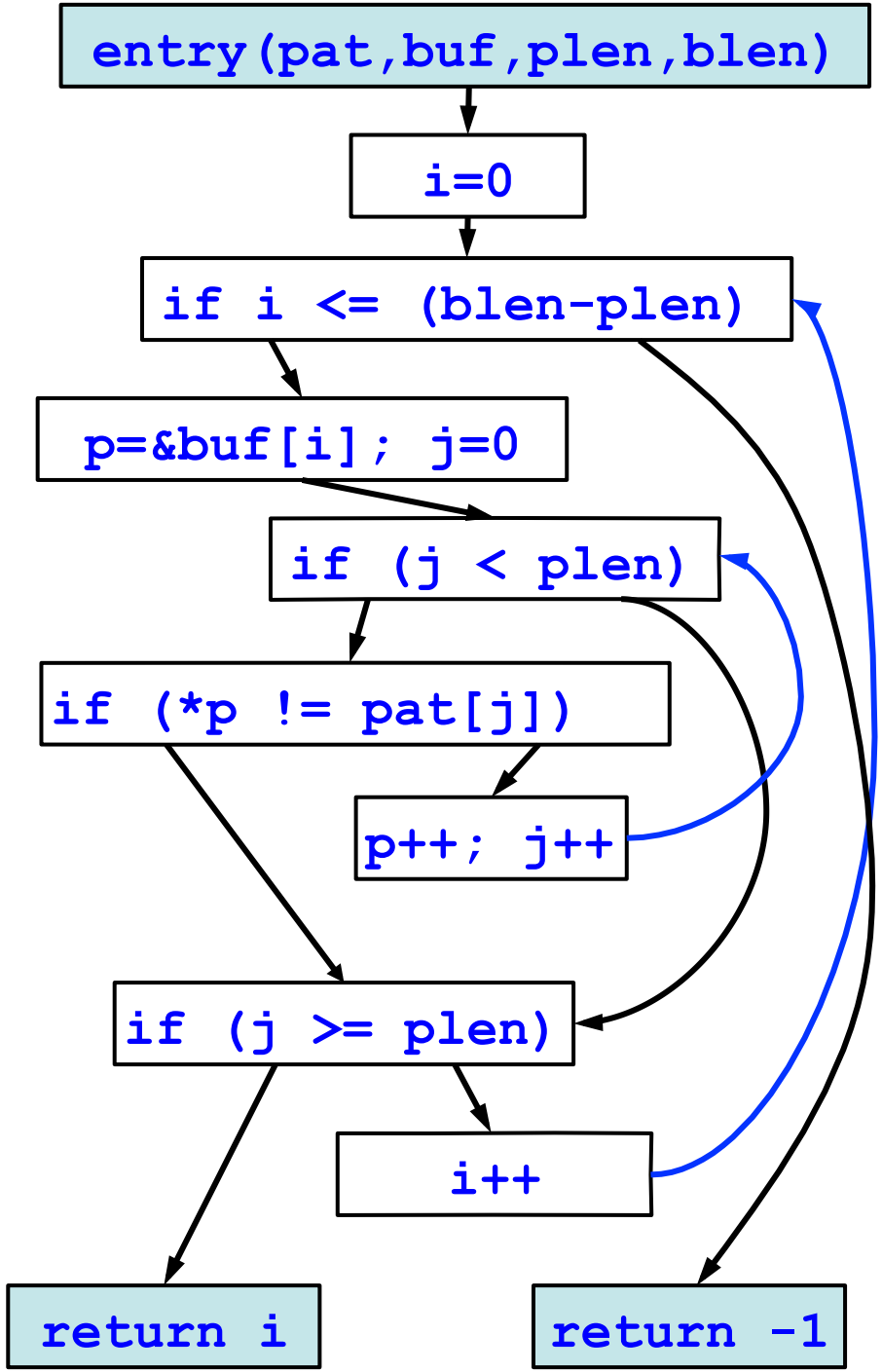
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
         if (*p != pat[j]) break;
         p++;
         j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



