Formal Verification of Authorization Policies for Enterprise Social Networks using PlusCal-2

Sabina Aktar, Ehtesham Zahoor, Olivier Perrin

Formal Verification of Authorization Policies for Enterprise Social Networks using PlusCal-2

Sabina Akhtar[1], Ehtesham Zahoor[2], and Olivier Perrin[3]

[1] Bahria University, Islamabad, Pakistan
sabina.buic@bahria.edu.pk
[2] Secure Networks and Distributed Systems Lab (SENDS),
National University of Computer and Emerging Sciences, Islamabad, Pakistan
ehtesham.zahoor@nu.edu.pk
[3] Université de Lorraine, LORIA BP 239 54506
Vandoeuvre-lès-Nancy Cedex, France
olivier.perrin@loria.fr

Abstract. Information security research has been a highly active and widely studied research direction. In the domain of of Enterprise Social Networks (ESNs), the security challenges are amplified as they aim to incorporate the social technologies in an enterprise setup and thus asserting greater control on information security. Further, the security challenges may not be limited to the boundaries of a single enterprise and need to be catered for a federated environment where users from different ESNs can collaborate. In this paper, we address the problem of federated authorization for the ESNs and present an approach for combining user level policies with the enterprise policies. We present the formal verification technique for ESNs and how it can be used to identify the conflicts in the policies. It allows us to bridge the gap between user-centric or enterprise-centric approaches as required by the domain of ESN. We apply our specification of ESNs on a scenario and discuss the model checking results.

Key words: Enterprise Social Network, Formal Verification, Model checking, PlusCal-2, TLA$^+$, TLC

1 Introduction

Information security research has been a highly active and widely studied research direction. In the last decade, the widespread usage of social networks has further amplified the need to protect user information. Recently, a number of organizations have started to use social networks as a tool for enhancing collaboration amongst their employees. Social network in an enterprise setting is termed as Enterprise Social Network (ESN). Implementations can be either homegrown systems built internally or tailoring existing social network implementations for an enterprise setting. One such implementation is Yammer[1] and one example

[1] https://www.yammer.com

of ESN usage for increase in productivity is Yammer usage within Boral Limited, a building and construction materials company. One important aspect to emphasize here is that an ESN is not just deployment of a social network in an enterprise environment. It may not be limited to the boundaries of a single enterprise and users from different ESNs can collaborate in a federated environment. For instance, Yammer has a concept of external groups, where external partners can collaborate with members of an organization. Addressing ESN security and privacy issues is a challenging domain as it requires to consider the perspective of both federated enterprises and the ESN users.

The security policy of an organization helps to better prepare for and address these security challenges. It specifies a high level specification of how to implement security principles and technologies. For instance, the Authentication policy of an organization specifies which users are allowed to use its services. Once a user has been authenticated, the authorization process allows to determine who can access what resources, under what conditions, and for what purpose. The authorization process can be based on temporal aspects and may involve delegation. An authorization policy is a high level description of access rules that will determine what rights an authenticated user has. While the federated authentication has been an active area of research with approaches such as SAML providing SSO for federated environments, the authorization capabilities, challenges and solutions are not thoroughly explored. The challenges are amplified in the case of ESN as authorization policies are not limited to the case of a single enterprise. We need to cater for authorization policies for each enterprise within the federation as well as user policies, as and ESN is essentially a social network allowing users to share and collaborate.

In this paper we address the challenges related to one important class of security policies, called the access control or authorization policies in a federated ESN setting. From an enterprise point of view, authorization policies are very important to express the access on resources. Their need is even more evident in the context of enterprise federation, where different enterprises trust each other. With the advent of Web 2.0, information sharing allowed users to collaborate more easily, thanks for the use of the social technologies. In this context, approaches such as OAuth and Lockr are based on the user perspective, rather than the classical enterprise point of view. In an ESN, the users share and access resources and at the same time, the enterprise imposes some polices to be taken care of. We address the problem of federated authorization for the ESNs and present an approach for combining user level policies with the enterprise policies. We present the formal verification approach for ESNs and detail how it can be used to identify the conflicts in the policies. It allows us to bridge the gap between user-centric or enterprise-centric approaches as required by the domain of ESN. We use PlusCal-2 [1, 12] as a formal modeling language for specifying the authorization policies. Formal methods are being used extensively at large scale distributed systems, for instance Amazon uses TLA$^+$ and PlusCal to model and verify AWS services such as *S3*, *DynamoDB* and *EBS*. The proposed specification approach is generic and expressive and is also based on TLA$^+$.

