Stack Policies

Ross Tate

WebAssembly Ecosystem

- Policy for interop
  - Free for all
    - no guarantees whatsoever when you call into or out from a wasm function
  - Cooperative
    - mechanisms for enabling interop assuming all parties play along
  - Hierarchical
    - mechanisms respect/enforce some notion of privilege

Running Example

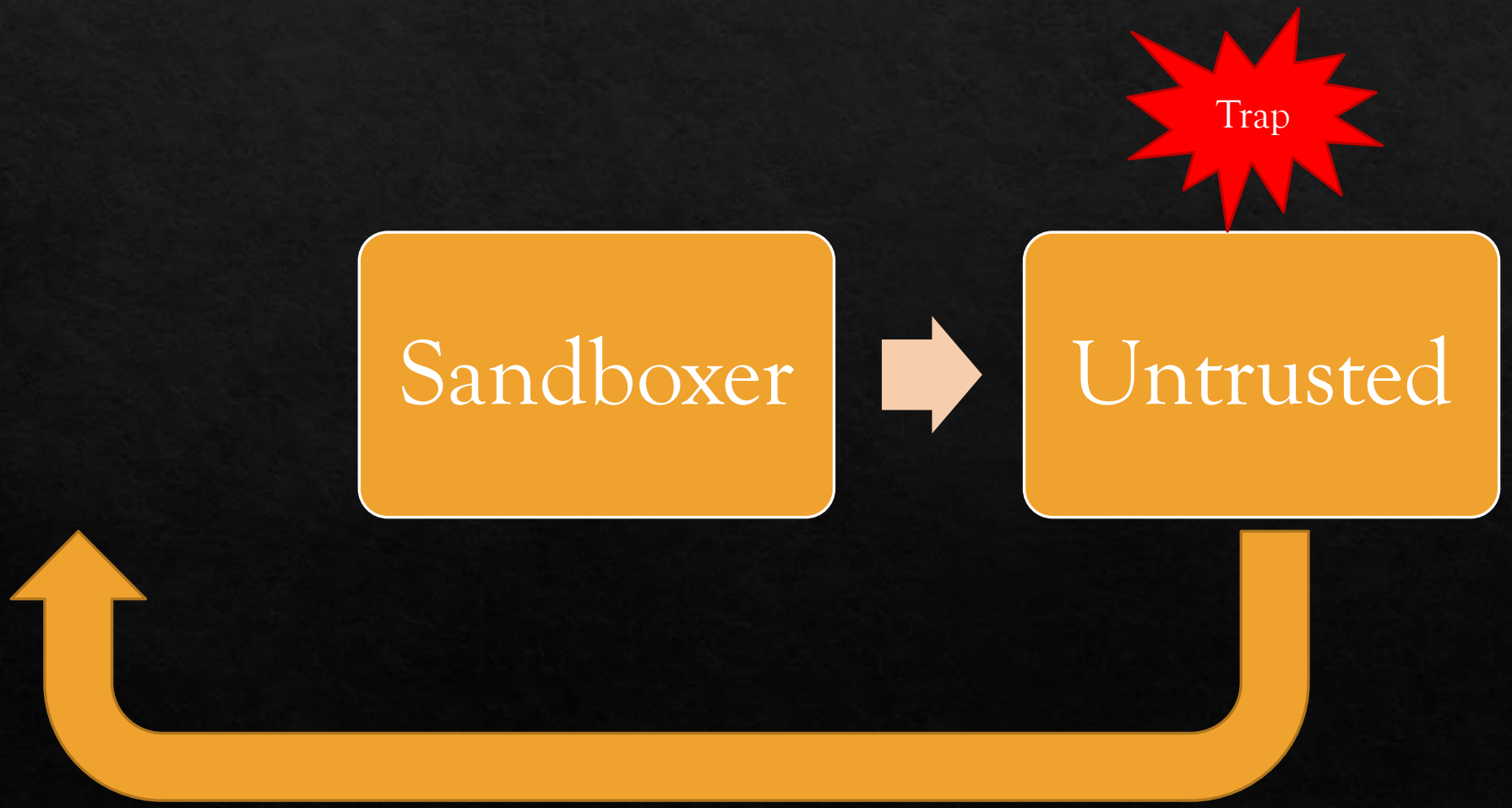
Stack Unwinding

Stack-Unwinding Policies

- Free for All
  - original wasm: no way to unwind other application's stacks or to get your own unwound
- Cooperative
  - mechanism for declaring unwind operations
  - mechanism for triggering unwind operations
  - Is it possible to bypass unwinders? Is it possible to ensure unwinders fire?
- Hierarchical
  - more privileged unwinding clauses cannot be bypassed by less privileged control flow

Sandboxing vs. Unwinding

Sandboxing vs. Trapping



Sandboxing

- How to bypass unwinders of untrusted code but not of sandboxed code?
  - Ideally without resorting to embedder
- Do we need a notion of privilege?
- Can we infer privilege from stack contents?
  - Closer to the root means more privileged
  - Can stack marks declare privilege?
  - “Understanding Java Stack Inspection” by Dan Wallach and Edward Felton in OAKLAND ’98

Questions

- What about first-class stacks?
- How do we maintain the integrity of the event loop?
- What guarantees can we provide applications?
- What mechanisms can we provide to facilitate cooperation between applications?
- What attack patterns will applications need to defend against?
- Is it possible to abstract over policies and yet enforce them?