Decentralised Runtime Monitoring for Access Control Systems in Cloud Federations

Md Sadek Ferdous, Andrea Margheri, Federica Paci, Mu Yang, Vladimiro Sassone
University of Southampton
{s.ferdous; a.margheri; f.m.paci; mu.yang; vsassone}@soton.ac.uk

***Abstract*—Cloud federation is an emergent cloud-computing paradigm where partner organisations share data and services hosted on their own cloud platforms.**

In this context, it is crucial to enforce access control policies that satisfy data protection and privacy requirements of partner organisations. However, due to the distributed nature of cloud federations, the access control system alone does not guarantee that its deployed components cannot be circumvented while processing access requests.

In order to promote accountability and reliability of a distributed access control system, we present a decentralised runtime monitoring architecture based on blockchain technology.

***Index Terms*—Access Control, Cloud Federation, Runtime Monitoring, Blockchain, Security.**

I. Introduction & Motivation

The advent of cloud computing has enabled new collaborative scenarios in which users and organisations share resources, information and services in order to achieve a common business goal or interest. An instance of this trend is provided by federated clouds [1], [2], [3] which are created dynamically to achieve such a business goal. However, its adoption can be hindered by security concerns such as who can access the shared resources, for what purposes, and what are the potential consequences of granting access.

An approach typically adopted to address these concerns is to deploy a federation-wise access control system to enforce access control policies attached to the federation by the resource owner [4]. This means that there will be distributed components that receive, exchange and process access requests and their corresponding access decisions with the possibility of being compromised. Indeed, it is possible that the components are compromised so that access requests or responses are modified, or the policies and the evaluation process are altered by a malicious user or software to gain unauthorised access to federated resources.

To detect such attacks, this paper proposes a runtime monitoring architecture for distributed access control systems: *Decentralised Runtime Access Monitoring System (DRAMS)*. This is achieved by including distributed logging probes which sense access control activities and intercept access requests and decisions. These logs are then processed to check the integrity of the monitored components. A key feature of DRAMS is that not only it is able to detect attacks to the components involved in an access control decision, but it is also resilient to attacks targeting the integrity of the logs or of the monitoring components. To achieve this, DRAMS leverages *blockchain* technologies [5] as an infrastructure for storing logs and performing non-repudiable monitoring checks. Blockchain is a novel technology that, besides its application to cryptocurrency, features fascinating properties of data integrity, distribution and control along with the support for so-called *smart-contracts*, which are arbitrarily complex programs deployed and executed autonomously on a blockchain.

DRAMS is proposed upon the access control system of *Federation-as-a-Service (FaaS)* [3], a recently proposed approach to cloud federation devised and developed by the H2020 project SUNFISH [6]. The FaaS access control system is based on the eXtensible Access Control Markup Language (XACML) [7] consisting of *Policy Decision Point* (PDP) and *Policy Enforcement Point* (PEP). Indeed, once a PEP receives a user's request, it forwards it to the PDP, which calculates the access decision. The decision is then enforced by the PEP.

In FaaS, the XACML components are deployed along with the tenants (i.e., virtual spaces of computing resources belonging to different clouds) underlying a FaaS federation. The PDP and the policy management is placed in the infrastructural tenant (i.e., the tenant owned by all federation clouds that enable the FaaS functionalities). PEPs are instead deployed in a distributed manner on the tenants edge, thus to intercept all communications, interact with the distributed sources of information and enforce the calculated accesses.

In what follows, we present DRAMS architecture and discuss the main challenges in implementing such architecture on top of blockchain technology.

II. DRAMS Architecture

DRAMS rests on a smart-contract blockchain to store logs and perform monitoring checks on them. Additionally, DRAMS is equipped with a formally-grounded policy analyser that evaluates whether an access decision is correct according to the semantics of the available policies. Its architecture is illustrated in Figure 1 which has the following components:

- *Logger*: consisting of *Probing agents* for intercepting and forwarding data to create access logs and *Logging Interface* (LI), which exposes to agents endpoints for storing data and managing security alert events generated by smart-contracts.
- *Smart-contract blockchain*: it is the smart-contract blockchain system storing and comparing logs, using

