

On the Design and Implementation of a Security Architecture for End to End Services in Software Defined Networks

Kallol Krishna Karmakar*, Vijay Varadharajan†, Udaya Tupakula†

Advanced Cyber Security Research Centre

Faculty of Science, Macquarie University, Sydney, Australia

kallol.karmakar@students.mq.edu.au*, [vijay.varadharajan, udaya.tupakula]@mq.edu.au†,

Abstract—In this paper, we propose a policy driven security architecture for securing end to end services across multiple autonomous domain based SDN environment. We develop a language based approach to designing a range of security policies that are relevant for SDN services and communications. The design of a security architecture that enables secure routing of packets based on the specified security policies in the SDN Controller is described.

Index Terms—Software Defined Networking(SDN) Security, OpenFlow, Policy Control.

I. INTRODUCTION

Software Defined Networks (SDN) is an emerging technology [1] that offers a promising approach to defend network attacks and complex management of the networking systems. The separation of the control plane from the data plane by SDN results in the network switches becoming simpler forwarding devices with the control logic implemented in software in a logically centralized Controller. This decoupling in the SDN enables the design of new innovative network functions and protocols. Although SDN offers several advantages to deal with complexities in current networks, a critical issue in SDN at present is that of security. The current state of the art in security in SDNs is still at its infancy [2]. Securing networks is becoming more challenging to businesses, especially with bring your own devices (BYOD), increased cloud adoption and the Internet of Things.

Inter-domain routing is the main backbone in today's network infrastructure. There are approximately 50000 autonomous domains. Policy based control in inter-domain SDN is a new area. Policy based routing for legacy network was first introduced by D. Clark in RFC1102[3] during 1989. This paper lags the security issues, which is introduced by Tsudik et al. [4] in the form of $n(n-1)$ symmetric key distribution. In current state of art many SDN controllers do not have full support for BGP. Some weaknesses and proposals for BGP use in SDN are made in [5]. On the other hand the most recent development for SDN inter-domain communication is SDNi[6] from OpenDaylight and SDN-IP[7] from ONOS. Both of them uses BGP and message passing for Autonomous Domains. Some important works regarding intra-domain policy control is found in [8], [9]. Inter-domain communication is not prioritized in these works. Moreover they have different weakness

like granular control, security and complexity in specifying the policy expression. Thus the necessity for secure inter-domain communication in SDN is in demand.

In this paper we present the design and implementation of security architecture for end-to-end services in SDN. A key component of the security architecture is the specification of security policies that are to be enforced on the SDN communications whether they are intra or inter-domain. First we will consider a Policy based Security Architecture for intra-domain interactions and then we will address the inter-domain communications enabling end to end SDN services across multiple domains. This short paper only represents a high level overview of the architecture.

In Section II, we describe the Policy based Security Architecture for end-to-end services. Section III discusses about policy specifications for intra-domain for our application and Section IV demonstrates Inter-domain communication behaviour for our application. Section V concludes.

II. POLICY BASED SECURITY ARCHITECTURE FOR SDNs

Each Autonomous Domain consist of an SDN controller and the hosts are attached to OpenFlow forwarding devices (routers, switches etc.). In an intra-domain communication, the traffic from source to destination passes through devices with a single SDN domain and the requested services are provided by the servers and devices in the same domain. In this case, the traffic and service requests are subjected to security policies in the SDN Controller of that domain. Inter-domain communications involving multiple domains requires cooperation between SDN Controllers, as the communications are subject to security policies in multiple Controllers. First we will consider a Policy based Security Architecture for intra-domain interactions and then we will address the inter-domain communications enabling end to end SDN services across multiple domains.