2 Related work

In this work, we address the challenges associated with authorization policies in the context of enterprise federation within an Enterprise Social Network[17]. Two major research directions in this domain have been to investigate the authorization from the enterprise or federation point of view, using approaches such as XACML, and the other is to address the challenge from a user point of view, with approaches such as OAuth and Lockr. Authorization issues in ESN is a relatively unexplored domain.

Access control and authorization in general has been an active research areas since decades. In this context, the focus of traditional research approaches has been the RBAC model [15] and its variations. Task based access control (TBAC) considers task based contextual information [18]. Team based access control [19, 8] introduces the concept of team to accomplish collaborative activities. Even though RBAC is a well defined model, it suffers from *role explosion* as too many rules (may even surpass the number of users) may need to be managed. In distributed environment (more precisely in an enterprise federation), RBAC may also introduce interoperability concerns as the semantics of different roles may be inconsistent across domains. Some approaches have investigated the use and challenges for RBAC in a distributed environment [6, 16]. One such approach is D-Role [14] in which authors propose extensions to XACML for distributed roles. In contrast to RBAC models, the Attribute based access control (ABAC) model is based on the attributes [10]. The resources, subjects and environment have attributes, and the policy rule is a boolean function on these attributes. ABAC provide more flexibility and expressiveness than RBAC in term of rules definition as a role itself can be an attribute in an ABAC model. For ESN authorization decisions, ABAC is thus the preferred model and it subsumes RBAC model.

XACML (eXtensible Access Control Markup Language) is a declarative, XML-based access control policy language for managing access to resources, based on ABAC model. As XACML is verbose and based on XML, a number of approaches to provide formal semantics of XACML using formal logic have been proposed [7, 11]. Further, a number of approaches have been proposed that build upon XACML for its usage in collaborative and distributed environments[23]. These include [5] in which the authors propose a distributed device access control architecture called *MPABAC*. In [21] authors have developed a formal policy language BelLog that can express both delegation and composition operators.

The second class of authorization policies specification is when the user determines the access for their resources. The most prominent approach being the OAuth [9] which allows users to share their personal resources, such as images, hosted on one Website with other sites without giving them their username and password. User-Managed Access (UMA)[2] is another user centric and it provides services for authorization, monitoring and changing data sharing. Lockr [20] is an access control system based on social relationships. Users can base ACLs for applications based on social relationships as defined in some social network.

[2] http://docs.kantarainitiative.org/uma/draft-uma-core.html

When it comes to specifying authorization policies within a social network, in [2] authors presented Persona, a system that promotes data privacy by allowing users to use attribute encryption for specifying policies themselves and not relying on the social network.

In the context of ESN, neither XACML nor User centric authorization mechanisms are sufficient on their own. Using XACML, an enterprise can easily define a set of policies for subjects within the enterprise, but it is more difficult for a user to define its own policy for managing access to its own data. User centric authorization mechanisms are well adapted for user defined access controls. However, in the context of ESN, it is not just about users, it is also about enterprises, and even more complex, enterprises federation. In this context, it is mandatory to allow the users to define their own policies, and to combine them with the enterprise ones, and to be able to decide if an access is granted or not. Our work builds upon our previous work in handling temporal, trust and delegation aspects in distributed environments [24, 3]. In this work we provide an approach that combines user level policies with the enterprise policies to bridge the gap between traditional federated authorization approaches.

3 Motivating Example

For the motivating example, we consider the case of a large municipality (we will call it a *commune* as in France) that is willing to take benefit from the rise of ESNs[3]. We consider a situation such as child protection case, where the relevant information could be scattered across different departments such as social services, education, healthcare, and police agencies. The information sharing between these different departments in an ESN would allow timely decisions to be made but comes with the challenge for managing such collaboration in a secure manner. We consider that each department has its own associated security policy and belong to a federation where they trust each other.

