Safe to the Last Instruction: Automated Verification of a Type-Safe Operating System

Jean Yang
MIT CSAIL

Chris Hawblitzel
Microsoft Research
Safe to the Last Instruction:

Jean Yang
MIT CSAIL

Chris Hawblitzel
Microsoft Research
Windows

A fatal exception 0E has occurred at 0020:C0011E36 in VXU VMM(O1) + 00010E36. The current application will be terminated.

* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue._
Memory Safety
Type Safety
Previously: “Safe” Systems

- Applications
- Drivers
- File System
- Microkernel

Unsafe code
(GC, stacks, drivers, ...)

Hardware

What currently exists
End-to-End Safe Systems

What we want

Verified code
(GC, stacks, drivers, ...)

Hardware

Microkernel

File System

Drivers

Applications

Type-checked OS

Untyped

Safe to the Last Instruction / Jean Yang
Verve, a Type-Safe OS

- Verify partial correctness of low-level Nucleus using Hoare logic based on a hardware spec.
- Verify an interface to typed assembly for end-to-end safety.
The Verve Nucleus

Interface specification

GC Heap
- Allocator and GC [POPL 2009]
- Stacks

Interrupt table
- Interrupt/error handling

x86 instructions
- Memory bounds

Devices
Thread Context Invariant

function StateInv
(s:StackID, state:StackState, ...)
returns (bool) {
    (!IsEmpty(state) → ...
    && (IsInterrupted(state) → ...
    && (IsYielded(state) → ...
    && state == StackYielded(
        StackEbp(s, tMems)
    , StackEsp(s, tMems) + 4
    , StackRA(s, tMems, fMems)) && ... 
}

Safe to the Last Instruction / Jean Yang
procedure Load(ptr:int)

returns (val:int);

requires memAddr(ptr);

requires Aligned(ptr);

modifies Eip;

ensures word(val);

ensures val == Mem[ptr];
Assembling Verve

- Source file
- Verification tool
- Compilation tool

Nucleus.bpl (x86) → Boogie/Z3 → Translator/Assembler → Verified

Safe to the Last Instruction / Jean Yang
Boogie to x86

implementation ReadKeyboard()
{
    call KeyboardStatusIn8();
    call eax := And(eax, 1);
    if (eax != 0) { goto proc; }
    call eax := mov(256);
    return;

    proc:
    call KeyboardDataIn8();
    call eax := And(eax, 255);
    return;
}

ReadKeyboard proc
    in al, 064h
    and eax, 1
    cmp eax, 0
    jne ReadKeyboard$proc
    mov eax, 256
    ret

ReadKeyboard$skip:
    in al, 060h
    and eax, 255
    ret

Safe to the Last Instruction / Jean Yang
Building Verve

Source file
Verification tool
Compilation tool

Kernel.cs

C# compiler

Kernel.obj (x86)

TAL checker

Linker/ISO generator

Verve.iso

Nucleus.bpl (x86)

Boogie/Z3

Translator/Assembler

Safe to the Last Instruction / Jean Yang
## Verve Performance

<table>
<thead>
<tr>
<th>Verve functionality</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round-trip yield</td>
<td>98</td>
</tr>
<tr>
<td>Round-trip wait + signal</td>
<td>216</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparisons</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4 (IPC)</td>
<td>224</td>
</tr>
<tr>
<td>SeL4 (IPC)</td>
<td>448</td>
</tr>
<tr>
<td>Singularity (yield)</td>
<td>2156</td>
</tr>
<tr>
<td>Linux (yield)</td>
<td>2390</td>
</tr>
<tr>
<td>Windows (yield)</td>
<td>3554</td>
</tr>
</tbody>
</table>
## Low Annotation Burden

<table>
<thead>
<tr>
<th></th>
<th>Copying</th>
<th>Mark-sweep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification Boogie lines</td>
<td></td>
<td>1185</td>
</tr>
<tr>
<td>Verified Boogie lines</td>
<td>3x code</td>
<td>4309</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4854</td>
</tr>
<tr>
<td>x86 instructions</td>
<td>1377</td>
<td>1489</td>
</tr>
</tbody>
</table>

9 person-months

Safe to the Last Instruction / Jean Yang
Verve vs. Sel4?

- **Sel4**
  - Verified microkernel
  - 8,700 lines of C
  - ~600 lines ARM assembly
  - 200,000 lines of Isabelle
  - 120-240 person-months

- **Verve**
  - Verified Nucleus
  - ~1500 lines of x86
  - C# kernel
  - 20x code
Contributions

- First **automatically, mechanically verified** OS for **type safety**.
- **Real system** running on x86 with **efficient code**.
- Approach for using **automated techniques** to verify safety.

http://www.codeplex.com/singularity