Figure 1 shows the Policy based Architecture(PbSA)for securing the communication in a SDN domain. We have designed and developed the Policy based Security Architecture as a Security Application running on top a SDN Controller for flexibility reasons. As PbSA is designed in a modular fashion, the components of PbSA can be implemented on a single host or distributed over multiple hosts. Before describing different

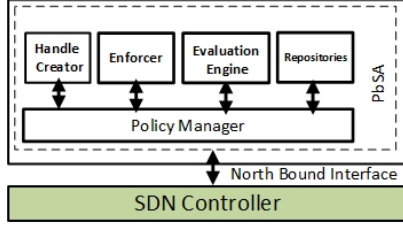


Fig. 1. Policy based Security Architecture for SDN

modules of PbSA, first we present a high level overview of the operation of the overall architecture. We assume that each AS domain is controlled by a SDN Controller. Each Controller maintains and updates a Topology Repository and a Policy Repository. The Topology Repository contains the network topology information discussed in Section II-A. The Policy Repository contains the policy rules and specifications which are expressed using a simple language based template described in Section II-B. The core component of PbSA is a Policy Manager, which manages every single operation of the security system. An Evaluation Engine is used to evaluate the incoming network traffic against the relevant security policies for that specific traffic. Following the evaluation, the Policy Manager determines the flow rules which are then conveyed to the Enforcer module. The Enforcer is used for enforcing the policies related to the flows. A Packet Handle Creator module creates the necessary handles for AS domains which is piggy backed with the payload from the Policy Manager. These handles are used to check the authenticity of the packet and the enforcement of policies at the switches which is discussed in detail in Section IV.

A. Topology Repository

For inter-domain routing of traffic across multiple domains requires an SDN Controller in one AS domain to have knowledge of other Controllers in other AS domains. We have used a pre-built topological maps of different AS domains. We have included an additional security attribute, a Trust Label for each AS domain SDN Controller. The intention of the Trust Label is to reflect the level of trust that particular Controller have. In our current architecture, this Trust Label is hard wired (static) and is specified at the time of installation of the Controller. Consider the distributed SDN environment shown in Figure 2 and the associated Topology Repositories are shown in Tables with in the Figure. Each hexagon in Figure 2 represents an AS domain. We have represented the AS Gateways using Gateway OpenFlow Switches. The OpenFlow Switches are represented using the notation ([source AS ID]SW[destination AS ID]). Hence the Switch connecting AS1 to AS2 will be represented as 1SW2. In each topology table, *AS_ID* is the identity of a particular AS, *Trust_Label* is the trust label of the AS domain, Hops is the distance from source to destination AS.

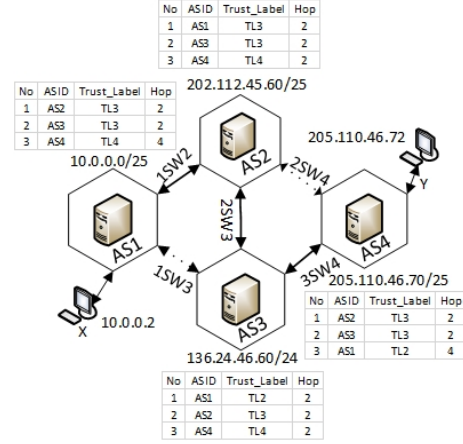


Fig. 2. Topology Views at the Controllers

B. Security Policy Specifications and Policy Repository

A key component of the security architecture is the specification of security policies that are to be enforced on the SDN communications whether they are intra or inter-domain. The specified security policies are stored in a Policy Repository in the PbSA. We have adopted a simple language based approach to specify the security policies; in particular, we have chosen the policy based routing syntax specified in RFC1102 [3] as the basis for our security policy specifications. Policies are rules that specify whether packets follow a particular path or paths in the network and the conditions under which the packets follow these paths. In our language, we have policy terms specifying a range of attributes associated with the flow and the entities in the SDN. These include the following:

- Flow Attributes: Flow ID, sequence of packets associated with the Flow, type of packets, Security Profile indicating the set of security services that are to be associated with the packets in the Flow
- Autonomous System Domain Attributes: AS Identities such as Source AS and Destination AS Identities, Identities of Entry and Exit Gateway/Switch to AS, AS Type (e.g. Commercial Domain, Government Domain) and Trust Label associated with the AS.
- Switches Attributes: Identities of the Switches and Trust Label of the Switches
- Host Attributes such as Identities of Hosts such as Source Host ID and Destination Host ID
- Constraints such as Flow Constraints (FlowCons) and Domain Constraints (DomCons) associated with a Flow; example: video traffic blocking in flows and specific switch trust label uses within a domain.
- Services (for which the policy expression applies),
- Time Validity (the period for which the policy expression is valid) and
- the Path (In case of intra-domain it indicates a specific sequence of switches and in the case of inter-domain communications it indicates the sequence of Autonomous

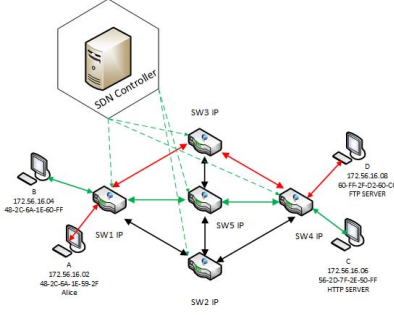


Fig. 3. Intra-Domain Example Scenario

Systems traversed by a Flow

The policy language has wild cards in its syntax enabling specification of policies that can apply to sets or groups of entities and services. When a policy expression is satisfied, then the associated action is performed which could be simple as just allow or deny the request. Hence using these policy terms, one is able to specify different sets of policy expressions to deal with a range of scenarios in both intra-domain and inter-domain communications in a distributed SDN environment.

III. INTRA DOMAIN SCENARIOS

First let us consider some simple intra-domain scenarios in a single SDN domain. Using the policy terms mentioned above, a simplified Policy Expression template could be as follows:
 $PE_i = \langle FlowID, SourceAS, DestAS, SourceHostIP, DestHostIP, SourceMAC, DestMAC, User, FlowCons, DomCons, Services, Sec - Profile, Seq - Path \rangle : \langle Actions \rangle$
 where i is the Policy Expression number.

This template will enable us to specify a range of policies for different users, from different locations, accessing different services using different devices following different paths. We will illustrate the use of the Policy Expression using an example scenario which specifies paths for specific services requested from the hosts. Example : Path Based Policy for Services: Consider for example, we are running HTTP and FTP servers on two machines with IP 172.56.16.06 and 172.56.16.08 respectively. We have created a OpenFlow switch matrix with five switches as shown in Figure 3. We want to guide the flows of end hosts 172.56.16.02 and 172.56.16.04 through this switch matrix based on the services they are trying to access. The Policy Expressions could be as follows:

$PE_5 = \langle *, *, *, 172.56.16.04, 172.56.16.06, 48 : 2C : 6A : 1E : 60 : FF, *, *, *, 80, \{Conf, Intg\}, (SW1; SW5; SW4) \rangle : \langle Allow \rangle$

The Policy Expression says that flows from host with IP 172.56.16.04 and MAC 48:2C:6A:1E:60:FF accessing the HTTP server on IP 172.56.16.06, should be protected for confidentiality and integrity and forwarded via OpenFlow switch SW1->SW5->SW4.

IV. INTER DOMAIN SCENARIOS

Let us now consider a simplified Policy Expression template using the policy terms mentioned above.

