Security Workshop

Dr. Carl Pulley
c.j.pulley@hud.ac.uk

Boron Tags

- Strings known to be user input can be used in memory searches to locate
  - input storage areas
  - code that accesses input string
- When strings are used like this, we are searching memory for **boron tags**
- Typically use strings such as *Aa0Aa1Aa2Aa3..* for boron tags

Soft and Hard Breakpoints

- Soft breakpoints implemented using an INT3 instruction
- Hard breakpoints implemented at the CPU level
  - limited number of these breakpoints available
  - registers DR0, DR1, DR2 and DR3 hold breakpoint address

Memory Breakpoints

- Implemented using guard pages
  - pages marked *PAGE_GUARD* generate exceptions when accessed
- Using memory breakpoints means we can locate code that reads or writes to contents of those pages
- By adding in conditional code to the breakpoint handler, we can detect when code handles boron tagged data

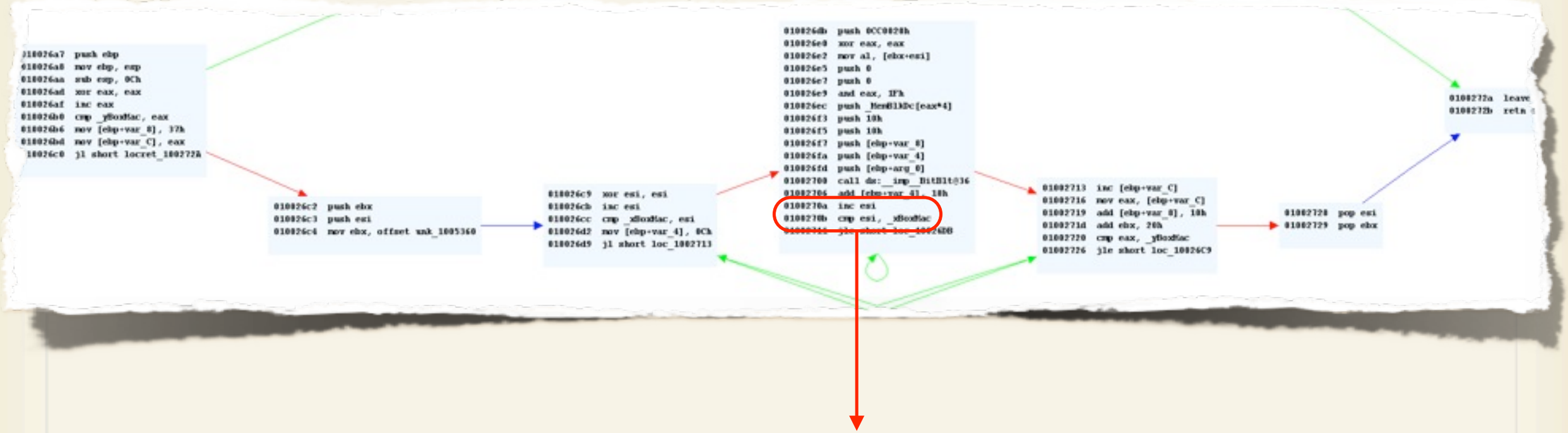
Basic Blocks

- *Basic block* is a contiguous set of assembler instructions that ends in
  - branch instruction
  - call instruction
  - return instruction
  - or a label referenced by a branch instruction elsewhere in the code

Control Flow Graphs

- Assembler code can be broken down into collections of basic blocks joined together by *jump* instructions
  - when laid out as a graph, this provides you with a *control flow graph*
  - Looking at the shape of a control flow graph allows the reverser to identify *high-level* language control-flow constructs
    - eg. *conditional* statements and *looping* constructs

Example



0100270a inc esi
0100270b cmp esi, _xBoxMac

Sunday, 7 February 2010

Control flow graph for winmine'exe's DrawGrid function.
Control flow graph shows a clear looping construct. Detail shows that looping construct could be a for-loop (ESI is being incremented and then compared against a value in _xBoxMac).

