Formal model and Policy specification for software defined networks

Dawood salman jasim al-farttoosi

kufa of University

Dawood.jasim@uokufa.edu.iq

Abstract—Software-defined networking SDN is

gaining a sharp increase in adoption by very well-known companies Like Google and Microsoft. Currently, the two stands out reliability and security are the most issues that hampering the SDN rapid growth. This paper aims to contribute to this growing area of research by exploring the SDN security issues. Novel approach will be proposed by utilizing the well-known access control approach called theusage control UCON model and also the Flog policy language of software defined networks SDNs. This work will be produced as a formal modelling via high-level abstract language like Flog.

Keywords—SDN, Usage Control, Flog, Reliability and Formal Model.

المستخلص :

تكتسب الشبكات المعرفة بالبرمجيات زيادة عالية ، وذلك من قبل شركات معروفة جدا غنية عن التعريف أمثال جوجل ومايكروسوفت لذا تبرز في الوقت الحالي مشكلتان اساسيتان هما:

-الموثوقية

-الامان

هاتان المشكلتان تعيقان النمو السريع ل(SDN) ان هذا العمل يهدف الى المساهمة في هذا المجال الحديث للبحث وذلك من خلال استكشاف القضايا الامنية في الشبكات المعرفة بالبرمجيات(SDN) يقترح البحث نهجا جديدا وذلك من خلال الاستفادة من نموذج التحكم بالوصول(Usage Control)وكذلك (flog policylanguagefor SDNS)

1. Introduction

In the digital world, nowadays security becomes more and more critical issue due to the sensitiveinformationand big amount of data transfer over theopennetwork as well as storethese data in cloud's repositories.Enterprise Strategy Group ESG survey shows that 79% of respondents claimed that network security is more challenging than it was [15].Currently, there are about 9 billion devices connected to the Internet and it is expected to be 24 billion in 2020 [1]. These networking devices exchange data in different types of networks such as wirelesssensor, Internet of Things IoT, Cloud networks …etc. The rigidity of traditional network architecture makes it difficult to handle the big amount of data in terms of resiliencemanagement and security policy orchestration [3], therefore the attention turned to find alternative solutions.

To surmount the inflexibility in the conventional network architecture, the emerging of Software defined networking SDN, since mid of 90s, gives hope to createaprogrammable network that can be managed like any computing

device and resource. In SDN*control* and *data* plane have been decoupled as depicted in figure 1 from [2].In this isolation the network configuration becomes easy to handle from single location usually called *controller*.

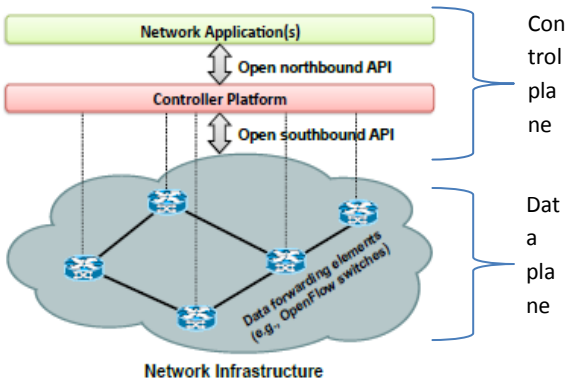

Figure 1: View of SDN architecture

The *controller* considered as the brain of network therefore any error in *control* plan would lead to network failure. Many *controllers* have been introduced till now as shown in table1. Due to this separation the networking physical equipment(like switch, router)become like a dump device that dedicated to perform packet forwarding by depending on set of rules that issue by the *controller*. Generally, SDN network consists of *control plane (controller and applications)* that performs various functions by using special purpose applications (like Intrusion Detection System IDS , Monitoring, Loud balancing... etc.) and *data plane (routers and switches)* which is responsible for packet forwarding [4]. Applications can be written in different languages and interact with *controller* through the northbound API, then centralized *controller* instructs the *data plane* in what action it should act via southbound API (usually Open Flow protocol) rather than each node in the network configure its own forwarding decision.

