A High-level Authorization Framework for Software-Defined Networks

Daniel Rosendo[1], Judith Kelner[1], and Patricia Takako Endo[2]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)

[2]Universidade de Pernambuco (UPE)

{daniel.rosendo,jk}@gprt.ufpe.br, patricia.endo@upe.br

Abstract. *Enterprise network managers need to control the access to their network resources and protect them from malicious users. Current Network Access Control (NAC) solutions rely on approaches, such as firewalls, VLAN, ACL, and LDAP that are inflexible and require per-device and vendor-specific configurations, being error-prone. Besides, misconfigurations may result in vulnerabilities that could compromise the overall network security. Managing security policies involve dealing with many access control rules, conflicting policies, rule priorities, right delegation, dynamics of the network, etc. This work presents HACFlow, a novel, autonomic, and policy-based framework for access control management in OpenFlow networks. HACFlow simplifies and automates the network management allowing network operators to govern rights of network entities by defining dynamic, fine-grained, and high-level access control policies. We analyzed the performance of HACFlow and compared it against related approaches.*

Resumo. *Gerenciadores de redes corporativas devem controlar o acesso aos recursos disponíveis na rede, assim como, protegê-los de usuários maliciosos. As soluções atuais de controle de acesso à rede consistem em firewalls, VLAN, ACL e LDAP. Tais tecnologias são inflexíveis e exigem configurações específicas por cada dispositivo e fornecedor diferente, sendo desta forma, propensa à erros de configuração podendo resultar em vulnerabilidades que comprometam a segurança geral da rede. Gerenciar políticas de segurança envolve lidar com um grande número de regras de controle de acesso, políticas conflitantes, prioridades de regras, delegação de papéis, assim como, lidar com a natureza dinâmica da rede. Este trabalho apresenta HACFlow, um framework autônomo e baseado em políticas para gerenciamento de controle de acesso em redes OpenFlow. HACFlow tem por objetivo simplificar e automatizar o gerenciamento de redes, permitindo que os operadores da rede governem os direitos das entidades da rede através de políticas de controle de acesso dinâmicas, em alto nível e de forma granular. Analisamos o desempenho do HACFlow e comparamos-o com soluções similares.*

1. Problem definition

During the last decade, advances in the Internet architecture and communication system technologies together with the introduction of the Fog Computing and Internet of Things (IoT) paradigms contributed to the growth of the network and the number of interconnected

heterogeneous devices. These devices exchange information and interact with each other and with humans and machines. Besides, they have different roles and different levels of access rights between them. Ensuring the security and privacy of these entities and defining and managing access rights to protect them from unauthorized access becomes a challenge [Sicari et al. 2015].

From the network operator perspective security tasks like authorizing the access between and for each network entity (users, sensors, printers, services, among others), are complex and challenging to manage due to many reasons. For example, misconfigurations may result in vulnerabilities that may compromise the overall network security. Besides, large and dynamic network environments and sensitive information also increase the management complexity. Due to those concerns, there is a need for a more sophisticated access control solutions based on high-level and autonomic policy implementations to minimize costs, and the network administrator effort and errors [Kreutz et al. 2015].

The Software-Defined Networking (SDN) paradigm stands to replace low-level configurations by high-level network access control mechanisms. Furthermore, it offers new opportunities (programmability, flexibility, dynamicity, and standardization) to overcome the above issues [ONF 2014]. Despite this, those problems in traditional networks perseveres in SDN.

Recently, researchers focused effort to address those problems as well as pointing out challenges and open problems regarding the NAC management in SDN. In [Ahmad et al. 2015] authors highlight challenges regarding the **Synchronization of Network Security and Network Traffic** (they need to be synchronized according to network changes and events) and the **Network Security Automation** (avoid human intervention and manual configurations, which are prone to errors).

