An operational framework to reason about policy behavior in trust management systems

Edelmira Pasarella[1]⋆ and Jorge Lobo[2]⋆⋆

[1] Dpto de L.S.I., Universitat Politècnica de Catalunya,
email edelmira@lsi.upc.edu
[2] ICREA - Universitat Pompeu Fabra
email jorge.lobo@upf.edu

Abstract. In this paper we show that the logical framework proposed by Becker et al. to reason about security policy behavior in a trust management context can be captured by an operational framework that is based on the language proposed by Miller to deal with scoping and/or modules in logic programming in 1989. The framework of Becker et al. uses propositional Horn clauses to represent both policies and credentials, implications in clauses are interpreted in counterfactual logic, a Hilbert-style proof is defined and a system based on SAT is used to proof whether properties about credentials, permissions and policies are valid in trust management systems, i.e. formulas that are true for all possible policies. Our contribution is to show that instead of using a SAT system, this kind of validation can rely on the operational semantics (derivability relation) of Miller's language, which is very close to derivability in logic programs, opening up the possibility to extend Becker et al.'s framework to the more practical first order case since Miller's language is first order.

1 Introduction

Trust Management Systems (TMS) [BFL96] are perhaps the most common model to describe distributed access control. In this model, there are (1) policies that define under what conditions a subject is able to access resources, (2) credentials that are provided by the subject in order to fulfill policies and (3) decisions of whether a subject has particular permissions. One of the most popular ways to describe TMS is to use logic programming-like languages for definition of policies and credentials (see for example [DeT02], [JIM01],[LGF00],[LM03]). Then, permissions are decided by inferences done combining policy and credentials.

Working under this framework, Becker et al. [BRS12] have recently proposed a logic system in which one can reason about TMS in general. In this

⋆ Partially supported by the Spanish CICYT project FORMALISM (Ref. TIN2007-66523) and by the AGAUR Research Grant ALBCOM (Ref. SGR20091137).

⋆⋆ Partially supported by the US Army Research Lab and the UK Ministry of Defence under agreement number W911NF-06-3-0001.

system, a Hilbert-style axiomatization is defined and a system based on SAT solvers is used to prove automatically TMS properties. In their logic, policies and credentials are formalized using propositional logic programs, while permissions are propositional Boolean formulas. Trust management behavior, i.e. to determine whether a permission p is true under a policy P when presenting a set of credentials Q, is captured by proving that the statement $Q \supset p$ *holds in the policy* P.[3] This statement is true if p holds in the policy P extended with those clauses in the credentials Q: $P \cup Q \vdash p$.

Example 1. To illustrate how policies, credentials and permissions are expressed, let us consider a very simple example about purchasing digital goods. The policy of the seller is:

$X.paid(Music, 1.99\$) \supset X.download(Music)$
$paypal(X,V)$
$wire\_transfer(X,V)$

that says "if X paid the fee *Fee* for the music *Music*, X is able to download *Music*" and, there are two ways in which X can pay a fee V: either by means of *paypal* or by means of a *wire_transfer*".[4] These statements represent schemes of policies for specific values of the arguments. Credentials for paying the fee presented by a subject X to request download permission can be written as follows:

$$(paypal(X, 1.99\$) \supset X.paid(song, 1.99\$)) \supset X.download(song)$$

To allow the subject X to download *Music*, the system has to join the policy and the credential $(paypal(X, 1.99\$) \supset X.paid(song, 1.99))$ in a single program and then verify that the permission $X.download(song)$ holds in it.

The important contribution of Becker et al.'s work is the definition of valid formulas in TMS. Informally, these are formulas that are true regardless of the policies and credentials that can be defined in the TMS. Hence, they are able to describe how to approach proofs such as *proving attacks* (i.e. discovering policies) in specific TMS systems, or general properties such as the transitivity of credential-based derivations.

The authors, however, argue that Hilbert style axiomatizations are difficult for building proofs because they are not goal oriented. Hence, they resort to an algorithm that interleaves syntactic transformations of formulas and calls to SAT solvers in order to do automatic verifications. In their paper there is an argument but not a proof that the mechanization is correct. A proof may be possible but probably not easy.

[3] In [BRS12] formulas like $Q \supset G$ are written as $\Box_Q G$ but we will follow Miller's notation.

