Costa and Costa *Journal of Internet Services and Applications* (2015) 6:18
DOI 10.1186/s13174-015-0034-4

Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

Vulnerabilities and solutions for isolation in FlowVisor-based virtual network environments

Victor T. Costa* and Luís Henrique M. K. Costa*

Abstract

In a virtualized environment, different virtual networks can operate over the same physical infrastructure. Each virtual network has its own protocols and share the available resources, thus highlighting the need of resource isolation mechanisms.

Investigating the isolation mechanisms provided by FlowVisor, we have discovered vulnerabilities previously unknown regarding addressing space isolation. We show that, in the presence of a malicious controller, FlowVisor's isolation can be broken allowing different attacks. This paper addresses these vulnerabilities by proposing an Action Slicing mechanism, that allows FlowVisor to limit which actions can be used by each virtual network controller, thus extending the virtual network definition. Our experimental results show that using the proposed Action Slicing mechanism can effectively neutralize the discovered vulnerabilities.

Keywords: OpenFlow; FlowVisor; Security; Network virtualization; Resource isolation

1 Introduction

With the objective of creating innovation opportunities in campus environments, the OpenFlow [1] platform was proposed, which allows the use of the physical infrastructure by both production and experimental networks, simultaneously. The FlowVisor [2] tool was also proposed, allowing the virtualization of an OpenFlow physical infrastructure into different virtual networks. Currently, FlowVisor provides isolation mechanisms for topology and addressing space, but lacks such mechanisms for bandwidth, device CPU, and forwarding tables. Resource isolation stands as one of the major challenges in SDN (Software Defined Networking) and OpenFlow virtualized network environments [3].

The FITS (Future Internet Testbed with Security) [4] testbed, discussed in detail in Section 4.1 and whose development is led by GTA/UFRJ, provides a virtualized OpenFlow network environment based on FlowVisor. In order to meet possible future demands of users, we have performed various tests to verify if the defined addressing space of each virtual network was being isolated correctly. From these experiments, we have discovered vulnerabilities in the FlowVisor's isolation mechanism, to the best of our knowledge, to this date unknown. Thus, the isolation can be broken by a malicious virtual network, allowing the manipulation of the traffic of other virtual networks. We show that this manipulation is mostly due to the fact that the FlowVisor is unable to control which actions should be allowed for each controller.

In this paper, we discuss the discovered vulnerabilities and the impacts of their exploitation in FlowVisor-based virtualized network environments. These vulnerabilities, summarized in three main cases regarding the addressing space isolation, allow a malicious controller to control or inject packets in other virtual networks. From this investigation, we propose modifications to FlowVisor that address these vulnerabilities and introduce the Action Slicing mechanism. Action Slicing allows FlowVisor to limit the actions that can be used by each virtual network controller, thus extending the virtual network definition. We have implemented the action slicing mechanism in the FlowVisor used in the FITS testbed, and our experiments confirm that the proposal can neutralize the attacks effectively.

This paper is organized as follows. Section 2 discusses the discovered vulnerabilities and related problems. In Section 3 we describe our proposed Action Slicing mechanism. In Section 4 we introduce the FITS testbed, describe

*Correspondence: torres@gta.ufrj.br; luish@gta.ufrj.br
Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ, Rio de Janeiro, Brazil

our experiments and show that the proposed Action Slicing mechanism neutralizes the discussed vulnerabilities. Section 5 discusses the trade-offs of our proposal. In Section 6 we describe the related work. Finally, Section 7 concludes the paper and presents future directions for this work.

2 Vulnerabilities and isolation problems

The addressing space of a virtual network represents what kind of traffic belongs to that network. In order to provide more flexibility, this definition can be more abstract, such as all HTTP traffic, or more specific, such as all traffic coming from a given IP address. In FlowVisor, the physical network's addressing space is sliced by using a structure called flowspace [2], that indicates which values of each header fields belong to a given network. To define the addressing space of a virtual network, it is necessary to specify which OpenFlow switches and which ports of each switch belong to that virtual network, in addition to the characteristics of the packets belonging to that network. In FlowVisor, the allowed header fields are defined according to the fields defined in the OpenFlow specification: allowed VLAN ID tags, source MAC addresses, destination MAC addresses, source IP addresses, destination IP addresses, IP protocol type (such as TCP or ARP), source Transport ports, and destination Transport ports.