Besides, authors in [Wickboldt et al. 2015] discuss challenges and management requirements in SDN such as **From High-level Rules to Network Configuration** (refers to the loss of low-level information when using high-level commands or rules, so the lost information needs to be reconstructed in the translation process) and **Autonomic and In-Network Management** (regards the autonomic reaction against network events). In addition, the Open Networking Foundation (ONF) document [TR-516 2015] specified requirements to be met by the SDN architecture such as the use of **Network Interaction Policies** (the need to create mechanisms to express, distribute, and manage interaction policies that define which operations can be performed by network entities).

Therefore, based on those challenges and problems, we point out the following research questions: ***i)*** How to simplify the Network Access Control management in current networks? ***ii)*** How to allow the definition of high-level access control policies to configure the network? ***iii)*** How to automate the reaction of security policies against network state changes and events? ***iv)*** How to maintain the synchronization between high-level policies and the network configurations?

2. Access control management in traditional networks

In traditional networks, the Network Access Control (NAC) management relies on a series of network devices like firewalls, routers, and switches, together with protocols, standards and technologies like RADIUS, IEEE 802.1x Port-based Network Access Control

(PNAC), Access Control List (ACL), Virtual Local Area Network (VLAN), Lightweight Directory Access Protocol (LDAP) (OpenLDAP and Active Directory), among others. Those solutions are inflexible and require per-device and vendor-specific configurations, being prone to errors.

Furthermore, changes in the network (dynamic networks) require manual reconfigurations in the network devices to comply with the established network security policy as well as network operator needs. Also, managing and maintaining them are expensive, which even in a small network, requires a management team [Liu et al. 2016]. Therefore, these approaches work well for stable networks and are hard to integrate and configure. Due to those concerns, there is a need for a more sophisticated access control solutions based on high-level and autonomic policy implementations to minimize costs and the network administrator effort and errors [Kreutz et al. 2015].

The aforementioned approaches (firewalls, PNAC, VLAN, LDAP, and Kerberos), normally rely on manual configurations (through human intervention) in firewalls, RADIUS server, routers, and switches, being highly exposed to misconfigurations, and resulting in a time-consuming task [Matias et al. 2014]. Therefore, the deploy of those technologies becomes harder in dynamic and large networks scenarios. Besides, there is a lack of granularity and expressiveness to implement the network access control, requiring the combination of different solutions. Figure 1a depicts the combination of these solutions (firewall, PNAC, VLAN, LDAP) to control the access in a traditional network.

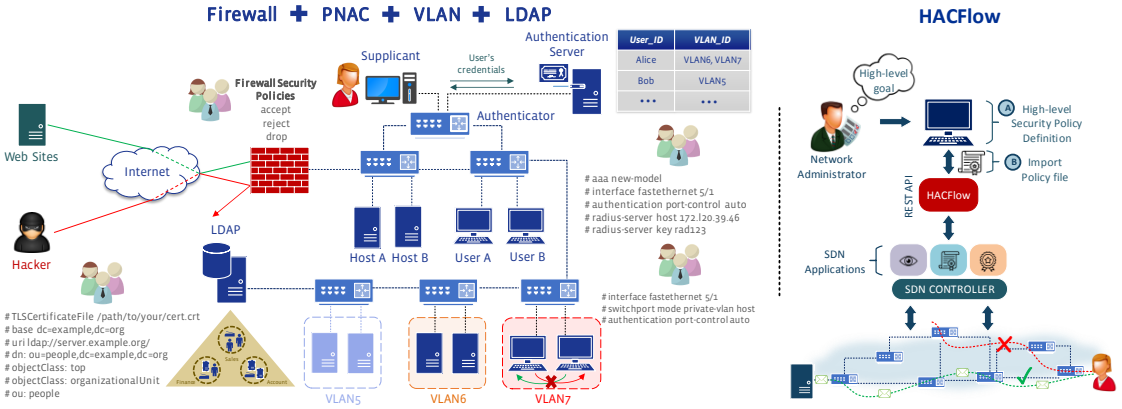

Figura 1. (a) Access control management in traditional networks. (b) Access control management in SDN with HACFlow.

