An Access Control Architecture for Securing Multi-Tenancy Cloud Environments

Ronald Beaubrun
Department of Computer Science and Software Engineering
Laval University
Quebec, Canada
e-mail: ronald.beaubrun@ift.ulaval.ca

Alejandro Quintero
Department of Computer and Software Engineering
Polytechnique Montreal
Montreal, Canada
e-mail: alejandro.quintero@polymtl.ca

***Abstract*— In multi-tenancy cloud environments, physical resources are transparently shared by multiple Virtual Machines (VMs) belonging to multiple users. Implementing an efficient access control mechanism in such environments can prevent unauthorized access to the cloud resources. In this paper, we propose an access control mechanism called CloudGuard that provides scalable and secure access control to the cloud in the context of multi-tenancy cloud environments. Such a mechanism prevents malicious tenants from generating and sending unauthorized traffic to the cloud network. Numerical results show that CloudGuard offers better system throughput in critical or high-risk multi-tenancy cloud network environments, where the amount of intrusion traffic is high.**

Keywords-access control; cloud computing; hypervisor; multi-tenancy; security, virtual machine.

I. INTRODUCTION

Cloud computing is a flexible and cost-effective platform for providing business and consumer services over the Internet [1][2][3]. Such a platform is utilized by multiple customers who share computing resources, including CPU time, network bandwidth, data storage space, with other users, which refers to multi-tenancy [4]. By multi-tenancy, Clouds provide simultaneous, secure hosting of services for various customers utilizing the same infrastructure resources [5][6]. However, in multi-tenancy cloud environments, one customer can gain unauthorized access to the information of other customers. In this context, it is important to control the access of network entities to such information.

Access control is a security feature that controls how users and systems communicate and interact with other systems and resources. In general, there are three types of access control: physical access control, technical access control and administrative access control [7][8]. Physical access control refers to the implementation of security measures in a defined structure in order to prevent unauthorized access to sensitive materials. Examples of such control include: security guards, picture IDs, locked and dead-bolted steel doors, biometrics, closed-circuit surveillance cameras and motion or thermal alarm systems. Technical access control employs the technology as a basis for controlling the access to sensitive information throughout a physical structure and over a network. Examples of technical access control are: encryption, smart cards, network authentication, Access Control Lists (ACLs), and file integrity auditing software. Administrative access control defines the human factors of security. All levels of the personnel within an organization are involved in such control. Administrative access control also determines which users have access to which resources and information.

The above types of access control can be integrated into security architectures in order to preserve the integrity, confidentiality and availability of resources that are collocated in multi-tenancy cloud environments. In this paper, we investigate the use of technical access control for proposing a secure access control mechanism in the context of multi-tenancy cloud environments. Such a mechanism will prevent malicious insiders from generating and sending unauthorized traffic to the cloud network.

The rest of the paper is organized as follows. Section II introduces the context and background related to access control in multi-tenancy cloud environments. Section III discusses the main existing methods and models for controlling access in multi-tenancy cloud environments. Section IV presents the main assumptions and principles of the proposed architecture. Section V illustrates and explains a use case scenario. Section VI evaluates the performance of the proposed access control architecture in terms of latency and system throughput. Section VII gives some concluding remarks and perspectives.

II. CONTEXT AND BACKGROUND

As illustrated in Figure 1, a multi-tenant cloud service provider has three essential elements: the cloud manager, the hypervisor and the Virtual Machines (VMs) [9]. The cloud manager is a console of management provided for clients in order to manage their cloud infrastructure, which means creating, shutting down, or starting the instances. The hypervisor, also called Virtual Machine Manager (VMM), allows multiple operating systems (guests or virtual machines) to run concurrently on a host server. Its main responsibility is to manage the application's operating systems (OSs) and their use of the system resources (e.g., CPU, memory and storage). Its role is to control the host processor and resources, and also to allocate what is needed to each operating system.

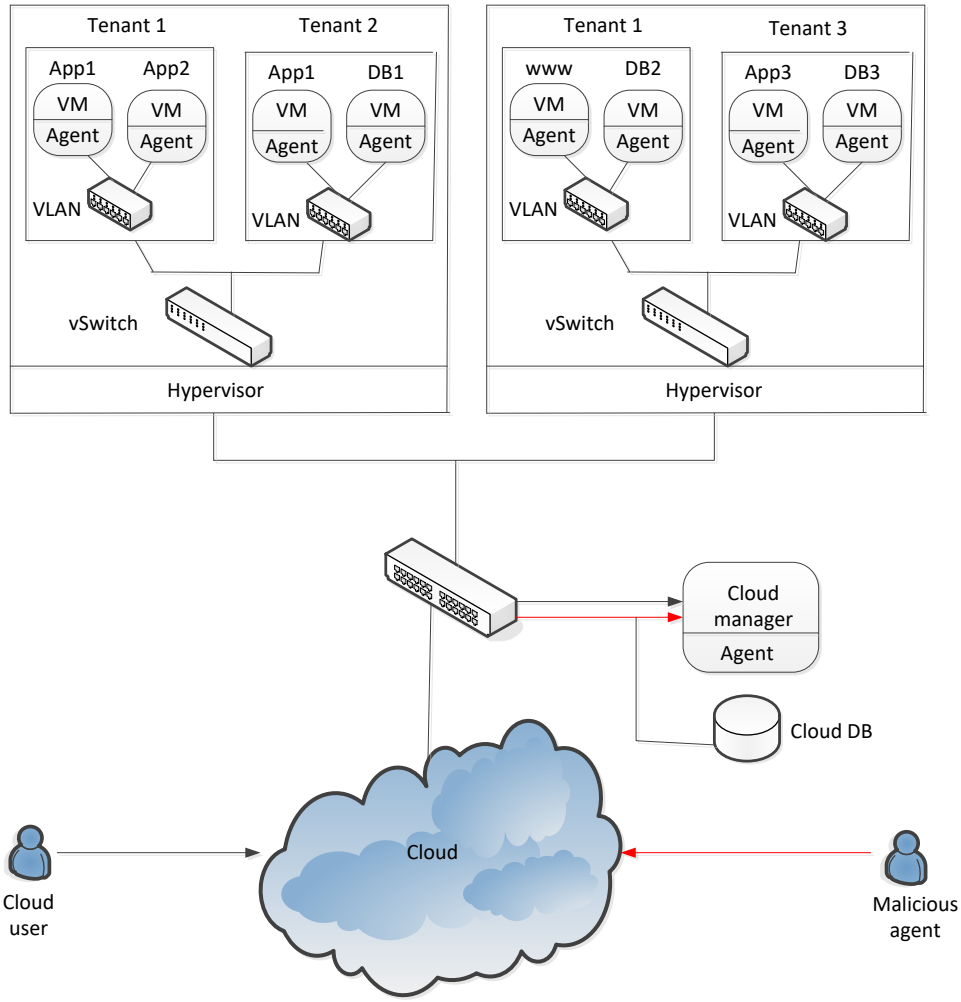

Figure 1. A model for a multi-tenant cloud service provider [9].

