The integration of access control levels based on SDN

Izzat Alsmadi

Computer Science Department,
University of New Haven,
West Have, CT, USA
Email: ialsmadi@newhaven.edu

Abstract: Systems and networks include several inputs and outputs from which they are accessed. Access controls exist to manage authentication and access controls through those inputs and outputs. One of the significant problems in this scope is the difficulty to have a global consistent system or network level access control. Such global centralised access control is needed to make sure that access control decisions taken by different applications at different levels are consistent. In this paper, we propose an SDN-based access control to approach this problem. Different access control methods are proposed to handle user and flow level access information. Access tables in those methods are initiated by network administrators. Concrete rules in those access tables can change dynamically based on network activities. We believe that ultimately all access control systems are going to converge in this direction.

Keywords: access control; authentication; authorisation; network security controls; SDN.

Reference to this paper should be made as follows: Alsmadi, I. (2016) 'The integration of access control levels based on SDN', *Int. J. High Performance Computing and Networking*, Vol. 9, No. 3, pp.281–290.

Biographical notes: Izzat Alsmadi is currently an Assistant Professor at University of New Haven, CT, USA. He has his PhD in software engineering from NDSU in 2008. His main research interests include software engineering, security and SDN.

1 Introduction

Systems, networks and databases are all include one or more levels of access controls. Authorisation or access control is considered the second layer of defence after the initial authentication system that tries to generally identify where the specific request, user, access, etc. represents a legitimate one or not. Access control comes next to specify in details for those legitimate users or access requests their control on the different system resources or assets. The term policy indicates a high level vision of system owners or administrators where those visions are translated in concrete actions using authentication and access control applications.

While policies are commonly used for security rules, however, they can be also extended to include: business processes' rules, quality assurance (QA) and service level agreement (SLA) rules. Companies include policies to regulate employees' activities, legal practices, codes of conducts, etc. Currently, most of those kinds of polices are not implemented or enforced through information systems. Nonetheless, future trends in policy automation may make this possible.

Security policies are about '*Who* can access *what*, under what *condition*(s), and for what *purpose*(s)?' Security policies largely exist in firewalls and access control applications. The objectives for those security controls can be largely divided into two parts: providing authentication and authorisation services. In authentication stage which usually comes first, focus is only whether to allow: user, host, program, traffic, etc. to access intended resource or not. Authorisation which usually comes as a second stage after authentication focuses on giving those authenticated entities the right level of permission. For example, operating systems may include users such as: administrators, guests, power users, etc. Those may all be allowed to access the computer. However, certain resources will be only accessed by particular 'rules'.

Rule-based access control (RBAC) systems maintain a list of authenticated users and the level of permission they have on the different computer resources. Attribute-based access control (ABAC) deals with entities rather than rules. Entities can be: users, hosts, virtual machines, applications, etc. Those all have attributes with different possible values. ABAC can be considered more rich and context driven than RBAC. For example, the same user can be given in one scenario an access to a resource and on another scenario will be denied access to the same resource if the context of 'permission request' is different. The context takes variables related to the entity itself, the resource to access, and the environment. RBAC has few context related attributes. It has also other limitations which limit its ability to successfully express a particular context with the right level of granularity.

RBAC advantage is that it is closer than ABAC to low level flow rules or firewall rules. Consequently, direct transformation can be drawn from the RBAC-based system to those low level rules. However, when it comes to user level policies that are expected to be very expressive, we think that ABAC can be better than RBAC. For example, a high level policy that says (local users should not be able to access ABC server resources from their BYODs or their smart devices), this policy looks very complicated to implement based on RBAC. On the other hand, ABAC can define objects with attributes. Users, smart devices, and servers can be defined in the ABAC system as objects with expressive attributes that can be network independent (i.e., no specific IP, MAC addresses, ports, etc.). RBAC has several other disadvantages in comparison with ABAC. For examples, in many networks, users can have different rules (e.g., graduate student and lecturer, network administrator and user, etc.). RBAC needs such users to keep having the different roles isolated from each other and used each one based on intention or convention. In addition, rules can be hierarchical (e.g. a manager is an employee). RBAC ignores also information related to the environment and how to handle those parts in the policies. A large system may produce a very large number of possible rules. RBAC is not granular enough to differentiate somewhat similar scenarios. ABAC, on the other hand, generalises everything into attributes and includes environment information or attributes. The subject can be described by one or more attributes. It can accept composition or hierarchy (e.g., user, group of users, policy, and policy set). ABAC includes also obligations which are actions that should be implemented upon fulfilling policy requirements.

The translation between high level policies and concrete implementation in security mechanisms is conducted in most cases manually by network administrators. However, SDN brings to this particular area a new opportunity for security policies to be interpreted, updated, evaluated and enforced by automatic tools with least human intervention. One significant challenge in this area is the difference of levels of abstraction between what tools can automatically interpret and between human languages or expressions that we like policies to be written in. Policy languages are consequently proposed to formalise the way policy rules are written and to enable those rules to be compiled and interpreted similar to programming languages' code. Policy languages can also help administrators to define permissions that can guarantee users the ability to perform tasks no more and no less.

