De l'Utilisation des Métagraphes pour la Vérification de Politiques de Sécurité

Loïc Miller[1 †], Pascal Mérindol[1], Antoine Gallais[2] et Cristel Pelsser[1]

[1] *Université de Strasbourg,* [2] *UPHF / INSA Hauts-de-France*

Les processus métier multi-agents aux interactions complexes sont généralement modélisés en tant que *workflows*. Le propriétaire des données confidentielles interagit avec des sous-traitants pour réaliser une séquence de tâches, en déléguant aux différents acteurs des droits limités sur les données sensibles. Cette délégation repose sur le contrôle d'accès aux données. Pour faciliter sa configuration, les administrateurs proposent une spécification des politiques d'accès et se reposent ensuite souvent sur un traducteur. Cependant, la traduction de la spécification vers l'implémentation peut mener à des erreurs lors d'un déploiement effectif entre les différentes entités du workflow et ainsi engendrer des failles de sécurité. Dans cet article, nous proposons des structures facilitant la détection et la correction d'erreurs potentiellement introduites en raison d'une traduction défectueuse ou d'un déploiement défaillant. En particulier, nous considérons une structure aux fondations formelles capables de modéliser naturellement et surtout très finement les politiques de sécurité : les **métagraphes**. Nous proposons une suite d'outils de traduction permettant de détecter ces erreurs potentielles et évaluons ses performances.

Mots-clefs : vérification de politiques, métagraphes, modélisation de politiques, rego, contrôle d'accès

1 Introduction

Authorization is a key aspect of security, regulating the interactions taking place in a given system. Since the systems to be secured by authorization can be highly complex, administrators often rely on policy-based management of authorization. Policies define the desired behavior of a system from a high-level perspective. Hence, this form of management allows to separate the problem of specification, i.e. defining the desired system behavior, from the problem of implementation, that is the enforcement of the desired system behavior.

In this paper, we deal with policy verification, i.e., we check whether the deployment of policies actually meets their high-level specification. Policy verification plays an important role since assisting tools are not free of errors, and policy deployments can become faulty. We propose to model policies with a generic yet rich structure : metagraphs. We use its formal foundations to verify whether the actual deployment of a policy (i.e., its implementation) matches its initial specification. We rely on this structure since, by design, it provides means to locate conflicts and avoid redundancy. Metagraphs provide a more fine-grained verification process than with other structures like usual graphs, and are more suited than other structures to represent business processes. To the best of our knowledge, metagraphs belong to the rare appropriate structures able to naturally model access control policies.

The contributions of our paper are as follows :

1. We are the first to use metagraphs to perform the verification of access control policies. We argue they represent one of the most appropriate form of policy modeling enabling refinement and verification. We also show how this verification allows us to pinpoint errors in the policy.

[†] This project has been made possible in part by a grant from the Cisco University Research Program Fund, an advised fund of Silicon Valley Foundation.

2. We propose a suite of translation tools enabling policy verification. More specifically, we introduce how to perform policy verification on a workflow-like policy specification. We rely on a policy implementation based on Rego, a high-level declarative language built for expressing complex policies.
3. Finally, we conduct a thorough performance evaluation. We verify that deployed policies match their specification in a very reasonable time, even for large workflows having a substantial number of rules.

Related Works Hughes and Bultan [HB08] as well as Bera et al. [BGD10] propose automatic verification of access control policies against a set of properties, using a SAT solver. Even though metagraphs have emerged as one of the most suited representation to reason about policies, only few existing works rely on them. Hamza et al. [HRG$^+$20] used metagraphs to model traffic profiles in IoT devices. Closer to our contribution, Ranathunga et al. [RRN20] use specific metagraph properties to detect redundancies and conflicts in network policies for distributed firewalls. Contrary to our work, they do not verify deployed policies against specifications.

After introducing metagraphs (Sec. 2), we show how to perform policy verification with them (Sec. 3). We evaluate our approach (Sec. 4), and finally conclude (Sec. 5).

2 Background : from Graphs to MetaGraphs

A metagraph is a generalized graph theoretic structure like directed hypergraphs, which is defined as a collection of directed set-to-set mappings. Each set (containing subsets or elements) in the metagraph is a vertex, and directed edges represent the relationship between sets. More formally, a metagraph can be defined as follows :
A metagraph $S = \langle X, E\rangle$ is a graphical construct specified by a generating set X and a set of edges E defined on the generating set. A generating set is a set of elements $X = \{x_1, x_2, ..., x_n\}$, which represent variables of interest. An edge e is a pair $e = \langle V_e, W_e\rangle \in E$ consisting of two sets, an invertex $V_e \subset X$ and an outvertex $W_e \subset X$.

