DS²: Declarative Secure Distributed Systems

Boon Thau Loo
University of Pennsylvania

Joint work with Wenchao Zhou, Bill Marczak, Micah Sherr, Mengmeng Liu, Matt Blaze, Zack Ives,

External collaborators: Martín Abadi (MSR), Yun Mao (AT&T), LogicBlox Inc.

This work is partially supported by NSF grant IIS-0812270 and CNS-0831376.
Motivation

- **Proliferation of new network architecture and protocols**
  - Overlay networks with new capabilities
    - Mobility, resiliency, anycast, multicast, anonymity, etc
  - Distributed data management applications
    - Network monitoring, publish-subscribe systems, content-distribution networks

- **Challenges - scalability and security threats**

- **Techniques proposed by security/networking community**
  - **Distributed debugging:** PIP [NSDI 06], FRIDAY [NSDI 07]
  - **Accountability:** IP traceback [SIGCOMM 00], IP forensics [ICNP 06], AIP [SIGCOMM 08]
  - **Distributed trust management:** SD3 [Oakland 01], Delegation Logic [TISSEC 03], Network capabilities [Hotnets’03]
Motivation

- **Problem: lacking generalized framework**
  - Designed for specific security threats
  - Implemented and enforced in different languages and environments
  - Lack of cross-layer integration (networks and higher layers)

- **Overall goal:**
  - Extensible platform for specifying and implementing *distributed systems* and their *security policies*
  - Support for a variety of existing and enable new *analysis techniques*
Outline of Talk

- Background: declarative networking and access control logic
- Unified declarative platform for secure distributed systems [ICDE’09]
- Network provenance [NetDB’08, CCS ’09 submission]
- Reconfigurable trust management [CIDR ’09]
- Other research highlights (http://netdb.cis.upenn.edu/)
Background: Declarative Network

Traditional Networks
- Network State
- Network protocol
- Network messages

Declarative Networks
- Distributed database
- Recursive Query Execution
- Distributed Dataflow
Background: Declarative Networking

- **Declarative query language for network protocols**
  - Network Datalog (NDlog) – distributed Datalog [SIGCOMM ’05, SIGMOD ’06]
  - Compiled to distributed dataflows, executed by distributed query engine
  - *Location specifiers* (@ symbol) indicate the source/destination of messages

- **Example: Network Reachability**

  \[ r1: \text{reachable}(\text{@}S, D) \leftarrow \text{link}(\text{@}S, D) \]

  \[ r2: \text{reachable}(\text{@}S, D) \leftarrow \text{link}(\text{@}S, Z), \text{reachable}(\text{@}Z, D) \]

  \( \text{link}(\text{@}a, b) \) – “there is a link from node \text{a} to node \text{b}”

  \( \text{reachable}(\text{@}a, b) \) – “node \text{a} can reach node \text{b}”

  If there is a link from \text{S} to \text{D}, then \text{S} can reach \text{D}.

  If there is a link from \text{S} to \text{Z}, AND \text{Z} can reach \text{D}, then \text{S} can reach \text{D}.

<table>
<thead>
<tr>
<th>Node a</th>
<th>Node b</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{link}(\text{@}a, b) )</td>
<td>( \text{link}(\text{@}b, c) )</td>
</tr>
<tr>
<td>( \text{link}(\text{@}a, c) )</td>
<td>( \text{reachable}(\text{@}b, c) )</td>
</tr>
<tr>
<td>( \text{reachable}(\text{@}a, c) )</td>
<td></td>
</tr>
</tbody>
</table>
Path Vector in Network Datalog

R1: \text{path}(\text{@S}, \text{D}, \text{P}) \leftarrow \text{link}(\text{@S}, \text{D}), \text{P}=(\text{S}, \text{D}).

R2: \text{path}(\text{@S}, \text{D}, \text{P}) \leftarrow \text{link}(\text{@S}, \text{Z}), \text{path}(\text{@Z}, \text{D}, \text{P}_2), \text{P}=\text{S} \bullet \text{P}_2.