Current access controls are not expressive enough. Languages or scripts used to write those rules should give users the ability to define: wild cards, sub categories, summation, subtraction, etc. They also need to contain user visible information. We then think that neither low level rules should be more expressive, nor high level policies should be more specific. The solution is to have a third abstraction layer or adaptor in between both of them. This mediator layer should be responsible to perform the two-way interpretation or information exchange.

We argued in this paper that real use cases justified using different access controllers for different scenarios. However, a global policy along with an access controller can organise and orchestrate the process between those different access points to ensure final decisions consider information from all those access points.

The rest of this paper is organised as the following: In Section 2, we will introduce several research papers that are relevant to the paper subject. In Section 3, we will present goals and approaches for policy architecture proposed in this paper for SDN networks. Paper is then concluded with a summary section.

2 Literature review

Networking trends are moving steadily toward more virtualisation and software-based network control or management. Cloud applications and environments take a significant portion of such direction. Since its early evolutions, many concerns were raised about security issues in the cloud (see as examples: Al-Said et al., 2015; Wang et al., 2015; Ficco, 2013; and many others). To focus this section, we will only consider a subset related directly to SDN.

There are many reasons for why current access control lists (ACLs) are not expressive enough to deal with evolving network requirements. For example, given the decisions, 'deny' and 'permit' related to making a decision about a flow or a packet; those binary decisions cannot be expressive in some particular scenarios. Research papers described examples related to connectivity or directionality cases. In addition, maybe we want to deny certain traffic for a certain time or deny it only from going to a certain destination. In other words, real cases may need some decisions that they are: neither completely deny nor completely permit.

Policies should also have auditing or meta data attributes related to who created the policy, when the policy was created, etc. Those attributes can be then used to handle certain cases of conflicts or auditing. A newly added policy or rule in already existing policies should be automatically evaluated against possible conflicts. This should happen automatically without the need for an administrator to go and edit or review the policies.

In their approach, tables are created and maintained to include a pair of (header, decision) records (Hinrichs et al., 2008, 2009). In order to improve performance, headers of packets are evaluated. If two packets have the same header, same decision can then be made about both of them (i.e., without the need to investigate the second one). Incoming packets are checked for possible match in those tables. FML that is proposed in authors' papers maintains also states related to lists of users and their devices or hosts.

In classical firewalls, eight fields about flows are created. Access control decision is then made based on the values of those flow fields or attributes. Those fields are: source and destination: hosts, users and access points, along with protocol and type (e.g., initial message or a response).

Those fields include information from higher lever layers in the network (in comparison to L2–L3 information in typical ACL). In addition to 'allow' and 'deny' decision, there are other decisions in FML: waypoints, avoid, and rate limit. Waypoints or reference points are defined to mark certain known points (e.g., hosts, a server, a gateway). 'Waypoint' and 'avoid' are opposite to each other. They both take one extra attribute in addition to the eight fields. This attribute is related to the node that must be visited or avoided. Rate limit includes also a 9th field that indicates a rate limitation (i.e., maximum allowance) on the traffic. Authors demonstrated how conflicts can be resolved based on the proposed fields using different examples.

Access control in wireless networks, home-networking and some other similar domain has special challenges related to the ability to distinguish (AAA): authentication, access control and accounting (Dangovas and Kuliesius, 2014). In classical networks, network applications do not distinguish or allow separate accounts between those three access control requirements. For example, a user who has a home wireless connection with possible neighbours using this network will not have the ability to identify, which activities in particular were accessed by those intruders. Researchers suggested that SDN can open the possibility to distinguish those three from each other. Eventually, this may open the opportunity for many new applications (Alsmadi and Xu, 2015).

While this early contribution in SDN policies showed a significant distinction from traditional ACLs, however, we think that it is not expressive enough to convey more complex scenarios than those described in the paper. In addition, FSL focuses only on end-to-end reachability without giving mechanisms to monitor the traffic.

In order to improve expressiveness in network and security policies, Voellmy et al. (2012) introduced Procera, a control architecture that includes a declarative policy language based on functional reactive programming. It can be used to describe reactive and temporal behaviours. Procera tries to help network designers to implement expressive policies without the need to use programming languages. This is since earlier OpenFlow policies are not expressive enough to handle reactive and temporal situations related to flow content, time issues, size of traffic, bandwidth consumption, etc. Reactive policies are policies that should be dynamic and be able to revisit or update themselves based on changes that are related to: traffic, time, etc. For example, we may have a particular policy that is dedicated to guest users which includes a timeout value to specify when this policy should expire. This policy will be frequently revisited as it is a temporal policy. If today is the day number 4, it will permit traffic. However, at the end of the fifth day, it will make a different decision (e.g., denies the host from accessing the network). In another example, a user should have a bandwidth quota for the current month. After reaching this quota, account will be banned or denied. As an alternative, user may get a message that future bandwidth consumption will be over-charged. Same policy should be reactivated or reinitialised at the beginning of the new month. Those examples show that the policy is interacting in two ways with the state of the traffic; affecting and being affected by it. It is also interacting with time or even with some other domain environments (e.g., start watching user web surf when they log-in). Currently, OF flow entries include idle_timeout and hard_timeout attributes in which flow rules themselves can be designed to be temporal.