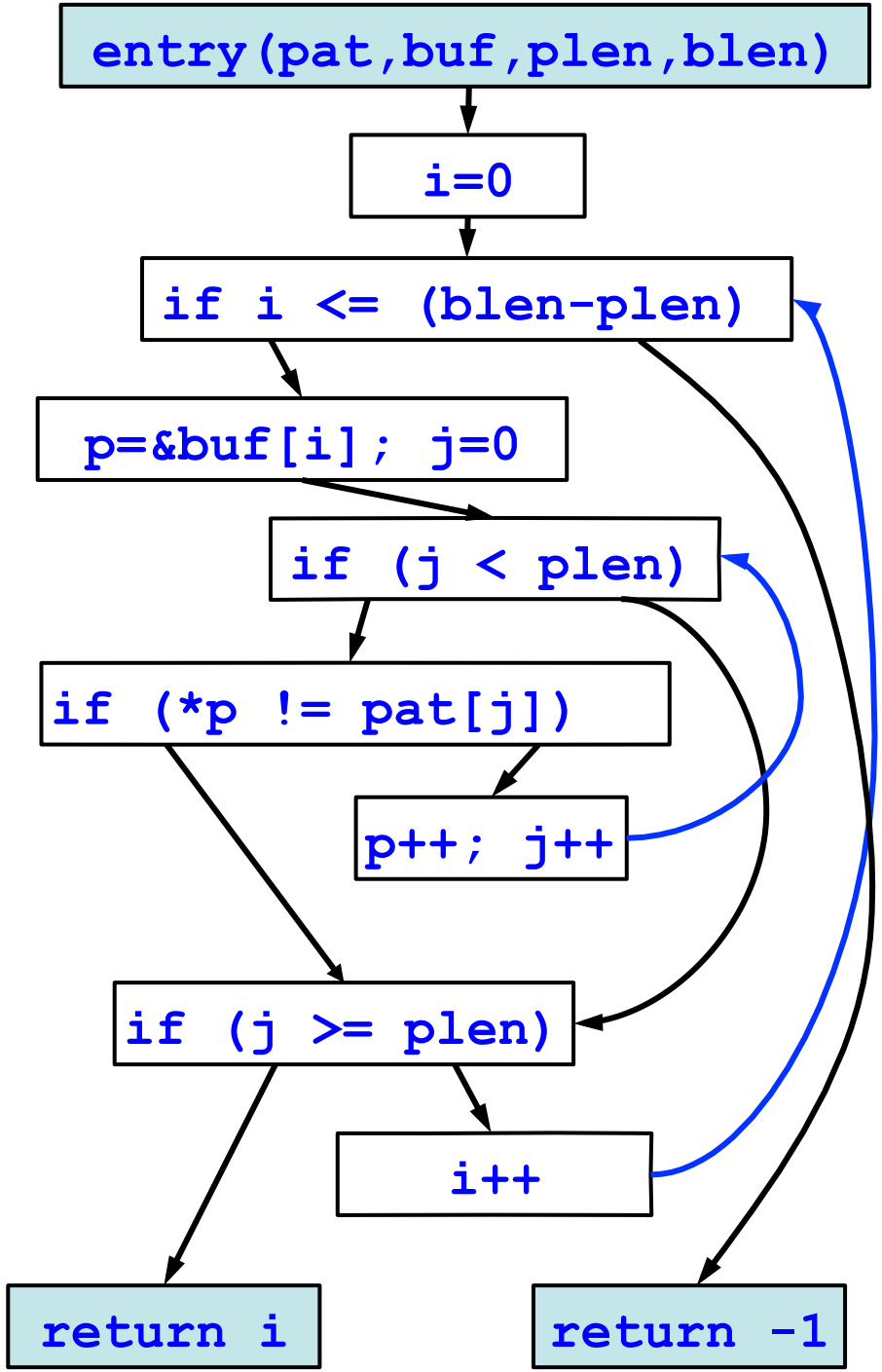
```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
        if (*p != pat[j]) break;
        p++;
        j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
         if (*p != pat[j]) break;
         p++;
         j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];
      j = 0;
      while (j < plen) {
        if (*p != pat[j]) break;
        p++;
        j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```



Data Flow Analysis

Goal: Is this code safe?

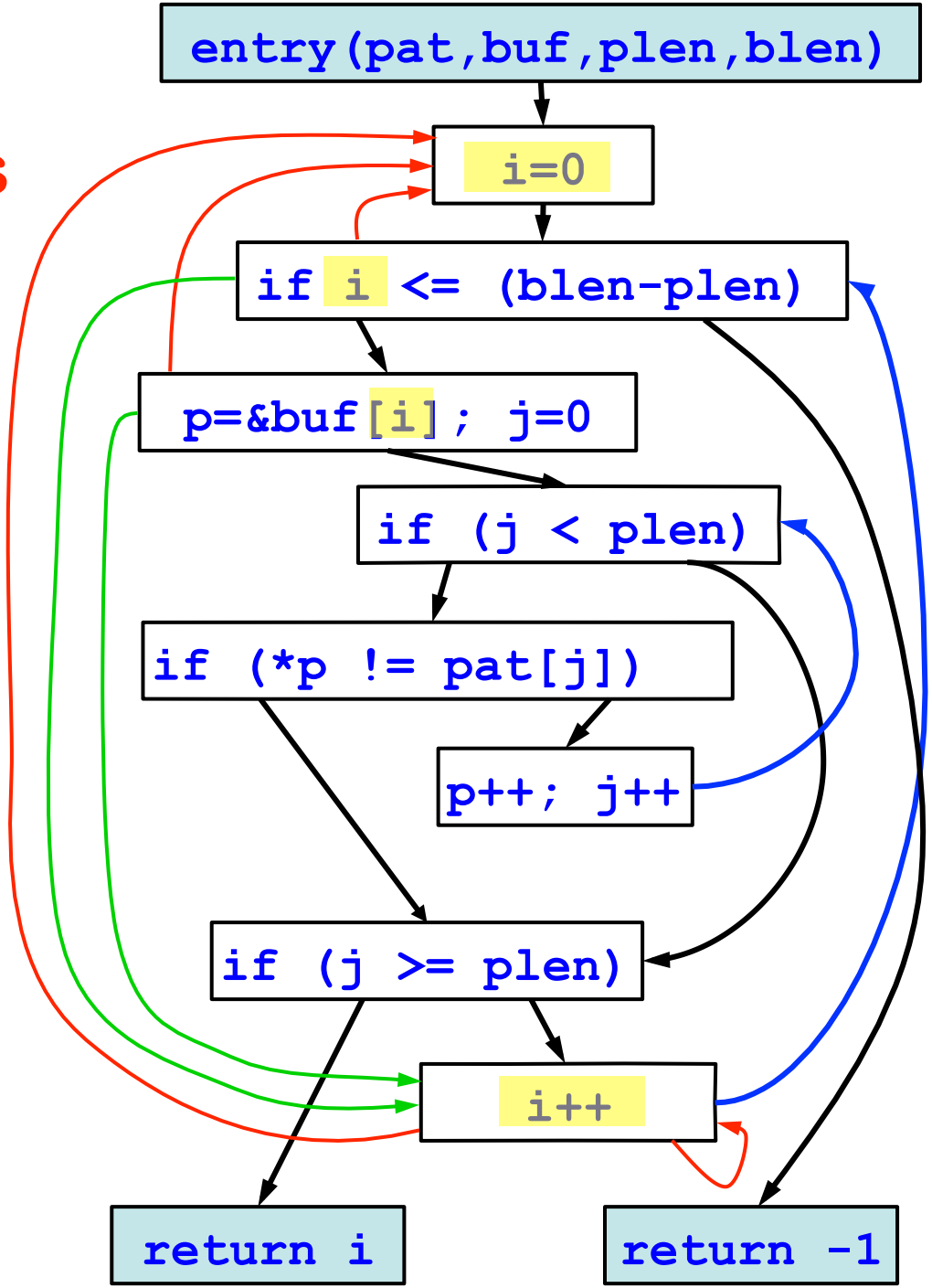
Subgoal:

Do we violate the borders of `buf` and `pat`?

- Simple dependences
- Flow insensitivity
- Loop carried dependences
- Pointers
- Aliasing

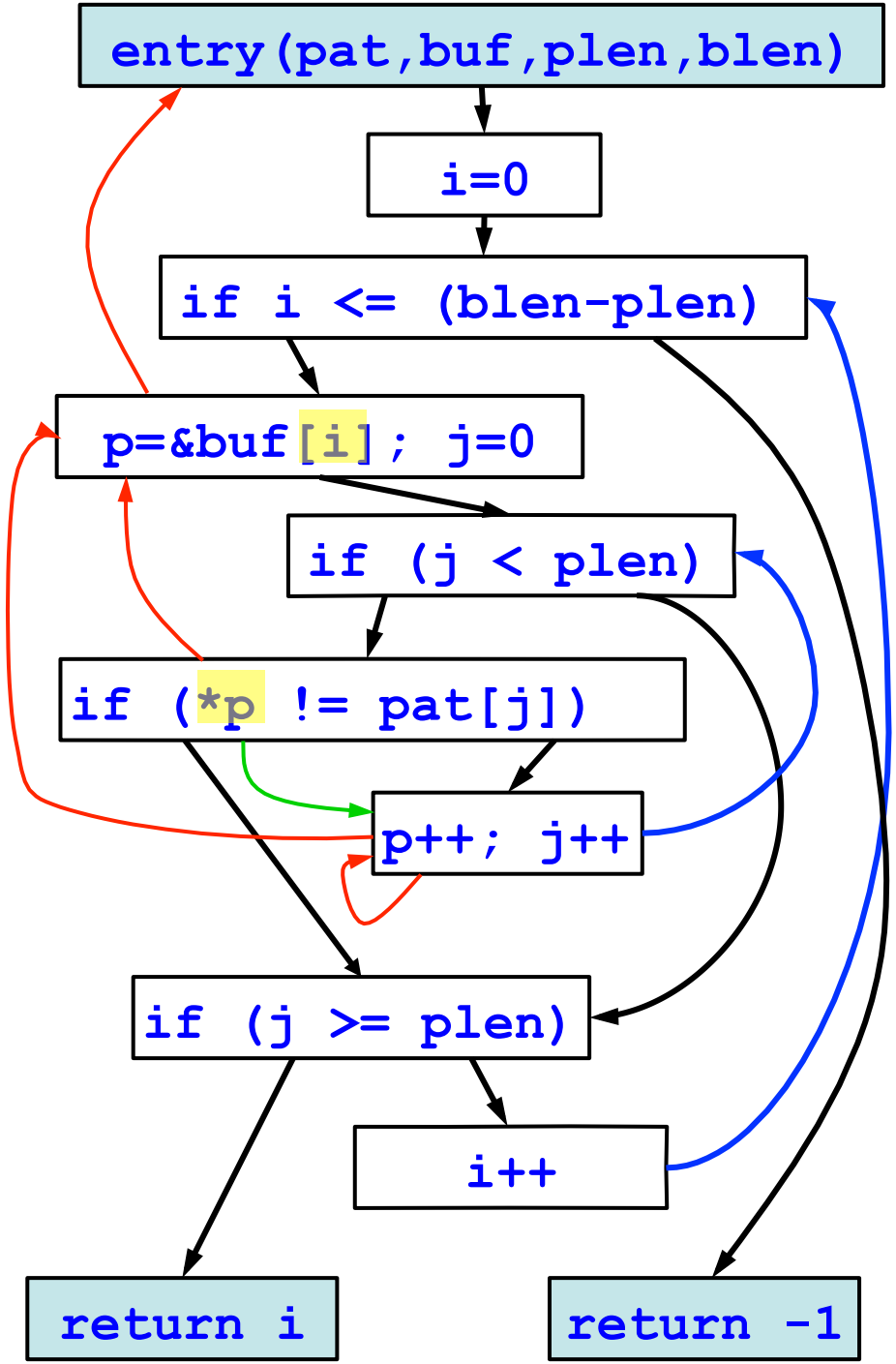
Data Flow Analysis

- Simple dependences
  - Back edges
  - Same node edges
- Loop carried dependences
- Need to understand the values for `i` to know that references to `buf[i]` are safe.
- Same for `j` and `pat[j]`.



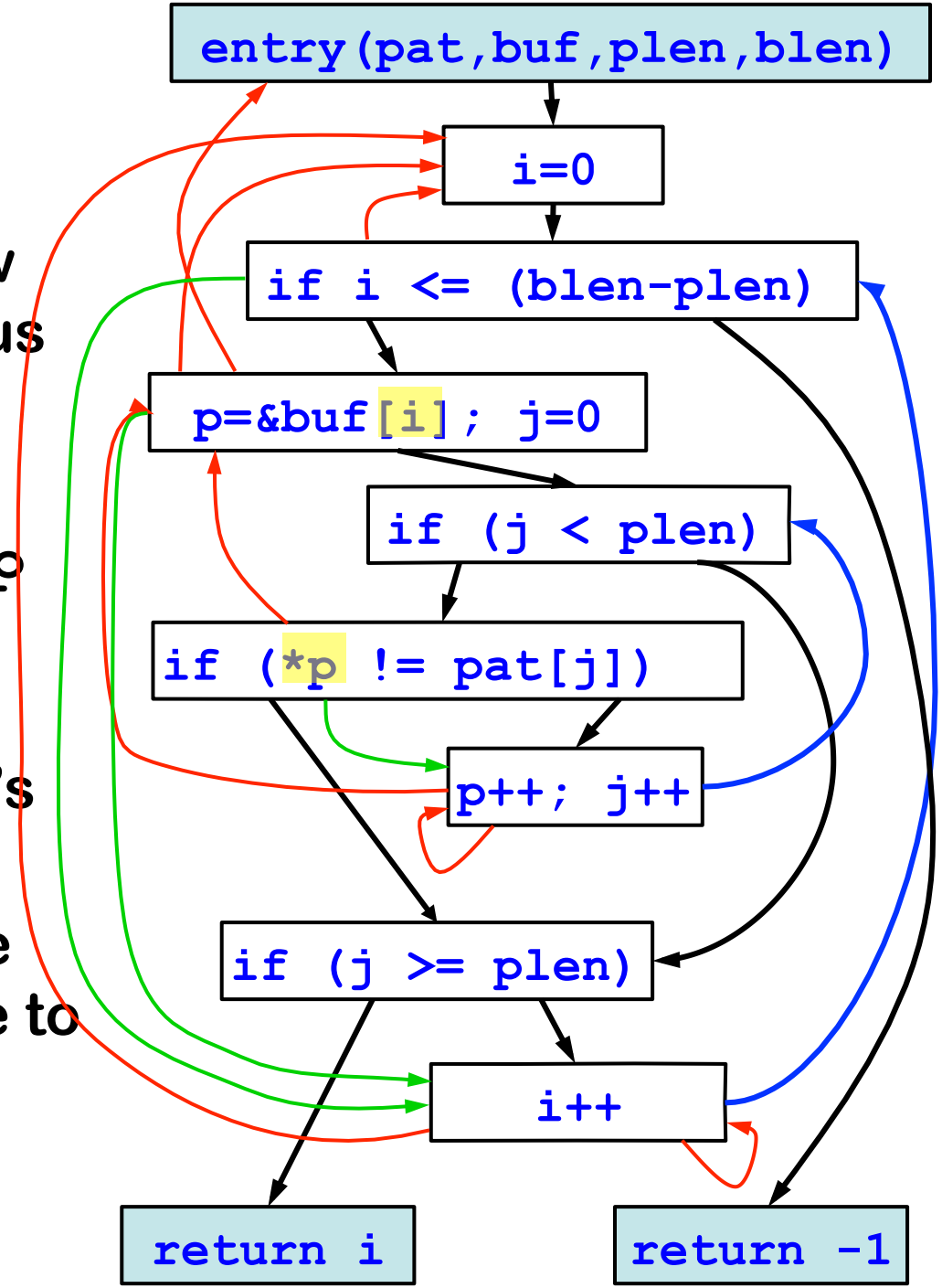
Data Flow Analysis

- Pointers
  - Similar to the data flow analysis on the previous slide.
  - Goal is to answer the question: where does `p` point? Are the references safe?
  - On what variables is `p`'s value based?
  - Of course, to calculate `p`'s value, we also have to know `i`'s value.



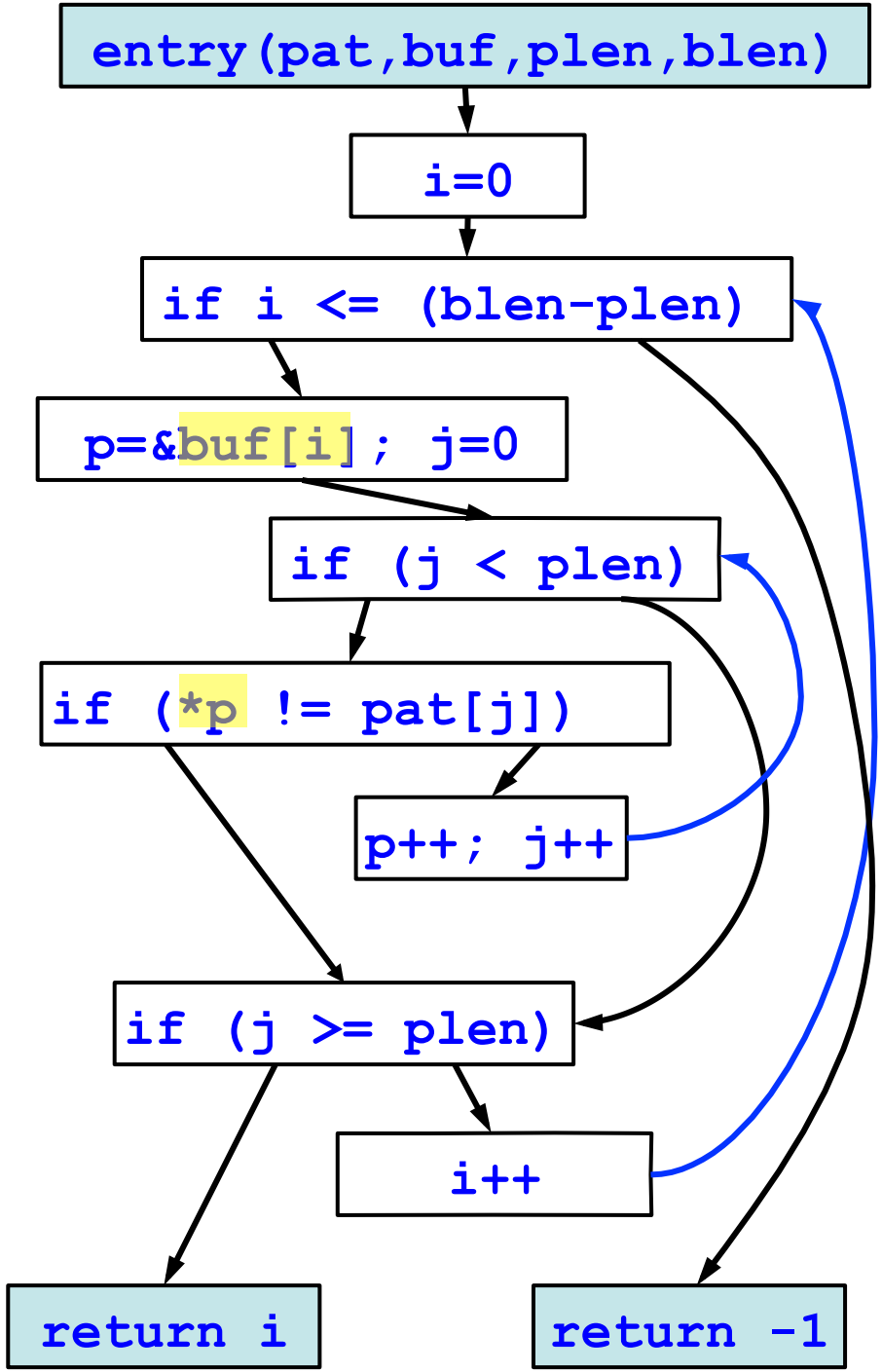
Data Flow Analysis

- Pointers
  - Similar to the data flow analysis on the previous slide.
  - Goal is to answer the question: where does `p` point? Are the references safe?
  - On what variables is `p`'s value based?
  - Of course, to calculate `p`'s value, we also have to know `i`'s value.



Data Flow Analysis

- Aliases
  - Note that there are two completely different ways to name the same memory locations.
  - Understand these aliases can be important to understanding how memory is being referenced.



