Lecture Notes
September 13, 2018

Code smell detectors are typically part of a more general static analysis "bug finder" tool that also finds other kinds of generic code problems

- These tools do not need to know anything about the intended functionality (features)

- Sometimes encode domain-specific rules, e.g., web client/server messaging formats and protocols, or knowledge of error specifications for widely used

- APIs, e.g., returns 0 on success and one of a list of predefined integers (macros) on failure

- May have option for users to add their own application-specific rules

Some Java bug finders: JDepend, Spotbugs (successor to FindBugs)

Some multi-language bug finders: PMD, SonarQube (combines multiple tools), Coverity (free for open-source projects)

In-class demo of PMD bug finder tool (Anthony)

Examples of what bug finders can detect:

Secret keys, tokens or passwords embedded in code

The bug finder could find this by looking for string constants containing domain-specific patterns known to appear in AWS configurations

The following was found in a tutorial on how to add password protection to a page using Javascript. What is wrong?

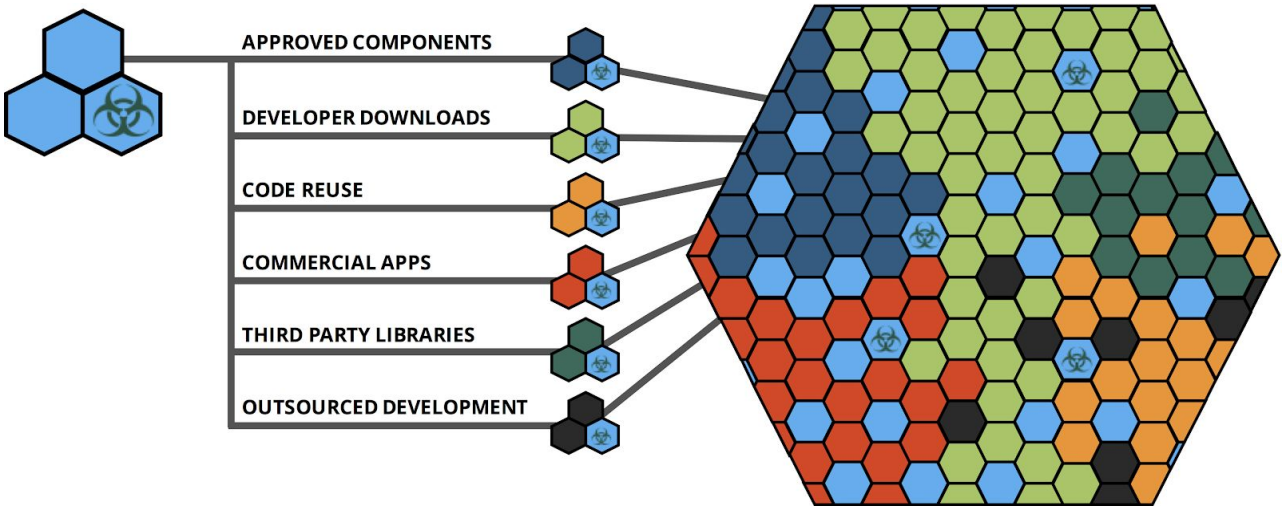
change.log new 1

```
1  <SCRIPT>
2  function passWord() {
3  var testV = 1;
4  var pass1 = prompt('Please Enter Your Password',' ');
5  while (testV < 3) {
6  if (!pass1)
7  history.go(-1);
8  if (pass1.toLowerCase() == "letmein") {
9  alert('You Got it Right!');
10 window.open('www.wikihow.com');
11 break;
12 }
13 testV+=1;
14 var pass1 =
15 prompt('Access Denied - Password Incorrect, Please Try Again.','Password')
16 }
17 if (pass1.toLowerCase()!="password" & testV ==3)
18 history.go(-1);
19 return " ";
20 }
21 </SCRIPT>
22 <CENTER>
23 <FORM>
24 <input type="button" value="Enter Protected Area" onClick="passWord()">
25 </FORM>
26 </CENTER>
```