2.1 Attack model

In a FlowVisor-based virtualized network environment, each slice is represented by its controller. Virtual networks are managed by different entities and all share the same physical resources. Thus, it is not possible to assume that all virtual networks are reliable and well-behaved. A virtual network may affect the traffic of other networks, intentionally or not, due to malicious actions of its controller or faulty behavior of the isolation mechanisms. We classify as malicious behavior when a virtual network controller intentionally acts in order to disturb other virtual networks, such as intentional creation or modification of flow rules to steal, deviate or inject packets. For faulty behavior, we denote controller actions that, due to a problem of the isolation mechanisms, end up disturbing other virtual networks, such as QoS mechanisms. Both behaviors are harmful and, for simplicity, we define all the controllers that perform such behaviors as malicious.

2.2 Vulnerabilities and attack cases

Although it is expected that FlowVisor provides isolation for the address space of different virtual networks, a different behavior was observed in our experiments. The key is that FlowVisor does not implement any kind of control over what types of actions each virtual network controller may define in its flows, allowing malicious controllers to explore breaches in the isolation mechanisms. Problems with the addressing space isolation mechanism were first discovered in the FITS testbed (Section 4.1), in which different virtual networks are defined by different VLAN ID tags. Given the results of tests, we have organized the discovered problems into three general cases, the VLAN ID Access Problem, the Field Rewrite Problem, and the Wildcard Rewrite Problem.

The VLAN ID Access Problem is specific to VLAN ID tags. In this case, we assume a scenario where FlowVisor is configured so that a given virtual network controller has no access to tagged packets, that is, it is only able to act upon packets without VLAN ID tags. The key point in the VLAN ID Access Problem is that the FlowVisor actually does not isolate the VLAN ID field, allowing the controller to match packets with any VLAN ID. Besides being able to create flows that control all packets, a malicious controller would also be able to define any types of actions to be applied to those packets, for example packet-drop actions or actions to modify a packet header field, such as the VLAN ID tag itself. As a consequence, this problem would allow a malicious controller to create black holes, for example, using packet-drop actions. In environments that use exclusively VLAN ID tags to define virtual networks, such as the FITS testbed, this problem would allow a malicious controller to steal, inject or deviate packets of other virtual networks, by means of header field modification actions. For example, a controller could match packets from other virtual networks (other VLAN IDs) and rewrite the VLAN ID tag of those packets, by VLAN ID rewrite actions, to make them packets of its own virtual network and thus steal them.

For the Field Rewrite Problem, we assume a scenario where the FlowVisor is configured so that a given network controller has only access to packets with a specific header field value, for example only packets with source IP address 192.168.1.1. If this controller tries to create a flow to control packets with any source IP address, using a wildcard value, FlowVisor rewrites the controller's request so that the flow's source IP address is the allowed value 192.168.1.1. The problem is that any types of actions are still allowed to the controller, such as header field modification actions. As a consequence, a malicious controller could modify the header of its own packets, disguising them as packets belonging to another slice. This would allow a malicious controller to inject packets in other virtual network's traffic. Although we have used source IP address in our example, the problem also exists for other fields: destination IP address, source MAC address, destination MAC address, source Transport port, destination Transport port or any combination of the these fields.

The Wildcard Rewrite Problem is similar to the previous one. Assume a scenario in which FlowVisor is configured so that a given network controller has only access to packets with a specific header field value, for example, only packets with source Transport port 80. The problem in this case is that, if the controller tries to create a flow rule to match packets with any source Transport port, using a wildcard value, FlowVisor does not rewrite the wildcard value to the configured value of 80. A malicious controller can thus create flows that control packets with any value of source Transport port. A malicious controller would also be able to define any actions to be applied to the packets, causing the same consequences mentioned in the first problem, allowing theft, injection, or deviation of packets of other virtual networks. This problem was first observed for the source Transport port field, but is the same for a few other fields, such as destination Transport port and IP protocol type. This problem is related to the FlowVisor implementation, and shows us the consequences that such mistakes bring to virtualized environments.

Table 1 summarizes the discovered vulnerability cases, their brief descriptions, and possible attacks.

2.3 Other related problems

In addition to the discussed vulnerabilities, there are other problems involving FlowVisor-based virtualized network environments that are also worth studying. Due to its project design, FlowVisor is currently a single mediation point between all OpenFlow switches and controllers, making it a single point of failure. FlowVisor is implemented in JAVA as server, receiving and responding messages from/to all OpenFlow network elements. This would make possible for a malicious controller to organize a denial-of-service attack, making FlowVisor unable to respond to legitimate requests and thus disrupting network operation. A similar analysis can also be considered for non-virtualized OpenFlow environments [5].

