Correct Compilation to WebAssembly

Ross Tate

Compiling to WebAssembly

```
extern void launch_missiles();

extern void foo_checkin();
extern void bar_checkin();

void (*mutable_global)() = &launch_missiles;

void benign() {}
void foo_internal(void (**func_ptr_ptr)()) { foo_checkin();}
/*export*/ void foo() {
   void (*func_ptr)() = &benign;
   foo_internal(&func_ptr)
   (*func_ptr)();
}

void bar_internal(int *int_ptr) { bar_checkin();}
/*export*/ int bar(int input) { bar_internal(&input); return input;}
```

Uses shadow stack

- Compile to wasm module
  - Imports $launch_missiles, $foo_checkin, and $bar_checkin: [] -> []
  - Exports $foo: [] -> [] and $bar: [i32] -> []
- Litmus test for correct compilation:
  - launch_missiles cannot be called using module

First Questions

Can imports/environment access unexported memory?

Can imports/environment access unexported functions?

Launching Missiles

JS: calls foo

foo calls foo_internal
  with &benign on shadow stack

foo_internal calls foo_checkin

JS: foo_checkin calls bar with &launch_missiles
  (&launch_missiles is just some i32)

bar calls bar_internal
  with &launch_missiles on shadow stack

bar_internal calls bar_checkin

JS: bar_checkin throws a trap
  bar_internal and bar popped off wasm stack

JS: foo_checkin catches trap and returns

foo_internal calls (*func_ptr)()
  value of func_ptr is taken from shadow stack
  current leaf of shadow stack is &launch_missiles
  call_indirect's to launch_missiles

```
extern void launch_missiles();

extern void foo_checkin();
extern void bar_checkin();

void (*mutable_global)() = &launch_missiles;

void benign() {}
void foo_internal(void (**func_ptr_ptr)()) { foo_checkin();}
/*export*/ void foo() {
   void (*func_ptr)() = &benign;
   foo_internal(&func_ptr)
   (*func_ptr)();
}

void bar_internal(int *int_ptr) { bar_checkin();}
/*export*/ int bar(int input) { bar_internal(&input); return input;}
```

Uses shadow stack

- Compile to wasm module
  - Imports $launch_missiles: [] -> [] and $checkin: [] -> []
  - Exports $foo: [] -> [] and $bar: [i32] -> []
- Litmus test for correct compilation:
  - launch_missiles cannot be called using module

Fixing Traps

- Add catch_trap and catch_trap_ref
  - (or combine catch_all/catch_trap into catch_everything)
- Enables efficient correct compilation
  - Compiled runtime has complete control of its wasm stack
  - Just like it has complete control over its internal memory
  - Just like it would have as a standard native process