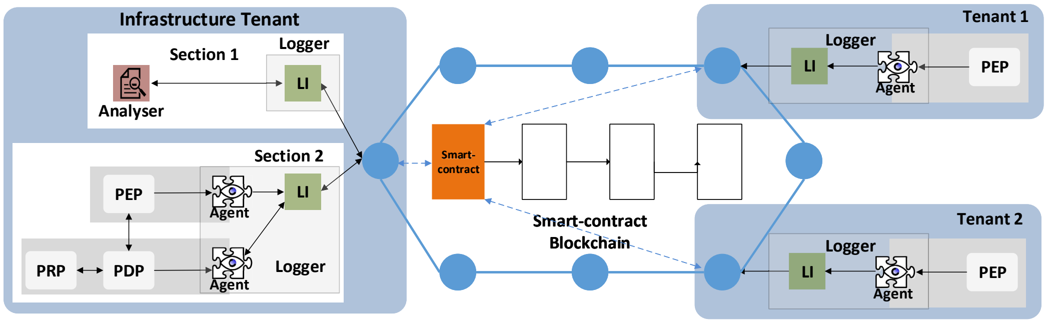

Figure 1: DRAMS architecture deployed on the access control system of a FaaS cloud federation (where 'Section i' stands for a set of computing resources belonging to a cloud 'i', while LI stands for Logging Interface)

expressly devised algorithms, thus to mitigate threat that modifies access control decisions or responses.

- *Analyser*: it checks the correctness of the access decisions calculated for the intercepted requests, with respect to the policies currently in force in the system to mitigate threats that alter the policy enforced or the policy evaluation process.

Indeed, the key element of the system is the blockchain infrastructure: it is connected via the LI, placed in each tenant, to all the other components. The agents are distributed in each tenant where the monitored access control components (i.e., PDP and PEP) are placed.

The LI also provides symmetric encryption and decryption functions, which are exploited by the other components to store/retrieve encrypted data in/from smart-contracts. Indeed, as data stored on a blockchain are visible to all users, encryption is used to protect data confidentiality.

The Analyser is a standalone entity logically placed within the Infrastructural Tenant, but deployed within a different cloud section with respect to the access control components. It dynamically consumes and evaluates the gathered logs to ensure the correct enforcement of access decisions. On the base of a logical representation of the access control policies evaluated by the PDP, the Analyser checks if for a given request the calculated response is the expected one using the formally-grounded analysis framework for XACML presented in [8].

III. Discussion

We discuss here the main challenges in implementing the DRAMS architecture.

System Integrity. Even though the smart-contract of DRAMS is immutable, the integrity of the other components, e.g. the LI, cannot be guaranteed by-design, because they are deployed off-chain. Similarly, as all the LI instances share a symmetric key K, its management is of paramount importance.

To mitigate both difficulties, we can introduce a trusted hardware platform (e.g., Trusted Platform Module) within the system. On the one hand, it can be leveraged to store the symmetric keys by increasing the overall system security. On the other hand, this platform can be utilised to guarantee the integrity of the off-chain components.

Log Size. The key parameter highly affecting the monitoring system is the size of the log. In fact, the bigger the size is, the higher is the latency to store the log on the blockchain. By relying on a private blockchain, where all PoW (Proof-of-Work) parameters can be dynamically tuned according to the needs, the latency can be maintained under control. However, due to the limited size of the network and a possibly lightweight PoW, this solution does not ensure strong integrity guarantees. Alternatively, a hybrid approach combining classical database with blockchain system should offer an adequate flexibility to find a trade-off between latency, integrity guarantees and, in case of public chain, cost. A preliminary design to such a system is presented in [9].

Acknowledgment

This work has been supported by the EU H2020 Programme under the SUNFISH project, grant agreement N. 644666.

References

[1] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *CLOUD*. IEEE, 2010, pp. 337–345.

[2] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud Federation," in *Cloud Computing, GRIDs, and Virtualization*, 2011, pp. 32–38.

[3] F. P. Schiavo, V. Sassone, L. Nicoletti, and A. Margheri (Eds.), "Faas: Federation-as-a-service," *CoRR*, vol. abs/1612.03937, 2016.

[4] B. Suzic, A. Reiter, F. Reimair, D. Venturi, and B. Kubo, "Secure data sharing and processing in heterogeneous clouds," *Procedia Computer Science*, vol. 68, pp. 116–126, 2015.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, available at https://bitcoin.org/bitcoin.pdf.

[6] "SecUre iNFormatIon SHaring in federated heterogeneous private clouds (SUNFISH)," Accessed on 16 January, 2017, http://www.sunfishproject.eu/.

[7] Bill Parducci,Hal Lockhart, "eXtensible Access Control Markup Language (XACML) Version 3.0," 22 January, 2013, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[8] A. Margheri, M. Masi, R. Pugliese, and F. Tiezzi, "A rigorous framework for specification, analysis and enforcement of access control policies," *CoRR*, vol. abs/1612.09339, 2016.

[9] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments," in *ITA-SEC*. CEUR-WS.org, To Appear.