As discussed in the vulnerability analysis, the key problem is due to the fact that currently it is not possible to specify which types of actions a virtual network controller may use or not in its flows, through FlowVisor. Besides the mainly discussed actions (forwarding, rewriting and discarding), the improper use of other actions can also allow a malicious controller to disrupt the operation of other virtual networks. Among these actions, we can highlight VLAN ID tag removal actions, VLAN PCP (Priority Code Point) rewrite actions and IP ToS (Type of Service) actions. These actions could, for example, mischaracterize traffic of virtual networks that use VLAN ID tags (such as in the FITS testbed) or disrupt a QoS (Quality of Service) system based in VLAN PCP [6] or IP ToS values.

3 The action slicing proposal

To address the aforementioned vulnerabilities, we have performed modifications to the FlowVisor and introduce an Action Slicing mechanism. The objective of the proposed mechanism is to extend FlowVisor's current virtual network definition, allowing network administrators to define which actions are allowed for each controller to use in their flows. Figure 1 shows the internal operation of the unmodified FlowVisor, when a command is sent by the controller to an OpenFlow switch. The controller commands are first received by a Slicer element (1), which is responsible for managing commands and messages from/to the OpenFlow controller. There is one Slicer for each virtual network controller. The Slicer then verifies if the received command complies with the virtual network definition (2), by means of its flowspace rules, and modifies the command when necessary. The resulting command is then sent to the switch (3) by means of the respective Classifier element, responsible for managing commands and messages to/from the OpenFlow switch. There is one Classifier for each OpenFlow switch.

By investigating FlowVisor's source code, we have concluded that the problems are focused on two points: the internal verification mechanisms of the Slicer element and the lack of control over which actions are allowed to a virtual network controller. The first refers to implementation problems in the JAVA source-code, causing some of the issues described in Section 2.2. The second is a limitation of the current isolation mechanism, that can be used maliciously, as discussed in Section 2.3. In order to solve this problem, we have developed an extension to the FlowVisor, the Action Slicing mechanism. This Action Slicing mechanism addresses this isolation issue and implements a control mechanism to limit which types of actions can be used by each virtual network controller.

Table 1 Summary of the discovered vulnerability cases

Vulnerability cases			
Problem	Hypothesis	Malicious action	Consequence
VLAN ID access	Controller *x* with no access to VLAN tagged packets	Controller *x* able to control packets with any VLAN ID tag	Theft, injection, or deviation of packets
Field rewrite	Controller *x* with access only to specific values of the *y* header field	Controller *x* can change the *y* header field of its packets to any other value	Packet injection
Wildcard rewrite	Controller *x* with access only to specific values of the *y* header field	Controller *x* able to control packets with any value of *y*	Theft, injection, or deviation of packets

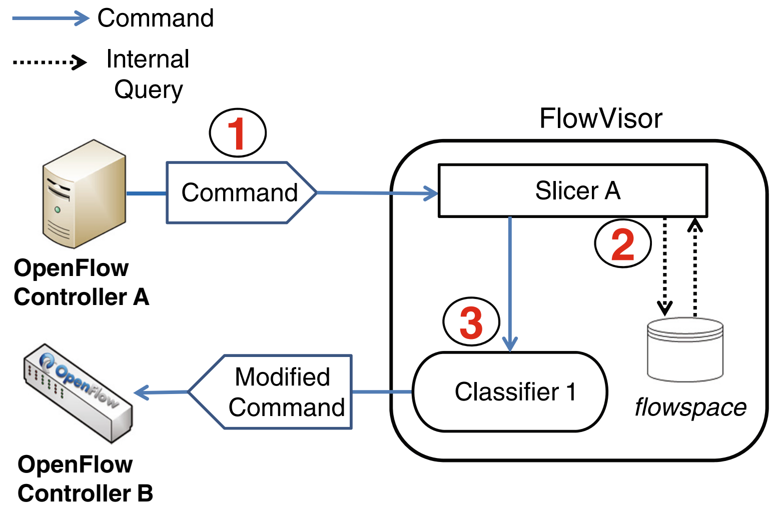

Fig. 1 Standard FlowVisor operation

3.1 Design and implementation

In this work we have implemented the Action Slicing mechanism as a JAVA module, independent of FlowVisor, but integrated to its request verification and validation mechanisms. As a consequence, the network administrator can define, based on the actions available in the OpenFlow protocol, which actions a given virtual network control may use in its flows. In case of requests containing invalid actions, the mechanism can be configured to modify those requests and keep only allowed actions in each request, or to promptly return an error message to the originating controller.