A VM is an isolated guest operating system installation within a normal host operating system. In this context, each client may have one or more VMs, as one physical server can host several VMs. In such an environment, one client can send unlimited amount of traffic to another client. Accordingly, a malicious agent can rent a VM on the same host where the target VM resides. This malicious agent can send unauthorized traffic to the target VM and violate the security of the target VM [10]. Intrusion traffic is the amount of traffic that is generated by a malicious VM, and the target of such traffic is to penetrate the vulnerabilities of the destination VM. In principle, intrusion traffic is unpredictable in a multi-tenancy cloud network.

The unauthorized traffic may contain some script or malware which violates the confidentiality or the integrity of the target VM data. Sending such traffic to another VM makes it possible to perform other sorts of attacks. For instance, a malicious agent who owns a VM can perform VM Hopping over another user who is co-located at the same host. With VM hopping, an attacker has the control of one VM and tries to gain the control of another VM. VM hopping allows an attacker to move from one virtual server to the next one, or even to gain the root access to the physical hardware. VM hopping is a considerable threat because several VMs can run on the same host, which makes them the targets for the attacker. By performing this attack, a malicious user can violate the security and steal the data of other users who are located at the same server while compromising the hypervisor file system [11].

In addition, the malicious insider can perform Denial of Service (DoS) attacks. These kinds of attacks exhaust the resources of the cloud network, such as bandwidth and computing power, by sending large amount of unauthorized traffic to other VMs.

III. EXISTING METHODS AND MODELS

In this section, we discuss the main existing methods and models for controlling access in the context of multi-tenancy cloud environments.

A. Distributed access control

The Distributed Access Control (DAC) architecture was proposed by Thomas *et al*. [12]. As illustrated in Figure 2, such an architecture has three main components: the Cloud Service Provider (CSP), the Cloud Service Consumer (CSC), and the Identity Provider (IdP). The CSC requests the resources or services hosted by the CSPs. In this stage, the CSC should be first authenticated to ensure that unauthorized users do not access the services from the CSP. The main responsibility of the CSP is to host and to provide various services or resources to the CSCs. As a result, for avoiding illegal and unauthorized access by CSCs, proper authorization and authentication of CSCs are required.

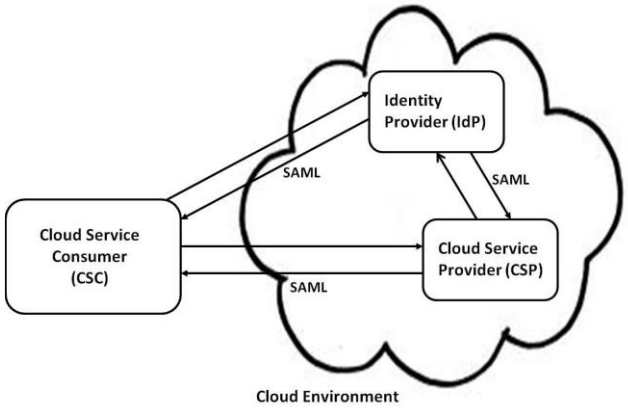

Figure 2. Distributed Access Control architecture [12].

Moreover, in DAC architecture, the IdP plays a great role since it generates identity tokens to the users. By using this identity token, a user can request the access to the cloud. Such a user may subscribe to services from multiple CSPs to meet the resource requirements. In this case, a federated identity management approach is required. The CSCs can use the identity tokens generated by the IdPs and these cloud users can exchange such tokens with various CSPs in the federation [12].

Analysis and results of DAC architecture reveal that using such an architecture is important in the domain of distributed applications or service computing. However, this model has some limitations. In particular, there is no effective mechanism which meets all access control requirements.

B. Adaptive access algorithm

Wenhui *et al*. [13] added trust management to the Role-Based Access Control (RBAC) in order to propose an adaptive access algorithm for cloud environments. This model is based on loyalty, *i. e.*, a user is restricted only when its behavior contains malicious behavior. More specifically, the user request is first analyzed, and based on trust evaluation, the user becomes dynamically authorized. Here, user's trust is calculated according to user's behavior. In other words, the user access to the resource is dynamically based on calculation. As a result, by establishing dynamic mapping between roles and trust values, this model is able to determine the security level and control the user's access to the resources.

The trust-role-based-access control model claims that it can efficiently control user's malicious behavior. However, this model depends on the trust values, as the trust evaluation process needs to be improved in order to become widely used.

C. Multi-tenancy access control model

Multi-Tenancy Access Control Model (MTACM) is a security architecture which embeds the security duty separation principle in multi-tenancy cloud environments [14]. The main idea of MTACM is based on limiting the management privilege of CSP and letting the customers manage the security of their own business. In this model, the duty separation mechanism between cloud service provider and cloud customer is handled by a management module. However, the management module is not user-friendly for customers, as the cloud customer has to take care of the data security.

D. Role-based multi-tenancy access control

Role-Based Multi-Tenancy Access Control (RB-MTAC) applies identity management to determine user's identity and applicable roles [15]. Such a model combines two important concepts in access control under multi-tenancy access environment: identity management and role-based access control. In this context, Yang *et al*. [15] believe that this combination makes it easier to manage privileges that protect the security of application systems and data privacy. Providing a set of privileges and identity management schemes for corporations in cloud computing environment is the main contribution of this security model.

This scheme can be used to easily change employee privileges when a personnel member leaves an organization or when we want to grant employees more access without the need to modify all employee privileges one by one. However, RB-MTAC is not independent, and for implementing it in a cloud computing system, a directory service is needed.