```
int Find(char *pat, char *buf,
         unsigned int plen,
         unsigned int blen) {

   int i, j;
   char *p;

   i = 0;

   while (i <= (blen - plen)) {
      p = &buf[i];

      j = 0;
      while (j < plen) {
        if (*p != pat[j]) break;

        p++;
        j++;
      }
      if (j >= plen) return i;
      i++;
   }

   return -1;
}
```

The goal is to understand the range of values for each variable:

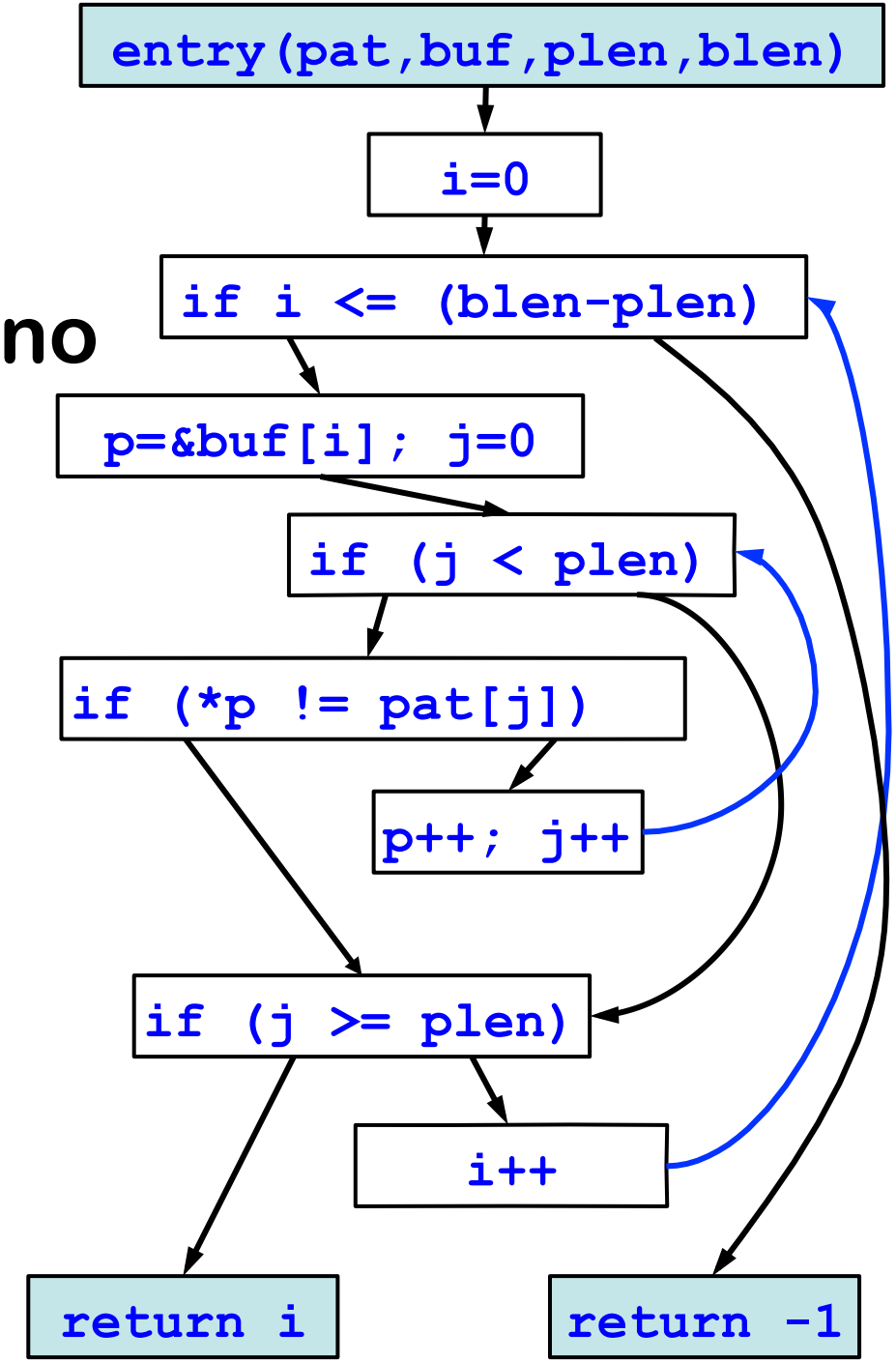
```
i:[0,0]

i:[0,blen-plen+1]
i:[0,blen-plen]
p:buf[0,blen-plen]
j:[0,0]
j:[0,plen-1]
j:[0,plen-1]
p:[buf[0,blen-plen+plen-1]
p:[buf[1,blen-plen+plen]
j:[0,plen]

j:[0,blen-plen]
i:[0,blen-plen+1]
```

Semantic Analysis

And this was a pretty simple example. It had no

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

These make program analysis slower, less precise, 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.

1. What You Need to Know about How Tools Work

2. The Tools And Their Use

Roadmap

- Motivation
- Source code example
- Tools for Java applied to the source code

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 the strong and weak points of using specific tools.

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_Containg_The_Code_yy.cpp)

CWE 601: Open Redirect

```
public void doGet(HttpServletRequest request,
                                HttpServletResponse response)
                        throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter returnHTML = response.getWriter();
  returnHTML.println("<html><head><title>");
  returnHTML.println("Open Redirect");
  returnHTML.println("</title></head><body>");

  String data;
  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) {
        URI uri;
        uri = new URI(data);
    // POTENTIAL FLAW: redirect is sent verbatim.
        response.sendRedirect(data);
        return;
   }
}
```

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

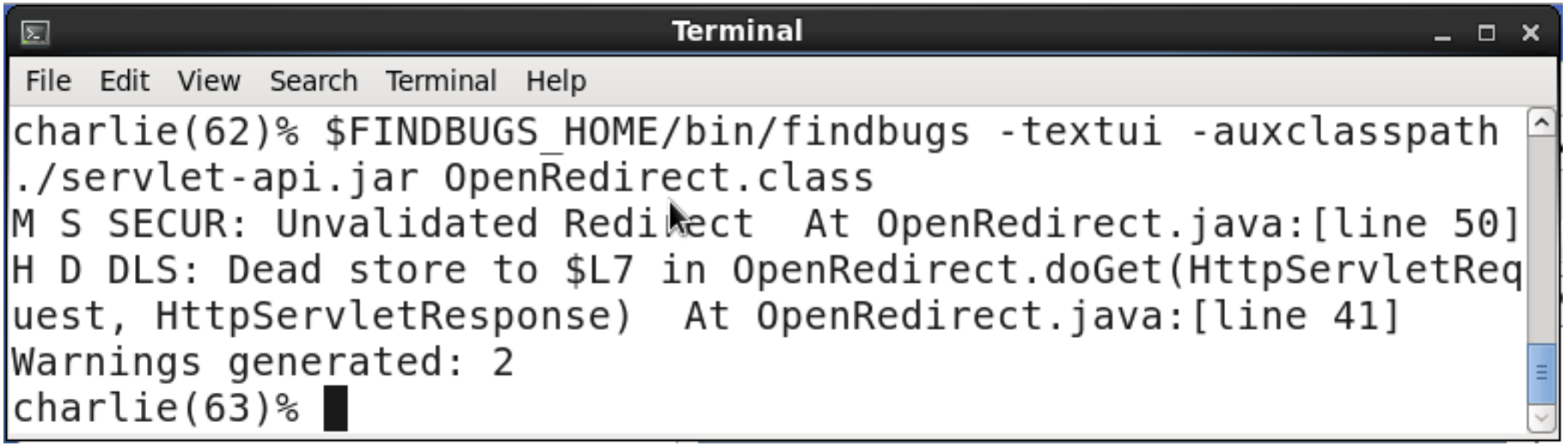
1. Define FINDBUGS_HOME in the environment.
2. Install the Find Security Bugs plugin.
3. Learn the command line instructions and also use the graphical interface.
4. Command line interface:

```
$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.**



FindBugs. Open Redirect

FindBugs

File Edit View Navigation Designation Help

Class name filter: Filter

Group bugs by: Category | Bug Kind | Bug Pattern

- Bugs (2)
  - Security (1)
    - Unvalidated Redirect (1)
      - Unvalidated Redirect (1)
        - Unvalidated Redirect
  - Dodgy code (1)
    - Dead local store (1)
      - Dead store to local variable (1)
        - Dead store to $L7 in OpenRedirect.doGet(HttpSer

No cloud selected | Enable cloud plugin...