Table 1: Some of SDN controllers

Controller name	Provider	Based	Year
OpenDayLight	Linux foundation	Java	2013
Beacon	Stanford University	Java	2010
Maestro	Rice University	Java	2013
NOX/POX	Nicira	Python/C++	2008/2012

The emergence of SDN changes the closed network system into open one in which many advantages can be offered such as:

a. Extract data from control plane to allow independent development
b. Provide global view of network (Network centralization)
c. Rapid configuration (more flexible than traditional)
d. Reduce the middle boxes
e. Simplify network management
f. Programmable network would support new innovations
g. Multi-tenancy (e.g. for data clouds and data centers)

The distinction between network virtualization and SDN is important. Network virtualization is the separation of logical network from the infrastructure; however the SDN refers to split forwarding hardware from control decisions. Virtualization offers many benefits to the service provider such as sharing resources, deploy services in customise virtual networks and decrease the cost. With SDN concept, the rapid deployment of services in high abstraction level can be facilitated. Hence, service providershave been encouraged to take the advantage of SDN in the network virtualization field, where some efforts have been invested in this context such as Flow Visor and SDN hypervisor [14].

Despite the advantages that offered by SDN, the emergence of new security issues might hamper the spreading of SDN by creating new potential security attacks, disable the synchronisation between controller and network devices such as switches [5]. The next section II

will show the literature survey about SDN security. Section III describes the Usage Control model $UCON_{ABC}$ as an access control method. In section IV shows the SDN Flog language specification and its benefits for SDN. Section V, will present some initial ideas on defining a new model for SDN security and robustness by combining the UCON and Flog. Finally, the summary of this work is provided in section VI.

1. SDN Security survey

Threats of cyber-attacks have become more prominent than ever, for this reason many efforts have been exerted to achieve data security, confidentiality and integrity. The new paradigm of SDN brings both new security mechanism and new threats, which are different from those that infect the traditional network [6]. Even though SDN is not widely deployed and as yet under development, it seems that it has a range of research that specially focused on security enforcement in both academia and industry field. In [7] authors have introduced SPHINX as a controller tool to detect any abnormal or malicious behaviour in real-time by depending on probabilistic and deterministic checking method[7].However, SPHINX does not tackle the whole issues that faced the SDN (e.g. when Open Flow switch deals with legacy switch). Hong et al. (2015) present TopoGuardthat extend the security of SDN controllers by making them able to deter topology poisoning attacks by automatically updating the network topology with the latest information that can be acquired from host tracker and port manager [8].

VeriFlow [9] produce an additional layer between *data plane* and *control plane* in order to check and prevent any faulty rule that issued by SDN applications from applying action in network, low latency has been considered in VeriFlow(debugging tool) to ensure that it will not affect the network performance. VeriCon[10] focused on verifying the controller applications in large-scale network, it has the ability not only to detect errors but also guarantee the absence of errors since the verification process begins at compile time. Some efforts invested in SDN policies, Batista and Fernandez[11] propose Poder Flow language in which knowing programming language (e.g. java, C++, Python…etc.) is not required to design SDN network policy, in addition it supports authorization policies (to define access stage) and obligation policies (to add restriction by network operator or controller).Lara and Ramamurthy [12] present OpenSec framework for automating security policy implementation, OpenSec represents a virtual layer between the controller and user to provide a view that reduces the difficulty during policy creation. In addition, it enables the network operator to write policies in readable form.

According to the latest survey about the benefits of SDN in terms of security found that 28% of organizations would use SDN-enabled network security to deter (block) malicious traffic from endpoints while keep normal ones; about 23%of organizations would use SDN to centralize their network security service policy and configuration management, about 23% of organizations would use SDN to automate network security handling purposes [15].

2. $UCON_{ABC}$ model

Usage Control (UCON) is a new version of access control models in which actions and events might cause changes in the authorization decision during accessing time [16]. In UCON the decision can be made by depends on authentication, obligation and condition also known as $UCON_{ABC}$. Service provision can be controlled by UCON not only before object usage but also during the usage time. UCON constructs predicates from variables attributes and then map from system status into Boolean form in order to make a decision. In $UCON_{ABC}$, Authorization can be specified by predicates of subject and object attributes, Obligation can be specified by predicates of subject actions while Condition can be specified by predicates on system attributes [16].