E. CloudPolice

Popa *et al*. [16] proposed *CloudPolice*, a system that implements a hypervisor-based access control mechanism for multi-tenancy cloud environments. CloudPolice operations are illustrated in Figure 3. More specifically, when a source VM initiates a new flow, the source hypervisor sends a control packet to the destination hypervisor. This control packet specifies the security group to which the source VM belongs (Step 1). As soon as the control packet reaches the source hypervisor, it will be checked by the destination hypervisor to verify the policy for the group of the destination VM (Step 2). If the policy allows the traffic, then the state of the traffic will be created for this flow by the destination hypervisor. However, if the traffic is not allowed or should be rate-limited, the control packet will be sent back to the source hypervisor to block or rate limit the flow or the VM (Step 3).

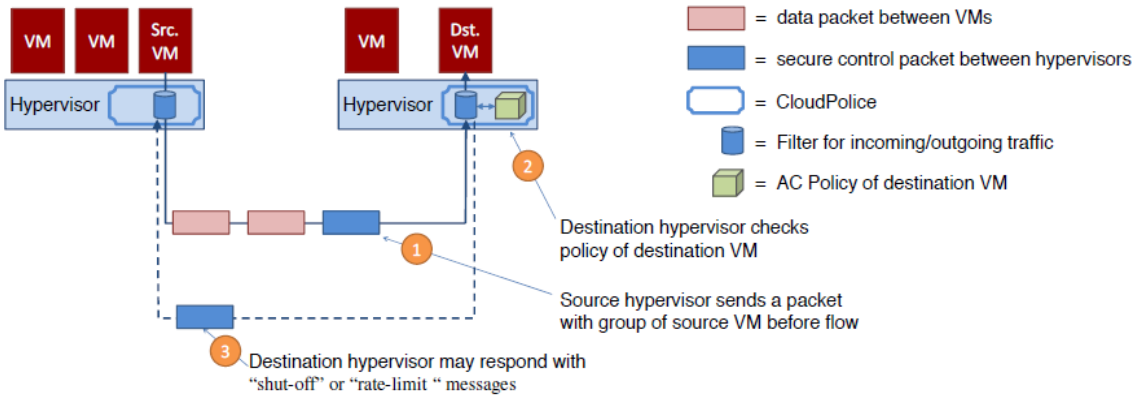

Figure 3. Overview of CloudPolice operations [16].

Since hypervisors are generally trusted, network-independent, close to VMs and fully software programmable, CloudPolice seems to be effective to prevent denial of service (DoS) attacks from malicious agents who send unauthorized traffic to their targets. As a result, CloudPolice acts as stateful firewalls and creates a state for each flow.

However, there are several major concerns for the feasibility of CloudPolice. The first concern is the ability for the hypervisor to act on per flow state, as the hypervisor should be ready to act on every single flow. The second concern is the ability to install new state with low enough latencies for new traffic flows, as we should make sure that the hypervisor is able to create a state for each new incoming flow very fast. As a result, the hypervisor should be able to create states for all new flows without latency (or at least with acceptable latency) and also act on the states that already exist in the buffer. Also, CloudPolice imposes overheads in the system, as the destination hypervisor receives all the traffic and decides to pass or drop the traffic based on the security attributes of the target virtual machines.

IV. THE PROPOSED ARCHITECTURE

This section defines the main assumptions, as well as the design and principles of the proposed architecture.

A. Main assumptions

The proposed architecture deals with the concept of Inter-VM traffic, which is the transmission of any data packet to and from one virtual machine. In other words, when the hypervisor encounters inter-VM traffic, the traffic does not pass through the physical switch or router, as the virtual switch that is located at the hypervisor forwards the packet to the destination VM. At this point, the following assumptions need to be done:

- The virtual machines and physical servers are co-located at the same cloud provider. If the entire system is not part of the Cloud, then for sending traffic to another Cloud, the traffic should pass through a real router or firewall. In this case, the policies that are implemented in the firewall should be enforced.
- Each physical server has only one hypervisor. In this case, the security attributes and access control lists of all virtual machines that belong to a physical server are located at one hypervisor. If we have multiple hypervisors on a physical server, we should apply an extra process for realizing which hypervisor contains the access control lists of certain virtual machines.
- Each physical server is hosting at least one tenant, and each tenant has at least one virtual machine. Since each virtual machine should be registered as a tenant, if a tenant is registered in the Cloud, a virtual machine should be assigned to that tenant.
- All access control lists are defined and stored in the hypervisor.
- In its startup process, a hypervisor sends an update message to the other hypervisors that are located at the same Cloud. This update message contains the IP address and the ID of virtual machines that are located at that hypervisor.

B. Architecture principles

The principles of the proposed architecture are based on control packets, which is the core element for verifying security permissions of virtual machines in multi-tenancy cloud environments. In the following, we explain the main elements of the proposed access control architecture, which is illustrated in Figure 4:

- Source (Src.) VM is a virtual machine that is installed on the source hypervisor, as the latter is located at the physical source server. The source VM is then sending traffic packets to a virtual machine in the same Cloud called Dst. VM.
- Destination (Dst.) VM is installed at the destination hypervisor, and this hypervisor is located at the destination physical server.
- A data packet is a packet that the source VM wants to send to the destination VM.

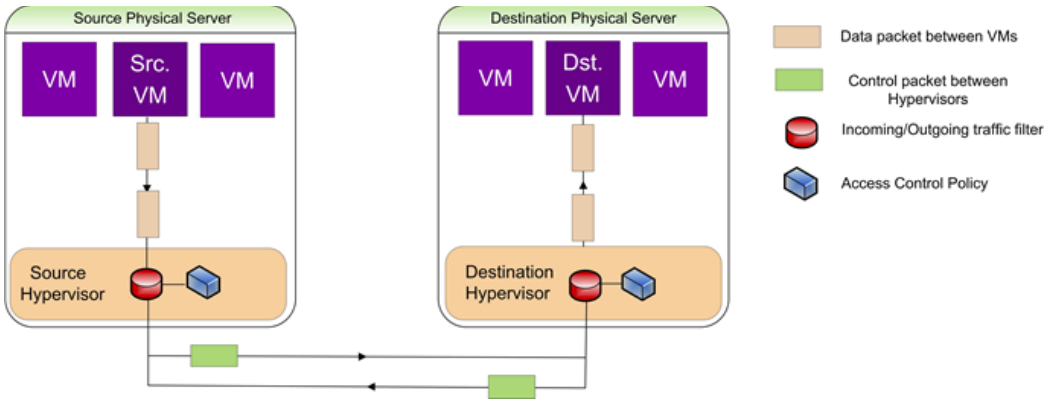

Figure 4. Principles of the proposed architecture.