OpenRedirect.java in — View in browser

```
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 exceptURISyntax)
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

Unvalidated Redirect
At OpenRedirect.java:[line 50]
In method OpenRedirect.doGet(HttpServletRequest, HttpServletR

Unvalidated Redirect

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.

Scenario

1. A user is tricked into visiting the malicious URL:
http://website.com/login?redirect=http://evil.vvebsite.com/fake/login
2. The user is redirected to a fake login page that looks like a site they trust. (http://evil.vvebsite.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.

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

Java - OpenRedir/src/OpenRedirect.java - Parasoft Jtest

File Edit Source Refactor Navigate Search Project Parasoft Run Window Help

OpenRedirect.java

```
    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);
        }
```

Task List

Find All Activate...

Outline

import declarations

OpenRedirect

doGet(HttpServletRequest, Ht

Problems | Javadoc | Declaration | CWE-SANS Top 25 Most Dangerous Programming Errors ... | Console | Quality Tasks

0 errors, 2 warnings, 0 others

Description

Warnings (2 items)

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.VPPD: 'getCookies()' is a ta...nd should be encapsulated by a validation

Jtest. Open Redirect

Java - OpenRedir/src/OpenRedirect.java - Parasoft Jtest

File Edit Source Refactor Navigate Search Project Parasoft Run Window Help

OpenRedirect.java

```
    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);
        }
```

Task List

Find All Activate...

Outline

import declarations

OpenRedirect

doGet(HttpServletRequest, Ht

Problems | Javadoc | Declaration | CWE-SANS Top 25 Most Dangerous Programming Errors ... | Console | Quality Tasks

0 errors, 2 warnings, 0 others

Description

Warnings (2 items)

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.VPPD: 'getValue()' is a dang...nd should be encapsulated by a validation

Jtest. Open Redirect

Java - OpenRedir/src/OpenRedirect.java - Parasoft Jtest

File Edit Source Refactor Navigate Search Project Parasoft Run Window Help

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;
```

Task List

Find All Activate...

Outline

import declarations

OpenRedirect

doGet(HttpServletRequest, Ht

Problems | Javadoc | Declaration | OWASP Top 10 2013 Security Vulnerabilities (Server Config | Console | Quality Tasks

0 errors, 4 warnings, 0 others

Description

Warnings (4 items)

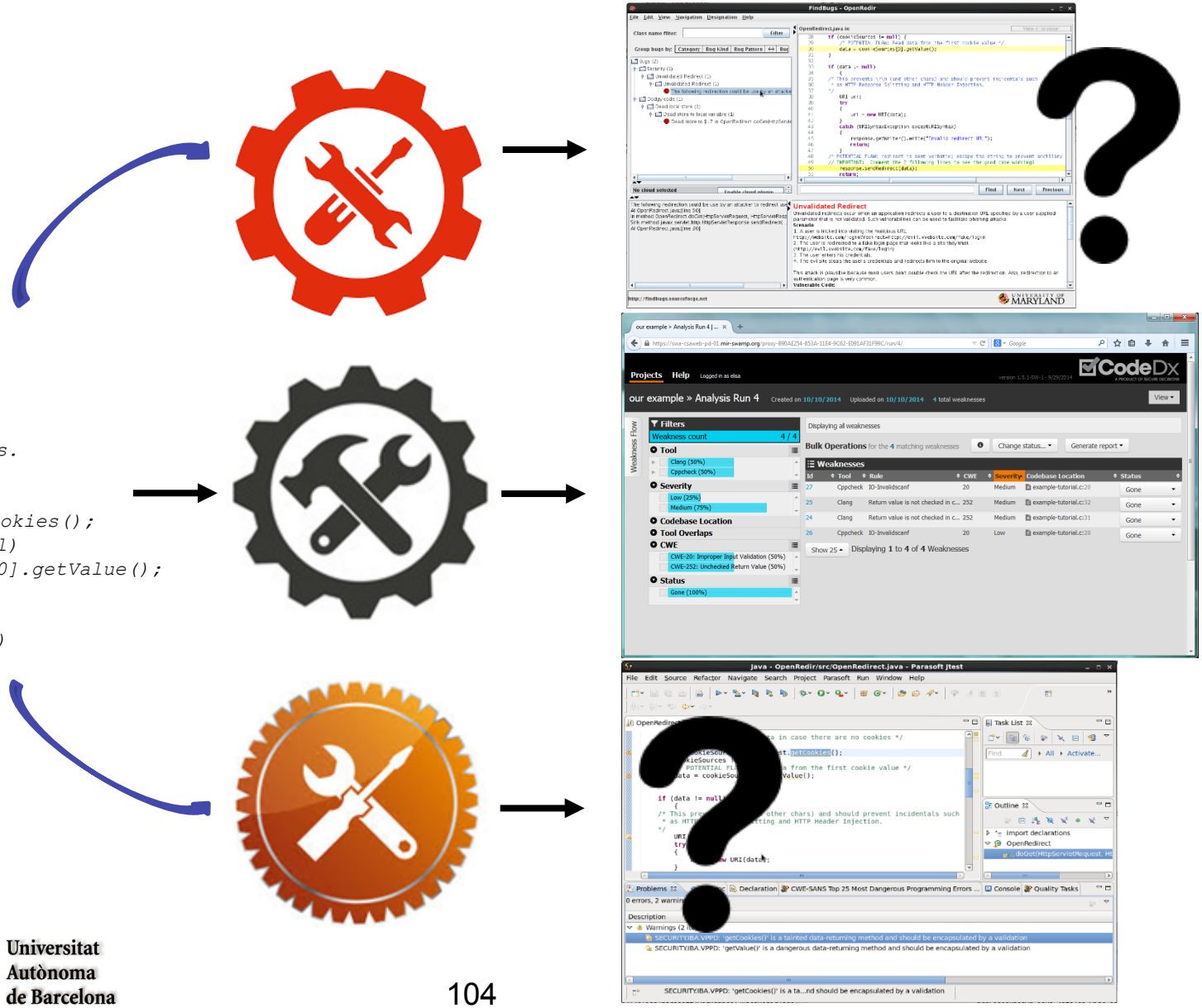
- SECURITY.IBA.VPPD: 'getCookies()' is a tainted data-returning method and should be encapsulated by a validation
- SECURITY.IBA.VPPD: 'getValue()' is a dangerous data-returning method and should be encapsulated by a validation
- SECURITY.IBA.VRD: No validation check in redirect URL
- SECURITY.IBA.VRD: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

SECURITY.IBA.VRD: No validation check in redirect URL

The SWAMP

Background.
Automated Assessment Tools

```
String data;
data = "";
// Read data from cookies.
Cookie cookieSources[] =

            request.getCookies();
if (cookieSources != null)
   data = cookieSources[0].getValue();
if (data != null) {
      URI uri;
      uri = new URI(data)
}
```



Background: Common Weakness Enumeration (CWE)

"CWE is a community-developed list of common software security weaknesses." cwe.mitre.org

Provides a unified and precise way to name software weaknesses.

Allows a more effective use of software security tools.

714 weaknesses in 237 categories.

Each CWE includes: ID, description, consequences, examples, potential mitigations.

https://cwe.mitre.org/

Background: Common Vulnerabilities and Exposures (CVE)

CVE is a standard way to name security vulnerabilities.

"Consists of a list of common identifiers for publicly known cyber security vulnerabilities".

Provides a baseline to be used for comparing and evaluating automated assessment tools.

Example: Heartbleed is CVE - CVE-2014-0160.

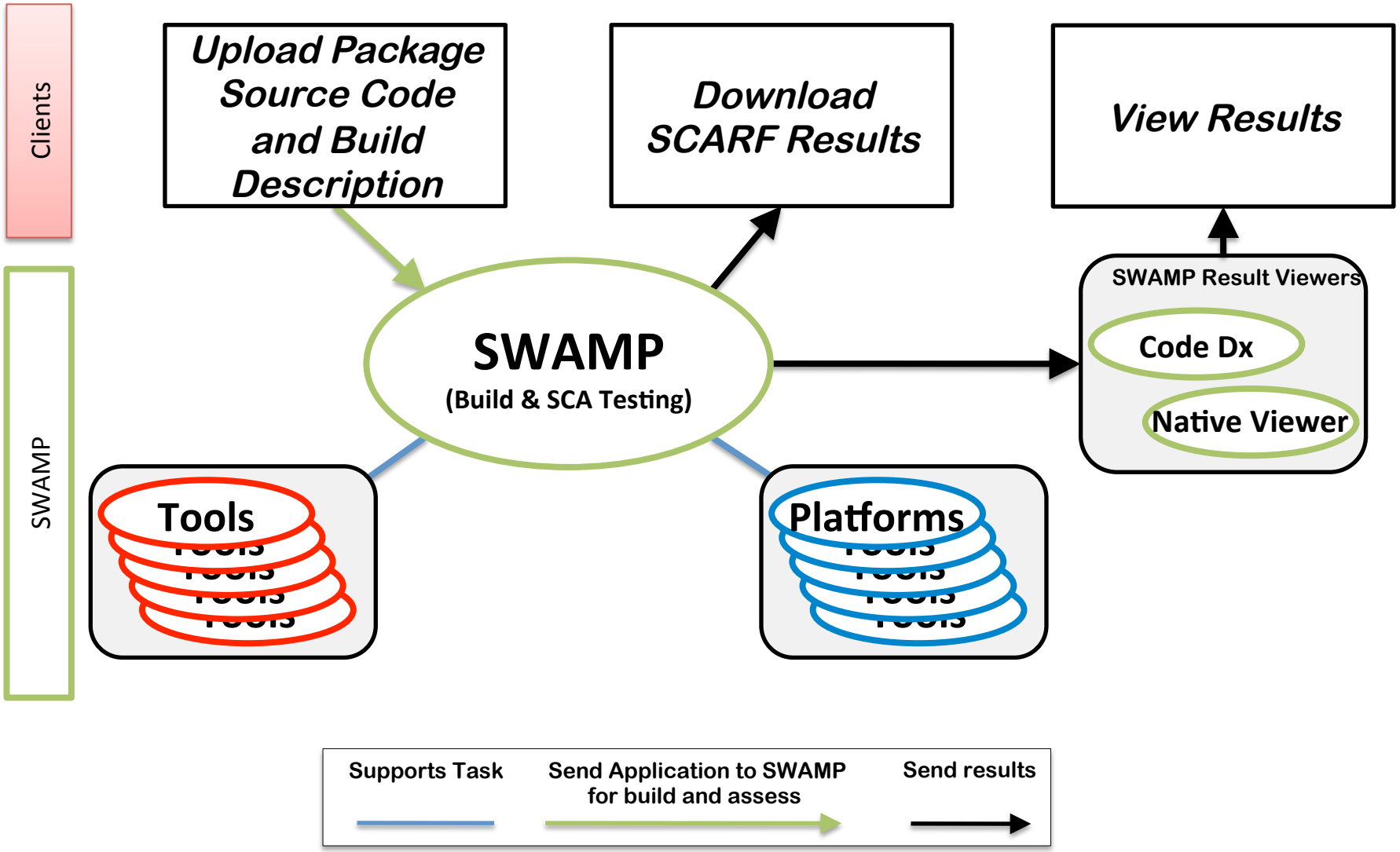
Over 90,000 CVEs.

https://cve.mitre.org/

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.