wikiHow to Password Protect a Web Page

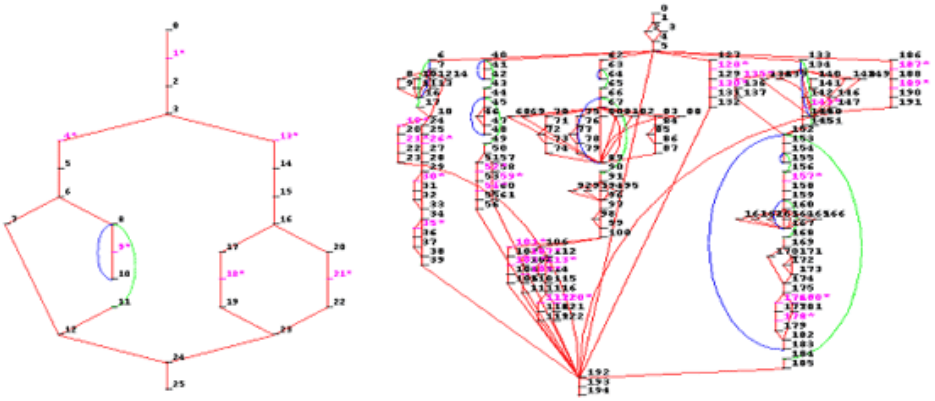
How do you think a bug finder tool could find the security bug in this code? The code could replace all uses of the identifier "password" with "xyzzy", and the bug finder should still be able to spot the problem.

Using third-party libraries with known security vulnerabilities



The bug finder tool compares code resources to databases of known security vulnerabilities, e.g., [whitesource]

Too many paths in a given function or method



Simple Code vs. Complex Code

This code has cyclomatic complexity 14, far beyond the typical threshold - but no human would consider this code complex. So a good bug finder needs a more sophisticated complexity metric

```
String getMonthName (int month) {
    switch (month) {
        case 0: return "January";
        case 1: return "February";
        case 2: return "March";
        case 3: return "April";
        case 4: return "May";
        case 5: return "June";
        case 6: return "July";
        case 7: return "August";
        case 8: return "September";
        case 9: return "October";
        case 10: return "November";
        case 11: return "December";
        default: throw new IllegalArgumentException();
    }
}
```

Client code that does not check/handle every possible error code or exception from a library or system call

Long list of linux error codes:
[errno - number of last error]

Errors vs. exceptions:

- A user entering the wrong data is not exceptional and does not need to be handled with an exception, but can still result in an unrecoverable state. Simple checks, on both front-end and back-end, can address this user error.

- A file won't open and is throwing FileLoadException or FileNotFoundException. This is an exceptional situation that should be handled by the application by catching the exception with appropriate processing code.

Dereferencing a null pointer

```
int a, b, c; // some integers
int *pi;     // a pointer to an integer

a = 5;
pi = &a; // pi points to a
b = *pi; // b is now 5
pi = NULL;
c = *pi; // this is a NULL pointer dereference
```

Static analysis can check whether its possible for a pointer variable to be set to NULL on any path to a given statement that dereferences the pointer. In this case there's only one simple path.

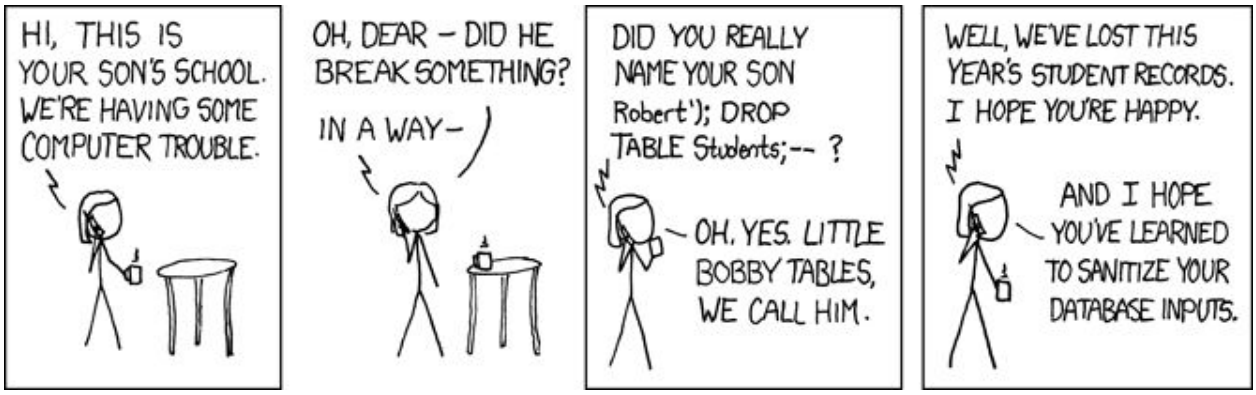
What should the bug finder do in the following case?

```
int a, b, c; // some integers
int *pi;     // a pointer to an integer

a = 5;
pi = &a; // pi points to a
b = *pi; // b is now 5
pi = abracadabra(a, b);
c = *pi; // could pi be a NULL pointer?
```

[You're dereferencing a null pointer!]