We further consider that a new child protection case has been forwarded to an employee at the commune, Alice, and she needs to create a case file and gather information from different departments to reach a decision about the case. As a standard practice she wants to give permissions to another person in her team, Genny, working on the case in another department. We assume that the case requires access to some database, thus Alice would grant permission to access it. However, there might be another policy already in the enterprise policies or user defined policies that doesn't allow Genny to access that particular database. These conflicts effect the overall performance and reliability of an enterprise social network where it is a major concern for the users. Our idea is to formally verify the set of policies before actually deploying the ESNs or anytime during the scheduled update of the network. This would reduce the need for the decision making algorithms to be invoked at the time of the request.

[3] Idea is based on real world example as briefly discussed in Social Networking In The Enterprise: Benefits And Inhibitors, a commissioned study conducted by Forrester Consulting on behalf of Cisco Systems

4 Proposed Approach

Our approach combines the two major classes of authorization policies; user level and enterprise level, in a federated environment and provides a federation-level authorization process. We use finite-state model checking [4] technique, used for formal verification of distributed systems, to address the challenges raised by the different enterprises in a federation. It allows to decide automatically whether the invariants and properties hold for the specified system or not. They are verified for finite instances of systems, described in a formal modeling language. We use PlusCal-2 [1, 12] as a formal modeling language for the formally specifying the enterprises in a federated environment. It is intended to provide the programmers the platform where they can specify their systems and generate the TLA$^+$ specifications. TLA$^+$ is a formal specification designed by Leslie Lamport provides a method for specifying the systems. It is supported by the model checker TLC [22] that verifies the system for the specified set of properties and invariants. PlusCal-2 features non-determinism, mathematical abstractions, and user-specified grain of atomicity. It emphasizes on the abstract model of the system instead of focusing on its efficient execution. Compared to PlusCal algorithm language [13] by Leslie Lamport, it introduces several other statements and flexibility for the programmer and reduces the dependency on TLA$^+$. They will be used in the specification of enterprise social networks to demonstrate their need in system specification. In the proposed architecture, policies are specified using ABAC model. As an ESN builds upon the notion of sharing and collaboration, the users can access authorization policies with the resources they are in control of (such as the ones they have created or the control has been delegated to them). This user level policy specification approach is consistent with the user-controlled authorization schemes that aim to put user in the control of authorization process.

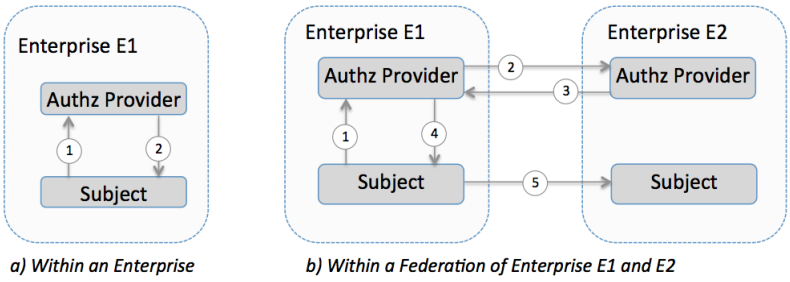

Fig. 1. Policies evaluation

However in an ESN setting, the collaboration is under the control of an enterprise and it has its own policies about the access to resources, for instance the enterprise policy that some critical resources can be accessed within some time frame. Thus the proposed approach allows to specify the policies also at

the enterprise level. Indeed the two set of policies may be inconsistent and a set of decision algorithms can decide the resolution scheme in case of conflict. The policy evaluation at an enterprise level is shown in the Figure 1-a. Inside an enterprise, when a user, *user*, tries to access a resource, *res*, its request, R, along with, its attributes and the attributes of the resource, composed formally as *(user, res)*, is sent to the Authorization Provider (AzP). The task of Authorization Provider (AzP) is to gather the user defined policies, UP and the enterprise policies, EP for the resource in order to process the request. Formally, we define a policy, P, as

$$P \triangleq \textit{(user, res, action)}$$

where an *action* can be to allow or deny the resource, *res*, for the user. Once, the Authorization Provider (AzP) has gathered all the policies, it filters out a subset of policies, SP, concerning the request, R, *(user, res, X)* from the set of policies for user defined policies, UP and enterprise policies, EP

$$SP \triangleq R \cap (UP \cup EP)$$

X in the request R allows to ignore all the actions in the set of policies and selects only the policies from entire set of policies related to the user and the resource. If the set SP is empty this means their is no policy written for that specific user and resource in the set of policies. In this case the request will be denied by the Authorization Provider (AzP). If there is only a single policy in the set SP, then in that case Authorization Provider (AzP) can take a decision immediately by applying the action mentioned in the policy. However, if there are more than one policies in the set then there might be conflict and the AzP will have to resolve it. AzP will deny the request if one of the policies in the set denies the resource formally written as

$$\exists\ p \in SP : p.action = \text{"}deny\text{"} \rightarrow \ \textit{Deny request}$$