Query: \text{path}(\text{@S}, \text{D}, \text{P}) \quad \text{Add S to front of P}_2

\text{Input: link(}@\text{source, destination})

\text{Query output: path(}@\text{source, destination, pathVector})
Large Library of Declarative Protocols

- Example implementations to date:
  - **Routing protocols:** DV, LS, DSR, AODV, OLSR, HSLS, etc.
  - **Overlay networks:** Distributed Hash Tables, Resilient overlay network (RON), Internet Indirection Infrastructure (i3), P2P query processing, multicast trees/meshes, etc.
  - **Network composition:** Chord over RON, i3+RON
  - **Hybrid protocols:** Combining LS and HSLS
  - **Others:** sensor networking protocols, replication, snapshot, fault tolerance protocols
Background: Access Control

- Central to security, pervasive in computer systems
- Broadly defined as:
  - Enforce security policies in a multi-user environment
  - Assigning credentials to principals to perform actions
  - Commonly known as trust management
- Model:
  - objects, resources
  - requests for operations on objects
  - sources for requests, called principals
  - a reference monitor to decide on requests

Principal $\rightarrow$ Do operation $\rightarrow$ Reference Monitor (‘guard’) $\rightarrow$ Object
Background: Access Control

- **Access control languages:**
  - *Analyzing* and *implementing* security policies
  - Several runtime systems based on distributed Datalog/Prolog

- **Binder [Oakland 02]: a simple representative language**
  - **Context:** each principal has its own context where its rules and data reside
  - **Authentication:** “says” construct (digital signatures)
    
    At alice:
    
    b1: access(P,O,read) :- good(P).
    b2: access(P,O,read) :- bob says access(P,O,read).

    “In alice's context, any principal P may access object O in read mode if P is good (b1) or, bob says P may do so (b2 - delegation)”

- **Several languages and systems:** Keynote [RFC-2704], SD3 [Oakland 01], Delegation Logic [TISSEC 03], etc.
Comparing the two

- Declarative networking and access control languages are based on logic and Datalog.
- Similar observation:
  - Martín Abadi. “On Access Control, Data Integration, and Their Languages.”
  - Comparing data-integration and trust management languages.
- Both extends Datalog in surprisingly similar ways:
  - Notion of context (location) to identify components (nodes) in a distributed system.
  - Suggests possibility to unify both languages.
  - Leverage ideas from database community (e.g. efficient query processing and optimizations) to enforce access control policies.
- Differences:
  - Top-down vs bottom-up evaluation.
  - Trust assumptions.
Outline of Talk

- Background: declarative networking and access control logic
- Unified declarative platform for secure distributed systems [ICDE’09]
- Network provenance [NetDB’08, CCS ’09 submission]
- Reconfigurable trust management [CIDR ’09]
- Other research highlights (http://netdb.cis.upenn.edu/)
Secure Network Datalog (SeNDlog)

- Rules within a context
  - Untrusted network
  - Predicates in rule body in local context
- Authenticated communication
  - “says” construct
  - Export predicate: “X says p@Y”
    - X exports the predicate p to Y.
  - Import predicate: “X says p”
    - X asserts the predicate p.

```
  r1: reachable(@S,D) :- link(@S,D).
  r2: reachable(@Z,D) :- link(@S,Z),
                   reachable(@Z,D).

  \[ \text{localization rewrite} \]

  At S:
  s1: reachable(@S,D) :- link(@S,D).
  s2: linkD(D,S)@D :- link(S,D).
  s3: reachable(Z,D)@Z :- linkD(@S,Z),
      reachable(@S,D).

  \[ \text{authenticated communication} \]

  At S:
  s1: reachable(@S,D) :- link(@S,D).
  s2: S says linkD(D,S)@D :- link(S,D).
  s3: S says reachable(Z,D)@Z :-
      Z says linkD(@S,Z),
      W says reachable(@S,D).
```
Example Protocols in SeNDlog