[4] We make the simplifying assumption that no third party is involved in the TMS and that the seller has access to paypal and the bank.

In this work we show that the logical framework proposed by Becker et al. can be captured by an operational framework that is based on a language proposed by Miller in 1989 to deal with scoping and/or modules in logic programming. *Our contribution is to show that we can rely on the operational semantics (derivability relation) of Miller's language, which is very close to derivability in logic programs, to do goal oriented formula verification.* This connection also open the possibility of extending Becker et al.'s framework to the more practical first order case since Miller's language is first order.

2 Trust management systems

In the following we assume the existence of an underlying propositional signature Σ that consists of a countable set of propositional variables. We call these propositional symbols Σ-atoms. For the sake of simplicity, in the following, we omit the prefix Σ- when it is clear from the context.

Definition 1. *Programs, clauses and goals are defined using the BNF presented below, where A, F, C, P and G range over (1) atoms, conjunctions of atoms, (2) clauses, (3) programs and (4) goals, respectively.*

(1) $F ::= \texttt{true} \mid A \mid F \wedge F$ *(2)* $C ::= F \supset A$

(3) $P ::= C \mid C;P$ *(4)* $G ::= F \mid \neg G \mid P \supset G \mid G \wedge G \mid G \vee G$

Perhaps the most unusual definition is the definition of goals. A *goal* is either an expression of the form $P \supset G$, where P is a program and (inductively) G a goal or an expression corresponding to a propositional formula built with the standard connectives $\neg$, $\wedge$ and $\vee$. We note that our goals are the formulas defined in Becker et al.'s. As usual, we define implication, $G_1 \rightarrow G_2$, as $\neg G_1 \vee G_2$ and equivalence, $G_1 \leftrightarrow G_2$, as $(G_1 \rightarrow G_2) \wedge (G_2 \rightarrow G_1)$. Throughout the rest of the paper, we adopt the following conventions: P and Q denote programs. Clauses may be embraced in parenthesis. In a clause of the form $\texttt{true} \supset p$ we simply write p.

2.1 Reasoning in trust management systems

In this section we introduce an operational framework to reason about policies, credentials and permissions. We denote this framework by $\hat{O}$-TMF. The $\hat{O}$-TMF framework is based on the language introduced in [Mil89]. The semantics is presented in terms of a derivation relation over sequents. A *sequent* is a pair of the form $P \vdash G$, where the program P is called the *antecedent* and the goal G the *succedent*. As explained earlier, the intuitive interpretation of embedded implications is that, given a program P, to prove the query $Q \supset G$ it is necessary

to prove G with the program $P \cup Q$. This is formalized in [Mil89] using the following inference rule:

$$\frac{P \cup Q \vdash G}{P \vdash Q \supset G}$$

2.2 $\hat{O}$- proof rules

The inference rules for sequents in $\hat{O}$-TMF are defined over $Programs \times Goals$ as follows

$$\frac{P \vdash_{\hat{O}} G_i}{P \vdash_{\hat{O}} G_1 \vee G_2}\ i = 1,2 \qquad \frac{P \vdash_{\hat{O}} G_1 \quad P \vdash_{\hat{O}} G_2}{P \vdash_{\hat{O}} G_1 \wedge G_2} \qquad \frac{P \cup Q \vdash_{\hat{O}} G}{P \vdash Q \supset G} \qquad \frac{P \vdash_{\hat{O}} Q \supset \neg G}{P \vdash_{\hat{O}} \neg(Q \supset G)}$$

$$\frac{P \vdash_{\hat{O}} \neg G_1 \wedge \neg G_2}{P \vdash_{\hat{O}} \neg(G_1 \vee G_2)} \qquad \frac{P \vdash_{\hat{O}} \neg G_1 \vee \neg G_2}{P \vdash_{\hat{O}} \neg(G_1 \wedge G_2)} \qquad \frac{P \vdash_{\hat{O}} A}{P \vdash_{\hat{O}} \neg\neg A}$$

An $\hat{O}$-proof for $P \vdash_{\hat{O}} G$ is a tree in which nodes are labeled with sequents such that (i) the root node is labeled with $P \vdash_{\hat{O}} G$, (ii) the internal nodes are instances of one of the above inference rules and (iii) the leaf nodes are labeled with *initial sequents*. An *initial sequents* is a sequent of the form $P' \vdash_{\hat{O}} G'$ where G' is a propositional formula that is $true$ in the minimal model of P'.