$PE_i^{AS_k} = \langle FlowID, SourceASAttributes, DestASAttributes, SourceHostIP, DestHostIP, SourceMAC, DestMAC, User, FlowCons, DomCons, Services, Sec - Profile, Seq - Path \rangle : \langle Actions \rangle$

where k is the AS ID and i is the Policy Expression number. In an inter-domain scenario, the various attributes in the Policy Expression are as follows:

Source AS Attributes include the following:

- AS Domain ID
- Source Subnet (SRC_{SUB}) = Source Subnet address from which the incoming packets originate.
- Source AS Controller Entry Switch (SRC_{ENT}) = Address of the entry OpenFlow Switch connected to the SDN Controller of the AS domain
- Source AS Type (SRC_{Type}) = Examples include government domain GOV or commercial domain COM or education domain EDU
- Trust Label (SRC_{TL}) = Trust Label of Source AS (optional)

Destination AS Attributes include the following:

- Destination Subnet (DST_{SUB}) = Destination Subnet address where the packets are to be transferred.
- Destination AS Type (DST_{Type}) = Type of the Destination AS
- Trust Label (DST_{TL}) = Trust Label of the destination AS (optional)

Source/ Destination Host specific parameters:

- Source Device IP (SRC_{IP}) = IP address of the host machine from where the packet originates.
- Source Device MAC (SRC_{MAC}) = MAC address of the host machine from where the packet originates.
- Destination Device IP (DST_{IP}) = IP Address of the destination host machine.
- Destination Device MAC (DST_{MAC}) = MAC Address of the destination host machine.

FlowCons and DomCons: FlowCons and DomCons represent conditions that need to be satisfied by the path taken by the traffic in the SDN environment. The Path (AS_{SEQ}) is used to explicitly specify a list of AS Domains that is to be traversed by the traffic.

Packet Type (PKT_{TYPE}) indicates the type of the incoming packets. In some cases, port numbers are used to indicate the type.

Time (T^{PE_i}) represents the duration time for which a particular Policy Expression is valid. A wild card implies the policy is valid from the start and there is no time limit.

Actions: Each Policy Expression has an action associated with it. The FlowCons and DomCons impose certain conditions associated with the actions. For instance, if there is a flow constraint which requires that this specific flow of packets of a specific type should only go through a set of switches that can provide a certain bandwidth, then the action requires that

appropriate flow rules be dynamically configured into a set of switches enabling a path that satisfies this constraint. Similarly, from a security point of view, if there is a flow constraint that requires only AS domains that are at a particular trust level be traversed by the specific traffic in question, then the action will require the selection of a path with appropriate AS domains. An action can also have some attributes. For instance, Destination Exit Switch (DST_{EXT}) attribute associated with an action indicates that the exit switch that should be taken by the traffic when the policy is satisfied.

Example: Secure Policy based Routing in Inter-Domain Scenarios:

Policy Expressions are stored in Policy Repository of each domain SDN controller. During packets transaction in each SDN domain controller Policy Manager checks incoming packets against the stored Policy Expression. With inter-domain communications, there is a need to transfer the packets to other AS domains. For such inter-domain scenarios, we introduce two further concepts. First is the notion of a *Handle*, which the PbSA creates and tags it to the Packet. The Handle consists of a list of visited AS domain IDs. The Packet + Handle is then transferred to the next AS Domain Controller. Similar process is being repeated as the packet goes through all the transit AS domain SDN Controllers until the packet reaches its destination. The second concept is that of a *Policy Transfer Token*. A Policy Transfer Token contains the constraints that are transferred from the current AS domain to the subsequent transit AS domains which need to be satisfied as the packet is being transferred. These constraints need to be taken account in addition to the constraints of the transit domains. For instance, if there is a constraint that the traffic should only pass through AS domains of trust level greater than a certain threshold, then this constraint needs to be satisfied by subsequent transit domains. Suppose an AS domain SDN Controller (with AS ID = 10) has a constraint that packets should only be forwarded through a path of AS domain SDN Controllers that have a trust level higher than say $TL3$. In distributed systems, in general, it is not possible for one domain Controller to know about policies of other domains. Hence there is a need to transfer the policy constraints, which are communicated via Policy Transfer Tokens. In this paper, we assume that the AS domain Controllers are trusted and that if and only if the policies can be satisfied in the domain, the receiving Controller will accept the packets. Let us now illustrate the above concepts using a simple inter-domain scenario in Figure 2, which shows two domains AS1 and AS4 connected via domains AS2 and AS3. Each AS domain has a SDN Controller and PbSA with its Policy and Topology Repositories as described above. The Tables beside each of the AS domains in Figure 2 represent the Topology Repository. We have not counted the intermediary node for hop counts between AS domain for simplicity. The Table I shows the policies in each of these AS domains stored in the Policy Repositories of PbSA.