- **Secure network routing**
  - Nodes import/export signed route advertisements from neighbors
  - Advertisements include signed sub-paths (*authenticated provenance*)
  - Building blocks for secure BGP

- **Secure packet forwarding**

- **Customizable anonymous routing**
  - Path selection and setting up “onion paths” with layered encryption
  - Application-aware Anonymity ([http://a3.cis.upenn.edu](http://a3.cis.upenn.edu))

- **Secure DHTs**
  - Chord DHT – authenticate the node-join process
  - Signed node identifiers to prevent malicious nodes from joining the DHT

- **P2P query processing – application layer**
  - PIER - built upon Chord DHT
  - Capability of *layered authentication*
Authenticated Path Vector Protocol

At Z,

- \( z_1 \_route(Z,X,P) :\) neighbor\((Z,X)\), \( P=f\_initPath(Z,X) \).
- \( z_2 \_route(Z,Y,P) :\) X says advertise\((Y,P)\), acceptRoute\((Z,X,Y)\).
- \( z_3 \_advertise(Y,P1@X) :\) neighbor\((Z,X)\), route\((Z,Y,P)\), carryTraffic\((Z,X,Y)\), \( P1=f\_concat(X,P) \).

- **Import and export** policies
- **Basis for Secure BGP**
  - Authenticated advertisements
  - Authenticated subpaths (provenance)
  - Encryption (for secrecy) with cryptographic functions
Authenticated Path Vector Protocol

At Z,
\[ z_1 \text{ route}(Z,X,P) \rightarrow \text{neighbor}(Z,X), \ P = f\text{\_initPath}(Z,X). \]
\[ z_2 \text{ route}(Z,Y,P) \rightarrow X \text{ says advertise}(Y,P), \ \text{acceptRoute}(Z,X,Y). \]
\[ z_3 \text{ advertise}(Y,P_1)@X \rightarrow \text{neighbor}(Z,X), \ \text{route}(Z,Y,P), \]
\[ \text{carryTraffic}(Z,X,Y), \ P_1 = f\text{\_concat}(X,P). \]

\[ p(@a,d,[a,b,c,d]) \quad p(@b,d,[b,c,d]) \quad p(@c,d,[c,d]) \]

\[ a \quad b \quad c \quad d \]

b says advertise(d,[a,b,c,d]) c says advertise(d,[b,c,d])
Authenticated Query Processing

- **Semi-naïve Evaluation**
  - Standard technique for processing recursive queries
  - Synchronous rounds of computation

- **Pipelined Semi-naïve Evaluation** [SIGMOD 06]
  - Asynchronous communication in distributed setting
  - No requirement on expensive synchronous computation

- **Authenticated Semi-naïve Evaluation**
  - Modification for “says” construct, in p’s context:
    
    \[
    a : - d_1, \ldots, d_n, b_1, \ldots, b_m, p_1 \text{ says } a_1, \ldots, p_k \text{ says } a_k, \ldots, p_o \text{ says } a_o.
    \]

  for kth import predicate, an authenticated delta rules is generated:

  \[
  p \text{ says } \Delta a : - d_1, \ldots, d_n, b_1, \ldots, b_m, p_1 \text{ says } a_1, \ldots, p_k \text{ says } \Delta a_k, \ldots, p_o \text{ says } a_o.
  \]
Architectural Overview of Dataflow

- **Dataflow Architecture**
  - Based on the P2 declarative networking system
  - Additional modules to support authenticated communication

s3: S says reachable(Z,D)@Z :- Z says linkD(@S,Z), W says reachable(@S,D).
Experimental Setup

- **P2 declarative networking system**
  - Extensions for security and provenance support

- **Workload**
  - Path-vector – network routing
  - Chord – distributed hash table
  - PIER – p2p query processing

- **Test-bed**
  - A local cluster with 16 quad-core machines
  - Planetlab testbed with 80 nodes

- **Metrics**
  - Communication overhead
  - Query completion time / lookup latency
Authentication Overheads