Input from user passed to framework, library API, logger, database, external system, etc. without sanitizing first



The school apparently stores student names in a table called Students. When a new student arrives, the school inserts his/her name into this table. The code doing the insertion might look like:

```
$sql = "INSERT INTO Students (Name)
        VALUES ('" . $studentName . "');";
execute_sql($sql);
```

This code first creates a string containing an SQL INSERT statement. The content of the $studentName variable is glued into the SQL statement. Then the code sends the resulting SQL statement to the database. Untrusted user input, i.e., the content of $studentName, becomes part of the SQL statement.

Say the user input is “John”, then the SQL is:

```
INSERT INTO Students (Name) VALUES ('John')'
```

This inserts John into the Students table.

Now say the user input is
“Robert'); DROP TABLE Students;--”
then the SQL statement becomes

```
INSERT INTO Students (Name) VALUES ('Robert');
DROP TABLE Students;--');
```

This inserts Robert into the Students table. But the INSERT statement is now followed by a DROP TABLE statement - which removes the entire Students table.

A bug finder could check for external inputs being used in constructing an SQL statement, rather than as arguments to a “prepared statement”. Every programming language commonly used with databases has some facility for prepared statements - which “sanitizes” all user-supplied data before inserting into the underlying SQL statements.

Sanitization is not just for SQL. Consider:

```
if (loginSuccessful) {
  logger.severe("User login succeeded for: " + username);
} else {
  logger.severe("User login failed for: " + username);
}
```

Say user has entered username "guest", the log gets

```
May 15, 2011 2:19:10 PM java.util.logging.LogManager$RootLogger log
SEVERE: User login failed for: guest
```

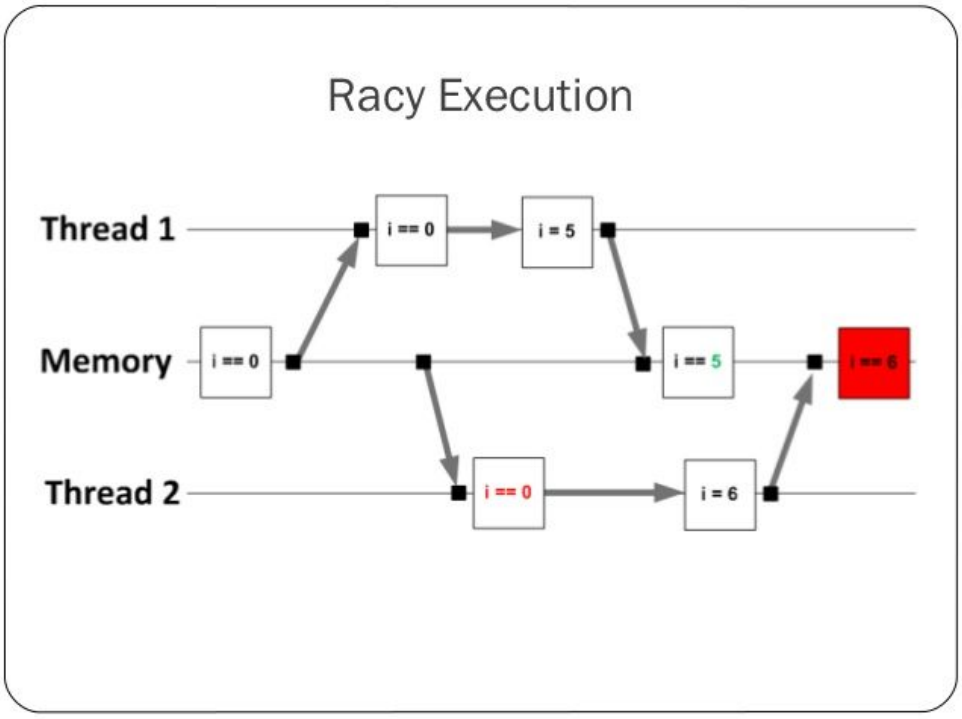
Say the user entered username
"guest
May 15, 2011 2:25:52 PM
java.util.logging.LogManager$RootLogger log
SEVERE: User login succeeded for: administrator"

