International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:8, 2014

Towards Security in Virtualization of SDN

Wanqing You, Kai Qian, Xi He, Ying Qian

***Abstract*—**In this paper, the potential security issues brought by the virtualization of a Software Defined Networks (SDN) would be analyzed. The virtualization of SDN is achieved by FlowVisor (FV). With FV, a physical network is divided into multiple isolated logical networks while the underlying resources are still shared by different slices (isolated logical networks). However, along with the benefits brought by network virtualization, it also presents some issues regarding security. By examining security issues existing in an OpenFlow network, which uses FlowVisor to slice it into multiple virtual networks, we hope we can get some significant results and also can get furtherdiscussions among the security of SDN virtualization.

***Keywords*—**FlowVisor, Network virtualization, Potential threats, Possible solutions.

I. Introduction

FLOWVISORis a network slicer [1] that acts as a transparent proxy between OpenFlow switches and controllers, which enables multiple tenants to share physical infrastructure. FlowVisor creates rich slices of network resources and delegate control of each slice to a different controller. By applying slicing policy on the entire network, it is useful to make different controllers with different responsibility to manage a big network. FlowVisor sits between controllers and switches, forwarding messages to the right controllers and helping control over slices according to slicing policies [2]. FlowVisor is a transparent layer between controllers and switches. From the controller's point of view, a FlowVisor behaves like a switch; and from the switch's point of view, it is like a controller. FlowVisor ensures that each controller touches only the switches and resources assigned to it. By pointing out this, those threats existing in ordinary SDN networks are still considered as the security issues after virtualization.

The architecture of the SDN after applying FlowVisor is shown as Fig.1. [1] FlowVisor sits between the OpenFlow controllers and switches. It configures the entire network into different slices according to the slice policy of different controller. The resources that can be isolated in FlowVisor are bandwidth, topology, traffic, switch CPU and forwarding table. The aforementioned infrastructure and resources are sliced into abstracted units by FlowVisor. FlowVisor operates as a transparent proxy controller between the physical switches of an OpenFlow network and other OpenFlow controllers and enables multiple controllers to operate the same physical infrastructure. The SDN network with FlowVisor consists of the following four main components:1) Guest controller, 2) Slice policy, 3) Control messages sent to switch, 4) Asynchronous messages to controller. The slice policy provides the method to isolate the network. With the aforementioned features, the advantages of FlowVisor summarized as [3]: Multi-tenancy; Better resource utilization; Simplified management; Rapid/Isolated service development. However, potential threats come along with those benefits cannot be ignored.

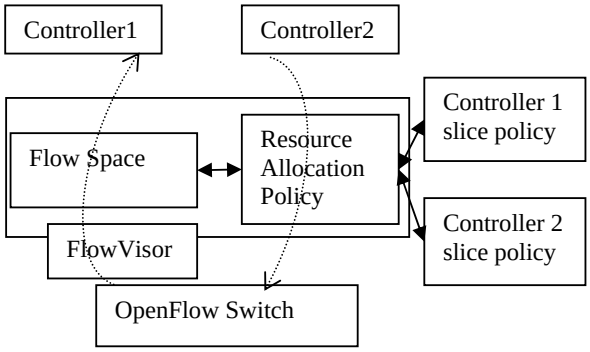

Fig. 1 FlowVisor Architecture [1]

II. Vulnerability Analysis

FlowVisor is a solution proposed to make the network management more effective and efficient. Also the cross-slice threat can be addressed by applying appropriate slicing policy. However, if FlowVisor is made use of by attackers, it will lead to disaster in the entire network.

Some of the potential threats are depicted in Fig. 2. In this section, we will discuss the detail about the possible threats and corresponding solutions.

A. Inference between Controller and Switch

FlowVisor provides various isolation mechanism to slice physical network from different dimensions; however, it does not implement action isolation, which is first analyzed in the prototype FITS [4]. This means that what kinds of actions that can be set on a flow entry are not well defined. Victor [4] proposed three possible threats that are related with the header fields' modification.