3. HACFlow: access control management in OpenFlow networks

HACFlow is a High-level Access Control management framework for OpenFlow networks and is based on the Organization Based Access Control (OrBAC) model. HACFlow aims to simplify and automate the NAC management providing mechanisms to define dynamic, fine-grained, and high-level access control policies, detect and solve conflicting policies, delegate roles, and react to network state changes and events (see Figure 1b).

The **HACFlow framework architecture** is composed of many components that work together to comply with all the previous issues. Next, we describe the core components of HACFlow. For a detailed description of all components and how they interact with each other refer to [Rosendo et al. 2017].

OrBAC API: Network Interaction Policies - the OrBAC API allows the definition of high-level and context-aware security policies. Besides, it provides mechanisms to detect and solve conflicting policies. The OrBAC API is composed of a policy implementer, policy checker, policy parser, and policy inference. The *policy implementer* allows the creation of predicates (*Organization, Role, Activity, View, and Context*), entities (*Subject, Action, and Object*), and abstract permission and prohibition policies. Then, the *policy checker* checks for constraints and conflicts in those abstract policies. Next, the *policy parser* generates the concretes rules from the abstract policies. Lastly, the *policy inference* infer the concrete rules considering its states, that can be active or inactive (in/out of context), and preempted or not preempted (lower/higher priority).

Event Listener: Synchronizing the Network Security and Network Traffic - HACFlow automatically processes network events and policy context changes. HACFlow receives those events from an SDN application and they can be the result of a user authentication, a vulnerability alert detected by a security service (IDS, DPI, or DDoS), among others. The main role of the event listener is to maintain the synchronization between the high-level security policies to network configurations. In order to HACFlow be able to react to those events, network operators must previously define context conditions (circumstances) and link it to a security policy. Therefore, HACFlow allows the network operator to describe how to react in case malicious traffic is detected.

The Policy Translator: From High-level Rules to Network Configuration - HACFlow allows network operators to define policies in a high-level way without taking care of how they will be implemented in the network. Such high-level of abstraction results in the loss of low-level network information (e.g. IP address, MAC address, port number, connected switch, among others). Therefore, these low-level data must be reconstructed in the translation process. This way, while translating a high-level security policy into a low-level OpenFlow flow rule, the HACFlow's components (OrBAC API, Policy Translator, and Entity Manager) must work together to perform the translation. Figure 2 depicts the whole translation process.

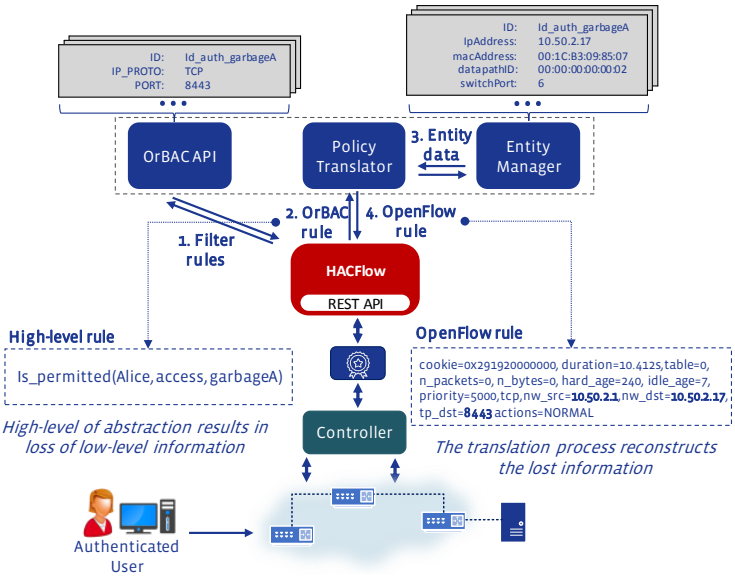

Figura 2. Policy Translator: From High-level Rules to Network Configuration.

