Programming Languages and Techniques (CIS120)

Bonus Lecture

April 17, 2015

“Code is Data”

Code is Data

Note: most images have been removed from this version of the presentation

M.C. Escher, Drawing Hands, 1948

Code is Data

- A Java source file is just a sequence of characters.
- We can represent programs with Strings!

```
String p_0 = “class C { public static void
  main(String args[]) {…}}”
String p_1 = “class D { public static void
  main(String args[]) {…}}”
…
String p_12312398445 = “…” // solution to HW09
…
String p_93919113414 = “…” // code for Eclipse
…
```

Consequence 1: Programs that manipulate programs

Interpreters

- We can create *programs* that manipulate *programs*
- An *interpreter* is a program that executes other programs
- interpret ("3 + 4") ➔ 7
- Example 1: javascript

Tools and Compilers

- Example 2: Eclipse
  - Note that Eclipse manipulates a representation of Java programs
  - Eclipse itself is written in Java
  - So you could use Eclipse to edit the code for Eclipse… ?!
- Example 3: Compiler
  - The Java compiler takes a representation of a Java program
  - It outputs a "low-level" representation of the program as a .class file (i.e. Java byte code)
  - Can also compile to other representations, e.g. x86 "machine code"

Example Compilation: Java to X86

```
class Point {
  int x;
  int y;
  Point move(int dx,
  int dy) {
    x = x + dx;
    y = y + dy;
    return this;
  )
}
```

```
.globl __fun__Point.move
__fun__Point.move:
      pushl %ebp
      movl %esp, %ebp
      subl $4, %esp
__5:
      movl 8(%ebp), %eax
      movl 4(%eax), %eax
      movl %eax, -4(%ebp)
      movl 12(%ebp), %ecx
      addl %ecx, -4(%ebp)
      movl -4(%ebp), %ecx
      movl 8(%ebp), %eax
      movl %ecx, 4(%eax)
      movl 8(%ebp), %eax
      movl 0(%eax), %eax
      movl %eax, -4(%ebp)
      movl 16(%ebp), %ecx
      addl %ecx, -4(%ebp)
      movl -4(%ebp), %ecx
      movl 8(%ebp), %eax
      movl %ecx, 0(%eax)
      movl 8(%ebp), %eax
      movl %ebp, %esp
      popl %ebp
      ret
```

Consequence 2: Malware

Rene Magritte, The Human Condition, 1933

Consequence 2: Malware

- Why does Java do array bounds checking?
- *Unsafe* language like C and C++ don’t do that checking;
  - They will happily let you write a program that “writes past” the end of an array.
- Result:
  - viruses, worms, “jailbreaking” mobile phones, Spam, botnets, …
- Fundamental issue:
  - Code is data.
  - Why?

Consider this C Program

```
void m() {
  char[10] buffer;

  char c = read();
  int i = 0;
  while (c != -1) {
    buffer[i] = c;
    c = read();
    i++;
  }
  process(buffer);
}

void main() {
  m();
  // do some more stuff
}
```

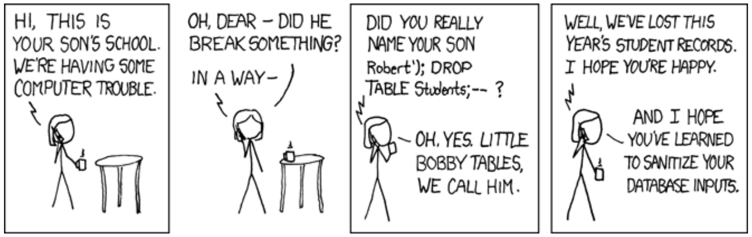
Notes:
- C doesn’t check array bounds
- Unlike Java, it stores arrays directly on the stack
- What could possibly go wrong?

Abstract Stack Machine

“Stack Smashing Attack”

Code Injection Attacks

```
void registerStudent() {
  print(“Welcome to student registration.”);
  print(“Please enter your name:”);
  String name = readLine();
  evalSQL(“INSERT INTO Students(‘” + name + “’)”  );
}
```


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY –
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

http://xkcd.com/327/

Consequence 3: Undecidability

Undecidability Theorem

Theorem: It is impossible to write a method

```
boolean halts(String prog)
```

such that, for any valid Java program P represented as a string `p_P`,

```
halts(p_P)
```

returns true exactly when the program P halts, and false otherwise.