Procera includes signals and signal functions as reactive concepts. Signals are like transient functions where functions are attached with a period of time. Signal functions or constructs cause transformations on signals. Event streams can have different operations: filtering, transforming, merging, and joining. Authors introduced several language constructs or signal functions and showed how they could be used to manipulate signals. Internet service providers (ISPs) have several policies related not only to security but also to; monitoring, billing, accounting, etc. SDN programmability can help ISPs provide services that can be customised for users based on their requests on demands and that can vary from one customer to another. Bismark (2015) project also aims at extending SDN to home networking by including OpenFlow technology in home small access points or routers. Based on such proposed modification, router or access points should then have modules to detect security attacks and automatically update flow tables to counter those attacks.

In Casado et al. (2007), ethane system described the interaction between the controller and security policies injected by the controller in switches. Those however were imitating traditional ACLs (Casado et al., 2007). In Nayak et al. (2009), resonance is a security mechanism for dynamic access control evaluation based on flow level information. It interacts with high level policies to make decisions on flows. Authors used a policy specification framework based on traditional or existing access control frameworks.

Feamster et al. (2010) used OpenFlow to solve policy problems in campus and enterprise networks. Specifically, they tackled two challenges; access and information flow controls. The decouple and the gap between high level expressive policies and low level access controls exist in switches or firewalls continue to be a serious challenge for administrators in dealing with large networks. Authors showed how the programmability nature of OpenFlow can help solving the described challenges. The paper presented two products: resonance; access control and Pedigree; information flow control. Authors described problems in those two controls (i.e., access and information flow controls) and traditional approaches to solve those problems. Authors then introduced OpenFlow-based solutions to those problems and showed how they can outperform traditional solutions. For information flow control, major problem is that traditional approaches to this type of control are host based. Hence, if the host is compromised, the information flow can go out of control. Tracking information leakage can be also very challenging. From the network side, only general features about the flow

such as IP address and port number can be tracked. If information control policy is integrated with the network layer, information can be tracked while traversing the network not only in the host or destination points. Pedigree is proposed with two components: A trusted tagger to tag packets with information extracted from the source process and an arbiter to make decisions on traffic. Same thing can be applied for incoming packets. Some of the open research issues authors discussed include: First, the need to have a language and a method to enforce such policies. The second challenge was related to the ability of decoupling network policies from topologies. In other words, there is a need for a flexible interface to isolate high level policies from low level implementation details. Policies themselves need not to have any reference to low level details (e.g., switch, port, IP address, MAC address, etc.).

SE-floodlight, an enhanced version of FRESCO (Shin et al., 2013) proposed a RBAC. Authors approach is considered the most mature one presented in literature related to policy management and enforcement. The framework includes six main security features that are implemented in SE-floodlight: least privilege module to allow northbound applications work outside controller context, RBAC for conflicts resolution, digital authentication for flow rules, packet-out control, inline flow rules' conflict resolution and security audit. Authors described several security directives to deal with the problem of translating high level policies to low level flow rules. However, we think that such solution is not scalable or reusable in different contexts of other security appliances. We described earlier the need to isolate high level rules or policies from low level flow rules using a separate module.

3 An SDN-based global access control

Our proposed solution focuses on proposing a system global access control solution. The apparent objective for such proposal is to reduce conflicts in decision between the different access control decision points. For example, a traffic that is allowed is expected by a layer 7 application can be possible blocked by an access control in layer 2 or 3. Not only rules are written separate in those different access controls, but also the information they can see about the network, topology, threats, etc. are typically at different levels of abstraction. As such, direct translation or interpretation from one access control to another is impossible.

The idea of fine grained access control was presented as one of the first show cases for SDN (Ethan; Casado et al., 2007). The goal was to enable access control decisions for example to admit certain user requests and deny others from the same user, based on the specific application, context, etc. Classically, such as controls are static and broader in context. We will revisit this paper in this section as it's the most relevant to our paper.

Fine grain access control can be utilised in several different manners. This includes for example, the ability to provide temporary access (e.g., BYOD or guests access) or also help ISPs provide customised services.