- A control packet is a special packet that is generated by the source hypervisor. Its content represents the specifications of the source and destination VMs.
- Incoming/outgoing traffic filter is a lightweight IDS that is integrated in the hypervisor. It compares the control packet with the access control lists of destination VM.
- An access control list is a set of security permission that defines the level of security of each virtual machine.

C. Architecture design

The main goal of the proposed architecture is to block and drop undesired packets as close as possible of the source hypervisor. As illustrated in Figure 4, when the source VM sends traffic to the destination VM, such traffic has to pass through the source hypervisor. As soon as a data packet reaches the hypervisor, it generates a control packet which consists of the necessary information for access control checking, such as the source IP address, the destination IP address, the port numbers, as well as the protocol type. Such a control packet has to be sent to the destination hypervisor which checks its content and decides whether the traffic can be delivered to the destination hypervisor. If the source VM is permitted to send the so-called traffic to the destination VM, the destination hypervisor adds a pass or drop value to the control packet payload, and sends it back to the source hypervisor. According to this value, the source hypervisor threats the awaiting traffic.

As illustrated in Figure 5, the process starts when a VM initiates to send some traffic to another VM. As soon as such traffic is received by the source hypervisor, it checks the packet and looks for the destination address that is located at the inserted IP packet header. If the destination address belongs to a virtual machine in the same cloud, we will have two possibilities. The first case considers that the destination address is located at the same physical server. In this case, the architecture checks the access control policy of the destination VM, and can decide whether to pass or drop the traffic. The second case occurs when the destination address is located at a different physical server. In this case, the source hypervisor generates and sends the control packet to the destination hypervisor. Then, it waits for the response control packet.

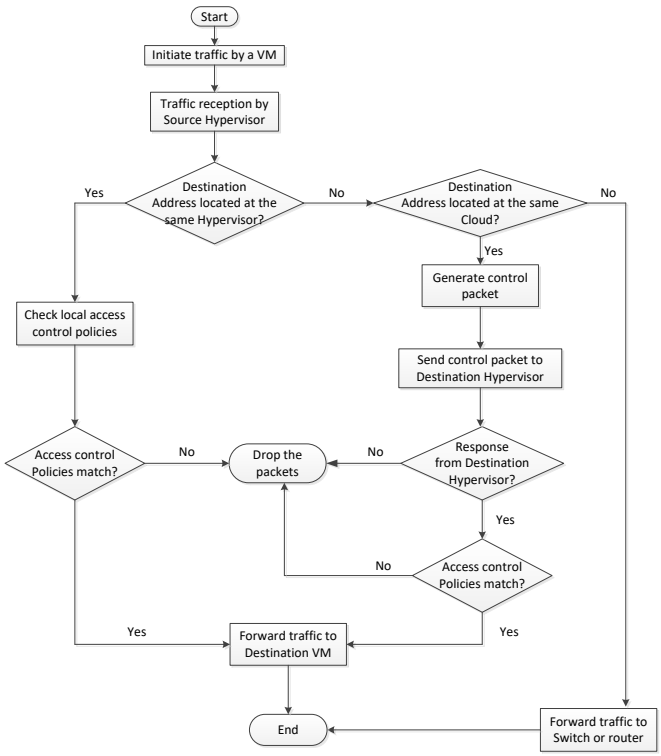

Figure 5. General mechanism flowchart.

Beside such possibilities, there may be an exception, when the destination address does not belong to any VM in this Cloud, which means that the source and destination addresses belong to two devices that are not co-located at the same Cloud. In this case, the architecture only has to pass the traffic to the default gateway of the source hypervisor (router, switch or firewall).

The main part of the mechanism starts if the destination address belongs to a VM that is located at a destination hypervisor. In this case, the whole traffic should wait until the source hypervisor generates and sends a control packet to the destination hypervisor. Hence, the decision will be made based on the response control packet. Figure 6 shows the main tasks of the destination hypervisor when it receives the control packet from the source hypervisor. More precisely, the destination hypervisor selects one of the following actions:

- Insert a pass value to the control packet if the access control policy of the destination VM matches, and accept the traffic from the source VM.
- Insert a drop value to the control packet if the access control policy of the destination VM does not match, as the source VM is not authorized to send the traffic to the destination VM.
- Insert a null value to the control packet if the destination address is not found in the destination hypervisor. This may happen if the control packet is sent to the hypervisor by mistake, or if the VM destination is migrated to another hypervisor, whereas the source hypervisor is not informed about such migration.

After inserting the proper value to the control packet, the destination hypervisor returns the edited control packet to the source hypervisor. The response control packet contains the decision and the action to be taken for the traffic. In the case of a drop value, the source hypervisor drops the traffic right away, as such traffic will not even exit the hypervisor, which means no wasted and unnecessary traffic in the network. Consequently, the network bandwidth does not suffer from extra and unwanted traffic. Finally, the pass value indicates that the access control policy matches between the source and destination, whereas the source VM and the traffic will pass throughout the destination hypervisor.

V. A USE CASE SCENARIO

In this section, we analyze a use case scenario which enables to tackle the problem of sending unauthorized traffic to a VM in the context of multi-tenancy cloud environments. This scenario is illustrated in Figure 7, where a public Cloud is connected to the Internet, using a router and three physical servers that are connected to a layer-2 switch. In this scenario, the function of the router is to route the internal cloud traffic to the Internet. Apparently, the router serves as a controller, enabling the networked devices to talk to each other efficiently.

In this scenario, there are 3 physical servers, as well as 10 virtual machines. These virtual machines belong to 4 tenants. The multi-tenancy topology of this Cloud is as follows:

- Server 1: Tenant 1 (VM1, VM2) and Tenant 2 (VM3);
- Server 2: Tenant 1 (VM4, VM5) and Tenant 3 (VM6, VM7);
- Server 3: Tenant 4 (VM8) and Tenant 3 (VM9, VM10).