Core SWAMP Functionality



SWAMP Tools and Platforms

Tools

C/C++
- Cppcheck
- Clang Static Analyzer
- Gcc Warnings
- Parasoft C/C++Test
- GrammaTech CodeSonar
- Synopsys Coverity

Java
- SpotBugs
- FindBugs with
  - FindSecurityBugs and
  - fb-contrib plug-ins
- Error Prone
- PMD
- Checkstyle
- OWASP Dependency-Check
- Parasoft Jtest

Android
- Android Lint
- RevealDroid

Python
- Bandit
- Flake8
- Pylint

Ruby
- Brakeman
- Dawnscanner
- Reek
- Rubocop
- Ruby-lint

PHP
- PHPMD
- PHP_Codesniffer

JavaScript
- ESLint
- Flow
- JSHint
- Retire.js

HTML
- HTML Tidy

CSS
- CSS Lint

XML
- XML Lint

Code Metrics (all)
- Cloc
- Lizard

Platforms

- Debian
- Ubuntu
- CentOS
- Scientific Linux
- Fedora

SWAMP Glossary

Package: A program, with all its source files and build ("make") commands. More than one user can share this package.

Project: A list of packages and a place to store the result of assessing those packages. Can be shared amongst different users.

Assessment: Running an analysis tool on a particular package.

Steps with the SWAMP

1. Create a new Project.
2. Add new Packages to that Project.

   Either:

   1. Upload a new package or
   2. Reference a package that already exists in the SWAMP.
3. Assess the Packages with the desired Tools.
4. View the results of the assessment.
5. Interpret the results and fix the problems.

CWE 601: Open Redirect

```
public void doGet(HttpServletRequest request,
                                HttpServletResponse response)
                        throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter returnHTML = response.getWriter();
  returnHTML.println("<html><head><title>");
  returnHTML.println("Open Redirect");
  returnHTML.println("</title></head><body>");

  String data;
  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) {
        URI uri;
        uri = new URI(data);
    // POTENTIAL FLAW: redirect is sent verbatim.
        response.sendRedirect(data);
        return;
   }
}
```

How to Describe a Weakness

- Attack point: How does the attacker affect the program.
- Impact point: Where in the program does the bad thing actually happen.

We describe these concepts in more depth in our module on "Thinking Like an Attacker".

Open Redirect (CWE 601)

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

Code with weakness:

- Attack Point: Reading data from the first cookie using `getCookies()`.
- Impact Point: `SendRedirect()` uses user supplied data.

Code without the weakness:

- Use a hard-coded string as argument to `SendRedirect()`.

CWE601_Open_Redirect__Servlet_getCookies_Servlet_01.java

Register to use the SWAMP

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl

https://www.mir-swamp.org

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

SWAMP | About | Contact | Resources | Policies | Help | Sign In

SWAMP

SOFTWARE ASSURANCE MARKETPLACE

Do It Early. Do It Often.

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!

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 code. Rest assured that it will remain private and secure.

2) Run your 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.

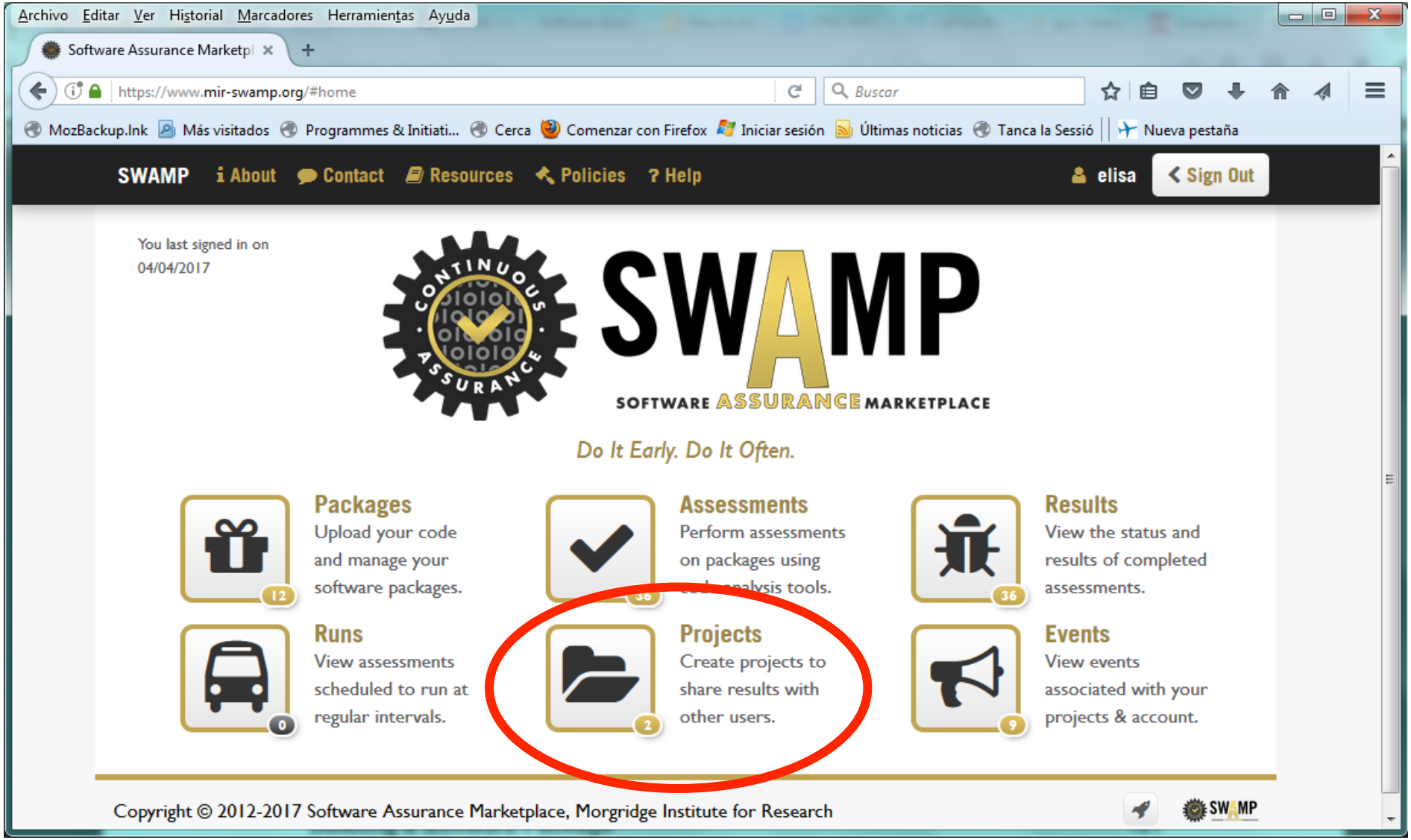
How Can you Identify Yourself

- Your SWAMP Login/Password.
- Your github account.
- Your Google account.
- Your university account though CILogon/ InCommon. http://www.cilogon.org/

  Check if you belong to a participating organization:

  https://www.incommon.org/participants/

What can I do in the SWAMP?



Create a Project

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl

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

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

SWAMP About Contact Resources Policies Help elisa Sign Out

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

Project	Description	Date Added
Tutorial Java	Tutorial Java	11/13/2014 15:59
Tools tutorial	Tools tutorial	10/09/2014 16:33

Projects I Joined

No projects.

Create a Project

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl | OpenRedirect2017 > Analysis | Weakness 34 Details | Code D | findbugs3.0.1 Report

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

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

SWAMP About Contact Resources Policies Help elisa Sign Out

Add New Project

Home / Projects / Add New Project

Please enter the details of your new project below.

Full name * MyProject

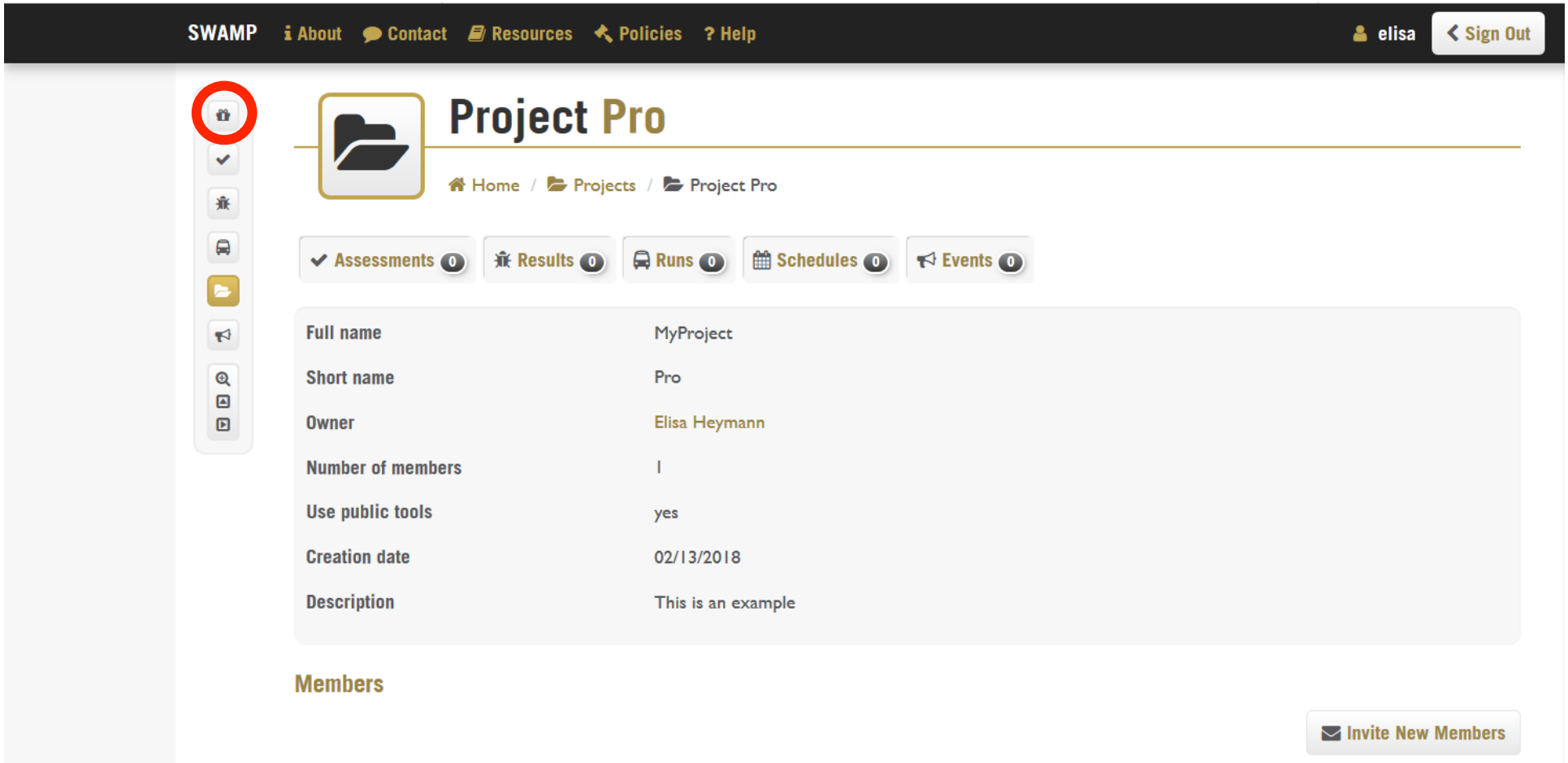
Short name * Pro

Description * This is an example.