We can prove the validity of formulas as follow. First, we need the following definition:

Definition 2. *Let G be a goal and Σ_G be the signature formed by the set of propositional atoms occurring in G. G is valid, $\vdash_{\hat{O}} G$, in the $\hat{O}$-TMF if and only if it is not possible to find a Σ_G-policy Δ such that $\Delta \vdash_{\hat{O}} \neg G$*

As we will see in the next section, our definition of validity in TMS is equivalent to Becker et al.'s definition. We now illustrate with an example how Definition 2 can be used to prove the validity of the formula given Example V.4. of [BRS12]: $G = \neg a \wedge d \supset \neg e \wedge (a \supset b \wedge c \supset d) \supset e \rightarrow c \wedge d \supset a$ where $\Sigma_G = \{a, b, c, d, e\}$.

Following Definition 2, $\vdash_{\hat{O}} G$ if and only if it is not possible to find Σ_G-policy Δ such that $\Delta \vdash_{\hat{O}} \neg G$. Hence, we have to prove that there is not Δ such that

$$\Delta \vdash_{\hat{O}} \neg(\neg a \wedge d \supset \neg e \wedge (a \supset b \wedge c \supset d) \supset e \rightarrow c \wedge d \supset a)$$

This is (classically) equivalent to:

$$\Delta \vdash_{\hat{O}} \neg a \wedge d \supset \neg e \wedge (a \supset b \wedge c \supset d) \supset e \wedge (\neg c \vee \neg(d \supset a))$$

Distributing $\wedge$ over $\vee$ we obtain that we have to prove there is no Δ such that:

1. $\Delta \vdash_{\hat{O}} \neg a \wedge d \supset \neg e \wedge (a \supset b \wedge c \supset d) \supset e \wedge \neg c$

(a) $\Delta \vdash_{\hat{O}} \neg a$: Since we are looking for a counter-example, we need to make this sequent an initial sequent, so we have to construct a policy Δ whose minimal model, M_Δ, contains the valuation $a = false$. This is $a \notin M_\Delta$. It is worth to notice that the scoping of Δ is the whole formula.

(b) $\Delta \vdash_{\hat{O}} d \supset \neg e$: if and only if $\Delta \cup \{d\} \vdash_{\hat{O}} \neg e$ and this is an initial sequent if the valuation $e = false$ is in the minimal model of $\Delta \cup \{d\}$. Hence, $e \notin M_\Delta$.

(c) $\Delta \vdash_{\hat{O}} (a \supset b \wedge c \supset d) \supset e$: if and only if $\Delta \cup \{a \supset b; c \supset d\} \vdash_{\hat{O}} e$ but this is not possible since $\neg e$ holds in M_Δ.

Therefore, $\Delta \nvdash_{\hat{O}} \neg a \wedge d \supset \neg e \wedge (a \supset b \wedge c \supset d) \supset e \wedge \neg c$

2. $\Delta \vdash_{\hat{O}} \neg a \wedge d \supset \neg e \wedge (a \supset b \wedge c \supset d) \supset e \wedge \neg(d \supset a)$. In this case, directly by item 1c above we get that $\Delta \nvdash_{\hat{O}} \neg a \wedge d \supset \neg e \wedge (a \supset b \wedge c \supset d) \supset e \wedge \neg(d \supset a)$

Since we cannot construct a (counter-example) policy for $\neg G$, then $\vdash_{\hat{O}} G$.

3 Equivalence of $\hat{O}$-TMF and Becker et al.'s TMS

In this section we briefly described how to show that our definition of validity is equivalent to Becker et al.'s. First, we recall some definitions from [BRS12]. Let *basic* goals be atoms and classical propositional compound formulas expressed in terms of $\wedge$ and $\neg$. Let P be a policy, M_P its minimal model and G a basic goal. Then, $P \Vdash G$ if and only if G holds in M_P. Additionally, Becker et al. inductively define that $P \Vdash Q \supset G$ if and only if $P \cup Q \Vdash G$, where Q is a policy. A goal G is valid, $\Vdash G$, if and only if for every policy P, $P \Vdash G$. In order to deal with goals that allow the evaluation of policies together with credentials, we need the following lemma.

