CS590U: Access Control: Theory and Practice Spring 2005

Assignment #6 Due: Thursday, March 31, 2005.

Problem 1 Consider the following example in which we have the following 4-tuples. Note that we use the syntax $(K, A, \omega, \cdot)$ to denote a 4-tuple that is signed (issued) by K and define the local name $K\ A$ to include the name string ω. The $\cdot$ represents the validity specification that we have omitted.

$$(K_{\text{Alice}},\ \text{friend},\ K_{\text{Alice}}\ \text{family friend},\ \cdot) \tag{1}$$
$$(K_{\text{Alice}},\ \text{family},\ K_{\text{Bob}},\ \cdot) \tag{2}$$
$$(K_{\text{Alice}},\ \text{friend},\ K_{\text{Diana}},\ \cdot) \tag{3}$$
$$(K_{\text{Bob}},\ \text{friend},\ K_{\text{Bob}}\ \text{friend friend},\ \cdot) \tag{4}$$
$$(K_{\text{Bob}},\ \text{friend},\ K_{\text{Alice}}\ \text{friend},\ \cdot) \tag{5}$$
$$(K_{\text{Bob}},\ \text{friend},\ K_{\text{Carl}},\ \cdot) \tag{6}$$
$$(K_{\text{Carl}},\ \text{friend},\ K_{\text{Eric}},\ \cdot) \tag{7}$$
$$(K_{\text{Diana}},\ \text{friend},\ K_{\text{Frank}},\ \cdot) \tag{8}$$

a. (3 pts) Show how to rewrite the local name K_{Alice} friend into K_{Eric} friend.

b. (8 pts) Write the semantic logic program corresponding to the above set of 4-tuples.

c. (6 pts) Compute the least Herbrand model of the above program.

d. (3 pts) Write down the values of all local names given by the least valuation.

e. (10 pts) Describe all name strings that K_{Alice} can rewrite into.

Hint: Think about regular grammars, i.e., regular string expressed in terms of grammars.

Problem 2 Consider the following example in which we have the following 4-tuples.

$$(K_{\text{Purdue}},\ \text{grad\_student},\ K_{\text{Purdue}}\ \text{dept grad\_sid},\ \cdot) \tag{9}$$
$$(K_{\text{Purdue}},\ \text{dept},\ K_{\text{Purdue}},\ \text{college dept},\ \cdot) \tag{10}$$
$$(K_{\text{Purdue}},\ \text{college},\ K_{\text{CoS}},\ \cdot) \tag{11}$$
$$(K_{\text{Purdue}},\ \text{college},\ K_{\text{CoE}},\ \cdot) \tag{12}$$
$$(K_{\text{CoS}},\ \text{dept},\ K_{\text{CS}},\ \cdot) \tag{13}$$
$$(K_{\text{CoS}},\ \text{dept},\ K_{\text{Math}},\ \cdot) \tag{14}$$
$$(K_{\text{CoE}},\ \text{dept},\ K_{\text{ECE}},\ \cdot) \tag{15}$$
$$(K_{\text{CS}},\ \text{grad\_sid},\ K_{\text{Alice}},\ \cdot) \tag{16}$$
$$(K_{\text{CS}},\ \text{grad\_sid},\ K_{\text{Bob}},\ \cdot) \tag{17}$$

a. (10 pts) Suppose that we want credentials (11) to (14) to be stored only by the issuers. That is, credentials (11) and (12) are stored only by K_{Purdue}, and credentials (13) and (14) are stored only by K_CoS. Further suppose that we want credential (16) to be stored by K_{Alice}. Assign the storage types to all role names in the example so that every credential is well-typed. There are different possibilities. Try to use as weak storage types as possible. For example, when "issuer-traces-def" is sufficient, avoid using "issuer-traces-all".

b. (10 pts) Using the storage type system in b, which credentials would be collected when performing a bi-directional search to try to determine whether $K_{\text{Purdue}}\ \mathrm{grad-student}$ can rewrite into K_{Alice}.