*Fields are required

Save Project Cancel

Create a Project



Packages

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl

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

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

SWAMP About Contact Resources Policies Help 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 all items

+ Add New Package

Package	Description	Type	Versions
buffer overflow		C/C++	1.0
command injection		C/C++	1.0
hard coded password		C/C++	1.0
info exposure		C/C++	1.0

Upload your Software Package

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl | OpenRedirect2017 > Analysis | OpenRedirect2017 > Analysis | Weakness 34 Details | Code D

https://www.mir-swamp.org/#packages/add 80% Buscar

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

SWAMP | About | Contact | Resources | Policies | Help | elisa | Sign Out

Add New Package

Home / Packages / Add New Package

Details | Source | Build | Sharing

PACKAGE INFO

Name * OpenRedirect2017

Description

File source
- Local file system
  The package source code is located on your local hard drive.
- Remote Git repository
  The package source code is located on a remote Git server.

PACKAGE VERSION INFO

File * Examinar... 10-b-OpenRedirect.tar

formats supported

Version * 1.0

Version notes

Upload your Software Package

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl

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

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

SWAMP | About | Contact | Resources | Policies | Help | elisa | Sign Out

Details | Source | Build | Sharing

Notice: This appears to be a Java bytecode package. You can set the language type if this is not correct.

Package path * 10-open-redirect/ Select

Language * Java

Show File Types

Java type

Java source
The package contains uncompiled Java code in its original source code format (.java files).

Java bytecode
The package contains Java code which has been compiled (.class, .jar, or .apk files).

Android APK
The package contains compiled Java code for the Android platform.

Java version

Upload your Software Package

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl

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

SWAMP | About | Contact | Resources | Policies | Help | elisa | Sign Out

Add New Package

Home / Packages / Add New Package

Details | Source | Build | Sharing

The following parameters are used to configure the build script which is used to build the package.

Class path *

Add

JAVA BYTECODE INFO

Advanced settings

*Fields are required

Package dependencies

No dependencies have been defined.

Add New Dependency

*Fields are required

Upload your Software Package

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl

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

SWAMP | About | Contact | Resources | Policies | Help | elisa | Sign Out

Add New Package

Home / Packages / Add New Package

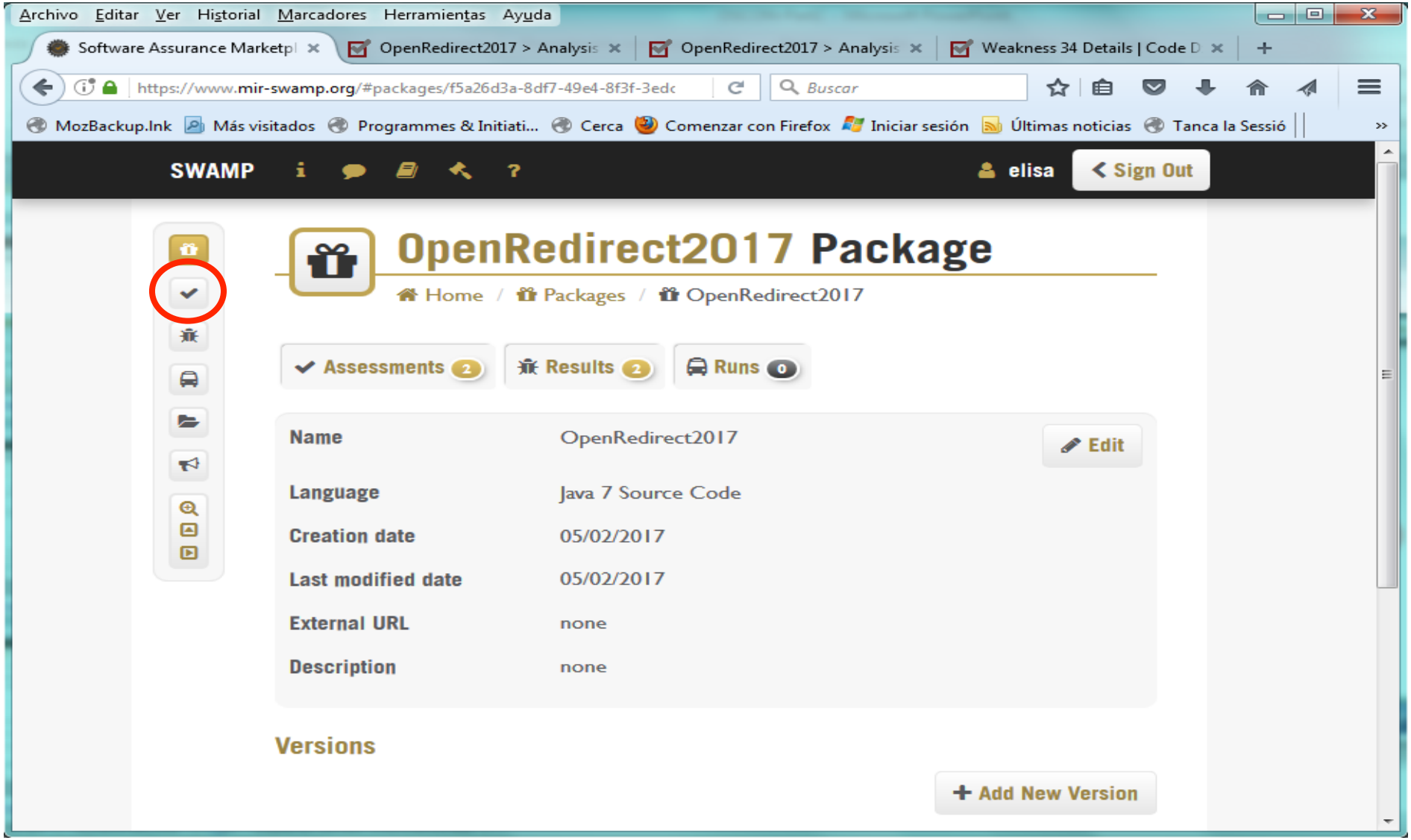
Details | Source | Build | Sharing

This package version is shared with members of the following projects:

	Project	Description
☑	Tutorial Java	Tutorial Java
☐	Tools tutorial	Tools tutorial

Save New Package | Prev | Cancel

Upload your Software Package



Run your Assessments

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Software Assurance Marketpl

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

SWAMP | About | Contact | Resources | Policies | Help | elisa | Sign Out

Assessments

Home / Assessments

Results 36 | Runs 0

Assessments are triplets of package, tool, and platform identifiers that together specify an assessment to be run. To run or schedule an assessment, select one or more assessments from the list below or add a new assessment.

Filters: any project | any package | any tool | any platform | all items

Run Assessments | Run New Assessment

Package	Tool	Platform	Results
buffer overflow latest	Parasoft C/C++test latest	Red Hat Enterprise Linux 6 64-bit latest	
	cppcheck latest		
	Clang Static Analyzer latest		
command injection latest	Parasoft C/C++test latest	Red Hat Enterprise Linux 6 64-bit latest	
	cppcheck latest		

Run your Assessments

File Edit View History Bookmarks Tools Help

Software Assurance Marketplac

https://www.mir-swamp.org/#assessments/run 90%

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

SWAMP About Contact Resources Policies Help 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: OpenRedirect2017