Possible solutions are developing a priority mechanism to set controllers with different access privileges; and a well-defined slice policy is needed.

Wanqing Youis with the Southern Polytechnic State University, Marietta, GA 30060 USA (e-mail: wyou@ spsu.edu).

Kai Qian, Dr., is with Southern Polytechnic State University, Marietta, GA 30060 USA. He is now with the Department of Computer Science (e-mail: kqian@spsu.edu).

Xi He is with the Computer ScienceDepartment,Georgia State University, Atlanta, GA USA (e-mail: hexi111@spsu.edu).

Ying Qianis with the Computer Science Department, East China Normal University, China (e-mail: yqian@cs.ecnu.edu).

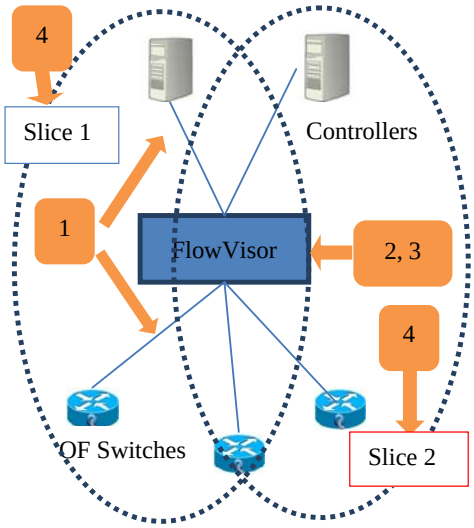

Fig.2 Threats Model

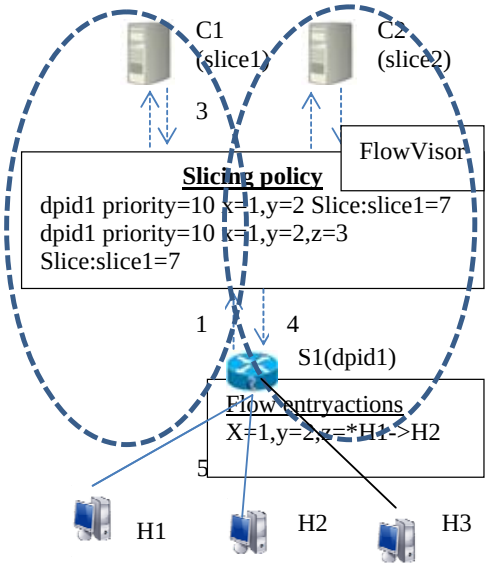

Fig. 3 Overlap of Flow Spaces

B. Denial of Service (DoS)

DoS is a prevalent network security issue, thus it is also a potential threat even in SDN with the use of FlowVisor [5], [6]. By generating DoS in FlowVisor, attackers are able to destroy the slicing policy performed in SDN network. The result may lead to different slices intervened by each other.

The solutions for this issue are as following.Rate limitation should be set both on controllers and switches; use FlowVisor to create a virtual black hole, like Cisco's interface *null0*, which can be used to suck in all malicious traffic.

C. Network-Wide Invariants

A network should be secure to forwarding loops and black holes in order to be steady. These were analyzed by Marco et al. that there is a forwarding loop if a packet goes through any given <switch-id, in_port> more than once. And a network is free from black holes if no packets are dropped in the network [7].

A prototype named VeriFlow was proposed by Ahmed et al. to check network-wide invariants [8].

D. Interference between Flow Spaces

A Flow space is a set of policies that describe the flow entries controlled by a particular controller. Since the flow tables of switches are shared by different controllers, the isolation mechanism may be violated if a Flow space belongs to Controller 1 is intervened by others, such as the rules maintained by Controller 1 can be modified by Controller2. This threat will destroy the isolation mechanism of FlowVisor. A scenario is shown as Fig.3.

III. Interference Testing

A. Attack Simulation

In this section, a flow space overlap in FlowVisor is tested. As the Fig. 3 describes, two flow spaces added by administrator may intersect with each other in the match fields, which is able to make controller 1 intervenes the traffic of controller 2. In this example, the match fields specified by flow space 1 include that in flow space 2 (x=1, y=2 is a subset of x=1, y=2, z=3).