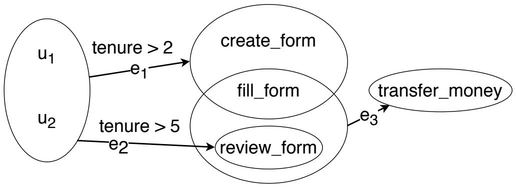

Figure 1: A simple example of conditional metagraph to model the following question : what are the necessary tasks for employees to perform a bank transfer ?

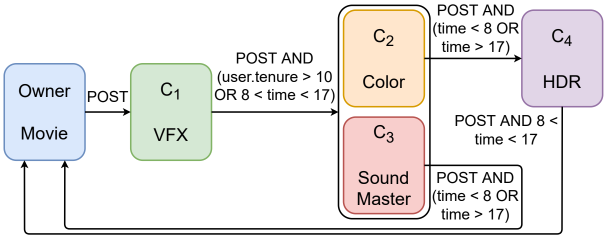

Figure 2: Movie workflow : special effects apply before color tuning and sound mastering. HDR is set up last.

Fig. 1 illustrates a conditional metagraph, augmented with propositions representing policy constraints. Overall, it represents the necessary tasks for employees to perform a bank transfer. Edges (e_1, e_2, e_3) relate sets of employees (u_1, u_2) and tasks ($create\_form$, $fill\_form$, $review\_form$, $transfer\_money$). They contain an arbitrary number of propositions, e.g. $tenure > 2$ for e_1. Using an edge depends on the evaluation of its proposition, e.g. both employees can $create\_form$ and $fill\_form$ via e_1 only if they have more than two years of experience.

3 Policy Verification Using Metagraphs

By modeling the high-level policy specification as well as the translated policy implementation as two metagraphs, we can compare both in order to track (distributed) deployment errors. When specification and implementation metagraphs match, the policy implementation has been correctly translated from the policy specification. If they do not match, the metagraphs are not equivalent : errors occurred during the refinement and/or deployment. Fig. 3 summarizes our approach.

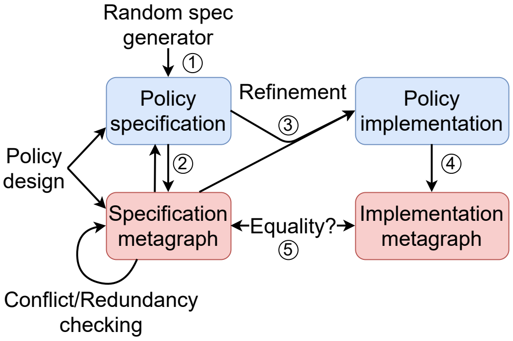

Figure 3: Enabling policy verification using metagraphs. We propose 4 tools : ① RandomWorkflowSpecGenerator ; ② ③ SpecToRego ; ④ RegoToMetagraph ; ⑤ SpecImplEquivalence.

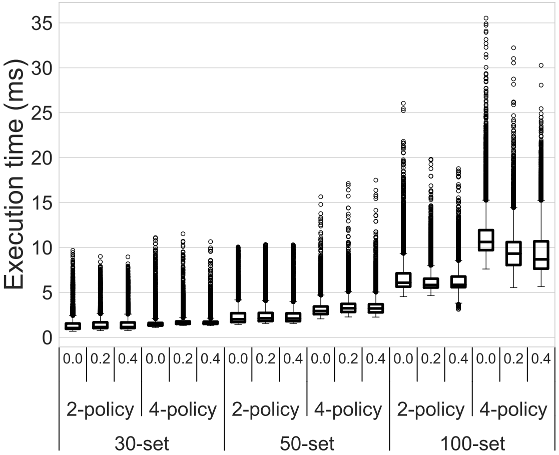

Figure 4: Execution time of our matching algorithm according to several different parameters. Values for 10-set and 20-set omitted for brevity.

For our purposes and evaluations, we consider the verification of policies enforcing workflows. For instance, in the post-production stage of making a movie, the owner of the data at risk wants to employ other companies to edit its video and audio components ; more specifically, the owner may want to add special effects (VFX), tune colors, set up High Dynamic Range (HDR) and master the audio. The intent of the owner can be modeled under the form of a workflow, as depicted in Fig. 2. Propositions on the edges constrain the communications. For example, C_1 can only send data to C_2 and C_3 if the communication is a `POST` request, and either the tenure of the user is greater than 10, or the request happens between 8 AM and 5 PM.