The log gets

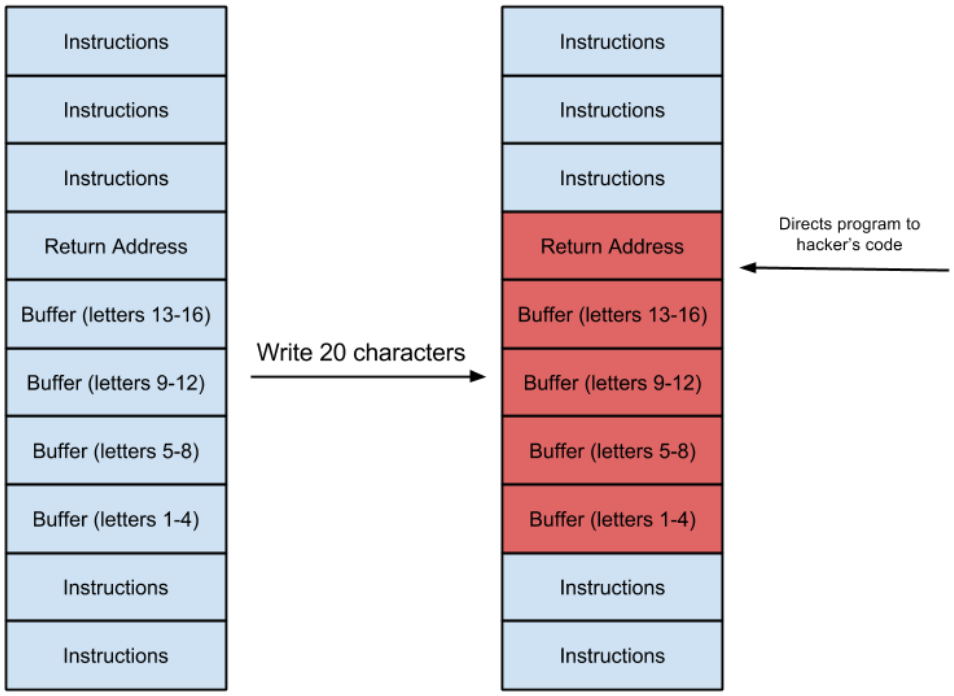
```
May 15, 2011 2:19:10 PM java.util.logging.LogManager$RootLogger log
SEVERE: User login failed for: guest
May 15, 2011 2:25:52 PM java.util.logging.LogManager log
SEVERE: User login succeeded for: administrator
```

How could a bug finder detect this problem?

There are many possible application-independent bug finder checks for multi-threaded code, e.g., matching locks with unlocks along every execution path and detecting possible data races

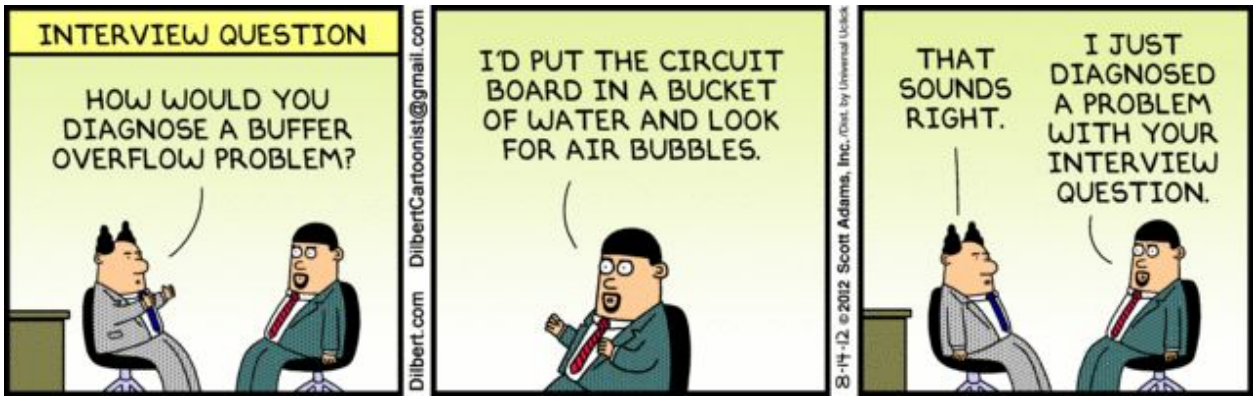


A bug finder for unmanaged languages should check array bounds, to prevent buffer overflow