When a host intends to send a packet to another host, if there is not a corresponding rule inside switch's flow table to direct this packet; then a Package_In event will be thrown from switch to controller to request for the commands. When a FlowVisor is included in the network, the request from switch will reach at FlowVisor first, and FlowVisor forwards the request to its controlling controller. After controller makes the decision to deal with the request, a Packet_Out event will be thrown to FlowVisor and then be forwarded to switch.

In the first case, a Package_In message with the header x=1, y=2 comes. This message would be forwarded to controller 1 by FlowVisor to request how to deal with this packet and then a new rule with match fields specify x=1, y=2, z=*, and actions is inserted into switch flow table. The labels from 1 to 5 indicate this procedure. When another Package_In message with the header x=1, y=2, z=3 comes. When this message arrives at switch 1, according to the previously installed rule, this packet will take the actions specified by that rule. It means slice 1 controls the traffic of slice 2. This is one of the interference that is possible to take place if flow space configuration is not implemented appropriately.

B. Threats and Solutions

The cases discussed above illustrated part of the potential vulnerability when FlowVisor is introduced into OpenFlow. FlowVisor is implemented to achieve network virtualization in SDN, so that production network and testing network work perfectly without interference. However, FlowViosr itself provides a tempting target for hackers, because FlowVisor acts both as a controller and a slicer in SDN, if it is down, the whole network is compromised. When referring to network security, CIA, which stands for confidentiality, integrity and availability, should be addressed. The prevention and detection of this kind of issue is vital in order to achieve a secure SDN network.

A possible strategy is to add an additional event handling mechanism in FlowVisor. The main concept is that a new_insert event will be thrown when adding a new flow space.When receiving the event, the event handler will check if there is overlap of flow spaces.

The possible strategies here to check the overlap of flow spaces mentioned above is shown as Fig.4. It means every time when a new flow space is added, the script below will be triggered to go through the existing flow spaces list and each flow space in the list will be compared with the new flow space. If there is an overlap, it is the network administrator's duty to make a decision either rewriting the existing flow space or giving up the new flow space.

```
//function used to check if there is a flow //space
overlap
//Input: array of flow spaces
//Output: if there will be flow spaces //overlap,
give a prompt to administrator //asking for
decision: rewrite or quit.
functioncheckOverlap(Array<FlowSpace> flows,
FlowSpacenewFlow)
START:
foreach flow in flows:
if(newFlow∩ flow ≠Φ ):
    prompt: rewrite or quit
    break;
end if
endforeach
END
```

Fig.4 Function for checking flow space overlapping

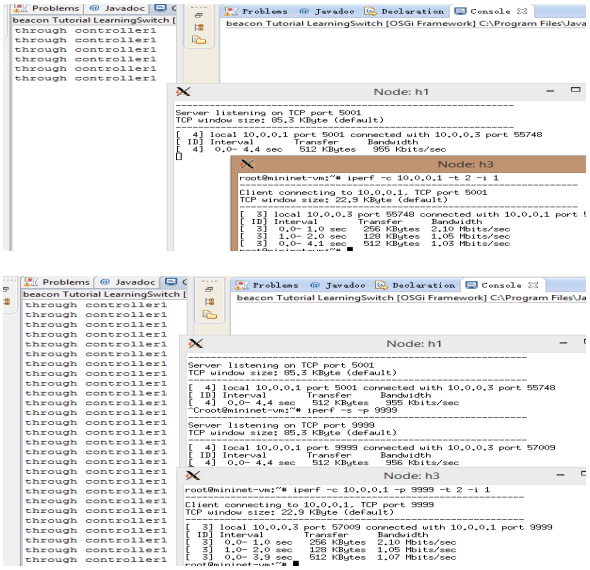

Fig. 5 Controller 1 interferences controller 2

C. Evaluation

In this section, theexperimentation we achieved in previous section is shown. The diamond topology was implemented in our testing, and it was sliced into an upper slice and a lower slice with two of the switches shared in difference slices. After the slicing policy was done by "fvctl" commands, we used "pingall" command to test the reachability between all pairs of hosts, which was shown in Fig. 5. After slicing, only hosts in the same slice are reachable from one to another.