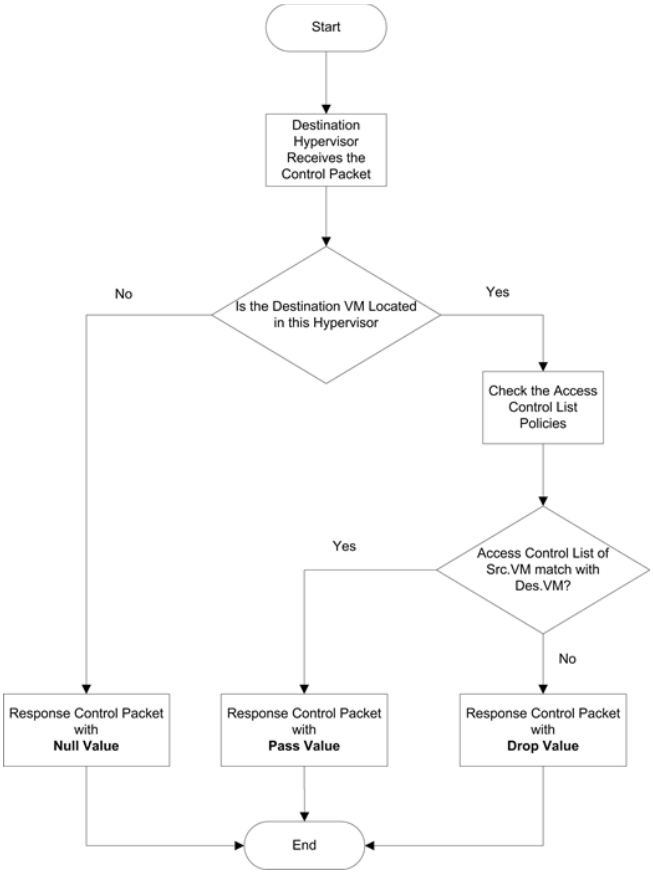

Figure 6. Destination hypervisor's tasks after control packet reception.

It is important to mention that the process of controlling the access is executed in the hypervisors. In this context, the scenario has two phases: the first phase consists of generating control packets, whereas in the second phase, the destination hypervisor investigates the information and decides about the destiny of the packet. More specifically, in phase one of the scenario, the VM Source sends a traffic flow to the hypervisor source, as illustrated in stage 1 of Figure 8. Then, the source hypervisor generates a control packet. The content of this control packet is based on the traffic to be sent from source VM3 to destination VM8.

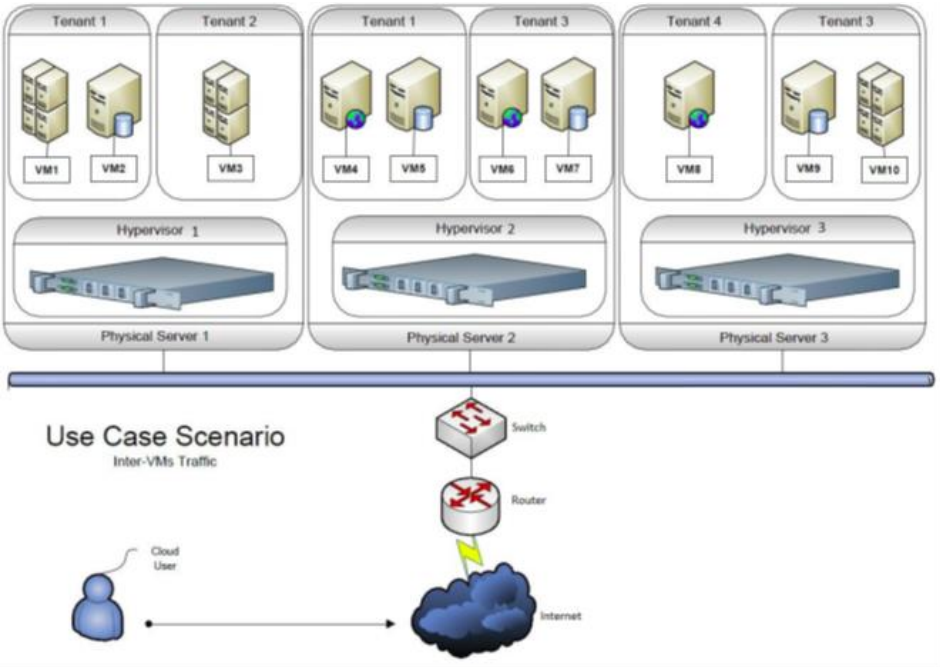

Figure 7. A use case scenario for multi-tenancy cloud access control.

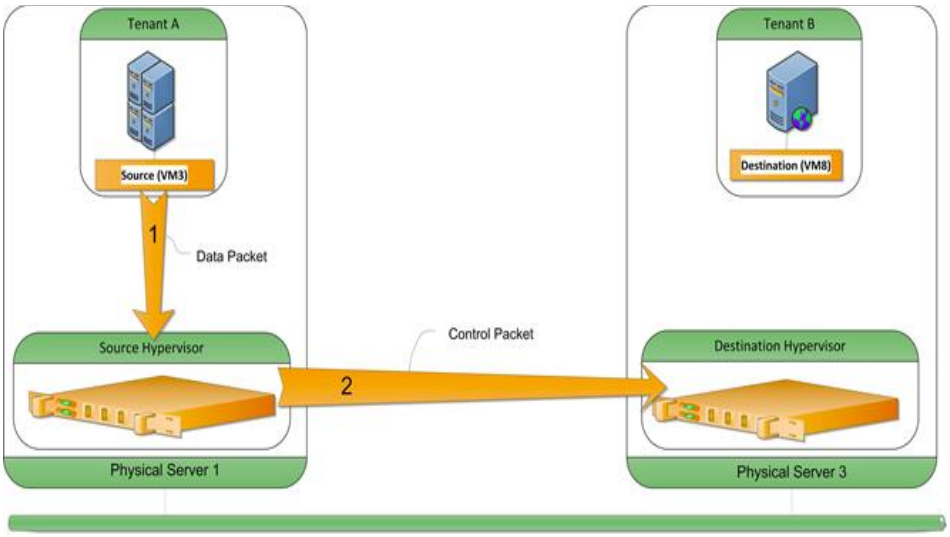

Figure 8. Illustration of phase one of the scenario.

As illustrated in stage 2 of Figure 8, the source hypervisor sends the control packet to the destination hypervisor in order to check the access control policy of the VM destination.