Path-vector protocol
- 128 nodes, 6 neighbors per node
- Auth-HMAC – 10% increase
- Auth-RSA512 – 20% increase
- Auth-RSA1024 – 40% increase

Chord DHT protocol
- 128 Chord nodes, random lookups
- Auth (with RSA1024) – less than 10% increase to finish 50% lookups

Proof-of-concept: a variety of secure network protocols with acceptable performance overhead

Tunable tradeoff between security and performance
Outline of Talk

- Background: declarative networking and access control logic
- Unified declarative platform for secure distributed systems [ICDE’09]
- Network provenance [NetDB’08, CCS ’09 submission]
- Reconfigurable trust management [CIDR ’09]
- Other research highlights (http://netdb.cis.upenn.edu/)
Network Provenance

- Naturally captured within declarative framework
- Explain the existence of any network state
- Similar notion in security community: *proof-trees*
Optimizing network provenance

- Two types of provenance: *local* and distributed
- Local provenance is expensive to maintain, relatively cheap to query
  - Tag entire derivation with each tuple
  - Can we make it more bandwidth efficient?
- Distributed provenance is expensive to query, cheap to maintain
  - Ongoing work: query on-demand and caching

- Modularization:
  - Combine common subtrees within a single provenance tree or across trees

- Store a compressed provenance structure
  - Binary decision diagrams (BDDs)
  - Sufficient for certain types of queries
  - Sacrifices some information for compactness
Binary Decision Diagrams [Bryant 86]

- Highly optimized libraries available: e.g. JavaBDD.

Boolean expression: \( x_1 \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \)
Compressed Provenance

- Compress the size of local provenance
  - Provenance semirings [PODS’07] annotates provenance in Boolean expressions
  - + means union, * means join
  - BDD encodings for compression

- Compressed:
  - Retain sufficient information for trust management.
  - Node-level provenance
  - Consider <a+a*b>, derivation reachable (a,c) is accepted as long as principal a is trusted
  - Principal b is inconsequential

Liu, Taylor, Zhou, Ives, Loo. Recursive Computation of Regions and Connectivity in Networks. [ICDE ’09]
Experimental Results

- Computing all-pairs shortest path cost.
- Modularization (Prov-Tree): 90% reduction in execution time over Prov-Naïve
- BDD (Prov-BDD): Additional 60% reduction in execution time
Wide Application of Network Provenance

<table>
<thead>
<tr>
<th>Provenance Taxonomy</th>
<th>Distributed Debugging</th>
<th>Accountability</th>
<th>Trust Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Derivation Tree / Algebra Expr.</td>
<td>Tree</td>
<td>Tree</td>
<td>Both</td>
</tr>
<tr>
<td>Local / Distributed</td>
<td>Both</td>
<td>Both</td>
<td>Local</td>
</tr>
<tr>
<td>Boolean/Quantifiable</td>
<td>Both</td>
<td>Boolean</td>
<td>Both</td>
</tr>
</tbody>
</table>

- Distributed debugging: PIP [NSDI 06], FRIDAY [NSDI 07]
- Accountability: IP traceback [SIGCOMM 00], AIP [SIGCOMM 08], IP forensics [ICNP 06]
- Distributed trust management: SD3 [Oakland 01], Delegation Logic [TISSEC 03]

Information hiding in provenance

At Z,

- \( sp1 \text{ pathCost}(S,D,C) :- \text{link}(S,D,C). \)
- \( sp2 \text{ pathCost}(D,Z,C1+C2)@D :- \text{link}(S,D,C1), \text{bestPathCost}(S,Z,C2). \)
- \( sp3 \text{ bestPathCost}(S,D,\text{min}<C>) :- \text{W says pathCost}(S,D,C). \)

*Miklau and Suciu. Controlling access to published data using cryptography. VLDB 03.*
Outline of Talk

- Background: declarative networking and access control logic
- Unified declarative platform for secure distributed systems [ICDE’09]
- Network provenance [NetDB’08, CCS ’09 submission]
- Reconfigurable trust management [CIDR ’09]
- Other research highlights (http://netdb.cis.upenn.edu/)
(Non-Exhaustive) Survey of Trust Management Languages