Select a version: Latest

Tool

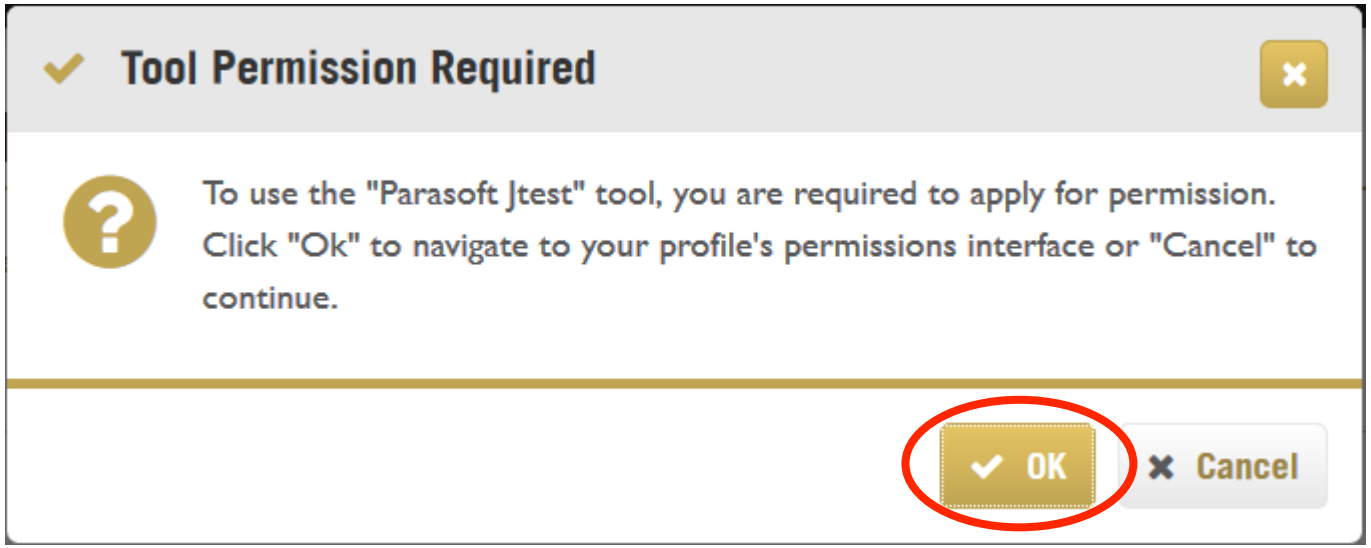
Select a tool to use: SpotBugs

Select a version: Latest

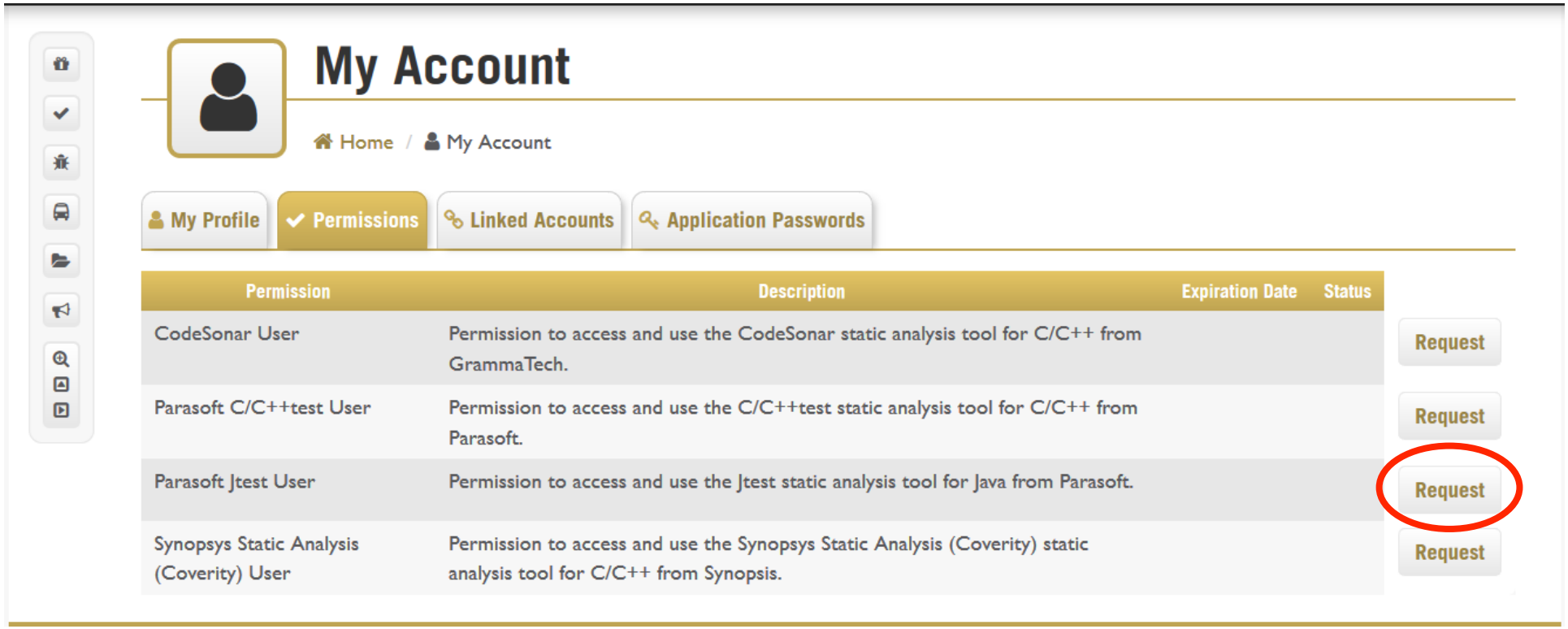
Save and Run Save Cancel

Run your Assessments

The first time you try to use a commercial tool you'll get this message:



Run your Assessments



Run your Assessments

In addition to the SWAMP web interface you can integrate SWAMP submissions into your workflow:

- IDE (Eclipse): Submission with a push of the SWAMP button. View results directly in Eclipse code window.
- CI (Jenkins): Submission with each build or periodically. View results in the Jenkins dashboard.
- Repositories (git/svn): Submission with each code commit. View results in the SWAMP.

Plugins publicly available for each of these.

My Assessments

File Edit View History Bookmarks Tools Help

Software Assurance Marketplac

https://www.mir-swamp.org/#results?package=f5a26d3 90% Search

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

SWAMP About Contact Resources Policies Help elisa Sign Out

Assessment results contain the results of an assessment run of a package using a tool on a particular platform. You may view the results of a single assessment run or you may view the output of several runs of a package using different tools in order to compare the results.

Filters any project OpenRedirect2017 any tool any platform any date 50 items

Viewer Code Dx Native

Notice: Click the view assessment results button to view the selected results using the selected viewer.

Auto refresh

View Assessment Results

Package	Tool	Platform	Date	Status	
OpenRedirect2017 1.0	SpotBugs 3.1.0	Ubuntu 16.04 64-bit	01/31/2018 11:43	finished	3
OpenRedirect2017 1.0	Parasoft Jtest 9.6.0	Ubuntu 16.04 64-bit	05/02/2017 14:00	finished	5
OpenRedirect2017 1.0	Findbugs 3.0.1	Ubuntu 16.04 64-bit	05/02/2017 13:08	finished	3

View your Results. SpotBugs - Native

File Edit View History Bookmarks Tools Help

Software Assurance Marketplac | Software Assurance Marketplac

https://www.mir-swamp.org/#results/a275a6dc-06ae-11 90% Search

MozBackup.lnk | Más visitados | Programmes & Initiati... | Cerca | Comenzar con Firefox | Iniciar sesión | Últimas noticias | Tanca la Sessió | Nueva pestaña

SWAMP | About | Contact | Resources | Policies | Help | elisa | Sign Out

Native Viewer Report

Home / Native Viewer Report

Summary

Package	OpenRedirect2017 **version 1.0**
Tool	SpotBugs **version 3.1.0**
Platform	Ubuntu **version 16.04 64-bit**
Number of weaknesses found	3
Create date	01/31/2018 11:46:16

Results

File	Line	Column	Severity	Group	Code
pkg1/10-b-OpenRedirect/OpenRedirect.java	41		1	STYLE	DLS_DEAD_LOCAL_STORE
	59		2	STYLE	RCN_REDUNDANT_NULLCHECK_OF_NONNULL_VALUE
	50		1	SECURITY	UNVALIDATED_REDIRECT

View your Results. SpotBugs - CodeDx

File Edit View History Bookmarks Tools Help

Software Assurance Marketplac | OpenRedirect2017 > Analysis R

https://swa-csaweb-pd-01.mir-swamp.org/proxy-91D6FF50-06C

MozBackup.lnk | Más visitados | Programmes & Initiati... | Cerca | Comenzar con Firefox | Iniciar sesión | Últimas noticias | Tanca la Sessió | Nueva pestaña

Projects

v1.8.3 SW 11/12/2015 CodeDx

OpenRedirect2017 » Analysis Run 24 Created on 1/31/2018 Uploaded on 1/31/2018 3 total weaknesses View

Weakness Flow

Filters clear all

nt 1 / 3

Tool clear filter

spotbugs (100%)

SECURITY (100%)

STYLE (0%)

Severity

Unspecified (100%)

Codebase Location

Tool Overlaps

CWE

Status

New (100%)

Displaying weaknesses matching a rule in SECURITY

Bulk Operations for the 1 matching weakness | Change status... | Generate report

Weaknesses

Id	Tool	Rule	CWE	Codebase Location	Status
7074	spotbugs	UNVALIDATED_REDIRECT	-	OpenRedirect.java:50	New

Show 25 Displaying 1 to 1 of 1 Weakness

https://codedx.com/Documentation/UserGuide.html

View your Results. SpotBugs - CodeDx

File Edit View History Bookmarks Tools Help

Software Assurance Marketplac | OpenRedirect2017 > Analysis R | Weakness 7074 Details | Code D

https://swa-csaweb-pd-01.mir-swamp.org/proxy-91D6FF50-06C0-

MozBackup.lnk | Más visitados | Programmes & Initiati... | Cerca | Comenzar con Firefox | Iniciar sesión | Últimas noticias | Tanca la Sessió | Nueva pestaña

Projects

v1.8.3 SW
11/12/2015

CodeDx

OpenRedirect2017 > Analysis Run 24 > Weakness 7074 **UNVALIDATED_REDIRECT** detected by **spotbugs**

First seen on **1/31/2018** **3** weaknesses in this file **1** similar weakness in this analysis run **Unspecified** severity

No Common Weakness Enumeration information available

jump to weakness

Status

New

Activity Stream

Post Clear Write comments with Markdown

Status set to **New** during Analysis Run 24 by **admin**

4 minutes ago

Description

```
The following redirection could be used by an attacker to redirect users to a phishing website.

Bug Path:

*** pkg1/10-b-OpenRedirect/OpenRedirect.java:50 ****** Primary Bug Location
  At OpenRedirect.java:[line 50]
*** pkg1/10-b-OpenRedirect/OpenRedirect.java:30 ***
  At OpenRedirect.java:[line 30]
```

Source Code

The weakness occurs in **10-b-openredirect.zip/10-b-OpenRedirect/OpenRedirect.java** on line **50**

```
1 import javax.servlet.*;
2 import javax.servlet.http.*;
3 import java.io.*;
4 import java.net.URI;
5 import java.net.URISyntaxException;
```

View your Results. SpotBugs - CodeDx

OpenRedirect2017 > Analysis Run 24 > Weakness 7074 UNVALIDATED_REDIRECT detected by spotbugs

jump to top

First seen on 1/31/2018 3 weaknesses in this file 1 similar weakness in this analysis run Unspecified severity

No Common Weakness Enumeration information available

jump to weakness

Status

New

Activity Stream

Post Clear Write comments with Markdown

Status set to New during Analysis Run 24 by admin

9 minutes ago

Source Code

The weakness occurs in 10-b-openredirect.zip/10-b-OpenRedirect/OpenRedirect.java on line 50

```
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,
10                      HttpServletResponse response)
11                 throws ServletException, IOException {
12
13 response.setContentType("text/html");
14     PrintWriter returnHTML = response.getWriter();
15
16         returnHTML.println("<html><head><title>");
17     returnHTML.println("OpenRedirect");
18     returnHTML.println("</title></head><body>");
19     returnHTML.println("<h2>Elisa: Bad case</h2>");
20
21         // bad
22
23     String data;
24
25     data = ""; /* initialize data in case there are no cookies */
26     /* Read data from cookies */
27     Cookie cookieSources[] = request.getCookies();
28     if (cookieSources != null) {
29         /* POTENTIAL FLAW: Read data from the first cookie value */
30         data = cookieSources[0].getValue();
31     }
```

View your Results. SpotBugs - CodeDx

OpenRedirect2017 > Analysis Run 24 > Weakness 7074 **UNVALIDATED_REDIRECT** detected by **spotbugs**

jump to top

First seen on **1/31/2018** **3** weaknesses in this file **1** similar weakness in this analysis run **Unspecified** severity

No Common Weakness Enumeration information available

jump to weakness

Status

New

Activity Stream

Post Clear Write comments with Markdown

Status set to **New** during Analysis Run 24 by **admin**

11 minutes ago

The weakness occurs in **10-b-openredirect.zip/10-b-OpenRedirect/OpenRedirect.java** on line **50**

```
23      String data;
24
25      data = ""; /* initialize data in case there are no cookies */
26      /* Read data from cookies */
27      Cookie cookieSources[] = request.getCookies();
28      if (cookieSources != null) {
29          /* POTENTIAL FLAW: Read data from the first cookie value */
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 exceptURISyntax)
44          {
45              response.getWriter().write("Invalid redirect URL");
46              return;
47          }
48      /* POTENTIAL FLAW: redirect is sent verbatim; escape the string to prevent ancillary issues like
        XSS, Response splitting etc */
49      /* IMPORTANT:  Comment the 2 following lines to see the good case working!
50          response.sendRedirect(data);
51          return;
52      }
53
```

View your Results. Multiple Tools.