In phase two, the control packet arrives at the destination hypervisor which checks the access control lists (ACLs) to verify if VM3 is authorized to send traffic to VM8. If the ACLs related to VM8 match, the destination hypervisor sends back a pass value within the control packet (called response control packet) to the source hypervisor, as illustrated in stage 3 of Figure 9. The response control packet enables the hypervisor source to decide what to do with the traffic that is waiting in the source hypervisor. Hence, if the security attributes of VM8 do not match the data packet, then the destination hypervisor sends a drop signal to the source hypervisor.

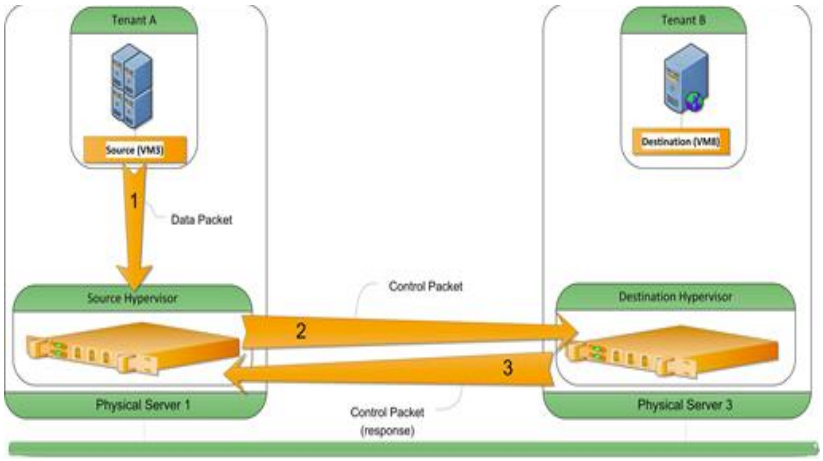

Figure 9. Illustration of phase two of the scenario.

VI. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of the proposed access control architecture that we call *CloudGuard*. More specifically, we will define the parameters that will enable to mathematically express the latency and system throughput. Then, we will present numerical results and analysis.

A. Parameter definition

Let us first define the most important parameters that characterize the proposed access control architecture:

- N_p: the number of packets transmitted from source VM to destination VM;
- S_p: the data size of each packet (bits);
- N_f: the number of traffic flows that are being transmitted from source VM to destination VM;
- S_f: the size of traffic flow (bits);
- T: the time taken by the traffic flow to be sent from source VM to destination VM (seconds);
- RTT: the time taken by a control packet to be sent from a source hypervisor to a destination hypervisor, and received back by the source hypervisor (seconds). This parameter also refers to as *Round Trip Time*;
- BW: The amount of data that can be carried from one point to another point in a given period in a Gigabit Ethernet (1 073 741 824 bits per second);
- P_i: the probability of having intrusion traffic.

B. Packet latency

We may now evaluate the latency of the proposed access control architecture and compare it with that of CloudPolice [16] that we studied Section III. While other different solutions may be found in the literature for comparison with the proposed architecture, CloudPolice is a hypervisor-based access control system that implements, like CloudGuard, the access control only within the hypervisor. In other words, both architectures are designed to enforce the access control list in the hypervisor.

However, a major difference between such architectures is that with CloudGuard, the source hypervisor waits for the control packet response from the destination hypervisor. In this case, if the security policies of source VM and destination VM match, the source hypervisor passes the traffic toward the destination hypervisor. As a result, the destination hypervisor passes the traffic to destination VM. On the other hand, according to CloudPolice, the source hypervisor does not wait for the control packet response, as it sends the whole traffic to the destination hypervisor. When the traffic reaches the destination hypervisor, the security policies will be checked between the traffic and destination VM. In this case, if the traffic is authorized, it will continue its journey to destination VM. However, if the traffic does not match with the security attributes of destination VM, it will be ignored and dropped in the destination hypervisor.

From the differences between CloudGuard and CloudPolice, we can generate two expressions for evaluating the latency related to each packet: one for the latency obtained with CloudGuard, and the other one for the latency obtained with CloudPolice. With CloudGuard, the control packet is generated by the source hypervisor, whereas the control packet response is generated by the destination hypervisor. When one VM wants to communicate with another VM in the same cloud environment, the source VM first sends the traffic to the source hypervisor. Then, the source hypervisor checks the destination address, and finds out the destination hypervisor on which the destination VM is hosted on.

In the next step, the source hypervisor generates a control packet, and sends such control packet to the destination hypervisor. Then, it keeps the traffic, and does not let the traffic to be passed to the destination hypervisor, unless it receives back the control packet response from the destination hypervisor. In this stage, the responsibility of the destination hypervisor is to check the traffic security attributes with the security policies of destination VM. If the traffic is authorized,

the control packet response asks the source hypervisor to pass the traffic.

As a result, for each traffic flow, one control packet needs to be generated with CloudGuard. Based on that, to evaluate the average packet latency, we need to multiply the number of traffic flows N_f with round trip time *RTT*. More specifically, to derive an expression for the latency, RTT is first added to the time it takes to transmit one traffic flow, while only considering non intrusion traffic. Then, we add this value to the round trip time of the control packet for intrusion traffic. Such process is repeated for each traffic flow in order to obtain the packet latency. Therefore, the average packet latency obtained with CloudGuard is expressed as follows:

$$D_{CG} = \left(RTT + \frac{S_f}{BW}\right)(1 - P_i) + RTT \times N_f \times P_i \qquad (1)$$

where *RTT*, S_f, *BW*, P_i and N_f are defined in subsection A.

Let us now evaluate the average packet latency while considering CloudPolice. With such access control architecture, when a VM needs to communicate with another VM in the same cloud environment, the source VM sends the traffic to the source hypervisor. The source hypervisor checks the destination VM address and passes the traffic to the proper destination hypervisor where destination VM is hosted. Then, the destination hypervisor checks the traffic security attributes with the security policies of the destination VM. If the traffic is not authorized, the destination hypervisor generates a control packet, and sends it to the source hypervisor in order to inform the source hypervisor to block the next stream of the same traffic, or to limit the bandwidth that must be allocated to this traffic. Therefore, CloudPolice generates control packets only for intrusion traffic flows. As a result, the number of control packets that are generated depends on the probability of intrusion traffic.