Ethan discussed a policy-based network where controller is expected to make decisions on flows. The paper also proposed a global policy that decides the fate of all flows. In addition, the controller plans the route of permitted flows. Entities (e.g., hosts, users, etc. register with the controller and their bindings with low level information (i.e., port number, MAC and IP addresses) are also recorded. Poleth policy language is proposed. Nonetheless, little information has been published since then about Poleth policy language and its implementation. In addition, while authors discussed issues related to access control, however, focus on flow tables where in routing tables and not access tables. In addition, little information is shown on how to handle difference in levels of abstraction between high level policies and flow information. For example, how does the controller make (permit, deny, forward decisions)? And what are the information controllers uses to deny or permit a flow? How does access control information is integrated with the controller to make decisions? In some controller, when a firewall module is added, it can override rules of the controller, what if there is another access controller? How could controller guarantee consistency of final decision? We think that Ethan focuses on decision related to traffic steering but not access control. This explains why most controllers include a firewall northbound API module. Nonetheless, access control has more attributes than those included typically in firewalls. Controller in general should receive the following information to help making routing and access control decisions: network registered hosts, users (applications), binding between those users/applications and (IP and MAC addresses) for hosts/users and port number for applications.

3.1 Flow rules vs. ACLs

In pure SDN, flow rules represent ACLs. They represent the lowest level of ACLs that are network specific. This is since in networking ACLs exist in several networking or security devices such as: firewalls, operating system active directories, authentication, authorisation or identity management systems, routers, port ACLs, etc. Computers are accessed by: user names, roles, IP, MAC addresses, and ports. Those are generally the identities, at different levels, that authentication or identity management systems use. Typically, an authentication system will have a white or black list. If it contains a black list, those in the list are denied access and everyone else is permitted. In the opposite terminologies, white list will be applied. So no matter of what level, authentication system will work, it will have this simple approach. There is a need for a global authentication system so that conflict based on different identities will not occur. For example, a particular flow will come from a particular host, user, IP address, MAC address, application and port. Hence, it does not make sense that this flow will be denied at one level based on IP address for example and then permitted based on user name. The

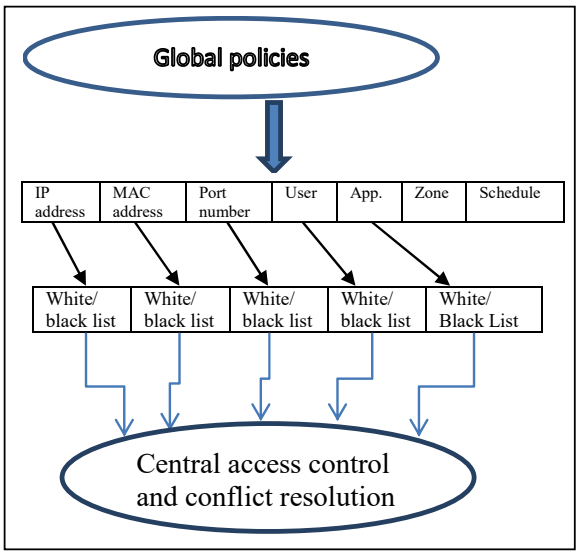
problem is that different information is evaluated at different levels and most likely by different authentication systems. There is one more challenge in this aspect between white and black listing. If one authentication system focuses on black listing, then in reality this system block only what it knows. Hence, if there is another identity management system in which a particular flow was permitted based on the first black list and black listed in the other one, it may make sense to block this flow assuming that there is a further knowledge in the later system that justified blocking this particular flow. In other words, as a defensive security approach, it makes sense to follow – deny-override – if two identity management systems have conflicting decisions for the same flow. In other scenarios, however, it can be more justifiable to follow 'permit-override' rather than 'deny-override'. Notice that this conflict cannot be discovered by one system and needs a global identity management system to be able to discover such conflict. Figure 1 shows general architecture for central access control. We call the modules to handle access information (i.e., user, application, port number, IP and MAC addresses) as access switches. We call their black/white list tables as access tables.

Globally identity management system should be extracted from high level or global policies. Those can be then interpreted for the different identity management systems based on their local terminologies. For example, in reference to the global policy: "Employees should not be able to use chat programs during working or office hours", for a port ACL, we need to identify known ports used by chat programs (e.g., instant messenger: 443, MS messenger 1,863, AOL: 5190). Users which this policy is applied on are employees. System can then have a list of users, and their MAC addresses. If no DHCP is used, system can also have a list of their IP addresses. In this specific example, the application and the port number are the most important information in this policy as white and black list will be based them.

Figure 1 shows depicted architecture of central access control based on global policy. Zone attribute refers to whether policy is applied to: incoming traffic, outgoing traffic or both. Schedule includes any time related attributes regarding the policy. Zone and schedule attributes do not include white/black lists.

User and application information are (L4–L7) information, IP, MAC and port values are (L2–L3) information. High level or global policies may have L4–L7 information but should not include network level information (i.e., L2–L3) information. Translation is hence necessary. Tools can be developed with rule sets and data to generate L2–L3 information that can also depend on run time information. For example, from current topology, information about network components and their IP addresses can be collected. Users, virtual machines or hosts IP and MAC addresses can be recorded and used. Port numbers that are typically used by different types of applications can be also collected and recorded.

