Using Information Flow Policies to Construct Secure Distributed Systems

Andrew Myers
Cornell University

Joint work with: Lantian Zheng, Stephen Chong, Steve Zdancewic, Nate Nystrom

Secure distributed systems?
• Goal: distributed systems that provide information security—confidentiality, integrity
  – Problem 1: hard to control information release
  – Problem 2: (mutual) distrust
• How to build?
  – distributed protocols, encryption, signing…
• How to validate?
• Our goal: programs secure by construction

Example: Battleship
• Two-player game in which each player tries to sink other’s ships
  – “A3”
  – “hit”
• General problem for multiplayer games/simulations: hard to prevent cheating
  – Distrust ⇒ Multiplayer code must change.
• Almost any distributed application contains distrust. auctions, financial transactions, online shopping, military information systems…
• Our goal: compiler transforms code to run securely on untrusted hosts

Secure partitioning

Security for distrusting principals
• Principals vs. hosts
  – Alice trusts hosts A & C
  – Bob trusts hosts B & C

Security Policies in Jif
• Confidentiality labels:
  \[\text{int}(\text{Alice:}) \ a_1; \quad \text{"a}_1\text{ is Alice's private int"}\]
• Integrity labels:
  \[\text{int}(\*:\text{Alice}) \ a_2; \quad \text{"Alice trusts a}_2\text{"}\]
• Combined labels:
  \[\text{int}(\text{Alice: ; } \*:\text{Alice}) \ a_3; \text{ (Both)}\]
• Enforced in Jif language using static information flow analysis:
  int(\text{Alice:}) \ a_1, a_2;
  int(\text{Bob:}) \ b;
  int(\*:\text{Alice}) \ c;
  \text{Insecure: } a_1 = b; \quad b = a_1; \quad c = a_1;
  \text{Secure: } a_1 = a_2; \quad b = a_1; \quad c = a_1;
**Battleship example**

- A's board is confidential to A but must be trusted by both A and B:
  \( \{A: \}; *:A,B \) 
- B's board is symmetrical:
  \( \{B: \}; *:A,B \)

- Secure, automatic placement of board data previously unsolvable
  \[ TOCS'02 \]

---

**Replication**

- **Idea 1:** replicate board onto both hosts so both principals trust the data.

- **Problem:** host B now has A's confidential data.
- **Idea 2:** host B stores a one-way hash of cells
  - Cleartext cells checked against hashed cells to provide assurance data is trusted by both A & B.
  - Compiler automatically generates this solution!

---

**Host labels**

- Trust in hosts described by **host labels**

- Battleship game:
  \( \{A: \}; *:A,B \) 
  \( \{B: \}; *:B \) 

- Data with confidentiality \( C \) and integrity \( I \) can be securely placed on host \( h \) if:
  \( C \cup C_h \) and \( I_h \cup I \)

- A's board:
  \( \{A: \}; *:A,B \) but \( \{A: \} \nsubseteq \{B: \} \) and \( \{*:A \} \nsubseteq \{*:B \} \)

---

**Secure replication condition**

- Data with confidentiality \( C \), integrity \( I \) can be securely placed on hosts \( h_i \) if:
  \( C \subseteq C_{h_j} \) for some host \( h_j \) 
  \( \bigcap_{h_i} I \) (instead of \( I_{h_i} \))

*Example* 

- A's board:
  \( \{A: \}; *:A,B \) 
  \( \{B: \}; *:B \) 

  - Confidentiality: \( \{A: \} \nsubseteq \{A: \} \) 
  - Integrity: \( \{*:A \} \nsubseteq \{*:A,B \} \)

---

**Downgrading in Jif**

**Declassification** (confidentiality)

\[
\text{int}(\text{Bob}; *:\text{Alice}) \ x; \\
y = \text{declassify}(x, \{\text{Bob}; *:\text{Alice}\} \text{ to } \{*:\text{Alice}\})
\]

**Endorsement** (integrity)

\[
\text{int}(\text{Bob}) \ x; \\
y = \text{endorse}(x, \{\text{Bob}\} \text{ to } \{\text{Bob}; *:\text{Alice}\})
\]

- Needed to describe richer confidentiality, integrity policies with intentional information flows
- Requires static authorization
- Requires protection integrity at downgrading point to ensure integrity of unsafe operations (robustness)

---

**Downgrading in Battleship**

- **Declassification:** board location \((i,j)\) not confidential once bomb dropped on it:
  \( \text{loc} = \text{declassify}(\text{board}[\text{move}.i][\text{move}.j], \{A: *:A,B \} \text{ to } \{*:A,B \}) \)

- **Endorsement:** opponent can make any legal move, and can initially position ships wherever desired.
  \( \text{move} = \text{endorse}(\text{move}, \{*:B \} \text{ to } \{*:A,B \}) \)

- \text{declassify}, \text{endorse} often correspond to network data transfers, hash value checks
Replicating computation

- Replicated data → replicated computation
- Computation must be placed on hosts that are trusted to observe, produce data
- Control transfers in original program may become transfers among groups of hosts

Results

- Implemented a variety of small programs in Jif and used Jif/split compiler to compile to distributed systems.
  - Battleship, three secure auction protocols, simple financial transactions, oblivious transfer
  - "Security-intensive", mutual distrust
  - Integrity limitations prevented automatic partitioning of most by original system.
- Implemented same programs with hand-crafted Java/RMI code.
  - Jif versions are 13-65% shorter, but send 2-4× more messages.

Related work

- Language-based security and static information flow
  - mostly ignores distribution, distrust
- Multilevel security and information flow
- Uniform replication for improved integrity and availability
  - replicated state machines, BFT, file systems
- Stack Inspection
  - protects downward control integrity

Program partitioning

- Methodology:
  - programmers write explicit security policies
  - compiler/splitter transform code to satisfy them
  - Input can be whole computation, or just part
- Recent work: automatic replication
  - Easier to obtain integrity assurance
  - Useful for realistic computations
- Mechanisms to make it work:
  - One-way hashing with nonces
  - Capability tokens with secret splitting
  - Two-phase synchronization protocol
  - Consistent global identifier generation
- Results: greater assurance, shorter code

Conclusions

- Methods are needed for obtaining system-wide security assurance in presence of mutual distrust
- Jif/split compiler automatically uses a variety of common techniques to solve secure compilation problems
  - encryption, digital signing, secure one-way hashing, nonces, agreement protocols, commitment protocols
  - Still more possibilities...
- Future work
  - More experience
  - Richer security policies
  - Formal validation
  - Optimization

http://www.cs.cornell.edu/jif

Using Information Flow Policies to Construct Secure Distributed Systems -- Myers