In this scenario, the host machine X (with IP address 10.0.0.2) wishes to communicate host machine Y (with IP address

TABLE I
STORED POLICY EXPRESSIONS

AS ID	Policy Expression
AS1	$PE_1^{AS1} = < *, (10.0.0.0/24, EDU, TL2), (205.110.46.70/25), 10.0.0.2, *, *, *, *, *, *, (80, 443), conf, * > : < (TL2+ =) >$
AS2	$PE_4^{AS2} = < *, (10.0.0.0/24, EDU, TL2), (205.110.46.70/25), 10.0.0.2, *, *, *, *, *, *, (80, 443), conf, (AS3) > : (TL2+ =) >$
AS3	$PE_2^{AS3} = < *, (10.0.0.0/24, EDU, TL2), (205.110.46.70/25), *, *, *, *, *, *, (80, 443), conf, * > : (TL2+ =) >$
AS4	$PE_1^{AS4} = < *, (10.0.0.0/24, EDU, TL2), (205.110.46.70/25), 10.0.0.2, 205.110.46.72, *, *, *, *, *, (80, 443), conf, * > : < Allow >$

205.110.46.72). As X and Y reside in two different AS domains, communication between them occurs via transit AS domains. At first, packets from X goes to the SDN Controller of AS1. As the policy expression PE_1 in AS1 matches with this particular network traffic, HTTP & HTTPS traffic originating from 10.0.0.2 is allowed to go to Y in the subnet 205.110.46.70/25. However let us assume that there is a flow constraint which specifies communications between X and Y must occur only through domains which have trust levels greater than or equal to $TL2$. This will result in the traffic routed through the AS2 domain via the OpenFlow Switch $ISW2$ (Connected to AS1). A similar process occur in AS2 and the traffic will be sent to Y in AS4.

V. CONCLUSION

In this paper we presented the design and implementation of security architecture for end-to-end services in SDN. A key component of the security architecture is the specification of security policies that are to be enforced on the SDN communications whether they are intra or inter-domain. We have presented Policy based Security Architecture for securing communications in single AS SDN domain and then extended the architecture for securing communication in multiple AS SDN domains.

REFERENCES

- [1] O. N. Foundation, "Sdn: The new norm for networks," <https://www.opennetworking.org/images/stories/downloads/sdnresources/white-papers/wp-sdn-newnorm.pdf> [Accessed 12Dec.2015].
- [2] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proc. of the 2nd ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013.
- [3] D. Clark, "Policy routing in internet protocols. request for comment rfc-1102," *Network Information Center*, 1989.
- [4] D. Estrin and G. Tsudik, "Security issues in policy routing," in *Security and Privacy, 1989. Proc., 1989 IEEE Symposium on*. IEEE, 1989.
- [5] C. E. Rothenberg et al., "Revisiting routing control platforms with the eyes and muscles of software-defined networking," in *Proc. of the first workshop on Hot topics in software defined networks*. ACM, 2012.
- [6] H. Yin et al., "Sdni: A message exchange protocol for software defined networks across multiple domains," *IETF draft, work in progress*, 2012.
- [7] A. Koshibe, "Application for onos," <https://wiki.onosproject.org/display/ONOS/SDN-IP+Architecture>.
- [8] J. Reich et al., "Modular sdn programming with pyretic," *Technical Reprint of USENIX*, 2013.
- [9] A. Voellmy et al., "Maple: simplifying sdn programming using algorithmic policies," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 87–98.