1. Flog SDN-Policy language

The evolution of the SDN accompanied with effort that aims to decrease the tedious low-level programming in which the programmer have to deal with situations explicitly through installing /uninstalling rules, packet processing in low-level. Flog "is designed as an event-driven, forward chaining logic programming language" that proposed in [13]. Flog adopted the idea of FML and Frenetic languages to introduce a new abstract logic programming language that enables the programmer to design policies in a very convenient manner. The authors in [13] focused on one event only, which is packet arrival at the controller. The idea of stateful firewall that has been mentioned in [13] can be depicted in figure 2.

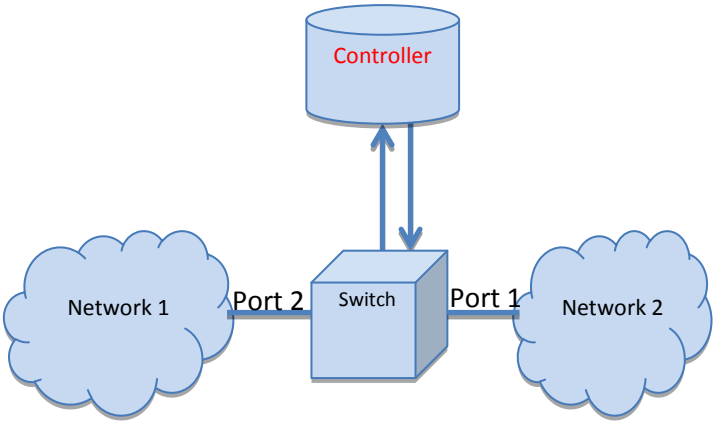

Figure 2: exchanging packets between Network 1 & 2

All packets from Network1 can be transferred to Network2 through port2, but before routing the controller has to store the destination IP (*destip*)of the packet. When any node from Network2 tries to send packet to Network1 (through port1), in such a case the controller will permit the access just for those nodes IP (*srcip*) that have been storedalready. In consequence, packets can send from Network1 to Network2 unconditionally, while the controller permits the packets from Network2 to Network1 if and only if the controller has seen node's IP before.

1. $UCON_{FLOG}$

The quiet open network nowadays leverage the concern of privacy and user authentication, we believe that SDN requires both rigorous access control mechanism and convenient abstracted programming language for two purposes firstly: to mitigate the malicious behavior since the network switches are lacking intelligence and decision-making, secondly: to decrease the complexity of policy creation. The existing techniques and mechanisms that have been used in SDN access control may not adequate to accommodate all risks (just focused on events that occur at run time).However, in our prospect it would be a great to embed the idea of usage control (UCON) into SDN languages(like Flog) to optimize the security of the network as well as to provide the continuity and mutability concepts in SDNpolicies. The following constructs (in terms of F() relation) show the form of $UCON_{Flog,}$ which are defining the concept of usage control to manage the behavior of flow packets that arrive at switch in order to access to the protected network:

F (*Tryaccess*) = Packet (*access*)
F (*Permitaccess*) = Controller (*Access*)
F (*Endaccess*) = Packet (*Endaccess*)
F (*Revokeaccess*) = Controller (*Revokeaccess*)

We will assume that each node in protected network is *Object o*, while any node from outside will be considered as *Subject s*.

A. Early Authorization models

1. Early-authorization without update
   Here there is no updating process for attributes before accessing; the following definition shows the model:

   $permitaccess(s,o) \rightarrow (tryaccess(s,o) \wedge (P_1 \wedge \ldots \wedge P_n)$
   Since $P_1,\ldots,P_n$ are predicates that can be built from subject and/or object.

Now let's put A into a plain text that more understandable, so we can do that by expressing every packet with remembered IP, stored in controller's access control list ACL, can access, otherwise the packet willbedropped. In other word, when any node from protected network (object) deals with node from outer network (subject) then subject's IP will stored in object's access control list as a trusted entity.