Problem 3 The Logic of Linked Name Containment (LLNC) [2] is one of the logics that has been developed in order to provide a semantics for SDSI. It has the following axioms. In the axioms, p, q, and r are SDSI name strings, a and a_1 are principals, and m denote a name identifier. A name string $a\ m_1\ m_2$ is encoded as $a's\ m_1's\ m_2$. A four tuple $(a, m, r, \cdot)$ is represented as "a $\texttt{says}$ $(m \longrightarrow r)$". The intuition is that if one can prove $p \longmapsto q$ using the axioms and a set of 4-tuples, then $p \longmapsto q$ is implied by the set of 4-tuples.

Propositional Logic:	All instances of propositional tautologies
Reflexivity:	$p \longmapsto p$
Transitivity:	$(p \longmapsto q) \Rightarrow ((q \longmapsto r) \Rightarrow (p \longmapsto r))$
Left-monotonicity:	$(p \longmapsto q) \Rightarrow ((p's\ \ r) \longmapsto (q's\ \ r))$
Associativity:	$((p's\ \ q)'s\ \ r) \longmapsto (p's\ \ (q's\ \ r))$
	$(p's\ \ (q's\ \ r)) \longmapsto ((p's\ \ q)'s\ \ r))$
Key Globality:	$(a's\ \ b) \longmapsto b$
	if b is a global identifier, *i.e.*, a principal
Key Linking:	$(a\ \texttt{says}\ (m \longrightarrow r)) \Rightarrow ((a's\ \ m) \longmapsto (a's\ \ r))$
	if m is a local name
Key Distinctness:	$\neg(a_1 \longmapsto a_2)$ if a_1 and a_2 are distinct keys, i.e., principals
Witnesses:	$\neg(p \longmapsto q) \Rightarrow \vee_a(\neg(p \longmapsto a) \wedge (q \longmapsto a))$
	$(p's\ \ q) \longmapsto a_1 \Rightarrow \vee_a((p \longmapsto a) \wedge (a's\ \ q \longmapsto a_1))$
Modus Ponens:	From ϕ and $\phi \Rightarrow \psi$ infer ψ

In the axioms, $\Rightarrow$ denote logic implication. The name strings considered in LLNC may contain principals in the middle of the string and may contain parenthesis, i.e., "$K_{\text{Alice}}'s\ (\mathrm{friend}'s\ (K_{\text{Bob}}'s$ friend$))$" is a name string. When we restrict ourselves to name strings having the form of one principal followed by a list of identifiers, the Associativity axioms and the Key Globality axiom are no longer needed. The Key Linking axiom interprets the 4-tuples.

a. (10 pts) Show how to prove $(K_{\text{Bob}}'s\ \ \mathrm{friend}) \longmapsto K_{\text{Eric}}$ using the above axioms and the 4-tuples in Problem 1.

Hint: Using the Transitivity and Left-monotonicity axioms suffices.

Problem 4 In SPKI/SDSI 2.0 [1], in addition to 4-tuples (which are also known as name certs), there are also auth certs and ACL entries.

An auth cert is a signed five tuple (K, H, D, T, V), where

- $K \in \mathcal{K}$ is the issuer principal, which signs the cert. The issuer grants a specific authorization through this 5-tuple.
- $H \in \mathcal{H}$ is called the subject, where $\mathcal{H}$ is defined to be the least set satisfying the following two conditions: (1) $SS \subseteq \mathcal{H}$, where S is the set of all names, and (2) $\theta(k, [H_1, H_2, \cdots, H_n]) \in \mathcal{H}$, where θ is a keyword, k and n are integers such that $1 \leq k \leq n$, and $H_1, H_2, \cdots, H_n \in \mathcal{H}$. The subject specifies principals that receive authorization from this 5-tuple.
- $D \in \{0, 1\}$ is called the delegation bit. When $D = 1$, the subject may further delegate the authorization it receives from this 5-tuple.
- T is the authorization tag, each tag represents a (potentially infinite set of) byte strings. T specifies the authorization that is granted by this 5-tuple.
- V is the validity specification, which is the same as in the case of a name cert.