Projects

v1.8.3 SW
11/12/2015
Code DX

OpenRedirect2017 » Analysis Run 29 Created on 2/8/2018 Uploaded on 2/8/2018 9 total weaknesses

View

Weakness Flow

Filters

Tool
- FindBugs (11.1%)
- Jtest (55.6%)
- spotbugs (33.3%)

Severity
- Unspecified (33.3%)
- Medium (11.1%)
- High (55.6%)

Codebase Location

Tool Overlaps

CWE

Status
- Unresolved (100%)

Displaying all weaknesses

Bulk Operations for the 9 matching weaknesses | Change status... | Generate report

Weaknesses

Id	Tool	Rule	CWE	Codebase Location	Status
39	Jtest	Avoid conditions that always evaluate to t...	-	OpenRedirect.java:59	Unresol...
38	Jtest	Protect against HTTP response splitting	79	OpenRedirect.java:50	Unresol...
37	Jtest	Protect against network resource injection	601	OpenRedirect.java:41	Unresol...
36	Jtest	Encapsulate all dangerous data returning ...	79	OpenRedirect.java:30	Unresol...
35	Jtest	Encapsulate all dangerous data returning ...	79	OpenRedirect.java:27	Unresol...
34	FindBugs	Non-validated redirect	601	OpenRedirect.java:50	Unresol...
7074	spotbugs	UNVALIDATED_REDIRECT	-	OpenRedirect.java:50	Unresol...
[illegible]	spotbugs	RCN_REDUNDANT_NULLCHECK_OF_NON...	-	OpenRedirect.java:59	Unresol...
7072	spotbugs	DLS_DEAD_LOCAL_STORE	-	OpenRedirect.java:41	Unresol...

Show 25 Displaying **1** to **9** of **9** Weaknesses

View your Results. Multiple Tools.

OpenRedirect2017 > Analysis Run 30 > Weakness 7074 **UNVALIDATED_REDIRECT** detected by **spotbugs**

jump to top

First seen on **2/8/2018** **9** weaknesses in this file **1** similar weakness in this analysis run **Unspecified** severity

No Common Weakness Enumeration information available

jump to weakness

The weakness occurs in **10-b-openredirect.zip/10-b-OpenRedirect/OpenRedirect.java** on line **50**

Status

Unresolved

Activity Stream

Post Clear Write comments with Markdown

Status set to **Unresolved** during Analysis Run 27 by **admin**

8 days ago

Status set to **New**

8 days ago

7074: UNVALIDATED_REDIRECT

Found by **spotbugs** on line **50**

34: Non-validated redirect

Found by **FindBugs** on line **50** with CWE **601**

38: Protect against HTTP response splitting

Found by **Jtest** on line **50** with CWE **79**

```
20
21      // bad
22
23      String data;
24
25      data = ""; /* initialize data in case there are no cookies */
26      /* Read data from cookies */
27      Cookie cookieSources[] = request.getCookies();
28      if (cookieSources != null) {
29          /* POTENTIAL FLAW: Read data from the first cookie value */
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 exceptURISyntax)
44          {
45              response.getWriter().write("Invalid redirect URL");
46              return;
47          }
48      /* POTENTIAL FLAW: redirect is sent verbatim; escape the string to prevent ancillary issues like XSS, Response splitting etc */
49      // IMPORTANT:  Comment the 2 following lines to see the good case working!
50          response.sendRedirect(data);
51          return;
52      }
```

Interpret your Results

- Go through the list of issues detected by the tool.

Projects

v1.8.3 SW
11/12/2015
Code Dx

WebGoat7-1--vm » Analysis Run 26 Created on 1/31/2018 Uploaded on 1/31/2018 1,006 total weaknesses

View

Weakness Flow

Filters

Tool
- spotbugs (100%)
  - BAD_PRACTICE (9.5%)
  - CORRECTNESS (9.2%)
  - EXPERIMENTAL (14.5%)
  - I18N (3.4%)
  - MALICIOUS_CODE (0.5%)
  - PERFORMANCE (10.6%)

Severity
- Unspecified (100%)

Codebase Location

Tool Overlaps

CWE

Status
- Unresolved (100%)

Displaying all weaknesses

Bulk Operations for the 1,006 matching weaknesses | Change status... | Generate report

Weaknesses

Id	Tool	Rule	CWE	Codebase Location	Status
7071	spotbugs	UCPM_USE_CHARACTER_PARAMETERIZ...	-	LessonUtil.java:54	Unresol...
7070	spotbugs	UCPM_USE_CHARACTER_PARAMETERIZ...	-	LessonUtil.java:52	Unresol...
7069	spotbugs	UCPM_USE_CHARACTER_PARAMETERIZ...	-	LessonUtil.java:50	Unresol...
7068	spotbugs	UCPM_USE_CHARACTER_PARAMETERIZ...	-	LessonUtil.java:48	Unresol...
7067	spotbugs	PATH_TRAVERSAL_IN	-	LessonUtil.java:140	Unresol...
7066	spotbugs	LSYC_LOCAL_SYNCHRONIZED_COLLECT...	-	LessonUtil.java:46	Unresol...
7065	spotbugs	LSYC_LOCAL_SYNCHRONIZED_COLLECT...	-	ExecResults.java:320	Unresol...
7064	spotbugs	LSC_LITERAL_STRING_COMPARISON	-	ExecResults.java:333	Unresol...
7063	spotbugs	LSC_LITERAL_STRING_COMPARISON	-	ExecResults.java:328	Unresol...
7062	spotbugs	LSC_LITERAL_STRING_COMPARISON	-	ExecResults.java:323	Unresol...
7061	spotbugs	SEO_SUBOPTIMAL_EXPRESSION_ORDER	-	Exec.java:227	Unresol...
7060	spotbugs	SEO_SUBOPTIMAL_EXPRESSION_ORDER	-	Exec.java:409	Unresol...
7059	spotbugs	RCN_REDUNDANT_NULLCHECK_WOULD...	-	Exec.java:100	Unresol...
7058	spotbugs	RCN_REDUNDANT_NULLCHECK_WOULD...	-	Exec.java:282	Unresol...
7057	spotbugs	MDM_STRING_BYTES_ENCODING	-	Exec.java:127	Unresol...

Status
- New (100%)

Interpret your Results

- **Try to address the most relevant first: high priority, security related, …**

Projects

v1.8.3 SW 11/12/2015 Code Dx

WebGoat7-1--vm » Analysis Run 26 Created on 1/31/2018 Uploaded on 1/31/2018 1,006 total weaknesses

View

Weakness Flow

Filters clear all

akness count 87 / 1,006

Tool clear filter

- CORRECTNESS (0%)
- EXPERIMENTAL (0%)
- I18N (0%)
- MALICIOUS_CODE (0%)
- PERFORMANCE (0%)
- SECURITY (100%)
- STYLE (0%)

Severity

- Unspecified (100%)

Codebase Location

Tool Overlaps

CWE

Status

- Unresolved (100%)

Displaying weaknesses matching a rule in SECURITY

Bulk Operations for the 87 matching weaknesses | Change status... | Generate report

Weaknesses

Id	Tool	Rule	CWE	Codebase Location	Status
7067	spotbugs	PATH_TRAVERSAL_IN	-	LessonUtil.java:140	Unresol...
7049	spotbugs	COMMAND_INJECTION	-	Exec.java:108	Unresol...
7021	spotbugs	SQL_NONCONSTANT_STRING_PASSED...	-	Login.java:127	Unresol...
7020	spotbugs	SQL_INJECTION_JDBC	-	Login.java:127	Unresol...
6974	spotbugs	SQL_NONCONSTANT_STRING_PASSED...	-	DefaultLessonAction.java:252	Unresol...
6973	spotbugs	SQL_INJECTION_JDBC	-	DefaultLessonAction.java:252	Unresol...
6938	spotbugs	SQL_INJECTION_JDBC	-	BackDoors.java:178	Unresol...
6937	spotbugs	SQL_INJECTION_JDBC	-	BackDoors.java:146	Unresol...
6936	spotbugs	SQL_INJECTION_JDBC	-	BackDoors.java:139	Unresol...
6910	spotbugs	SQL_INJECTION_JDBC	-	BlindNumericSqlInjection.java:112	Unresol...
6896	spotbugs	PATH_TRAVERSAL_IN	-	BlindScript.java:210	Unresol...
6895	spotbugs	PATH_TRAVERSAL_IN	-	BlindScript.java:206	Unresol...
6876	spotbugs	SQL_INJECTION_JDBC	-	BlindStringSqlInjection.java:112	Unresol...
6855	spotbugs	SQL_NONCONSTANT_STRING_PASSED...	-	ChallengeScreen.java:226	Unresol...
6854	spotbugs	SQL_INJECTION_JDBC	-	ChallengeScreen.java:226	Unresol...
6849	spotbugs	PATH_TRAVERSAL_IN	-	ChallengeScreen.java:419	Unresol...
6848	spotbugs	PATH_TRAVERSAL_IN	-	ChallengeScreen.java:417	Unresol...
6847	spotbugs	PATH_TRAVERSAL_IN	-	ChallengeScreen.java:380	Unresol...
6846	spotbugs	PATH_TRAVERSAL_IN	-	ChallengeScreen.java:379	Unresol...

Interpret your Results

```
20
21      // bad
22
23      String data;
24
25      data = ""; /* initialize data in case there are no cookies */
26      /* Read data from cookies */
27      Cookie cookieSources[] = request.getCookies();
28      if (cookieSources != null) {
29          /* POTENTIAL FLAW: Read data from the first cookie value */
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 exceptURISyntax)
44          {
45              response.getWriter().write("Invalid redirect URL");
46              return;
47          }
48      /* POTENTIAL FLAW: redirect is sent verbatim; escape the string to prevent ancillary issues like XSS
, Response splitting etc */
49      // IMPORTANT: Comment the 2 following lines to see the good case working!
50          response.sendRedirect(data);
51          return;
52      }
53
```

Interpret your Results

- Determine if it's a real problem or a false positive.
- If it's a true positive, fix the problem.
- If it's a false positive mark it so it won't be raised again when running again the assessment.
- Upload a new version of the Package, in the same Project.
- Run the assessment again.

Summary

- The SWAMP allows easy access to multiple automated tools for software assurance.
- Every project has now access to a great suite of tools for software assurance.
- Scanning your software for weaknesses should be part of the software development life cycle.
- Assess your software periodically to prevent code changes introduce new weaknesses.
- If you are not comfortable uploading your software, consider using SWAMP-in-a-Box.

Questions?

Elisa Heymann

Elisa.Heymann@uab.es

Barton P. Miller

bart@cs.wisc.edu

https://continuousassurance.org

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

THE UNIVERSITY of WISCONSIN MADISON

Universitat Autònoma de Barcelona

TRUSTED CI THE NSF CYBERSECURITY CENTER OF EXCELLENCE