<table>
<thead>
<tr>
<th></th>
<th>Authentication</th>
<th>Delegation</th>
<th>Conditional Re-Delegation</th>
<th>Threshold Structures</th>
<th>Type System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aura</td>
<td>Y</td>
<td>Y*</td>
<td>Y</td>
<td>Y?</td>
<td>Y</td>
</tr>
<tr>
<td>Binder</td>
<td>Y</td>
<td>Y*</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Cassandra</td>
<td>Y</td>
<td>Y*</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>D1LP</td>
<td>Y</td>
<td>Y</td>
<td>Y (depth/width)</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>KeyNote</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>SD3</td>
<td>Y</td>
<td>Y*</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SeNDLog</td>
<td>Y</td>
<td>Y*</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>SPKI/SDSI</td>
<td>Y</td>
<td>Y*</td>
<td>Y (boolean)</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

- Problem: many languages, features, separate runtime systems, hard to compare and reuse
- Our goal: A unified declarative framework to enable all of these languages
LBTrust: Reconfigurable Trust Management

- Constraints: type safety, program correctness, security
- Meta-programmability.
  - Meta-model: rules as data [VLDB 08]
  - Meta-rules (code generation)
  - Meta-constraints (constraint + reflection)
- Customizable partitioning, distribution, and communication
- Extensible predicates for cryptographic primitives
- Developed using LogicBlox (http://www.logicblox.com), a commercial Datalog engine
Constraints and Types

\[
\text{fail}() \leftarrow \text{access}(P,O,M), \neg \text{principal}(P).
\]

“let fail() whenever access(P,O,M) and not principal(P)”

\[
\text{access}(P,O,M) \rightarrow \text{principal}(P).
\]

“whenever access(P,O,M), require principal(P)”

\[
\text{access}(P,O,M) \rightarrow \text{principal}(P), \text{object}(O), \text{mode}(M).
\]

type constraint
Meta-Model Schema

\[
\begin{align*}
\text{rule}(R) & \rightarrow . \\
\text{active}(R) & \rightarrow \text{rule}(R). \\
\text{head}(R,A) & \rightarrow \text{rule}(R), \text{atom}(A). \\
\text{body}(R,A) & \rightarrow \text{rule}(R), \text{atom}(A). \\
\text{atom}(A) & \rightarrow . \\
\text{functor}(A,P) & \rightarrow \text{atom}(A), \text{predicate}(P). \\
\text{arg}(A,I,T) & \rightarrow \text{atom}(A), \text{int}(I), \text{term}(T). \\
\text{negated}(A) & \rightarrow \text{atom}(A). \\
\text{term}(T) & \rightarrow . \\
\text{variable}(X) & \rightarrow \text{term}(X). \\
\text{vname}(X,N) & \rightarrow \text{variable}(X), \text{string}(N). \\
\text{constant}(C) & \rightarrow \text{term}(C). \\
\text{value}(C,V) & \rightarrow \text{constant}(C), \text{string}(V). \\
\text{predicate}(P) & \rightarrow . \\
\text{pname}(P,N) & \rightarrow \text{predicate}(P), \text{string}(N).
\end{align*}
\]

ensures rules are well-structured
Rules as Data

foo(x) ← bar(x).

“let foo(x) whenever bar(x)”
Meta Rules for Security

- **Meta**
  - Code generation (insert new rules that must be evaluated)
  - Reflection (query for program structure)

- **Meta-Syntax**
  - Embedded rule/bounded constants (~P2 and ~P1)

```
active([\l active(R) \leftarrow says(~P2, ~P1, R). I]) \leftarrow delegates(P1, P2).
```