$permitaccess(s,o) \rightarrow (tryaccess(s,o) \wedge ((s.ip) \in o.acl))$

2. Early-authorization with pre-update
   Here the authorization decision requires one or more pre-update of attributes as the following definition:
   $permitaccess(s,o) \rightarrow (tryaccess(s,o) \wedge (P_1 \wedge \dots \wedge P_n) \wedge preupdate(attribute))$

3. Early-authorization with ongoing update
   Here the authorization decision requires updates during accessing, which means there is no revoke at usage time. The model can be defined as follow:

   $permitaccess(s,o) \rightarrow (tryaccess(s,o) \wedge (P_1 \wedge \dots \wedge P_n)) \wedge (onupdate(attribute) \wedge endaccess(s,o))$

4. Early-authorization with post update
   Here the controller should update the subject and/or object attributes at the end of access, the model can be defined as follow:

   $permitaccess(s,o) \rightarrow (tryaccess(s,o) \wedge (P_1 \wedge \dots \wedge P_n)) \wedge (postupdate(attribute) \wedge endaccess(s,o))$

   One of the main advantages of this model is that it can be utilized to count how many times that particular *subject* (from outer network) sends request to *object* (from protected network), in consequence it might be used to mitigate (detect) the DOS attack by counting the number of requests during a period of time as the following:

   $permitaccess(s,o) \rightarrow (tryaccess(s,o) \wedge ((s.ip) \in o.acl) \wedge postupdate(s.Time\ request) \wedge endaccess(s,o))$

   $postupdate(s.Time\ request)$: $s.Time\ request' = s.Time\ request + 1$

B. On-authorization models

1. On-authorization without update
   Here the controller has the rights to revoke any access during the run time due to unsatisfied predicate, the following definition shows the model expression:

   $(\neg(P_1 \wedge \dots \wedge P_n) \wedge (state(s,o)=running) \rightarrow revokeness(s,o))$

   Or finish normally with satisfied predicate as follow:

   $((P_1 \wedge \dots \wedge P_n) \wedge (state(s,o)=running) \rightarrow endaccess(s,o))$

2. On-authorization with pre-update
   This is a combination between model A2&B1, which means there is an update process starts before accessing to enforce the authorization decision.

3. On-authorization with on-update
   Here the controllerimposes theupdateactionfor subject and/or object attributes duringthe access operation until the end or revoke access, the model can be defined as follow:

   $(state(s,o)=running) \wedge (P_1 \wedge \dots \wedge P_n) \rightarrow (onupdate(attribute) \wedge (endaccess(s,o) \vee revokeness(s,o))$

4. On-authorization with post-update

Here the controllerpostpone the update action of subject and/or object attributes till the end oftask(either normally or abnormally), the model can be defined as follow:

$(state(s,o)=running) \wedge (P_1 \wedge \ldots \wedge P_n) \rightarrow (postupdate(attribute) \wedge (endaccess(s,o) \vee revokeness(s,o))$

Now we will illustrate the block of $UCON_{Flog}$ by borrowing the idea of SDN Flog language and UCON model:

 # Events Capturing
packet-in, packet-out, flow-mod,online/offline switch, active/inactive ports…etc.
#EventsHandling
After capturing events, SDN programmer should write a proper logic program to process the fact that generated by events.
 # Policy generating
It is the final stage in which the routing and access control policy will be generated for network infrastructure (e.g. switch). As the following syntax:
(Particular packets) |〉 (Re-action), Priority (integer number)
Also, UCON policy model can be written in this part.

According to [13], *flow* keyword has been specified to define rules by depending on some packets' properties such as (srcip, dstip, VLAN, etc.) to capture new network events. After defining properties, programmer can add constrains then pass the captured facts to predefined database (either to use for one time only or make it valid for future use). The following expression will clarify the figure 2 that has been mentioned earlierby applying the scheme of UCONFlog in below:

 # Events Capturing
Flow (s.destip=ip,o.srcip=ip′),inport=2→ condb(ip,ip′)
 # Events Handling
condb(ip,ip′) +→ acl(ip,ip′)
acl(ip,ip′) + → acl(ip,ip′)
 # Policy generating
inport(2) |〉fwd (1), priority(0)
Do
acl(ip) → src(ip), inport (1) |〉fwd (2), priority (0)
When

$(\neg(ip=ip') \wedge ip \in acl \wedge (state(ip,ip')=running) \rightarrow revokeness(ip,ip'))$

Wherecondb = controller database,acl = access control list.

