A Certified Core Policy Language

Bahman Sistany*† and Amy Felty*
*School of Electrical Engineering and Computer Science
University of Ottawa, Ottawa, Canada
†Irdeto Canada Corporation, Ottawa, Canada
Email: bahman.sistany@irdeto.com, afelty@uottawa.ca

***Abstract*—We present the design and implementation of a Certified Core Policy Language (ACCPL) that can be used to express access-control policies. We define formal semantics for ACCPL and use the Coq Proof Assistant to state theorems about this semantics, to develop proofs for those theorems and to machine-check the proofs ensuring correctness guarantees are provided. The main design goal for ACCPL is the ability to reason about the policies written in ACCPL with respect to specific questions such as safety. In addition, ACCPL and the established proofs are integrated such that extensions to expressive power may be explored by also extending identifiable proof statements in the direction of the added expressivity. To this end, ACCPL is small (the syntax and the semantics of ACCPL only take a few pages to describe), although we believe ACCPL supports the core features of many access-control policy languages.**

I. Introduction

We describe the design of a Certified Core Policy Language (ACCPL) for expressing access control policies and its implementation in the Coq Proof Assistant [1].[1]

Using Coq to implement ACCPL was an important factor in its design, allowing us to address the trade-off between expressive power and ease of formal proof of correctness. The semantics of ACCPL are specified by a translation from policy statements together with an access request and an environment containing all the relevant facts, to decisions. We present results showing the translation functions behave correctly with respect to the decision question that asks whether a request to access a resource may be granted or denied, given a policy. The translation functions also cover the case where a given policy does not apply to a request in which case a decision of non-applicable is rendered. Our results show that for each access request, the translation algorithm terminates on all input policies with a decision of granted, denied or non-applicable.

To motivate the design of ACCPL, let us review the definition of "access-control": Authorization refers to the process of rendering a decision about whether to permit or deny access to a resource or asset of interest, hence the term "access-control."

In order to harmonize access control in large environments with many subjects and objects and disparate attributes, the Policy-based Access Control (PBAC) [8] model has been proposed. PBAC allows for a more uniform access-control model across the system. PBAC systems help create and enforce policies that define who should have access to what resources, and under what circumstances. Because of the cited advantages, along with its generality and widespread use, PBAC is the model ACCPL implements.

[1]Our complete proof development in Coq is available at http://www.site.uottawa.ca/~afelty/accpl/.

A. A Core Policy Language for PBAC Systems

Currently, the most popular Rights Expression Languages (REL)s include the eXtensible rights Markup Language (XrML) [16], and Open Digital Rights Language (ODRL) [5]. Both of these languages are XML-based and are considered declarative languages. RELs, or more precisely Digital Rights Expression Languages (DRELs) deal with the "rights definition" aspect of the Digital Rights Management (DRM) ecosystem of digital assets. DRM refers to the digital management of rights associated with the access or usage of digital assets.

The eXtensible Access Control Markup Language (XACML) [9] is another access control policy specification language that is general, high-level, and allows policies to be defined in a wide variety of domains. Like ODRL and XrML, it is based on XML and the PBAC model. ODRL and XrML differ from XACML by their focus on digital assets protection and in general DRM, hence the term Digital Rights Expression Languages (DREL).

For a variety of reasons, we found XACML, ODRL and XrML all to be ill-suited as the basis for a core policy language. First, they are all large languages that provide numerous features but suffer from a lack of formal semantics. Second, all of these languages cover much more than policy expressions leading to access decisions; they also address enforcement of policies. Third, they are limited in terms of what can be built on top of them.

A policy language that was designed with logic and formal semantics in mind and also one that was small and extensible was clearly needed. We use Pucella and Weissman's subset of ODRL [12] as the basis for ACCPL and in doing so treat digital rights as our main access-control application without loss of generality with respect to other applications, with the final goal of performing formal verification on policies written in ACCPL.

B. Formal Semantics for PBAC Languages

Formal methods help ensure that a system behaves correctly with respect to a specification of its desired behavior [10]. This specification of the desired behavior is what is referred to as *semantics* of the system.

To formalize the semantics of PBAC languages several approaches have been attempted by various authors. Most

are logic based [3], [12] while others are based on finite-automata [4], operational semantics based interpreters [13] and web ontology (from the Knowledge Representation Field) [6]. As described below, our work can be viewed as an extension of [12].

C. Specific Problem

Policy languages and the policies, sometimes called *agreements*, written in those languages are meant to implement specific goals such as limiting access to specific assets. The tension in designing a policy language is usually between how to make the language expressive enough, such that the high-level and often service-oriented goals for policies may be expressed in the policy language, and how to make the policies verifiable with respect to the stated goals.