An ACL entry is a locally stored 5-tuple $(\texttt{Self}, H, D, T, V)$. It is very similar to an auth cert, except that the issuer is a special symbol `Self` instead of a key and that the ACL entry is not signed. We will treat `Self` as a special principal in $\mathcal{K}$.

Roughly speaking, a 5-tuple (K, H, D, T, V) means that the principal K delegates the authorization encoded in T to every principal that is associated with H, and when $D = 1$, allows such a principal to further delegate this authorization.

The 5-tuple reduction rule in Section 6.3 of RFC 2693 is as follows: the two 5-tuples $(K_1, S_1, D_1, T_1, V_1)$ and $(K_2, S_2, D_2, T_2, V_2)$ yield the 5-tuple $(K_1, S_2, D_2, \mathsf{AIntersect}(T_1, T_2), \mathsf{VIntersect}(V_1, V_2))$, provided that $S_1 = K_2$, $D_1 = 1$, and $\mathsf{AIntersect}(T_1, T_2)$ succeeds. $\mathsf{VIntersect}$ is an operator that takes two validity specifications V_1 and V_2 and returns a validity specification that is valid only when both V_1 and V_2 are valid. $\mathsf{AIntersect}$ is an operator that takes two authorization tags T_1 and T_2, and returns their intersection; $\mathsf{AIntersect}(T_1, T_2)$ may fail when the intersection of T_1 and T_2 is empty.

The 5-tuple reduction rule only applies when the subject is a principal. When the subject is a name, the way to use the 5-tuple reduction rule is to first replace a 5-tuple that has a name S as the subject with a set of 5-tuples, each of which has one principal in the valuation of the S as the subject. The procedure for handling thresholds is much more complicated; and we ignore thresholds for now.

An authorization tag is a list of byte-strings or sub-lists. Two tags intersect by matching, element for element. If one list is longer than the other but matches at all elements where both lists have elements, then the longer list is the result of the intersection. This means that additional elements of a list must restrict the permission granted. For example, `(ftp (host ftp.clark.net))` may represent the permission of ftp access to every file and every directory on the host `ftp.clarke.net`. This is more general than `(ftp (host ftp.clark.net) (dir /pub/cme))`, and the intersection of the two tags results in the latter. SPKI also has a small number of special expressions.

`(*)` stands for the set of all tags and byte-strings. In other words, it will match anything. When intersected with another tag, the result is that other tag.

`(* set <tag-expr>*)` stands for the set of elements listed in the *-form.

`(* prefix <byte-string>)` stands for the set of all byte strings that start with the one given in the *-form.

`(* range <ordering> <lower-limit>? <upper-limit>?)` stands for the set of all byte strings lexically (or numerically) between the two limits. The ordering parameter (alpha, numeric, time, binary, date) specifies ordering.

a. (5 pts) Suppose one wants to write a policy that grants a permission described using a tag "`(ftp (host ftp.clark.net) (* set read write))`" to any user whose age is over 18. Further suppose that the age is documented using a digital driver license. Can this policy and the digital driver license be represented in SPKI/SDSI 2.0? If yes, give the related 5/4-tuples. If no, explain why.

b. (5 pts) Suppose one wants to write a policy that grants a permission described using a tag "`(ftp (host ftp.clark.net) (* set read write))`" to any user who is a student. Can this policy be represented in SPKI/SDSI 2.0? If yes, give the related 5/4-tuples. If no, explain why.

c. (5 pts) How would you represent the policy and digital driver license in part a using RT_1?

Problem 5 (25 pts) Come up with an example scenario in which you think trust management can be applied. Describe the principals involved and the policy statements that exist. Make the scenario as interesting as you can.

References

[1] Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. SPKI certificate theory. IETF RFC 2693, September 1999.

[2] Joseph Halpern and Ron van der Meyden. A logic for SDSI's linked local name spaces. *Journal of Computer Security*, 9(1-2):47–74, 2001.