Figure 1 Global polices – central access control (see online version for colours)



All earlier objects (or access switches) should support also groups. For example, we may need to refer to a group of IP addresses, ports, MAC addresses or users. Certain constructs should support this. Those include:

1 Any, or do not care, where the input can accept all valid values.

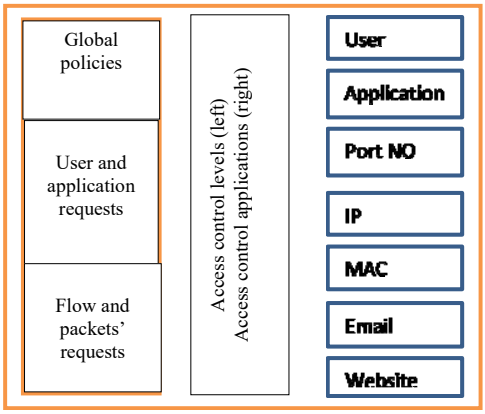
2 A continuous range where input will include first and last values.

3 Non-contiguous range where input will include all values enumerated.

Zone values relate to which interfaces policy should be applied on.

Access requests come from either users or applications or from the network. In addition to global policies, those are the inputs to the access controller (Figure 2).

Figure 2 Access controller architecture (see online version for colours)



Here is policy 1 (from policy examples) expressed in earlier terms:

- "IP_address=any, port=any, MAC_address=any, user=employee, application="chat-programs", zone=In/Out, schedule=8-5h/5d/12m".

Each policy should contain only one instance of objects described earlier. Group policies may combine more than one policy together. Application object can include in addition to the application name other information such as: Protocol, port number, etc.

An action field should be also included in each policy. In addition to permit and deny actions, count and log can be also added and those can be selected in addition to permit or deny.

L2–L3 information is usually extracted at later stages and should not be included at first level global policies.

As another example, for the policy to deny internet connections to websites with improper contents, the real challenge is how we define improper content. The easiest way is to have a black list of those websites and keep adding to it. Content-based approaches are also used to search for certain keywords that can be used as flags for websites with improper content. For simplicity, if we assume one known website to black list (and then repeat the process for all black listed websites), policy can be written as: "IP_address=any, port=80, MAC_address=any, user=employee, application="browser", website=A, action=deny".

3.1.1 Predicate access control

The central access control and conflict resolution module acts as a policy controller. Access switches (IP, MAC, port, user and application) include access rules. In addition to the particular access switch information, access rule includes also information related to zone and schedule. Tables 1 and 2 include examples of access rules that can exist in IP and port access table.

Table 1 IP access rules

IP	*Action*	*Zone*	*In-activity timeout*	*Schedule*
192.168.1.1	Deny	In/out	2000 s	24h/7d/12m
212.33.45.6	Permit	In	2000 s	8-5 SMTWTF

Table 2 Port access rules

Port	*Action*	*Zone*	*In-activity timeout*	*Schedule*
80	Permit	Out	2000 s	24h/7d/12m
80	Deny	In	2000 s	24h/7d/12m
1,030	Deny	In/Out	2000 s	8–5 SMTWTF

Different access tables are maintained by the central access controller. Access requests are sent to the central access controller.

In comparison with firewall access controls or rule-sets, this approach can be called firewall tables, or more generic access control tables. The major difference is that those tables will be dynamic and not static as in firewalls. The idea extends flow tables in switches that largely make decision for routing and not access control. Access control switches should focus on access control or authentication.

3.1.2 Access registration and binding

Different access switches proposed in our access control architecture should have the ability to register new entities. New applications can register as they are installed, users are registered when they are created and logged in, and so on. Journaling and tracking this information is a key process to the NAC process. Binding is about relating the different access entities or switches with each other. For example, a user can be associated with a host, a host with an IP and MAC addresses, an application with a host and port number and so on.

3.1.3 Access controller – security mode

The access controller should analyse incoming access request and come up with one of the following decisions. The access controller has two modes: security and normal modes.

Security mode is considered as a defensive approach. For a flow to be permitted, it should be permitted by more than one access control. This is since many security attacks target access controls. For example, ARP spoofing typically attack IP-MAC binding and a legitimate MAC can be falsely claimed by an illegitimate IP address. IP spoofing also occurs when illegitimate IP addresses impersonate legitimate ones. Hence, a defensive mode should require more than permit decision and deny decisions should be dominant. In addition to attacks, sometimes a specific access list information can be changed and others are necessary to verify it. For example, a DHCP server changes IP-host binding frequently. Users may change their Ethernet cards. Applications may bind to different ports.

To demonstrate the idea of dynamic binding, let us consider the following example. By having port 80 permitted by a rule, such generic rule may not limit intruders who are trying to use this port. In reality, such intruders were not and should not be denied due to using port 80 only. In other words, there are other information that should be added to this to make a deny decision. Let us assume that this addition is the IP address (192.168.1.1).

SDN should be able to deal dynamically with changing hosts, accounts, IP addresses, etc. However, the ability to dynamically handle such cases should not compromise security. In other words, such cases should not opportunities for illegitimate users, hosts, etc. to access the network. Security binding of different access controls can be an affective counter measure to many security attacks. Binding here is not static. In other words, IP addresses are not linked to hosts, MAC addresses, etc. Each one of them is stored in a different access switch table. Binding only occurs based on the current access request (i.e., based on what the request includes). Information in access switch tables are not

statically connected to neither high level policies, nor low level or physical network devices.