And check that every memory allocation is followed (in every execution path) by a corresponding memory deallocation, and there are no further uses of the memory after it has been deallocated

CWE-416: Use After Free

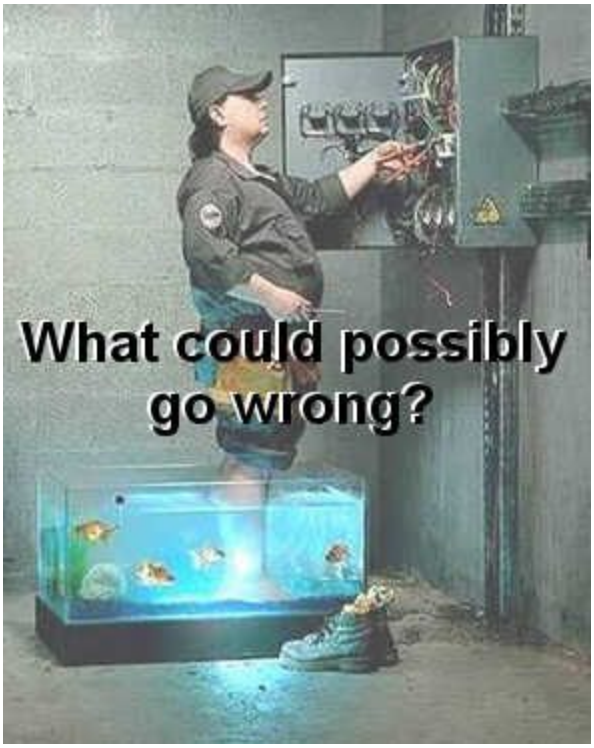

INTERVIEW QUESTION
HOW WOULD YOU DIAGNOSE A BUFFER OVERFLOW PROBLEM?
Dilbert.com DilbertCartoonist@gmail.com
I'D PUT THE CIRCUIT BOARD IN A BUCKET OF WATER AND LOOK FOR AIR BUBBLES.
8-14-12 ©2012 Scott Adams, Inc./Dist. by Universal Uclick
THAT SOUNDS RIGHT.
I JUST DIAGNOSED A PROBLEM WITH YOUR INTERVIEW QUESTION.

Bug finders typically produce many *false positives*, e.g., reporting a missing free or unlock on an execution path that is actually infeasible at runtime

So it is necessary to be able to *suppress* certain error/warning messages. For example, once a human has marked a particular warning as false positive, the bug finder should never give that same warning for the same code or "similar" code

Static bug checkers cannot avoid *false negatives*, they cannot find "all" bugs and cannot "prove" no bug exists

Static analysis does not replace dynamic analysis, particularly testing - static analysis finds bugs that *might* happen with some input. Testing finds bugs that *did* happen with a specific input, but it's infeasible to consider all possible inputs


What could possibly
go wrong?


What could possibly
go wrong?


gif-finder.com

First individual/pair homework assignment: Practice with Github and submitting assignments on Canvas due before class on Tuesday, to make sure you can use github and post assignment submissions on canvas.

First team assignment: Everyone will need a team (of 4 students = 2 pairs) to develop your team project, so start looking for teammates - talk to other students after class, post advertisement on piazza, etc. The team formation assignment is due next Thursday, 11:59pm.

In most cases, team assignments will be due just before midnight on the due date and individual/pair assignments will be due before class on the due date. If that's not what the assignment says, check with the instructor.

Please do not join a team if you plan to drop the class. If you drop the class after already joining a team, please tell your former team members asap so they can recruit new teammates. If you do not already have a full team before class on Thursday next week, ask for help at the beginning (not end!) of class.

Important team formation considerations: Agree on same platform (Linux, Mac, Windows) and same programming language (Java, C/C++, Python). Make sure all prospective teammates have compatible schedules and locations for frequent team meetings (and even more frequent pair meetings).

You do not need to propose your project yet, but it would be wise to brainstorm with prospective teammates about potential project ideas.

Your team project does **not** need to be a web or mobile app, which was required in recent previous years, and instead I discourage web and mobile apps.

Your project **does** need to be something that you will be able to demo - there needs to be a command line console or GUI.

Your project should also use some kind of persistent and structured data store (e.g., SQL, NoSQL, key-value store) and use some third-party library, framework or API (not just the built-in libraries that come with the programming language).