Alonzo Church, April 1936 Alan Turing, May 1936

Halt Detector

- Suppose we could write such a program:

```
class HaltDetector {
  public static boolean halts(String javaProgram) {
    // ...do some super-clever analysis...
    // return true if javaProgram halts
    // return false if javaProgram does not
  }
}
```

- A correct implementation of HaltDetector.halts(p) always returns either true or false
  - i.e., it never raises an exception or loops
- HaltDetector.halts(p) ⇒ true means “p halts”
- HaltDetector.halts(p) ⇒ false means “p loops forever”

Do these methods halt?

```
“boolean m(){ return false; }”
=> YES
“boolean m(){ return m(); }”
=> NO
“boolean m(){
   if (“abc”.length() == 3) return true;
   else return m(); }”
=> YES
“boolean m(){
   String x = “”;
   while (true) {
      if (x.length() == 3) return true;
      x = x + \“a\”;
    }
  return false;
}”
=> YES
```

Consider this Program called Q:

```
class HaltDetector {
  public static boolean halts(String javaProgram) {
    // ...do some super-clever analysis...
    // return true if javaProgram halts
    // return false if javaProgram does not
  }
}

class Main {
  public static void Q() {
     String p_Q = ???;   // string representing method Q
     if (HaltDetector.halts(p_Q)) {
       while (true) {}  // infinite loop!
     }
  }
}
```

What happens when we run Q?

```
public static void Q() {
     String p_Q = ???;   // string representing method Q
     if (HaltDetector.halts(p_Q)) {
       while (true) {}  // infinite loop!
     }
  }
```

if HaltDetector.halts(p_Q) ⇒ true then Q ⇒ infinite loop

if HaltDetector.halts(p_Q) ⇒ false then Q ⇒ halts

Contradiction!

- Russell's Paradox (1901)
- Gödel's Incompleteness Theorem (1931)
- Both rely on *self reference*

Bertrand Russell, 1901

Potential Hole in the Proof

- What about the ??? in the program Q?
- It is supposed to be a String representing the program Q itself.
- How can that be possible?
- Answer: code is data!

- See Quine.java

YO, I'M M.C. QUINE AND I'M HERE TO SAY, "YO, I'M M.C. QUINE AND I'M HERE TO SAY!"

Profound Consequences

- The "halting problem" is *undecidable*
  - *There are problems that cannot be solved by a computer program!*

- Rice's Theorem:
  - Every "interesting" property about computer programs is undecidable!

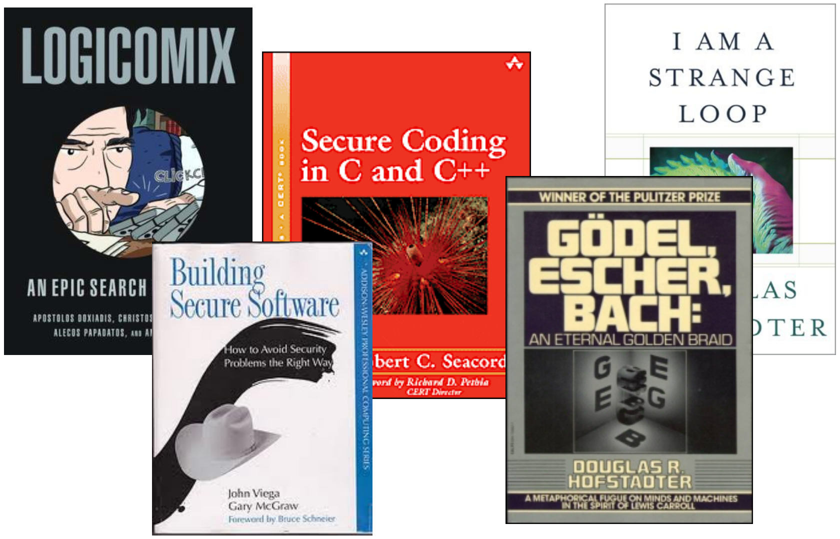
- You can't write a perfect virus detector!
  1. virus detector might go into an infinite loop
  2. it gives you false positives (i.e. says something is a virus when it isn't)
  3. it gives you false negatives (i.e. it says a program is not a virus when it is)

- Also: You can't write a perfect autograder!

Recommended Courses

- Programs that manipulate Programs
  - CIS 341: Compilers and interpreters
- Malware
  - CIS 331: Intro to Networks and Security
- Undecidability
  - CIS 262: Automata, Computability and Complexity

Recommended Reading


LOGICOMIX
CLICK
AN EPIC SEARCH
APOSTOLOS DOXIADIS, CHRISTOS
ALECOS PAPADATOS, AND A
Secure Coding in C and C++
A CERT BOOK
bert C. Seacord
word by Richard D. Pethia CERT Director
Building Secure Software
How to Avoid Security Problems the Right Way
John Viega
Gary McGraw
Foreword by Bruce Schneier
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
I AM A STRANGE LOOP
LAS
DTER
WINNER OF THE PULITZER PRIZE
GÖDEL, ESCHER, BACH:
AN ETERNAL GOLDEN BRAID
G E B
DOUGLAS R. HOFSTADTER
A METAPHORICAL FUGUE ON MINDS AND MACHINES IN THE SPIRIT OF LEWIS CARROLL