In security mode, deny override is considered if for the same request there are different decisions from the different access controls. In addition, if access control mode is 'security' access requests that can find no match in any access table, will be denied. On normal mode, the two cases are reversed. Namely, if there are match conflicts between the different access controls, permit-override is enabled and access request will be permitted. Similarly, if access request has no match in any access control, access request is allowed. Here is the step access controller takes to make a request decision.

1 Upon request arrival, access controller uses a special module will extract the following information from the access request: (user, application, port number, IP address, MAC address, zone and schedule). Information that has no value in the policy description is skipped or left empty.

2 For each access table of the five tables (user, application, port number, IP and MAC addresses), access controller looks for a match between the current access request and access rules in each access table. Decisions for matched records are extracted. In security mode, one 'deny' is enough to cause the whole request to be denied. Security mode is a defensive mode where access control system can be switched to this mode in occasions of security breaches, or when this access control system is running in a server hosting important, sensitive information. This mode hence adopts a white list approach where only those explicitly allowed to access will be allowed to access. In addition, no single access switch should explicitly deny the subject access request.

Security modes typically should work with a proactive mode where network administrator is expected to add access rules to switches. This is security mode by default deny flows with no records in access switches. Hence, if all access switches are empty, no flow will be allowed at all.

For example, in terms of ports, the administrator may decide to open the following ports only: 80, 8080, 1030, etc. Those are opened only based on the known permitted applications. This means that in security mode and based on port access only, no flow will be permitted to access the network if it is not coming from or destined to one of those ports regardless of the other values in the other access switches.

This mode can be very defensive and secure where it will not permit unauthorised access. However, it may prevent legitimate flows from accessing the network. How real-time information can change access tables? If flows are added with values that are not in access switches, those values are added to the access switch with their decision.

The most important aspect in security mode is that while rules can be added dynamically, however, rules added manually in the initial mode by administrators (as default values) can never be overridden or removed by dynamic flows).

3.1.4 Access controller – normal mode

In normal mode, one 'permit' is enough to cause the request to be permitted. In any table, no match is considered 'deny' in security mode and 'permit' in normal mode. Normal mode can be selected in regular working time. If the subject request is not listed in any access table, it will be granted access (i.e., black list mode). Further, if different access switches have conflicting decisions, permit decision is assumed.

3 If the access request has no match at all, in security mode, it will be denied and in normal mode it will be permitted.

4 Based on the result of the access request, the access controller writes a record in each access table.

In addition to security and normal modes, access controller can take active or proactive mode. In active mode, initial generic access rules can be added to all access tables to decide on access requests that have no match. On the other hand, in proactive mode, access tables are left empty and rules are added dynamically based on real-time access requests. Proactive mode allows also network administrators to inject manually certain rules to override any possible decisions made at real-time. Based on special cases, the list of access switches can be extended. For example, it is possible to have an e-mail-access switch for an e-mail server, or websites access switch for a web server and so on.

Normal mode can work with proactive network where access switches can start empty and decisions are made based on traffic. Rules can be added by network administrators in normal mode. However, those rules can be possibly overridden by dynamic flows. For example, if a network admin has one rule in ports access control to deny port 80. And we have a flow of (IP:192.168.1.1, port:80) while the IP addresses was permitted in the IP access table, the flow will be permitted and the record of denying port 80 will be changed to 'permit'.

3.1.5 Examples

We will consider several examples to demonstrate how data is added to access switches and how access authorisation is applied.

- Let us start with a normal mode where all tables are empty (i.e., switches' flows tables and entities access tables). For simplicity assume zone is (in), i.e., flow is coming to the network and that there is no time constraint in (schedule). Flow attributes: IP: 192.168.1.1, MAC:a.b.c, and port number=8000. As access controller is in security mode, and as there are no records in switches, flow will be permitted and access controller will write a rule in each one of the three access switches (IP: 192.168.1.1, MAC: a.b.c and port: 8000), with "permit" decision zone="in" and

schedule = “any”. In normal mode, those can work as a bulk permission. In other words, any flow coming to port “8000” will be permitted, any flow coming to MAC “a.b.c” will be permitted and so for the IP address. It is clear that this mode favours performance on security where security breaches may happen if for example an IP-spoofing case occurs and the flow will be authorised based on the IP address only without verifying the port, MAC address, etc.

3.1.6 A hybrid approach

While we think that both earlier described modes have their applicable use cases, however, in most scenarios, security mode is very rigid and many false positive cases may occur where access controller will be denying legitimate flows. On the other hand, ‘normal’ mode is very lenient, and many false negative cases where access controller will allow illegitimate traffic may occur. We assume that if the flow matches no record in any switch, it will be permitted in the normal mode. In the hybrid mode, this will not be allowed. This means that administrators should have some initial default values that can be used as default values.

