

CS590U

Access Control: Theory and Practice

Lecture 7 (September 23rd)
Trust Management: SPKI and Logic-based Semantics for TM Languages

The Trust-Management (TM) Approach

- Multicentric access control using delegation
 - access control decisions are based on distributed policy statements issued by multiple principals
- Common characteristics of TM systems:
 - treat public keys as principals
 - use digitally signed credentials

2

Defining a TM Language

- Syntax for policy statements and queries
- Semantic relation \blacktriangleright (proves)
 - Given a set P of policy statements and a query Q, $P \blacktriangleright Q$ means that Q is true given P
 - when Q corresponds to an access request, the request should be authorized in the state P
 - \blacktriangleright defines the notion of proof-of-compliance

3

Automated Trust Negotiation (Interactive Deduction)

- Credentials may contain sensitive information
 - need protection just as other resources
 - deduction must be interactive

4

The PolicyMaker TM Language

- Policy statements (Assertions)
 - *Source ASSERTS AuthorityStruct WHERE Filter*
- Queries
 - *key₁, key₂, ..., key_n REQUESTS ActionString*
- Semantics relation \blacktriangleright
 - find a delegation chain

5

The KeyNote TM Language

- Queries:
 - Requesting principals
 - The action attribute set
 - The set of compliance values
- Policy statements (assertions):
 - $\langle \text{AuthField} \rangle$, $\langle \text{LicenseesField} \rangle$, $\langle \text{ConditionsField} \rangle$

6

History of SPKI/SDSI

- SDSI (Simple Distributed Security Infrastructure)
 - SDSI 1.0 and 1.1 (Rivest & Lampson'96)
- SPKI (Simple Public Key Infrastructure)
 - SPKI 1.0 (Ellison 1996)
- SPKI/SDSI 2.0
 - RFC 2693 [1999]
 - [Clarke et al. JCS'01]

7

Work on Semantics of SPKI/SDSI (1)

- Develop specialized modal logics
 - Abadi [CSFW-10, JCS'98]
 - Halpern & van der Meyden
 - LLNC [CSFW-12, JCS'01]
 - Extension of LLNC to handle SPKI [CSFW-14]
 - Howell & Kotz [ESORICS'00]

8

Work on Semantics of SPKI/SDSI (2)

- Other approaches
 - Graph-based: Aura [CSFW-11]
 - Logic-programming-based: Li [CSFW-13]
 - Automata-based: Jha & Reps [CSFW-15] [CSFW-16]
 - First-Order-Logic-based: Li & Mitchell [CSFW-16]

9

Policy Statements in SPKI 1.0: 5-tuples

- A 5-tuple (K, S, D, T, V)
 - K is the issuer principal
 - S is the subject
 - can be a principal or a threshold
 - $D \in \{0, 1\}$ in the re-delegation bit
 - T is the authorization tag
 - V is the validity specification

10

Queries in SPKI 1.0

- Queries are not explicitly defined in SPKI/SDSI documentation
- Instead, a 5-tuple reduction mechanism is specified, which can be viewed as implicitly defining syntax of queries and the semantic relation

11

5-tuple Reduction in RFC 2693

- The 5-tuple reduction rule:
 - Given $(K_1, S_1, D_1, T_1, V_1)$ and $(K_2, S_2, D_2, T_2, V_2)$
 - if $S_1 = K_2$ and $D_1 = 1$,
 - derive $(K_1, S_2, D_2, AIntersect(T_1, T_2), VIntersect(V_1, V_2))$
- Implicit queries:
 - Does one 5-tuple follow from a set of 5-tuples?

12

A First Look at Tags

- A tag is a list of byte-strings or sub-lists.
- Two tags intersect by matching element by element
- Example:
 - $\text{AIntersect}(\text{ftp}(\text{host ftp.clark.net}), \text{ftp}(\text{host ftp.clark.net})(\text{dir /pub/cme})) = \text{ftp}(\text{host ftp.clark.net})(\text{dir /pub/cme})$

13

Comparing KeyNote and SPKI 5-tuples (1)

- KeyNote and SPKI 1.0 has almost the same delegation structures
- Example: KA delegates the ftp permission to KB, and KB to KC.

KeyNote: $KA \xrightarrow{\text{access=ftp}} KB \xrightarrow{\text{access=ftp}} KC$

SPKI: $KA \xrightarrow[\text{D=1}]{(\text{ftp})} KB \xrightarrow[\text{D=1}]{(\text{ftp})} KC$

14

Comparing KeyNote and SPKI 5-tuples (2)

- Differences
 - Re-delegation control (boolean vs. no)
 - Encoding of requests and conditions
 - SPKI uses tags
 - KeyNote uses name/value pairs and condition expressions
- Same limitation
 - all statements in a chain must talk about one permission

15

Logical Semantics for TM Languages

- A TM Language has
 - Syntax for policy statements and queries
 - Semantic relation \blacktriangleright (proves)
 - Given a set P of policy statements and a query Q, $P \blacktriangleright Q$
- The semantic relation can be defined using logic if both policy statements and queries are logical statements

16

Towards a Logical Semantics for SPKI

- Simplifying assumptions
 - treat tags as constants
 - assume that subjects are not thresholds
 - ignore validity periods
- Use one predicate grant (K, K_S, T) to denote that K grants the right T to K_S.
- What about re-delegation?
 - $\forall x (\text{grant}(K, x, T) \Leftarrow \text{grant}(K_S, x, T))$

We are only using a subset of FOL!