As an example, once a user authenticates on the network, a third party SDN application responsible for authenticating the user will notify HACFlow (through its REST API) to start the authorization process. At first, once notified HACFlow will (step one) get the user's high-level policies through the OrBAC API and (step two) pass them to the Policy Translator. Next, the Policy Translator (step three) gets from the Entity Manager the low-level data and then translates the high-level rule (OrBAC policy) into the low-level rule (OpenFlow flow rule). Lastly, HACFlow (step four) returns the OpenFlow rules to the third party SDN application to enforce the user's rules in the network. Finally, the user will be able or not to access the network entities (hosts, printers, services, among others).

4. Evaluation and Comparison

Our testbed consists of a physical machine with Ubuntu 15.04 with a Core i7-3770 CPU 3.40GHz and 16GB of RAM. On top of it, we configured three virtual machines. The first virtual machine refers to the HP VAN SDN controller version 2.5.15. The second one is the Mininet network emulator version 2.2 with Open vSwitch 2.4.0. Lastly, the third one refers to the HACFlow framework. Next, we developed a SDN application for the HP VAN SDN controller that interacts with HACFlow through its REST API. This application enforces the OpenFlow security rules provided by HACFlow. The network topology consists of one SDN controller, three OpenFlow-enabled switches, and two hosts.

4.1. HACFlow Performance Evaluation

Network State Change - Vulnerability Alert: we evaluated the required time to HACFlow block the access of a user to a server after a vulnerability alert is detected. When a network alert is triggered by a security monitoring system the SDN application notifies HACFlow to reconfigure the network. From the results (see Figure 3a) HACFlow needed 8.24 ms in average (32.1% of the total time) to react to a vulnerability alert.

Dynamic Security Policies: are context-aware policies that may have their state changed (active or inactive) depending on some circumstances (day of a week, an hour of a day, and so on). HACFlow is able to automatically react to these contextual changes, not requiring any manual per-device reconfiguration of the network devices. In this experiment, we simulate a security policy being out of context, that means, being out of a circumstance imposed by the network operator. According to results (see Figure 3b), we point out that HACFlow required 7.57 ms in average (30.9% of the total time).

Role Delegation: once a delegation occurs, HACFlow assigns the new security rules to the granted network entity and sends the rules to be enforced in the network. According to the results (see Figure 3c) HACFlow required 119.45 ms in average (30.8% of the total time) to delegate a single role linked to a single security policy.

High-level to Low-level Policy Inference: we analyzed the scalability of HACFlow to infer the OpenFlow flow rules. In this analysis, HACFlow infers 1, 4, 16, 32, 64, 128, and 256 rules. We divided the flow rule inference into two steps. The first (*1. Security rule filter*) refers to the filtering of rules through the OrBAC API and the extraction of low-level data about the network entities. The second one (*2. Policy translation*) refers to the process that obtains the OpenFlow flow rules from the OrBAC rules. From results, we highlight that HACFlow needed 0.4791ms to translate 256 rules. Furthermore, the whole process (filter and translate 256 rules) required 1.1 seconds on average.

4.2. Comparison Against Existing Solutions

Policy Translation: here, we compared HACFlow against Frenetic and OpenSec frameworks regarding the required time to translate a high-level security policy into low-level OpenFlow flow rule. The experiment was executed 256 times and the results are the mean and standard deviation. From Figure 3d, we point that HACFlow, Frenetic, and OpenSec require similar times to translate a single security rule. But, Frenetic required a lower time.

Event Reaction Delay: next, we compared HACFlow against OpenSec regarding how long each one requires to react to a network state change and event. The reaction time includes the moment that the framework receives the alert until it returns the OpenFlow flow rules to reconfigure the network. The experiment was run 256 times and our results represent the mean and standard deviation. From Figure 3e, OpenSec required a lower time, 8.1ms against 8.5ms by HACFlow.