The Authorization Provider (AzP) will allow the resource access if all of policies allow the resource for the user. It can be formally specified as

$$\forall\ p \in SP : p.action = \text{"}allow\text{"} \rightarrow \ \textit{Allow access}$$

When a user wants to access a resource that is not within the enterprise but belongs to another enterprise within the same federation, the request is first validated by the local AzP of the enterprise to which the user belongs. The enterprise checks if the request conforms to the enterprise policy, for instance to enforce that the request might be not allowed for the users with a specific set of attributes. The request is then forwarded to the AzP of the enterprise to which the resource belongs and the AzP checks if the request conforms to the user specified resource policies and the enterprise policies, Figure 1-b.

5 ESNs Specification in PlusCal-2

Our specification[4] of enterprises in a federation allows us to identify the inter enterprise or intra enterprise level conflicts in the policies. Figure 2 shows the

[4] Available at https://github.com/sabinaakhtar/ESNSpecs

```
1  algorithm ESNSpec
2  extends Naturals, Sequences , TLC, FiniteSets
3  variable  EP1Users = {"Alice", "Tim"}, EP2Users = {"Bob", "Genny"},
4            AllUsers = EP1Users ∪ EP2Users,
5            EP1Res = {"R1", "R2"}, EP2Res = {"R3", "R4"},
6            AllRes = EP1Res ∪ EP2Res,
7            ReqPool = [u ∈ AllUsers, r ∈ AllRes ↦ [status ↦ {}]]
8  process Enterprise1[1]
9  variable  rules = 0, status = 0, statusRecord = 0,
10           Policies = [u ∈ AllUsers ↦
11                 CASE (u = "Alice") →  {[res ↦ "R2", act ↦ "deny"], [res ↦ "R3", act ↦ "allow"]}
12                      □ (u = "Tim") → {[res ↦ "R4", act ↦ "deny"], [res ↦ "R3", act ↦ "allow"]}
13                      □ (u = "Bob") → {[res ↦ "R3", act ↦ "allow"], [res ↦ "R2", act ↦ "deny"]}
14                      □ (u = "Genny") → {[res ↦ "R4", act ↦ "deny"], [res ↦ "R4", act ↦ "deny"]} ]
15 begin
16     atomic
17     for u ∈ AllUsers  for r ∈ AllRes
18                 rules := Policies[u];
19                 for rule ∈ rules
20                      if rule.res = r then
21                           statusRecord := ReqPool[u,r];
22                           statusRecord.status := statusRecord.status ∪ {rule.act};
23                           ReqPool[u,r] := statusRecord;
24                      end if; end for; end for; end for;
25     end atomic;
26 end process
27 process Enterprise2[1]
28 variable  rules = 0, status = 0, statusRecord = 0,
29           Policies = [u ∈ AllUsers ↦
30                 CASE (u = "Alice") →  {[res ↦ "R2", act ↦ "deny"], [res ↦ "R1", act ↦ "allow"]}
31                      □ (u = "Tim") → {[res ↦ "R1", act ↦ "deny"], [res ↦ "R2", act ↦ "allow"]}
32                      □ (u = "Bob") → {[res ↦ "R1", act ↦ "deny"], [res ↦ "R4", act ↦ "deny"]}
33                      □ (u = "Genny") → {[res ↦ "R1", act ↦ "deny"], [res ↦ "R4", act ↦ "allow"]} ]
34 begin
35     atomic
36     for u ∈ AllUsers  for r ∈ AllRes
37                 rules := Policies[u];
38                 for rule ∈ rules
39                      if rule.res = r then
40                           statusRecord := ReqPool[u,r];
41                           statusRecord.status := statusRecord.status ∪ {rule.act};
42                           ReqPool[u,r] := statusRecord;
43                      end if; end for; end for; end for;
44     end atomic;
45 end process end algorithm
46 invariant ∀ u, ∈ AllUsers, r ∈ AllRes : (ReqPool[u,r].status ∩ {"allow", "deny"}) ≠ {"allow", "deny"}
```