A hybrid approach should be able to compromise between the two modes. The idea of the hybrid approach is that we assume that most security attacks succeed in breaking one access control at a time. Hence, each flow is only required to get two kinds of verifications to be allowed to pass through the access controller.

This mode should also work in active and proactive cases. However, access switches cannot be empty. For example, network administrators can flush IP and MAC addresses of all their local users. They can also select very specific ports in port access control based on known applications.

Permit override is decided if flow succeeds in acquiring two ‘permit’ decisions from two different access switches.

Example

In the hybrid mode, default values are expected in all access switches. Those however can be overridden dynamically. Assume that local network IP addresses are all virtual 192.x.x.x. Assume also that network administrators included all registered MAC addresses in the MAC access controller. Further, the following ports are open: 80, 8080, and 1030. Network can handle normal traffic and if a user wants to add a new application that wants to open a new port, they need to have legitimate IP and MAC addresses. In that case, the access controller will automatically allow or permit this request. In addition, this new port can be open for other users given that it was opened by a legitimate user.

The same scenario can be applied if request is from users or applications. At the high level, request should include user and application. In addition, at least one of the low level information (i.e., MAC, IP addresses or port number) should be included, or collected using the request.

For example, a user may install a new application that has no record in the access control. In the first time, they can use it only from existing legitimate user account and also verified IP or MAC address. While such hybrid approach can ensure secure access control verification, it can also accommodate changes dynamically.

Global policy and central access controller can hence achieve the following goals:

- Policies should be read from all access control systems. Decisions made about an access request should consider all access control systems.
- A policy driven network that is driven by high level policies.
- The ability to automate policy different activities including policy enforcement, implementation, orchestration, configuration, etc.

For simplicity, it is possible to have two tables rather than one in each access switch (black and white list tables). In that case, decision field will be eliminated and this may accelerate the process of finding a possible match. Of course, each table can be also divided into source and destination or based on the zone attribute value (i.e., in/out/both).

Some firewalls or access controllers handle conflict between different access lists by giving priorities. For example this is a sequence in one of Cisco firewalls (MAC egress, IP egress, MAC ingress, and IP ingress). In addition, specific rules supersede general rules. For example, a rule that deny a specific IP address has more priority on a rule that permits all IP addresses (i.e., using any keyword).

4 Three-layer policy architecture

Policies orchestrate and translate interactions between humans and machines. On the other hand, this should not be a one way interaction (i.e., instructions from humans to machines). In the network case, in particular, network should be able to pass back information about traffic and network status and such information should impact policies.

There are two modes in OpenFlow networks to add policies from the controller to flow tables. In the reactive mode, no initial rules are inserted and controller inserts rules in real-time based on incoming or outgoing traffic. The advantage of this approach is that it is real-time and is optimised to the exact traffic network. The disadvantage is that initially or for a starting period (that can be short or long) network response time will be slow as each flow will be sent to the controller to make judgement about. In addition, controller will be overwhelmed with many flows.

In the proactive mode, network administrators can define based on their network a file of initial flow rules. The controller can then flush them one time to switches flow tables. Unlike the reactive approach, those rules may not be completely optimised to the current network. However, from a performance perspective, this can be a better approach lowering the overhead on the network and the controller in particular.

Proactive controllers can then have a very intelligent list of policies in which it knows exactly what to do with each traffic. Occasions should rarely occur when the controller is not sure what to do with a particular flow. False positive decisions can result in dropping very important traffic and false negative decisions may cause serious malicious traffic to go through the network. The ability for the controllers to make such proper decisions depends on the tool it has to make such decisions. A controller supporting module (i.e., an ABAC authorisation module) can be one of those very important supporting modules.

We can summarise objectives that should exist in future policies:

- A global policy architecture where high level and low level policies are connected to each other.
- Policy sets should be written in a language close to human languages and use expressive text understood by administrators.
- Policy sets should be abstract and should be location independent and not be tied to a particular network, topology, etc. We mentioned earlier that network and traffic should be monitored and this information should impact policies. This may contradict the requirement of having policies that are network independent. We will elaborate on this later on.
- Rules should be implemented with location, network and topology variables.
- Users manually write or modify policies.
  Everything else should be automated through software programs. The policy activities to automate include: implementation (i.e., translating policy sets to policies and then to flow rules), enforcement (i.e., check based on incoming and outgoing traffic whether policies are observed or violated and permit/deny based on that), migration (transferring policies from other networks or systems), reconfigured or transformed (i.e., if the network is changed, where a switch or a host is added for example, policy or policies should be accommodate that automatically).

Policies in network security serve three different levels:

1 At the application level, users write policies to regulate users-applications-systems interactions. They can specify who can do what, when and how. However, at this level, users are not identified as individuals but as groups. Network, systems and applications are only identified by general names without any technical terms. Typically, we expect policies such as:
   - Employees should not be able to access accounting services remotely.
   - Students should not be allowed to use smart devices during exams.
   - Users can have unlimited internet download speed only after working hours.