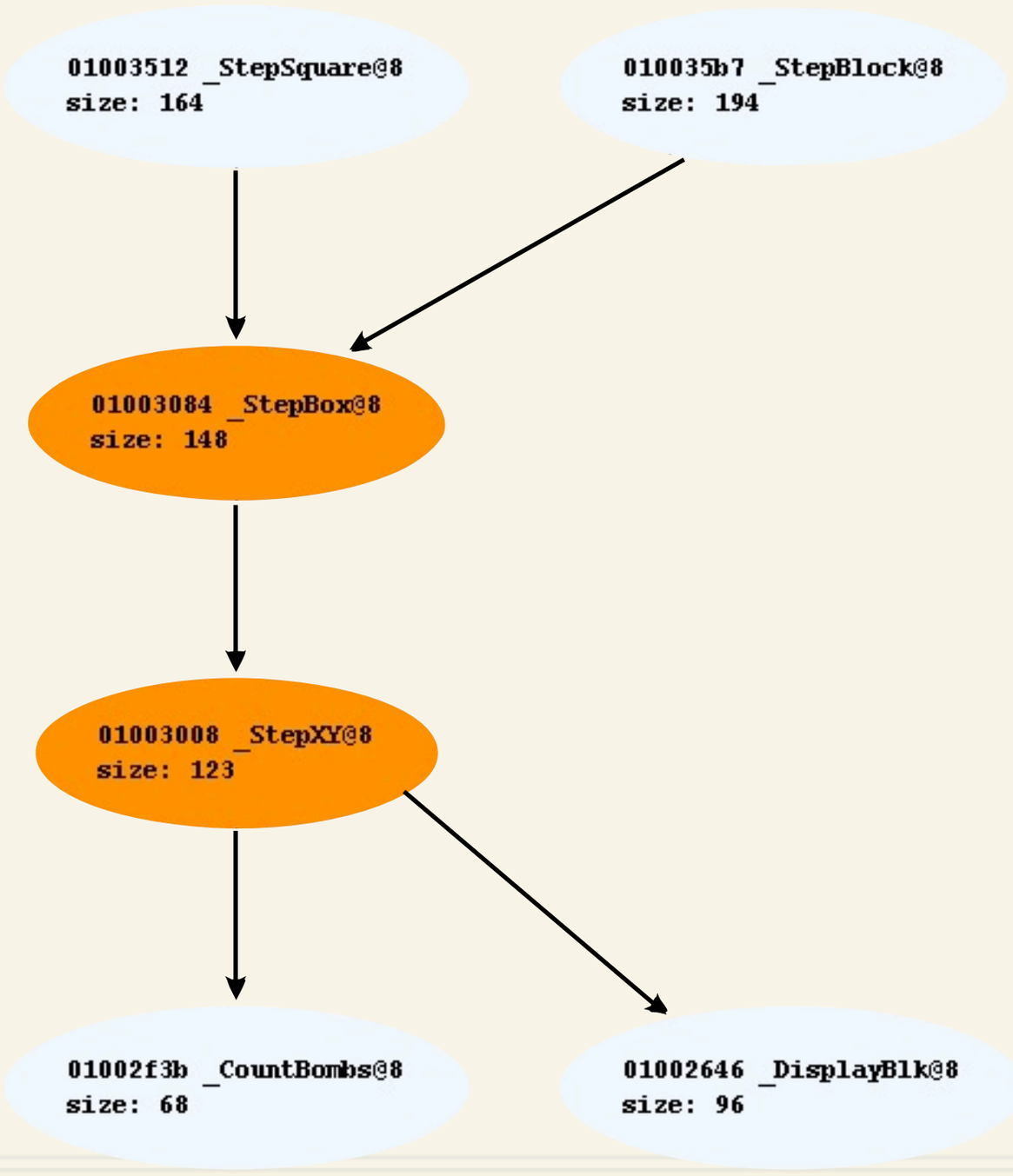
Call Graphs

- Functions are units of code that
  - pop return addresses from the stack
  - utilise stack frames for holding arguments and local variables
- Functions have *prologues* and *epilogues* to initialise *stack frames*
- Call graph is the graph with
  - function *names* as nodes
  - edges exist when a *source* function calls a *target* function

Sunday, 7 February 2010

Can use call graphs to get a quick overview as to what a function may do

Example



Sunday, 7 February 2010

Part of winmine.exe's function call graph. Here we see the winmine.exe functions responsible for iterating through grid squares when an non-mined square is clicked.

String Searching

- Function names may be googled to gain an insight as to what they may do
  - *REGoogle* is an IDA plugin that does this automatically!
- Binaries often have names stripped out of them for performance reasons
  - symbol servers allow debugging symbols to be reinserted into code!

Process Stalking

- Place breakpoints on all functions and basic blocks
- When breakpoint hit
  - log basic block/function to database
    - possibly save CPU state
- Breakpoint hits are tagged
- By merging or diff'ing tagged breakpoint sets can filter out GUI code and locate core code!

Example

- GUI applications tend to use callbacks to relate GUI events to core/interesting code

Breakpoints on all
functions/basic blocks

GUI interactions => breakpoints hit with GUI code + **minimal** non-GUI code

GUI interactions => breakpoints hit with GUI code + **functional** non-GUI code

Subtract the two sets of breakpoints!