More specifically, for evaluating the latency obtained with CloudPolice, we first need to multiply the number of intrusion traffic flows with RTT. Then, we need to evaluate the time it takes for intrusion traffic to pass to the network since the amount of intrusion traffic may technically consume the available network bandwidth. Based on that, we first calculate the round trip time RTT of all control packets. Then, RTT is added to the time that it takes to pass the intrusion traffic from source to destination. As a result, the average packet latency obtained with CloudPolice is expressed as follows:

$$D_{CP} = \left(RTT \times P_i + \frac{S_f}{BW}(1 - P_i)\right) \times N_f \qquad (2)$$

where *RTT*, P_i, S_f, *BW* and N_f are defined in subsection A.

C. System throughput

In order to evaluate the proposed access control architecture, the system throughput is also taken into consideration. In this context, we develop two expressions for the system throughput: one expression for the system throughput with CloudGuard and another one for the system throughput with CloudPolice. For evaluating the first expression, we consider that CloudGuard generates a control packet for each traffic flow before passing such traffic to the destination hypervisor. The traffic remains in the source hypervisor until the source hypervisor receives back the control packet response. In this case, only authorized traffic flows can pass through the network.

As a result, for evaluating the system throughput with CloudGuard, we first need to divide the number of traffic flows by the average latency. Then, we multiply the obtained results with the probability of not having intrusive traffic. The system throughput with CloudGuard is expressed as follows:

$$R_{CG} = \frac{S_f N_f}{D_{CG}}(1 - P_i) \qquad (3)$$

where S_f, N_f, and P_i are defined in subsection A, whereas D_{CG} is given by (1).

Moreover, with CloudPolice, no control packet is sent before passing traffic packets to the destination hypervisor. In other words, all traffic flows pass from the source hypervisor to the destination hypervisor. As a result, such traffic flows include unwanted and unauthorized traffic. In this context, when evaluating the system throughput with CloudPolice, we should deduct from such traffic flows the amount of unauthorized traffic, since such unauthorized traffic that is generated by an intruder may occupy the available network bandwidth. Then, the number of traffic flows is divided by the average latency obtained in (2), as we multiply such a result with the amount of intrusion traffic. The obtained expression is valid for the situations where the probability of intrusion traffic is between 0 and 0.5. Further, because of the nature of CloudPolice, we consider the system throughput equal to zero if the probability of intrusion is between 0.5 and 1.

As a result, the system throughput with CloudPolice is expressed as follows:

$$R_{CP} = \begin{cases} \frac{S_f N_f}{D_{CP}}(1 - 2P_i), if\ 0 \le P_i \le 0.5 \\ 0, \qquad\qquad if \quad 0.5 < P_i \le 1 \end{cases} \qquad (4)$$

where S_f, N_f, and P_i are defined in subsection A, whereas D_{CP} is given by (2).

D. Numerical results and analysis

For numerical results, we will compare the system throughput obtained with CloudGuard with that obtained with CloudPolice. For such comparison, MATLAB Release 2021b (R2021b) [17] will be used as a fundamental research tool that can mathematically computes the system throughput for both architectures according to several scenarios. In this context, a number of parameters must be calculated.

First, it is important to evaluate the size of traffic flows S_f. Based on [18], S_f may be expressed as follows:

$$S_f = S_p \times N_p \qquad (5)$$

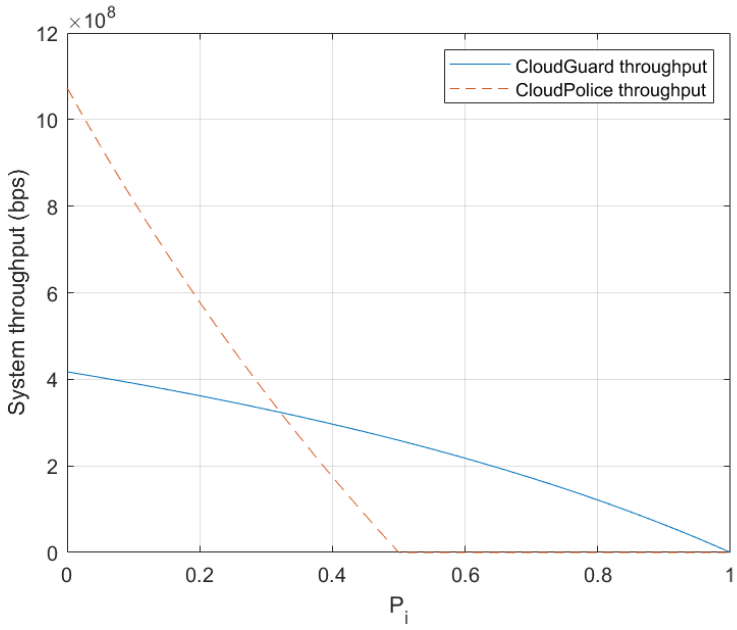

Figure 10. System throughput comparison (CloudPolice vs CloudGuard).

where S_p represents the packet size, and N_p represents the number of packets transmitted from source VM to destination VM. We then consider packet sizes of 128 bytes (*i.e.*, 1 024 bits), and we assume that each traffic flow contains 1 000 packets. As a result, the flow size considered for numerical results will be set to 1 024 000 bits.

Another important parameter for numerical results is the number of traffic flows N_f that are sent from source VM to destination VM. In order to verify how CloudGuard and CloudPolice performs in different situations, we consider N_f as random numbers from 1 000 to 10 000, and we choose the same N_f value for system throughput comparison.

Moreover, in [16], *RTT* is set to 1.5 *ms* with CloudPolice. Since the process of generating control packets with CloudPolice is similar to that of CloudGuard, we assume that RTT is the same for both access architectures, *i.e.*, 1.5 *ms*. Further, as intrusion traffic is unpredictable in a multi-tenancy cloud network, the probability of intrusion traffic P_i will be chosen between 0 and 1 for numerical results.

Figure 10 illustrates the system throughput comparison (CloudPolice vs CloudGuard) in function of P_i. We realize that, if the probability of intrusion is low (P_i < 0.3215), CloudPolice offers better system throughput than CloudGuard. Moreover, for P_i = 0.3215, both architectures have the same throughput, whereas when the amount of intrusion traffic is higher in the cloud system (P_i > 0.3215), results show that CloudGuard offers better system throughput than CloudPolice.