“activate a rule active(R) \leftarrow says(P2, P1, R). for every delegates(P1, P2).”
A Concrete Example: The “ Says” Authentication Construct

\[
\begin{align*}
says(P1, P2, R) &\rightarrow prin(P1), prin(P2), rule(R). \\
rulesig(R, S) &\rightarrow rule(R), string(S). \\
rsapubkey(P, K) &\rightarrow prin(P), string(K). \\
rsaprivkey(P, K) &\rightarrow prin(P), string(K). \\
says(bob, alice, R).
\end{align*}
\]

\[
\begin{align*}
r1: \text{rulesig}(R, S) &\leftarrow \text{says}(P1, _, R), \\
&\text{rsapubkey}(P1, K), \\
&\text{rsasign}(R, S, K).
\end{align*}
\]\n
\[
\begin{align*}
r2: \text{says}(P1, _, R), \\
&\text{rsapubkey}(P1, K), \\
\end{align*}
\]
Delegation (Basic)

alice “speaks-for” bob == “if alice says something, bob says it too.”

speaks-for is a special form of delegation:

dele

g

gates(P1,P2) → prin(P1), prin(p2).

ds

egates(bob,alice).

“i will believe (i.e. say)

any rule that alice says”

r1: active([ | active(R) ← says(P2,P1,R). | ]) ← delegates(P1,P2).

r2: active(R) ← says(alice,bob,R).
Other cool features


- **Conditional Delegations:**
  - Constraint by width, depth, or predicates
  - Detecting delegation violations (use of provenance)
- **Customizable distribution/partitioning policies**
  - Partition data and rules by principals
  - Distribute principals across machines
- **Same security policy rules can run in local/distributed environment**
  - Use meta-rules to rewrite top-down access control to execute in a bottom-up evaluation engine
- **Example languages:**
  - Binder, Delegation logic, D1LP,
  - Secure Network Datalog [ICDE 09],
- **Usage:** Authenticated routing protocols, access control in distributed databases, distributed file systems
Summary of Contributions

- **Key ideas:**
  - Declarative framework for networks and security specifications
  - Authenticated query processing techniques for distributed settings
  - Network provenance: usage, maintenance and optimizations
  - LBTrust: Distributed reconfigurable trust management

- **Future Work**
  - Optimizing network provenance maintenance and querying
    - Performance / security tradeoff, distributed provenance
  - Applications:
    - Extensible secure routing ([http://a3.cis.upenn.edu](http://a3.cis.upenn.edu))
    - Securing cloud data (multi-user across network administrative domains)
  - Verification
Relevant Publications

http://www.cis.upenn.edu/~boonloo/pubs.html

- **Recursive Computation of Regions and Connectivity in Networks.**
  Mengmeng Liu, Nicholas E. Taylor, Wenchao Zhou, Zachary Ives, and Boon Thau Loo. 25th International Conference on Data Engineering (ICDE), Apr 2009.

- **Unified Declarative Platform for Secure Networked Information Systems.**
  Wenchao Zhou, Yun Mao, Boon Thau Loo, and Martín Abadi. 25th International Conference on Data Engineering (ICDE), Apr 2009.

- **Declarative Reconfigurable Trust Management.**

- **Provenance-aware Secure Networks.**
  Wenchao Zhou, Eric Cronin and Boon Thau Loo. 4th International Workshop on Networking meets Databases (NetDB), Apr 2008.

- **Scalable Link-Based Relay Selection for Anonymous Routing.**
  Micah Sherr, Matt Blaze, and Boon Thau Loo. 9th Privacy Enhancing Technologies Symposium (PETS), Aug 2009.
Other Research Highlights

- **DAWN: Declarative Adaptive Wireless Networks**
  - In collaboration with BBN Technologies under the DARPA Wireless Networks After Next (WNaN) program
  - Deployment on Orbit wireless testbed
  - SIGCOMM ‘09 demonstration (Declarative toolkit integrated with ns-3)

- **Verifiable networking**
  - Combining theorem proving, model checking and declarative network verification/synthesis [PADL’09, TPHOL’09, AFM’09]

- Visit [http://netdb.cis.upenn.edu](http://netdb.cis.upenn.edu) for more details! 😊
Thank You …

http://netdb.cis.upenn.edu