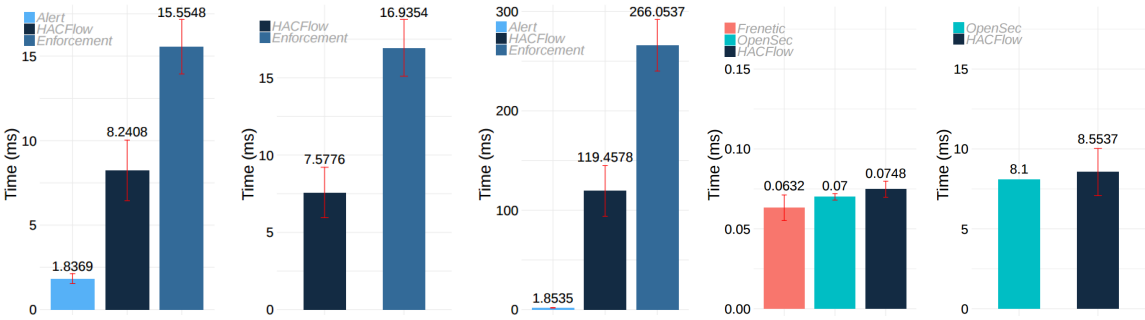

Figura 3. (a) Reaction of HACFlow against a vulnerability alert. (b) Reaction of HACFlow against a dynamic policy. (c) Role delegation in HACFlow. (d) Policy translation comparison. (e) Event reaction comparison.

5. Related work

Frenetic [Foster et al. 2011] is a high-level language for OpenFlow networks based on Functional Reactive Programming (FRP) and SQL-like queries. The Frenetic architecture consists of an implementation of the FRP operations (to define high-level policies), a run-time system (to translate high-level policies into low-level packet-processing rules and manage the policy enforcement), and the NOX SDN controller. HACFlow and Frenetic manage the network traffic by defining high-level policies. In both, network managers do not take care of how those policies will be implemented and enforced on the network.

FRESCO [Shin et al. 2013] is a security framework focused on enforcing security constraints to SDN applications. With FRESCO, those applications can replicate security functions like firewalls and attack deflectors. In FRESCO, network operators define high-level security policies based on a scripting language. This language relies on the block, deny, allow, redirect, and quarantine security primitives. While, in HACFlow, the high-level policies are based on the OrBAC model. To detect and solve conflicts, FRESCO includes a Security Enforcement Kernel (SEK) module integrated to the NOX SDN controller. Therefore, supporting FRESCO in other SDN controllers require the implementation of this module, not being a straightforward integration. On the other hand, the OrBAC component in HACFlow framework detects and solve conflicting policies. The HACFlow deployment does not require any extension or modification in SDN controllers, as FRESCO requires.

OpenSec[Lara and Ramamurthy 2016] is a security framework to automate the implementation of security policies. In OpenSec, the network operator defines high-level goals (high-level security rules) to determine by which processing units (DDoS, DPI, spam detection, among others) a traffic must be monitored. While HACFlow implements high-level security policies to determine which actions network entities can perform. OpenSec and HACFlow react dynamically to network alerts by enforcing switch-level rules. Besides, they allow network managers to previously define how this reaction must be implemented/enforced according to the alert received.

6. Conclusions

This work proposes HACFlow, a novel SDN framework that aims to simplify and automate the management of access control policies in OpenFlow networks by providing mechanisms to *i)* express, distribute, delegate, and manage interaction policies; *ii)* define dynamic, fine-grained, and high-level access control policies; *iii)* translate high-level security policies into network configurations; *iv)* maintain the synchronization between the security policies and the network traffic. We analyzed the performance of HACFlow and the results showed that it reduces the effort (HACFlow is faster and less prone to errors than manual configurations) to implement a variety of network configurations. We also compared HACFlow against related approaches and the results showed that HACFlow offers more management features (based on the points motivated). Besides, results showed that HACFlow requires a similar time to translate security policies and react to network events.

