Compile Jif Programs for Distributed Systems

[ZZNM 01, ZCMZ 03]
Lantian Zheng
Cornell University
zlt@cs.cornell.edu

November 10, 2003
Problem

• Jif: well-typed programs are secure. (Wow!)
• But the execution platform is in TCB.
  – Do we have to trust Microsoft Windows?
  – What if my laptop is stolen?
• Let users decide: blue pill or red pill.
  – Users bear the risks associated with their decisions.
• What if there is no single host trusted by all the participating users (principals) of a program?
Distributed Systems as the Platform: Opportunity and Challenge

- Potential to be more secure
  - Decentralized trustiness
    * Run Alice’s code on Alice’s host, and run Bob’s code on Bob’s host.
  - Boost security: replication, secret sharing.
  - Avoid single point of failure.

- Weaker assumption: partial failure is a given.
  → requires fault detection or tolerance

- Synchronization
Architecture

Jif Source Program

Jif Compiler

Splitter

Distributed System (with trust specifications)

Frontend

Backend

Execution platform
A Simple Example

\[ x = a + b \]

- \( x, a \) and \( b \): trusted by Alice and Bob.
- \( h_1 \): trusted by Alice
- \( h_2 \): trusted by Bob
A Simple Example

\[ x = a + b \]

- \( x, a \) and \( b \): trusted by Alice and Bob.
- \( h_1 \): trusted by Alice
- \( h_2 \): trusted by Bob

\( H \): trusted by Alice and Bob
A Simple Example

\[ x = a + b \]

- \( x, \ a \) and \( b \): trusted by Alice and Bob.
- \( h_1 \): trusted by Alice
- \( h_2 \): trusted by Bob

\( H \): trusted by Alice and Bob
A Simple Example

\[ x = a + b \]

- \( x, a \) and \( b \): trusted by Alice and Bob.
- \( h_1 \): trusted by Alice
- \( h_2 \): trusted by Bob

\( H \): trusted by Alice and Bob
Overview of Code Generation

- **Phase 1:** \([e_1; \ldots; e_n] = e_1@H_1; \ldots; e_n@H_n\)
  - \(H_i\) is trusted to run \(e_i: P(H_i) \leq P(e_i)\)
  - \(H_i\) is a virtual host.
    - provides a hook for applying replication.

- **Phase 2:** \([e_i@H_i] = e_{i1}@h_1 \parallel \ldots \parallel e_{im}@h_m\).

- **Phase 3:** insert calls to the run-time system after \(e_{ij}\)
  - Transfer control between hosts
  - Transfer data between hosts
Virtual Host

- Single host [ZZNM, SOSP 01]
- Simple replication (with hashing) [ZCMZ, Oakland 03]
- Quorum systems [future work]
- Secret sharing [future work]
Security Labels and Hosts

- General security policy: $\{ o : f_1, \ldots, f_n \}$.
  - You can only hurt by friends.
  - Confidentiality labels: $\{ o : r_1, \ldots, r_n \}$.
  - Integrity labels: $\{ o : w_1, \ldots, w_n \}$

- Host labels: the trustworthiness of hosts.
  - E.g. $C(h) = \{ o : A, B \}$ and $I(h) = \{ o : A \}$
Simple Replication with Hash

- Replication increases integrity.
  - Replicate data \( d \) on \( h_1 \) and \( h_2 \).
  - Replicas need to be consistent.
  - \( H = \{h_1, h_2\} \): \( I(H) = I(h_1) \cap I(h_2) \)
  - Sufficient trustiness: \( I(H) \subseteq I(d) \)
  - E.g. \( I(d) = \{o : congress\} \), \( I(h_1) = \{o : senate\} \), \( I(h_2) = \{o : house\} \).

- Replication may jeopardize confidentiality.
  - E.g. \( C(d) = \{o : senate\} \)
  - \( h_1 \leftarrow d \quad h_2 \leftarrow md5(d, nonce) \)
  - \( H = \langle\{h_1, h_2\}, \{h_2\}\rangle \): \( I(H) = I(h_1) \cap I(h_2) \quad C(H) = C(h_1) \)
  - Implicit flow: \( C_{if}(H) = C(h_1) \cap C(h_2) \)
Replicating Computation

- $H = \{h_1, \ldots, h_n\}$
  $\rightarrow [e@H] = e@h_1 \parallel \ldots \parallel e@h_n$

- $H = \langle\{h_1, \ldots, h_n\}, \{h_{i1}, \ldots, h_{im}\}\rangle$
  - If $e$ is $x = y$, then $[e@H] = e@h_1 \parallel \ldots \parallel e@h_n$.
  - Otherwise, $e@H$ cannot be compiled.
Run-time System: Control Transfer (I)

- \( e_1 @ H_1 \rightarrow e_2 @ H_2 \)
  - \( H_1 \) sends a request (run \( e_2 \)) to \( H_2 \).
  - \( H_2 \) checks \( I(H_1) \subseteq I(e_2) \).

- Simple replication: \( H_1 = \{h_1, \ldots, h_n\} \quad H_2 = \{h'_1, \ldots, h'_m\} \)
  - \( h_1, \ldots, h_n \) send the request to \( h'_j \).
  - \( h'_j \) checks \( G_j: \prod_{1 \leq i \leq n} I(h_i) \subseteq I(e_2) \cup I(h'_j) \).

- Correctness: \( G_1 \land \ldots \land G_m \Rightarrow I(H_1) \subseteq I(e_2) \)
Run-time System: Control Transfer (II)

- What if $I(e_1) \nsubseteq I(e_2)$?

- Consider the whole control flow: $\ldots e_0; e'_1; \ldots; e'_m; e_1; e_2$.
  - $I(e_0) \subseteq I(e_2)$ and $\forall i \in [1..m] I(e'_i) \nsubseteq I(e_2)$

1: run $e'_1$, T
2: return with T
• Simple replication: $H_1 = \{h_1, \ldots, h_n\}$
  - Each $h_i$ generates a token $t_i$.
  - $T = \{t_1, \ldots, t_n\}$
  - Return to $h_i$ by presenting $t_i$. 

Partitioning and Replication 15
Conclusion

● Hypothesis: it’s impossible or too expensive to implement a provably secure platform.

● Key ideas:
  – Let users specify the trustworthiness of hosts and take the corresponding risk.
  – Use distributed systems as the platform.
    → analyze and apply existing techniques: replication, secure hashing, nonces...

● Technical contributions: splitter, run-time protocols.