2.Conclusion

To sum up, it can be clearly seen that this work aims to highlight the benefits of the new networking architecture of SDN, especially in terms of centralization and programmable features. We thengave an initial idea of how to utilize the usage control model in order to increase the SDNs reliability and security. The logic programming language Flog has been used as an abstract language example to draw the new model that we call$UCON_{Flog}$. As a first step, two predicates of a new authorization model have been written formally. For future work, weplanto apply the same methodology practically through using the most well-known SDN emulator Mininet as a testbed environment.

References

[1] ValdiviesoCaraguay, Á. L., Benito Peral, A., Barona López, L. I., &GarcíaVillalba, L. J. (2014). SDN: Evolution and Opportunities in the Development IoT Applications. *International Journal of Distributed Sensor Networks, 2014*.

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

[3] Raghu/Castro-leonYeluri (Enrique). (2014). *Building the Infrastructure for Cloud Security*. Springer Verlag.

[4] Monsanto, C., Reich, J., Foster, N., Rexford, J., & Walker, D. (2013, April). Composing Software Defined Networks. In *NSDI* (pp. 1-13).

[5] Hu, F., Hao, Q., &Bao, K. (2014). A survey on software defined networking (SDN) and openflow: From concept to implementation.

[6] Schehlmann, L., Abt, S., &Baier, H. (2014, November). Blessing or curse?Revisiting security aspects of Software-Defined Networking. In *Network and Service Management (CNSM), 2014 10th International Conference on* (pp. 382-387). IEEE.

[7] Dhawan, M., Poddar, R., Mahajan, K., & Mann, V. (2015). SPHINX: Detecting security attacks in software-defined networks. In *Proceedings of the 2015 Network and Distributed System Security (NDSS) Symposium*.

[8] Hong, S., Xu, L., Wang, H., &Gu, G. (2015). Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures.NDSS.

[9] Khurshid, A., Zhou, W., Caesar, M., & Godfrey, P. (2012). Veriflow: verifying network-wide invariants in real time. *ACM SIGCOMM Computer Communication Review*, *42*(4), 467-472.

[10] Ball, T., Bjørner, N., Gember, A., Itzhaky, S., Karbyshev, A., Sagiv, M., ...&Valadarsky, A. (2014, June). Vericon: Towards verifying controller programs in software-defined networks. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation* (p. 31).ACM.

[11] Batista, B., & Fernandez, M. (2014, February). PonderFlow: A Policy Specification Language for Openflow Networks. In *ICN 2014, The Thirteenth International Conference on Networks* (pp. 204-209).

[12] Lara, A., & Ramamurthy, B. (2014, December).OpenSec: A framework for implementing security policies using OpenFlow. In *Global Communications Conference (GLOBECOM), 2014 IEEE* (pp. 781-786).IEEE.

[13] Katta, N. P., Rexford, J., & Walker, D. (2012, September). Logic programming for software-defined networks.In*Workshop on Cross-Model Design and Validation (XLDI)*.

[14] Bozakov, Z., & Papadimitriou, P. (2014, May). Towards a scalable software-defined network virtualization platform. In *Network Operations and Management Symposium (NOMS), 2014 IEEE* (pp. 1-8). IEEE.

[15] SN blogs: The security benefits of an SDN-enabled network. (n.d.). Retrieved April 17, 2015, from http://searchnetworking.techtarget.com/news/4500244535/SN-blogs-The-security benefits-of-an-SDN-enabled-network.

[16] Zhang, X., Parisi-Presicce, F., Sandhu, R., & Park, J. (2005). Formal model and policy specification of usage control. *ACM Transactions on Information and System Security (TISSEC)*, *8*(4), 351-387.