6.1. Contributions

Our main contributions can be highlighted as **(1)** We demonstrate how SDN, OpenFlow, and the OrBAC model can be used together to improve and automate the network access control management. **(2)** We propose a framework for the definition of high-level and human-readable policies, trying to simplify the management of access control policies and minimize misconfigurations. **(3)** We propose a novel solution to define high-level network security policies that dynamically react to network events and rule state changes, taking advantage of SDN flexibility and programmability features to reconfigure the network. **(4)** We show that SDN may be leveraged to offer ACL at a much finer granularity where more flexible rules may be defined, as opposed to existing port and VLAN based rules only. **(5)** We improve network access control management without modifying the SDN architecture (e.g., the OpenFlow protocol, controllers, and switches). **(6)** We present a quantitative and qualitative analysis of HACFlow framework and compare it against related solutions.

(7) Publications: this research resulted in two publications. The first publication [Aschoff et al. 2017] was accepted in IFIP/IEEE International Symposium on Integrated Network Management (IM 2017). In this paper, we proposed an SDN-based Network Access Control (S-NAC) solution that authenticates and authorizes network entities. In the second publication [Rosendo et al. 2017], we evolved the idea and proposed HACFlow, a novel SDN framework with much more management capabilities. It was accepted in the International Conference on Network and Service Management (CNSM 2017).

(8) OrBAC API bugfix: we helped to improve the OrBAC model API by notifying a bugfix. Such bug regards the *NotifyContextStateChange()* method in the *AbstractOrbacPolicy* class which was not correctly monitoring concrete rule state changes, making

it impossible to create dynamic security policies. The OrBAC API developers corrected this bug, then we tested it to validate the bugfix, and lastly, we confirmed the correction to them. Then, a new version was released. In the MotOrBAC web page (MotOrBAC version 2.5 and OrBAC API 1.5.1 from 12/04/2016 at http://motorbac.sourceforge.net), inside the *changelog.txt* file they thank us: *"Big thanks to Daniel Rosendo for pointing this out!"*.

Referências

Ahmad, I., Namal, S., Ylianttila, M., and Gurtov, A. (2015). Security in software defined networks: A survey. *IEEE Communications Surveys & Tutorials*, 17(4):2317–2346.

Aschoff, R., Rosendo, D., Machado, M., Santos, A., and Sadok, D. (2017). A network access control solution combining orbac and sdn. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 483–489. IEEE.

Foster, N., Harrison, R., Freedman, M. J., Monsanto, C., Rexford, J., Story, A., and Walker, D. (2011). Frenetic: A network programming language. In *ACM Sigplan Notices*, volume 46, pages 279–291. ACM.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Lara, A. and Ramamurthy, B. (2016). Opensec: Policy-based security using software-defined networking. *IEEE Transactions on Network and Service Management*, 13(1):30–42.

Liu, J., Li, Y., Wang, H., Jin, D., Su, L., Zeng, L., and Vasilakos, T. (2016). Leveraging software-defined networking for security policy enforcement. *Information Sciences*, 327:288–299.

Matias, J., Garay, J., Mendiola, A., Toledo, N., and Jacob, E. (2014). Flownac: Flow-based network access control. In *2014 Third European Workshop on Software Defined Networks*, pages 79–84. IEEE.

ONF (2014). Software-defined networking: The new norm for networks, https://www.opennetworking.org. *White Paper*.

Rosendo, D., Endo, P. T., Sadok, D., and Kelner, J. (2017). An autonomic and policy-based authorization framework for openflow networks. In *Network and Service Management (CNSM), 2017 13th International Conference on*, pages 1–5. IEEE.

Shin, S., Porras, P. A., Yegneswaran, V., Fong, M. W., Gu, G., and Tyson, M. (2013). Fresco: Modular composable security services for software-defined networks. In *NDSS*.

Sicari, S., Rizzardi, A., Grieco, L. A., and Coen-Porisini, A. (2015). Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164.

TR-516, O. (2015). Framework for SDN: Scope and Requirements. `https://www.opennetworking.org`. Version 1.0. Last access: December, 2016.

Wickboldt, J. A., De Jesus, W. P., Isolani, P. H., Both, C. B., Rochol, J., and Granville, L. Z. (2015). Software-defined networking: management requirements and challenges. *IEEE Communications Magazine*, 53(1):278–285.