This form of policy specification can be generically expressed as a list of rules : each describing an edge of the metagraph, as a triplet of the form of ⟨*source, destination, policy*⟩. To implement those policies, we consider Rego, a high-level declarative language built for expressing complex policies. Once we have the policy specification and the implementation, we transform both into conditional metagraphs. For this, we develop three generic policy translators : from specification (raw) to specification metagraph, from specification to implementation (Rego), and from implementation to implementation metagraph.

Policy specification into a conditional metagraph ② We need to define the variable set, the proposition set and the edge set defining the conditional metagraph. To this end, we parse a policy specification file providing rules in the raw format.

Policy implementation (i.e. Rego) into a conditional metagraph ④ We use ANTLR4, Another Tool for Language Recognition, which is a parser generator used for translating structured files. After constructing our lexer rules and parser grammar for Rego, we were able to generate the Abstract Syntax Tree (AST) for any Rego policy file. The AST is used to generate the metagraph.

Comparing metagraphs ⑤ To compare metagraphs, we tag edges in one metagraph upon a match with edges in the other metagraph until we have treated all edges. Non tagged edges correspond to errors/mistakes in the implemented policies, singled out by our comparison. This approach assumes identical literals in both representations.

4 Performance analysis

To profile our policy verification algorithm, we measure the time required to compare the specification and implementation metagraphs.

Methodology To obtain generally representative results, we chose to generate random workflows for the policy specifications (instead of relying on few small real cases). Values for those parameters were chosen to be in line with real-world policies [RRN20].

— **Size of the workflow**, i.e., number of elements in the generating set : 10, 20, 30, 50 or 100.

— **Policy size**, i.e., number of conditional propositions on each edge for the policy : 2 or 4.

— **Error rate**, i.e., fraction of errors we generate in propositions of the metagraph. A value of 0.4 means that 40% of the elements/propositions of the metagraph are tampered with ; we consider the following error rates : 0.0, 0.2 and 0.4.

Overall, we obtain 300 different policy specifications and 27,000 policy implementations (300 specifications, 3 error rates, 30 repetitions). After generating the policy implementations, we translate those into metagraphs to finally perform the comparison.

Evaluation For each of the 27,000 scenarios, we measure the cumulative time of both sorting and matching for 30 runs, ending up with a total of 810,000 measures. We remove outliers due to peak machine load (Z-score superior to three) : 9367 values out of 810000 (1.16%). We ran our measurements on commodity hardware with an Intel Core CPU 3.5-GHz, 16GB of RAM.

Fig. 4 shows that the error rate produces a negligible effect on the time required for the comparison, whereas the computing time increases with the number of elements in the generating set (as well as with the policy size). The main complexity of our metagraph comparison is inherent to edge sorting : $O(m \cdot log(m))$, with m the number of edges. Predicting execution time using an OLS regression, we found that the number of edges ($\beta = 0.0025$; $p < 0.001$) is a significant predictor. The overall model fit is : $R^2_{adj} = 0.868$, with the *post hoc power analysis* indicating a power greater than .999. We argue that our verification technique can be efficiently implemented as long as the number of rules is reasonable. The complete results and code are publicly available [‡].

5 Conclusion

In this paper, we detailed to what extent metagraphs are appropriate structures to naturally model access control policies. Their formal and graphical foundations guide the reasoning to manipulate such policies. Those constructs are suitable modeling tools, and while they enable policy analysis, we have proposed here to use them for a practical verification of access control policies.

Références

[BGD10] Padmalochan Bera, Soumya Kanti Ghosh, and Pallab Dasgupta. Policy based security analysis in enterprise networks : A formal approach. *IEEE Transactions on Network and Service Management*, 7(4) :231–243, 2010.

[HB08] Graham Hughes and Tevfik Bultan. Automated verification of access control policies using a sat solver. *International journal on software tools for technology transfer*, 10(6) :503–520, 2008.

[HRG+20] Ayyoob Hamza, Dinesha Ranathunga, Hassan Habibi Gharakheili, Theophilus A Benson, Matthew Roughan, and Vijay Sivaraman. Verifying and monitoring iots network behavior using mud profiles. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[RRN20] Dinesha Ranathunga, Matthew Roughan, and Hung Nguyen. Verifiable policy-defined networking using metagraphs. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[‡] See https ://zenodo.org/record/4426675.