17

Introduction of Datalog (1)

- Example Datalog rules:
 - $p(a,b), p(b,c)$
 - $p(x,y) \leftarrow p(x,z), p(z,y)$
 - $\forall x \forall y \forall z (p(x, y) \Leftarrow p(x, z) \wedge p(z, y))$
- Semantics of a Datalog program
 - model-theoretic
 - fix-point-based
 - proof-theoretic

18

Introduction of Datalog (2)

- Efficiency of evaluation
 - worst-case linear in grounded program

19

Semantics for Tags

- A tag T represents a (potentially infinite) set of strings
- $(K_1, K_2, 0, T, \bullet)$ means that K_1 grants to K_2 any rights encoded in strings in T
- To handle tags in a logical semantics, we use constraints
 - $t \in T$ to represent that the variable t ranges over all strings in T

20

A FOL Semantics for SPKI (w/o thresholds)

- Each statement is translated into a FOL sentence
 - $(K, K_S, 0, T, \bullet)$ maps to $\forall t (g(K, t, K_S) \leftarrow t \in T)$
 - $(K, K_S, 1, T, \bullet)$ maps to $\forall t (g(K, t, K_S) \leftarrow t \in T) \wedge \forall z \forall t (g(K, t, z) \leftarrow (g(K_S, t, z) \wedge t \in T))$

We are not in Datalog anymore !

21

Introducing Constraint Datalog

- What is Constraint Datalog:
 - Special form of CLP; query language for Constraint DB
- A Constraint Datalog rule:
 - $R_0(x_0) :- R_1(x_1), \dots, R_n(x_n), \varphi(x_0, x_1, \dots, x_n)$
 - x_0, x_1, \dots, x_n are tuples of variables
 - φ is a constraint in all the variables

22

Example

- A grants to B the permission to
 - connect to hosts in "stanford.edu"
 - at any port from 8000 to 9000;
 - and allows B to further delegate
$$\text{grant}(A, B, \text{'connect'}, h, p) :-$$
$$h \subset (\text{edu,stanford}), p \in [8000,9000].$$
$$\text{grant}(A, x, \text{'connect'}, h, p) :-$$
$$\text{grant}(B, x, \text{'connect'}, h, p),$$
$$h \subset (\text{edu,stanford}), p \in [8000,9000].$$

23

Example (continued)

- B grants to D the permission to connect to the host "cs.stanford.edu" and any host in this domain at any port
 - $\text{grant}(B, D, \text{'connect'}, h, p) :-$
 $h \subset (\text{edu,stanford,cs}).$
- Query: which hosts does A allow D to connect to at port 8080?
 - $\text{query}(h) :- \text{grant}(A, D, \text{'connect'}, h, p), p=8080.$

24

Using Constraint Datalog to Analyze KeyNote (1)

- Example assertions one could write in KeyNote:
 - A delegates to B all actions (a,h,p) in which $action='connect' \wedge host=\sim'^*.stanford.edu' \wedge port \leq 8443 \wedge port \geq 8000$
 - A delegates to B all actions (x,y,z,n) in which $x^n + y^n = z^n$, where x, y, z, n are integers.

25

Using Constraint Datalog to Analyze KeyNote (2)

- Theorem: it is undecidable to compute the set of all requests that one KeyNote assertion authorizes.
- How to view the theorem:
 - given any specific request, one can still determine whether it is authorized
 - cannot determine whether an assertion authorizes any request at all

26

Returning to SPKI: A Closer Look at Tags

- Special Tags
 - (*)
 - (* set <tag-expr>*)
 - (* prefix <byte-string>)
 - (* range <ordering> <lower-limit> <upper-limit>)

27

Incompleteness of 5-tuple Reduction

- 5-tuple reduction is incomplete
 - An example
 - Given $(K_1, K_2, 0, (* \text{ set read write}), \bullet)$ and $(K_1, K_2, 0, (* \text{ set delete}), \bullet)$; it follows (in the logical semantics) that $(K_1, K_2, 0, (* \text{ set read delete}), \bullet)$;
 - however, this cannot be derived using 5-tuple reduction

28

Another Problem of 5-tuple Reduction

- AIntersect is not defined when intersecting
 - a range tag with a prefix tag
 - or two range tags using different ordering
- AIntersect cannot be defined in a sound and complete way [Howell]
 - such intersections may result in sets not finitely representable by tags

29

A Solution

- Source of the problem:
 - interaction among special tags in SPKI cause difficulties
- Solution:
 - Instead of using the tag $(ftp (*prefix /pub/software) (* \text{ set read write}))$
 - Use constraint-based terms $ftp(dir \subseteq (pub,software), access \in \{read,write\})$

30