Proposition 1. *Let $Q \supset G$ be a goal. Then, for all policies P*

$$P \Vdash Q \supset G \quad \textit{if and only if} \quad P \vdash_{\hat{O}} Q \supset G \quad \blacksquare$$

The proof follows by induction by showing that $P \Vdash Q_1 \supset \ldots Q_k \supset G$ iff $P \vdash_{\hat{O}} Q_1 \supset \ldots Q_k \supset G$ using the definition of $\Vdash$ and the $\supset$-$\hat{O}$ rule. As a corollary we also have

$$P \Vdash G \quad \text{if and only if} \quad P \vdash_{\hat{O}} G \qquad (1)$$

We use one of the main results in [BRS12] as well. This is, the equivalence between their derivability and their proof system validity . Given a formula φ,

$$\Vdash \varphi \quad \text{if and only if} \quad \vdash \varphi \qquad (2)$$

Theorem 1. *Let G be a goal. Then, $\vdash G$ if and only if $\vdash_{\hat{O}} G$*

Proof. From (1) and (2) it follows that $\vdash G$ if and only if $\Vdash G$ if and only if $\Vdash_{\hat{O}} G$ $\blacksquare$

4 Final remarks

In this work we have presented a very operational definition of validity in TMS. Based on this result we have designed a top-down proof procedure of validity. This procedure works similar to abduction in logic programs with the addition that not only atoms but also rules can be assumed in order to find Δs (see Def. 2). We are able also to describe a model theoretic semantics based on Kripke structures following Miller's models. In particular, Miller interprets a world of a Kripke's model as a program and the knowledge at each world as its minimal model. This intuition can be explained in terms of two basic ideas of modal logic. The first one is the notion that a *world* may be considered to represent the "knowledge" that we have at a certain moment. The second idea is that a formula can be considered to hold if we can infer its truth from the knowledge that we have now or one that we may acquire in the "future", capturing the idea of credentials. Details will appear in the full version of this paper.

An important consequence of the connections between Miller's language and the propositional logic for reasoning in TMS is the possibility of lifting the results to policies, credentials and permissions with variables. We cannot apply directly Miller's results because his logic doesn't deal with negation. There is, however, an extensions to Miller's logic that deals with normal logic programs [POPN12], but we need to work out the details of the axiomatization since the approach in [POPN12] uses a notion similar to Clark's completion as opposed to minimal models. Complementary to these extensions we will also like to check how an implementation of validity using our approach will compare to the implementation of Becker et al.

References

[BRS12] Becker, M. Y. and Russo, A. and Sultana, N.: Foundations of logic-based trust management. In Proc. of IEEE Symposium on Security and Privacy, 161–175 (2012)

[BFL96] Blaze, M. and Feigenbaum, J. and Lacy, J.: Decentralized trust management. In Proc. IEEE Symposium on Security and Privacy, 1996. 164–173 (1996)

[DeT02] DeTreville, J.: Binder, a logic-based security language. In Proc. of IEEE Symposium on Security and Privacy, 2002, 105–113 (2002)

[JIM01] Jim, T.: SD3: A trust management system with certified evaluation. In Proc. of IEEE Symposium on Security and Privacy, Security and Privacy, 2001. 106–115 (2001)

[LGF00] Li, N. and Grosof, B. and Feigenbaum, J.: A practically implementable and tractable delegation logic In Proc. of IEEE Symposium on Security and Privacy, 27–42 (2000)

[LM03] Li, N. and Mitchell, J. C.: Datalog with constraints: A foundation for trust management languages. In Practical Aspects of Declarative Languages, 58–73, 2003. Springer

[Mil89] Miller, D.: A logical analysis of modules in logic programming. The Journal of Logic Programming, Vol. 6, 1, 79–108 (1989). Elsevier

[POPN12] Pasarella, E. and Orejas, F. and Pino, E. and Navarro, M.: Semantics of structured normal logic programs. The Journal of Logic and Algebraic Programming, Vol. 81, 5, 559–584 (2012). Elsevier