Fig. 2. Specification of a federation with two enterprises in PlusCal-2.

specification of a federation with two ESNs. The specification can be used for one or more than one enterprises. It only requires to add the information about the new enterprise along with the set of policies for each one added in the federation. In PlusCal-2, the specification starts with the keyword **algorithm** followed by the name of the specification. Then we have standard modules for specific constructs used from TLA$^+$ that are required by the TLC model checker. The global variables are defined at line 3. In our specifications, we assume that we have two enterprises *Enterprise1*, *Enterprise2* representing Alice's and Genny's departments respectively. Then we specify their users globally as *EP1Users* and *EP2Users*. *AllUsers* is the union of all the users in all the enterprises. Each

enterprise owns its own set of resources, specified in our specification as *EP1Res* and *EP2Res*. Similarly, we have *AllRes* as a union of all the resources. The two processes *Enterprise1* and *Enterprise2* will represent Authorization providers in our specifications. To keep the specification at an abstract level, we removed the role of a user that sends the request for the resources and replaced it with a pool of requests that includes all possible requests that can be made by any user in a federation. At line 7, we define *ReqPool* as function whose domain is in *AllUsers* and *AllRes* specifies the pool of requests that can be made by the users. Its range is represented by a record with a field *status* that is initially an empty set. Its purpose is to contain all the actions that can be applied to a request by any policy in an enterprise/federation. The functionality of the enterprises remains same except for the policies that they have defined at their own level. The policies are specified at line 10 for *Enterprise1* as a function whose domain is in *AllUsers* and range represents a set of rules for each users corresponding to the resources. For example, user *Tim* has an access to some resource *R3* but does not have an access to resource *R4* specified a line 12.

The specification of the enterprise is at line 15 till line 25. For each possible request, it consults its own set of policies and if it finds a policy it updates the field *status* of the *ReqPool* with action specified in the policy. This role of an enterprise is enclosed in a construct **atomic** to avoid the interleavings with the other process. This makes our specification more coarse grained to avoid the state space explosion problem. The specification ends with an invariant at line 46 of Figure 2. The TLC model checker verifies the invariant at each state and if the invariant is violated, it stops the execution and shows the counter example. The invariant specifies that for each user, $u \in AllUsers$, and for each resource, $r \in AllRes$, the specified formula must hold and the intra or inter enterprise policies must never allow and deny the access at the same time.

As per our motivating example introduced a scenario where an employee, Alice, was working on a case from a department, in collaboration with another employee, Genny, of another department. They had to access a database, lets call it *R4*, to complete the processing of the case. Now, even if Alice grants access of this resource to Genny, there might be a possibility that she cannot access this resource. We can formally verify this scenario through our specifications and identify the policies with the conflict using the counter example. In figure 2, we have modeled such a scenario to find out that counter example. PlusCal-2 compiler translates our specifications to TLA$^+$ specifications. Then we model checked those specifications using TLC model checker and it stops with the following message

```
Error: Invariant Inv0 is violated. The behavior up to this point is:...
```

Inv0 is the name of the invariant used in the TLA$^+$ specifications for the invariant at line 46 of Figure 2. The TLC model checker gives a counter example that clearly shows where it was voilated. In the last state of the counter example, it shows the state of the variable *ReqPool*.

```
... /\ ReqPool = ( <<"Alice", "R1">> :> [status |-> {"allow"}] @@ ...
  <<"Tim", "R1">> :> [status |-> {"deny"}] @@ ...
  <<"Bob", "R1">> :> [status |-> {"deny"}] @@ ...
  <<"Genny", "R4">> :> [status |-> {"deny", "allow"}] ) ...
```

The output clearly show that the policies had conflict for request by user *Genny* for the resource *R4*. However, if there will be no such conflict in the policies, then this specific invariant will never be violated. Similarly, we can use formal verification techniques for identifying various other issues in the federated environment as well. The PlusCal-2 compiler has not been released yet as it is in testing phase and it will be released in few months time. The current version of PlusCal-2 compiler along with the TLC model checker are also available at InriaForge[5] .