   In those examples, we showed that at this level, policies or policy sets should be for groups and not individuals (as individuals represent instances of their groups which can be specified in level 2). Similarly, applications and devices are known by general categories that can have several instance examples (e.g., accounting, smart devices, and internet). For simplicity we will call them at the first level as policy sets, at the second level as policies and at the third level as rules. Policy sets include policies and policies include rules.

2 Level two should include information typically included in ACLs. This is an intermediate stage between high level policy sets and low level rules. Every authorised person, application, or service should have an entry in this access control system. There are currently several examples of ACLs such as those that exist in operating systems, databases or websites active directory or user management, ACLs in firewalls, port control, and routers. Information in those control lists can be rule-based (RBAC) or attribute-based (ABAC).

3 Rules in flow tables and firewalls in particular. Those should have the same attributes exist in flows so that it is possible to check and match those rules with flows can be simple, dynamic and direct. Since those rules will talk to and direct low level network components, for performance issues, they need to be simple and straightforward.

Two-way communication should be designed between each two consecutive layers. From top to bottom, special tools should be developed to allow automatic translation from high to low level terminologies. On the other hand, information from bottom up should be used to improve policies. ACLs in the middle layer provide constraints on flows at the low level. On the other hand, a special module should be developed to support a feedback control where information from network flows can be used to trigger future rules in ACL. Data mining, AI and patterns' recognition methods can be used to analyses network traffic and make rules' recommendations. Those can be triggered for security purposes such as security breaches or attacks or they can be triggered for QA purposes (e.g., performance). Between ACLs and high level policies, modules should be developed to allow automatic translation of policies to ACLs. On the other hand, feedback control is also recommended to re-evaluate existing policies or trigger adding new ones based on network traffic and environment.

5 Conclusions

Access control is one of the most important tasks in the management of systems, networks, databases, etc. Ultimately, the goal is to allow all legitimate users to have exactly access levels they are supposed to have and also prevent any illegitimate user or request to access internal assets or resources. This is all should be accomplished with

both extremely high level of accuracy and also performance. While those two quality attributes typically contradict each other, such access control systems should achieve high percentages in both quality attributes.

Most classical access control systems work in different layers or levels of abstractions. For many practical reasons, a global central access control system is seen unrealistic. In this paper, we proposed a global central access control system utilising SDN. We demonstrated the design and feasibility of such a system. We believe that the advantages of achieving such global access control systems surpass any challenges to design or implement them.

References

Al-Said, T., Rana, O. and Burnap, P. (2015) 'VMInformant: an instrumented virtual machine to support trustworthy cloud computing', *International Journal of High Performance Computing and Networking*, Vol. 8, No. 3, pp.222–234, DOI: 10.1504/IJHPCN.2015.071257.

Alsmadi, I. and Xu, D. (2015) 'Security of software defined networks: a survey', *Computers & Security*, Vol. 53, pp.79–108.

Bismark (2015) *The Broadband Internet Service Benchmark* [online] http://projectbismark.net/ (accessed November 2015).

Casado, M., Freedman, M.J., Pettit, J., Luo, J., McKeown, N. and Shenker, S. (2007) 'Ethane: taking control of the enterprise', in *ACM SIGCOMM Computer Communication Review*, ACM, Vol. 37, No. 4, pp.1–12.

Dangovas, V. and Kuliesius, F. (2014) 'SDN-driven authentication and access control system', in *The International Conference on Digital Information, Networking, and Wireless Communications (DINWC2014)*, The Society of Digital Information and Wireless Communication, pp.20–23.

Feamster, N., Nayak, A., Kim, H., Clark, R., Mundada, Y., Ramachandran, A. and Tariq, M.b. (2010) 'Decoupling policy from configuration in campus and enterprise networks', in *2010 17th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*, IEEE, pp.1–6.

Ficco, M. (2013) 'Security event correlation approach for cloud computing', *Int. J. of High Performance Computing and Networking*, Vol. 7, No. 3, pp.173–185.

Hinrichs, T.L., Gude, N., Casado, M., Mitchell, J. and Shenker, S. (2008) *Expressing and Enforcing Flow-based Network Security Policies*, University of Chicago, Tech. Rep.

Hinrichs, T.L., Gude, N., Casado, M., Mitchell, J.C. and Shenker, S. (2009) 'Practical declarative network management', in *WREN*, pp.1–10.

Nayak, A., Reimers, A., Feamster, N. and Clark, R. (2009) 'Resonance: inference-based dynamic access control for enterprise networks', in *Proceedings of the Workshop on Research on Enterprise Networking (WREN)*, Barcelona, Spain, 21 August, pp.11–18.

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', *NDSS 2013*.

Voellmy, A., Kim, H. and Feamster, N. (2012) 'Procera: a language for high-level reactive network control', in *Proceedings of the 1st Workshop on Hot topics in Software Defined Networks*, ACM, August, pp.43–48.

Wang, H., Zheng, Z. and Yang, B. (2015) 'New identity-based key-encapsulation mechanism and its applications in cloud computing', *Int. J. of High Performance Computing and Networking*, Vol. 8, No. 2, pp.124–134.