Also, it is important to mention that a lot of intrusion traffic (P_i > 0. 5) completely paralyzes the cloud system with CloudPolice, since the system is unable to offer any throughput for P_i > 0. 5. Such results prove that CloudGuard is more appropriate than CloudPolice in high-risk multi-tenancy cloud computing environments. In other words, CloudGuard is an effective security architecture which is suitable for high-risk environments. A high-risk multi-tenancy cloud computing environment refers to a cloud environment in which the system is the target of many attacks, as lots of intrusion attempts and unwanted traffic are being sent to the VMs. Financial institutions and game centers are examples of networks that are always targeted by attackers and malicious agents. On the other hand, one can hardly imagine a network environment with zero intrusion traffic, which means that P_i = 0.

VII. Conclusion

The access control architecture (called CloudGuard) proposed in this paper for multi-tenancy cloud environments satisfies a number of requirements, such as scalability and security. This architecture is scalable in the sense that, if the number of VMs grows, we only need to implement this architecture in the hypervisor of each physical server without any extra changes in the system. Besides that, the architecture enables to maintain the security of information in the cloud system by controlling the traffic sent from one hypervisor to another hypervisor and enforcing the security policies in the hypervisor.

Using MATLAB R2021b, we built a mathematical model in order to evaluate CloudGuard performance. More specifically, we compared the system throughput obtained with CloudGuard with that obtained with CloudPolice. Numerical results prove that CloudGuard obviously offers better system throughput in critical or high-risk multi-tenancy cloud network environments, where the amount of intrusion traffic is high. As a result, CloudGuard leads to better

performance by avoiding unnecessary traffic and dedicating the cloud resources to necessary traffic.

Future works will focus on intra-cloud traffic. In real world cloud environment, the traffic may pass through lots of intermediate devices, such as switches, routers and firewalls. In this context, when a VM needs to send traffic to another VM that is located at another Cloud, we need more complex access control updates between hypervisors that are located at different cloud environments. Hence, in the context of intra-cloud traffic, the process of propagation of security attributes and updates between hypervisors should be taken into consideration. Future works will also focus on implementing a prototype of the proposed architecture on a real cloud environment.

REFERENCES

[1] R. Beaubrun and A. Quintero, “A Secure Access Control Architecture for Multi-Tenancy Cloud Environments,” CLOUD COMPUTING 2021: The Twelfth International Conference on Cloud Computing, GRIDs, and Virtualization, 18-22 Apr. 2021, pp. 48-53.

[2] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez, "An analysis of security issues for cloud computing," Journal of Internet Services and Applications 4:5, vol. 4, pp. 5-18, 2013.

[3] M. Auxilia and K. Raja, “Dynamic Access Control Model for Cloud Computing,” Sixth International Conference on Advanced Computing (lCoAC), 17-19 Dec. 2014, pp. 47-56.

[4] Z. Minqi, Z. Rong, X. Wei, Q. Weining, and Z. Aoying, "Security and Privacy in Cloud Computing: A Survey," in 6th International Conference on Semantics Knowledge and Grid (SKG), Beijing, 1-3 Nov. 2010, pp. 105-112.

[5] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, "A framework for native multitenancy application development and management," in 9th International Conference on E-Commerce Technology/4th International Conference on Enterprise Computing, Ecommerce and E-Services, Tokyo, 23-26 July 2007, pp. 551-558.

[6] G. Kappes, A. Hatzieleftheriou, and S. V. Anastasiadis., “Multitenant Access Control for Cloud-Aware Distributed Filesystems,” IEEE Transactions on Dependable and Secure Computing, Vol. 16, No. 6, pp. 1070-1085, 2019.

[7] S. Harris, CISSP All-in-One Exam Guide, Sixth ed. New York: McGraw-Hill, 2013.

[8] K. Albulayhi, A. Abuhussein, F. Alsubaei, and F.T. Sheldon, “Fine-Grained Access Control in the Era of Cloud Computing: An Analytical Review,” 10th Annual Computing and Communication Workshop and Conference (CCWC), 6-8 Jan. 2020, pp. 748–755.

[9] K. Benzidane, S. Khoudali, and A. Sekkaki, "Autonomous Agent-based Inspection for inter-VM Traffic in a Cloud Environment," in 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012), London, 10-12 December 2012, pp. 656-661.

[10] S. J. De and S. Ruj, “Efficient Decentralized Attribute Based Access Control for Mobile Clouds,” IEEE Transactions on Cloud Computing, Vol. 8, No. 1, pp. 124-137, 2020.

[11] A. Jasti, P. Shah, R. Nagaraj, and R. Pendse, "Security in Multi-Tenancy Cloud," in International Carnahan Conference on Security Technology (ICCST), San Jose, CA, 5-8 October, 2010, pp. 35-41.

[12] M. V. Thomas and K. C. Sekaran, "An Access Control Model for Cloud Computing Environments," in 2nd International Conference on Advanced Computing, Networking and Security (ADCONS), Mangalore, 15-17 Dec 2013, pp. 226-231.

[13] W. Wenhui, H. Jing, S. Meina, and W. Xiaohui, "The design of a trust and role based access control model in cloud computing," in 6th International Conference on Pervasive Computing and Applications (ICPCA), Port Elizabeth, 26-28 Oct. 2011, pp. 330-334.

[14] X.-Y. Li, Y. Shi, Y. Guo, and W. Ma, "Multi-Tenancy Based Access Control in Cloud," in International Conference on Computational Intelligence and Software Engineering (CiSE), Wuhan, 10–12 Dec. 2010, pp. 1-4.

[15] S.-J. Yang, P.-C. Lai, and J. Lin, "Design Role-Based Multi-tenancy Access Control Scheme for Cloud Services," in International Symposium on Biometrics and Security Technologies (ISBAST), Chengdu, 2–5 Jul 2013, pp. 273-279.

[16] L. Popa, M. Yu, S. Y. Ko, S. Ratnasamy, and I. Stoica, "CloudPolice: Taking Access Control out of the Network," in ACM Workshop on Hot Topics in Networks. HotNets, Monterey, CA, USA, 20-21 Oct. 2010, pp. 1-6.

[17] Available from https://www.mathworks.com/

[18] Internet Engineering Task Force (IETF), “IPv6 Flow Label Specification”, in Request for Comments (RFC): 6437, ISSN: 2070-1721, 2011.