6 Conclusion

In this paper, we have presented a formal specification of federated environment and how formal verification can be used for the authorization challenges associated with Enterprise Social Networks (ESNs). The traditional approaches for authorization are either user-centric or enterprise-centric and we have advocated to bridge the gap between them. In contrast to traditional XML based authorization policy specification languages, our approach is formal and provides a precise, expressive, flexible and non-ambiguous representation. It also allows for reasoning about authorization policies (e.g. to find inconsistencies or hard constraints). We have also presented the model checking results for our specifications and discussed how these tools can be used to identify conflicts by stating invariants and properties. In future, we plan to automate the translation of actual user defined and enterprise level policies to PlusCal-2 so that we can formally verify them and identify conflicts in the policies. We also plan to investigate other challenges and issues in Enterprise Social Networks from the perspective of formal verification techniques.

References

1. Sabina Akhtar, Stephan Merz, and Martin Quinson. A high-level language for modeling algorithms and their properties. In *13th Brazilian Symp. on Formal Methods (SBMF 2010)*, Natal, Brazil, 2010. Springer.
2. R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. In *SIGCOMM*, 2009.
3. Ahmed Bouchami, Olivier Perrin, and Ehtesham Zahoor. Trust-based formal delegation framework for enterprise social networks. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland*, 2015.
4. Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, Cambridge, Mass., 1999.

[5] https://gforge.inria.fr/projects/pcal2-0/

5. Feng Liang et al. An attributes-based access control architecture within large-scale device collaboration systems using xacml. In *Green Communications and Networks*. Springer, 2012.
6. Tao Wu et al. A distributed collaborative product design environment based on semantic norm model and role-based access control. *J. Network and Computer Applications*, 2013.
7. Tuan Ngoc Nguyen et al. Towards a flexible framework to support a generalized extension of xacml for spatio-temporal rbac model with reasoning ability. In *ICCSA (5)*, 2013.
8. Christos K. Georgiadis, Ioannis Mavridis, George Pangalos, and Roshan K. Thomas. Flexible team-based access control using contexts. In *SACMAT*, 2001.
9. Dick Hardt. The oauth 2.0 authorization framework. 2012.
10. Vincent C Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to attribute based access control (abac) definition and considerations. *NIST Special Publication*, 800:162, 2014.
11. Vladimir Kolovski, James A. Hendler, and Bijan Parsia. Analyzing web access control policies. In *WWW*, pages 677–686, 2007.
12. Leslie Lamport. *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.
13. Leslie Lamport. The PlusCal algorithm language. In *6th Intl. Coll. Theoretical Aspects of Computing (ICTAC 2009)*, Kuala Lumpur, Malaysia, 2009. Springer.
14. Hannah K. Lee and Heiko Luedemann. lightweight decentralized authorization model for inter-domain collaborations. In *SWS*, 2007.
15. Joon S. Park, Ravi S. Sandhu, and Gail-Joon Ahn. Role-based access control on the web. *ACM Trans. Inf. Syst. Secur.*, 4(1):37–71, 2001.
16. C. Ruan and V. Varadharajan. Dynamic delegation framework for role based access control in distributed data management systems. *Distributed and Parallel Databases*, 2014.
17. Gerald Stei, Sebastian Sprenger, and Alexander Rossmann. Enterprise social networks: Status quo of current research and future research directions. Springer, 2016.
18. R. K. Thomas and R. S. Sandhu. Task-based authorization controls (tbac): A family of models for active and enterprise-oriented autorization management. In *DBSec*, 1997.
19. Roshan K. Thomas. Team-based access control (tmac): a primitive for applying role-based access controls in collaborative environments. In *ACM Workshop on Role-Based Access Control*, pages 13–19, 1997.
20. Amin Tootoonchian, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: better privacy for social networks. In *CoNEXT*, 2009.
21. Petar Tsankov, Srdjan Marinovic, Mohammad Torabi Dashti, and David A. Basin. Decentralized composite access control. In *POST*, 2014.
22. Yuan Yu, Panagiotis Manolios, and Leslie Lamport. Model checking TLA+ specifications. In *Correct Hardware Design and Verification Methods (CHARME'99)*, Bad Herrenalb, Germany, 1999. Springer.
23. Ehtesham Zahoor, Zubaria Asma, and Olivier Perrin. A formal approach for the verification of AWS IAM access control policies. In *6th European Conference on Service-Oriented and Cloud Computing , ESOCC 2017*, pages 59–74, 2017.
24. Ehtesham Zahoor, Olivier Perrin, and Ahmed Bouchami. CATT: A cloud based authorization framework with trust and temporal aspects. In *CollaborateCom 2014, Miami, Florida, USA*.