More flow spaces were added to make flow space overlapping. In our case, the newly added flow spaces would be the same as previously added flow spaces while also specifying the communication protocols used when sending packets and assigned to the other controller. The result was that the first controller would take care of the requests that should have been responded by the second controller, which was illustrated below in Fig. 5.

IV. Future Work

We have simulated a port based slicing policy and a possible strategy is proposed. In the future, we will make effort to implement this solution and have the performance been tested. Moreover, we will slice the network according to other slicing policies and try to figure out potential threats existing in a SDN network with FlowVisor, come up with solutions corresponding.

In the future, we plan to implement a network slicing experimentation by FlowVisor to validate the feasibility of aforementioned threats as well as to explore more possible vulnerabilities in OpenFlow with FlowVisor in depth.

V. Conclusion

In this paper, we focus on the security issues on SDN virtualization. We have explored and analyzed potential attacks possibilities and possible defense strategy.Our work was based on a switch ports slicing. If ports belong to different slices and the flow spaces configuration is not proper, it is very likely that a cross-slices threat will take place. And we figured out an event handling mechanism in FlowVisor to avoid flow spaces overlapping. As mentioned in the last section, in the future, we will dedicate to employing the strategy we proposed in FlowVisor. Moreover, we will try to implement other slicing policy based on VLAN ID and other slicing mechanism and try to explore the potential vulnerability and try to make SND network more secure

References

[1] Rob Sherwood, Glen Gibb, Kok-kiongYap, Guido Appenzeller, Martin Casado, Nick Mckeown, and Guru Parulkar. FlowVisor: A Network Virtualization Layer.OpenFlow Switch, page 15, 2009
[2] Rowan Kloeti, OpenFlow: A Security Analysis, 2012 ftp://yosemite.ee.ethz.ch/pub/students/2012-HS/MA-2012-20_signed.pdf
[3] incntre.iu.edu/sites/default/files/FlowVisor%20Intro.pptx, accessed 2014
[4] Victor T. Costa, Luıs Henrique M. K. Costa, Vulnerability Study of FlowVisor-based Virtualized Network Environments http://www.gta.ufrj.br/wnetvirt13/papers/ts5-02.pdf
[5] Romão, Daniel, et al. "Practical security analysis of OpenFlow implementation." ,2013
[6] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM,pp. 55–60, 2013
[7] Canini, Marco, et al. "A NICE way to test OpenFlow applications." NSDI,04/2012

[8] Khurshid, Ahmed, et al. "Veriflow: Verifying network-wide invariants in real time." ACM SIGCOMM Computer Communication Review 42.4: 467-472, 2012

Wanqing You is a graduate student in the department of computer science & software engineering at Southern Polytechnic State University. She got her bachelor degree in software engineering at Xiamen University, China, 2013. She has industrial experience in the related field and has published papers in her research areas.

Dr. Kai Qian is a computer science professor in the department of computer science & software engineering at Southern Polytechnic State University. He got his Ph.D in computer science and engineering at University of Nebraska-Lincoln, 1990. His research areas include computer network and mobile security, big data analysis for security, machine learning, and pattern recognition. He has published about 100 research papers in these areas in many journals and conferences. He has received a number of research projects on the cybersecurity from NSF these years.

Xi He is a CS research assistant and instructor at Georgia State University. He is specialized in parallel and distributed computing, network architecture, and grid computing. He has much year industrial experience as a software engineer and has published papers in his research areas.

Dr. Ying Qian is an Associate Professor in the Department of Computer Science and Technology, at East China Normal University, Shanghai, China. She received her Master and Ph.D. degree in Department of Electrical & Computer Engineering from Queen's University, Kingston, Ontario, Canada. Her research interests include Software Defined Network, high-performance scientific computation, and parallel programming. Sh e has published about 20 research papers in these areas in many journals and conferences.