Unified Platform for Secure Networked Information Systems

Duke CPS296.1 Spring10
Xuanran Zong

Background & Motivation

- Underlying network
  - Accountability
  - Efficient packet tracing
  - Flow analysis
- No integration with networked information system
  - Focus on specific threat
  - Different environment

Objective

- Unified Declarative platform
  - Specify
  - Implement
  - Analyze
  - Audit
- Large-scale secure information system

Building blocks

- Logic-based trust management system (Binder)
- Declarative networking (NDlog)
- Data analyze via provenance

Binder

- Query language based Datalog
- Access control in untrusted network
- Context
- ‘Says’
- Example:
  b1 may-access(P,O,read) :- good(P).
  b2 may-access(P,O,read) :-
    bob says may-access(P,O,read).
- Why do they choose Binder?

Binder vs NDlog

<table>
<thead>
<tr>
<th></th>
<th>Binder</th>
<th>NDlog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Assumption</td>
<td>Untrusted</td>
<td>Trusted</td>
</tr>
<tr>
<td>Export of derived tuples</td>
<td>No restriction</td>
<td>Restricted (Location specifies)</td>
</tr>
<tr>
<td>Evaluation Order</td>
<td>Top-down (Why?)</td>
<td>Bottom-up (Why?)</td>
</tr>
</tbody>
</table>
SeNDlog

- Unification of Binder and NDlog
- Features
  - Rules bind to particular node
  - Example
    
    \[
    \text{At } N, \ c_1, c_2, \ldots, c_n
    \]
    
    \[
    r_1 p : p_1, p_2, \ldots, p_n.
    \]
    
    \[
    r_2 p_1 : p_2, p_3, \ldots, p_n.
    \]

SeNDlog

- Communication
  - Explicit control of import and export tuples
  - Import predicate (body): \( N \text{ says } p \)
  - Export predicate (head): \( N \text{ says } p@X \)
  - Why do we need these restriction? Why not just simply use NDlog style?
  - Example
    
    \[
    \text{At } N,
    \]
    
    \[
    e_1 p(X,Y) : p_1(X), p_2(Y).
    \]
    
    \[
    e_2 p(X,Y,W) : X \text{ says } p_1(X), Z \text{ says } p_2(W), Z=\text{N}.
    \]
    
    \[
    e_3 p(Y,Z)@X : p_1(X), Y \text{ says } p_2(Z).
    \]
    
    \[
    e_4 Z \text{ says } p(Y)@X : Z \text{ says } p(Y), p_1(X).
    \]

SeNDlog

- Honesty Constraint
  - \( X \text{ says } p \) in head \( \Rightarrow \) \( X \text{ says } p \) in body
  - Why?

- Extensions
  - Security level: efficiency security tradeoff
  - Is it necessary?

Some SeNDlog Examples

- Authenticated path-vector protocol
  
  \[
  \text{At } Z,
  \]
  
  \[
  z_1 \text{ route}(Z,X,P) : \text{neighbor}(Z,X), P=\text{f_initPath}(Z,X).
  \]
  
  \[
  z_2 \text{ route}(Z,Y,P) : X \text{ says } \text{advertise}(YP),
  \]
  
  \[
  \quad \quad \quad \text{acceptRoute}(Z,Y,P).
  \]
  
  \[
  z_3 \text{ advertise}(YP)@X : \text{neighbor}(Z,X), \text{route}(Z,Y,P),
  \]
  
  \[
  \quad \quad \quad \text{carryTraffic}(Z,Y,P), P_1=\text{f_concat}(X,P).
  \]
  
- Can also implement BGP, P2P, CDN

Another Example

- Secure Chord DHT
  
  \[
  \text{At } NI,
  \]
  
  \[
  ni_1 \text{ requestCert}(NI,K)@CA : \text{startNetwork}(NI),
  \]
  
  \[
  \quad \quad \quad \text{publicKey}(NI,K), \text{MyCA}(NI,CA).
  \]
  
  \[
  ni_2 \text{ nodeID}(NI,N) : CA \text{ says } \text{nodeIDCert}(NI,N,K)
  \]
  
  \[
  ni_3 CA \text{ says } \text{nodeIDCert}(NI,N,K)@LI : \text{CA says }
  \]
  
  \[
  \quad \quad \quad \text{nodeIDCert}(NI,N,K), \text{landmark}(NI,LI).
  \]
  
  \[
  \text{At CA},
  \]
  
  \[
  ca_1 \text{ nodeIDCert}(NI,N,K)@NI : NI \text{ says } \text{requestCert}(NI,K),
  \]
  
  \[
  \quad \quad \quad \text{S=secret}(CA,NI), N=\text{fGenerateID}(K,S).
  \]
  
  \[
  \text{At LI},
  \]
  
  \[
  li_1 \text{ acceptJoinRequest}(NI) : CA \text{ says nodeIDCert}(NI,N,K).
  \]

One more example

- Secure DHT-based join processing
  
  \[
  \text{At alice,}
  \]
  
  \[
  a_1 \text{ storeA}(X,Y)@NI : \text{tableA}(XY), K=\text{f_sha}(X),
  \]
  
  \[
  NI=\text{Chord::X}.
  \]
  
  \[
  \text{At bob,}
  \]
  
  \[
  b_1 \text{ storeB}(X,Y)@NI : \text{tableB}(XY), K=\text{f_sha}(X),
  \]
  
  \[
  NI=\text{Chord::X}.
  \]
  
  \[
  \text{At NI,}
  \]
  
  \[
  r_1 \text{ results}(XY)@r : alice \text{ says storeA}(X,Y),
  \]
  
  \[
  bob \text{ says storeB}(Y,Z).
  \]
  
- One more layer of authentication
Secure Query Processing

- Pipelined semi-naive evaluation (PSN)
  - Asynchronous
- Authenticated PSN
  - Two more operator: SigGenerator, SigChecker
  - Example
    
    2a. \( \Delta \text{route}(Z, Y, P) : \Delta \text{advertise}(Y, P), \Delta \text{acceptRoute}(Z, X, Y). \)
    
    2b. \( \Delta \text{route}(Z, Y, P) : \Delta \text{advertise}(Y, P), \Delta \text{acceptRoute}(Z, X, Y). \)

Layering support and security extensions

- Extract payload, discard multiple headers
- LocSpecDemux to support local overlay dataflow
- Security extension
  - Optional attribute to SigGenerator and SigChecker

Network Provenance

- Capture the how each predicate is derived
  - Diagnosis, forensics, trust management
- Naturally fit the bottom-up evaluation
- Local vs Distributed
- Online vs Offline

Network Provenance

- Authenticated provenance

Evaluation

- Comparing the performance between auth and non-auth
- Results
  - Authentication introduces latency, especially in PlanetLab
  - Auth + Provenance doubles the completion time
  - Small message transfer tends to have negligible latency overhead (Chord DHT)
  - Bandwidth intensive query tends to have more overhead (best path)

Discussion point

- Untrusted node?
- Query optimization?
- Compilation overhead? Compilation efficiency?