As stated earlier, an important part of fulfilling the verifiability goal for policy languages, is to define formal semantics, based on which theorems of interest may be declared and proven. However as is the case for other paper-proofs, often the language used to do these proofs, is based on intuitive justifications. As such these proofs are difficult to formally verify.

D. Contributions

We have designed a policy based access-control language called ACCPL based on ODRL and starting with definitions in [12]. The ACCPL framework has been encoded in Coq which is both a programming language and a proof-assistant. We have specified and proved ACCPL correct with respect to properties of interest in Coq which will allow us to extract programs from the proofs; the executable programs can be used on specific policies and a query, to render a specific decision such as "a permission has been granted".

In addition, ACCPL and the established proofs are integrated such that extensions to expressive power to ACCPL may be explored by also extending identifiable and corresponding proof statements in the direction of the added expressivity. This integration has been implemented in what we call the Translation Function Framework (TFF) used for handling semantics as part of our Coq encodings for ACCPL.

To qualify ACCPL "easy to reason about" we used Tschantz and Krishnamurthi's [15] reasonability properties as metrics of reasonability and designed the language such that ACCPL would satisfy these properties.

We have made significant modifications to the semantics of Pucella and Weissman's language such that an answer to a request for access to a resource may be determined unambiguously and for all cases.

Our executable program returns a list of decisions. We define what it means to extract a single decision from this list and show that it is always possible to extract a *coherent* decision.

Given that ACCPL is a core policy language with semantics that have been certified correct, we could use ACCPL to implement various (more expressive) policy languages. In addition ACCPL could be used as an intermediate language to reason about interoperability between those policy languages [11], [7].

II. ACCPL Semantics

We specify the semantics of ACCPL as a translation function from an agreement together with an access request and an environment containing all relevant facts, to decisions. The reader is referred to the Coq code implementing the semantics, along with all the auxiliary types and infrastructure which implement the semantics for ACCPL.

A. Types of Decisions

As mentioned, in ACCPL, evaluating a request against an agreement renders a granted, denied or non-applicable decision. Including the non-applicable decision was important for generality and for defining the semantics correctly.

B. Translations

Intuitively a query or request asks the following question given an agreement: "May subject `s` perform an action `ac` to asset `a`?" We represent a query by its components, namely the subject, action and asset that form the query question.

III. Correctness of ACCPL

In this section, we discuss the main theorem and some important supporting theorems, expressing the most important properties we have proved about ACCPL. For all other supporting theorems and for all proofs, the reader is referred to [14] and the accompanying Coq code, respectively.

A. Correctness of Translation

The `trans_agreement_dec` theorem is the declaration of the main correctness result for ACCPL. Together with proofs for other theorems and lemmas, we have "certified" ACCPL correct by proving this theorem. The list that `trans_agreement` returns will contain results one per each primitive policy found in the agreement. Specifically the predicate `isResultInQueryResult` checks for the existence of a particular result in the given list of results. The theorem states that for all environments, agreements and queries, the list that `trans_agreement` produces contains either a `Permitted` or a `NotPermitted` result or the list will contain neither `Permitted` nor `NotPermitted` results. By mentioning the agreement translation function (`trans_agreement`) directly in the statement of the theorem `trans_agreement_dec`, we tie the correctness property to how the translation functions work. To prove the theorem, and with each successive subgoal during the interactive proof process, the definition of the translation function in scope gets unfolded and used so the translation functions have to be defined such that each subgoal is discharged and the proof is completed.

According to the declaration of the `trans_agreement_dec` theorem, there are three cases that are mutually exclusive. The first case is when the set has a at least one `Permitted` result; we answer the access query in this case with a result of `Permitted`. The second case is

when the set has at least one `NotPermitted`; we answer the access query in this case with a result of `NotPermitted`. In the case where all the results are `Unregulated` we answer the access query with a result of `Unregulated`.

B. Mutual Exclusivity of `Permitted` and `NotPermitted`

The proof of the theorem `trans_agreement_not_Perm_and_NotPerm_at_once` establishes that both `Permitted` and `NotPermitted` results cannot exist in the same set returned by `trans_agreement`. This result also establishes the fact that in ACCPL rendering conflicting decisions is not possible given an agreement.

The proof of the theorem `trans_agreement_not_NotPerm_and_not_Perm_implies_Unregulated_dec` shows that in the case where neither a `Permitted` nor a `NotPermitted` result exists in the set returned by `trans_agreement`, there must exist at least one `Unregulated` result.

IV. Future Work

We describe in [14] how ACCPL meets the reasonability properties of Tschantz and Krishnamurthi [15] mentioned earlier. However, we have not yet certified (using formal proofs) that ACCPL has these properties. We defer formally proving these properties for ACCPL as future work.

Another direction for future work is to explore different ways ACCPL could be made more expressive. For example, we can add various policy combinators and their semantics to ACCPL using the TFF. As mentioned earlier, the TFF we have developed for ACCPL is meant to keep the delicate balance between addition of expressiveness while maintaining provability of established results.

Another design goal for ACCPL is to make it a target language for deploying policies written in other languages. We could capture, implement and study the semantics of these other policy-based access-control systems using the TFF and ultimately certify the semantics of those languages with respect to their specifications, the same way that ACCPL has been certified correct. In fact, our current work includes extending ACCPL to handle the expressive power of SELinux [2].

V. Conclusion

We have presented the design and implementation of ACCPL as a small and certifiably correct policy language. ACCPL is a PBAC system that can be used to express general access-control rules and policies. In addition we have defined formal semantics for ACCPL, where we have discovered and added all possible cases when answering a query on whether to allow or deny an action to be performed on an asset. We have subsequently used the Coq Proof Assistant to state theorems about the expected behavior of ACCPL when evaluating a request with respect to a given agreement, to develop proofs for those theorems and to machine-check the proofs ensuring correctness guarantees are provided. We have in particular stated, developed and proved correctness results for the semantics of ACCPL.

References

[1] Y. Bertot and P. Castéran, *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Springer Science & Business Media, 2004. [Online]. Available: http://www.labri.fr/perso/casteran/CoqArt/index.html

[2] A. Eaman, B. Sistany, and A. Felty, "Review of existing analysis tools for SELinux security policies: Challenges and a proposed solution," in *E-Technologies: Embracing the Internet of Things, Proceedings of the 7th International MCETECH Conference*, ser. Lecture Notes in Business Information Processing, vol. 289. Springer, 2017, pp. 116–135.

[3] J. Y. Halpern and V. Weissman, "Using first-order logic to reason about policies," *ACM Transactions on Information and System Security*, vol. 11, no. 4, 2008. [Online]. Available: http://doi.acm.org/10.1145/1380564.1380569

[4] M. Holzer, S. Katzenbeisser, and C. Schallhart, "Towards formal semantics for ODRL," in *Proceedings of the First International Workshop on the Open Digital Rights Language (ODRL), April 22-23*, 2004, pp. 137–148.

[5] R. Iannella., "Open digital rights language (ODRL) version 1.1," 2002, [accessed 05-August-2016]. [Online]. Available: http://www.w3.org/TR/2002/NOTE-odrl-20020919/

[6] A. Kasten and R. Grimm, "Making the semantics of ODRL and URM explicit using web ontologies," in *The 8th International Workshop for Technical, Economic and Legal Aspects of Business Models for Virtual Goods*, Namur, Belgium, 2010, pp. 77–91.

[7] X. Maroñas, E. Rodríguez, and J. Delgado, "An architecture for the interoperability between rights expression languages based on XACML," in *Proceedings of the 7th International Workshop for technical, economic and legal aspects of business models for virtual goods*, 2009.

[8] NIST, "A survey of access control models," 2009, [accessed 05-August-2016]. [Online]. Available: http://csrc.nist.gov/news_events/privilege-management-workshop/PvM-Model-Survey-Aug26-2009.pdf

[9] OASIS, *XACML Version 3.0*, 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[10] B. C. Pierce, *Types and Programming Languages*. Cambridge, MA, USA: MIT Press, 2002.

[11] J. Prados, E. Rodriguez, and J. Delgado, "Interoperability between different rights expression languages and protection mechanisms," in *First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*. IEEE, 2005, pp. 145–153.

[12] R. Pucella and V. Weissman, "A formal foundation for ODRL," *CoRR*, vol. abs/cs/0601085, 2006. [Online]. Available: http://arxiv.org/abs/cs/0601085

[13] N. P. Sheppard and R. Safavi-Naini, "On the operational semantics of rights expression languages," in *Proceedings of the 9th ACM Workshop on Digital Rights Management, November 9*. ACM, 2009, pp. 17–28. [Online]. Available: http://doi.acm.org/10.1145/1655048.1655052

[14] B. Sistany, "A certified core policy language," Ph.D. dissertation, University of Ottawa, 2016. [Online]. Available: https://www.ruor.uottawa.ca/handle/10393/34865

[15] M. C. Tschantz and S. Krishnamurthi, "Towards reasonability properties for access-control policy languages," in *SACMAT 2006, 11th ACM Symposium on Access Control Models and Technologies, June 7-9, Proceedings*. ACM, pp. 160–169. [Online]. Available: http://doi.acm.org/10.1145/1133058.1133081

[16] X. Wang, G. Lao, T. DeMartini, H. Reddy, M. Nguyen, and E. Valenzuela, "XrML - eXtensible rights markup language," in *Proceedings of the 2002 ACM Workshop on XML Security, November 22*. ACM, pp. 71–79. [Online]. Available: http://doi.acm.org/10.1145/764792.764803