The operation of Action Slicing is illustrated in Fig. 2. Similarly to the original FlowVisor operation, controller commands are initially received by a Slicer element (1). A Slicer element is the FlowVisor module that interfaces with controllers, and there is one for each controller in the network. This Slicer element then verifies if the command complies with the virtual network definition (2), by matching with FlowVisor's flowspace, and the command is rewritten if necessary. Until this point, the message has followed the standard FlowVisor processing. The Slicer element then activates our Action Slicing mechanism, and the Action Filter verifies if the actions defined by the controller are permitted, according to the Permission Data Base (3). In case a forbidden action is included in the message, the Action Filter can then rewrite the command so that the action is removed, or send back an error message to the controller. The resulting command is then forwarded to the respective Classifier element (4), which sends the command to the destined OpenFlow switch.

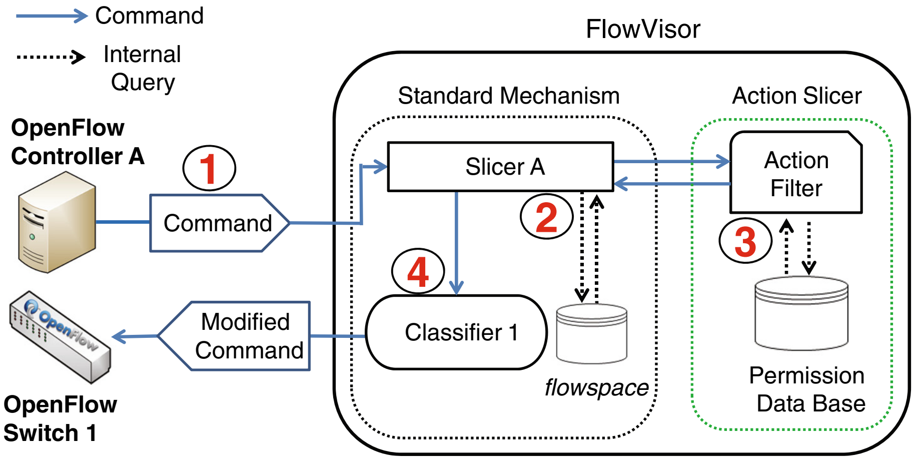

Fig. 2 FlowVisor operation with action slicing

A Classifier element is the FlowVisor module that interfaces with switches, and there is one for each switch in the network.

The Action Slicing mechanism is integrated to FlowVisor by means of JAVA function calls, and the Slicer elements communicate directly with the Action Filter. The Permission Data Base (example in Table 2) is divided into rules, which indicate if a given action is allowed or not over a given virtual network controller. The developed mechanism is currently in use in the FITS testbed.

The key reason behind the effectiveness of the proposed mechanism is the fact that each action in the OpenFlow protocol is atomic and other actions can not be combined to realize the effect of one action. Therefore, by blocking a specific action that might be used maliciously, a malicious controller has no way within the OpenFlow protocol to execute that same attack.

4 Evaluation

In this section we present the testbed used as motivation for this study and the experimental results of our evaluation.

4.1 The FITS network Testbed

The test platform used in this paper for the experimentation of the FlowVisor's isolation mechanisms is FITS (*Future Internet Testbed with Security*) [7]. The main goal of FITS is to provide a general-purpose, open, and shared network experimentation infrastructure. FITS assumes a pluralist approach, in which virtual networks are located in one or spread across various physical nodes. The FITS testbed allows the user to choose between a conventional packet forwarding approach, through the user's virtual machines (VMs), or a plane separation approach [8], in which packet forwarding is performed by the physical machine, according to control information provided by the VMs. Plane separation increases the forwarding capacity of the virtual network and also allows the VMs to spend resources executing other processes, instead of processing packet forwarding.

Forwarding in the FITS testbed is performed by the Open vSwitch [9] software switch, deployed in all FITS nodes. The software switch performs packet forwarding between VMs and through the physical network. Open vSwitch implements a Flow Table, allowing network management through the use of flows. One of the possible interfaces for interaction with Open vSwitch is the OpenFlow protocol. Therefore, the integration of the VMs with the OpenFlow network is possible through the use of Open vSwitch. Other OpenFlow-enabled switches can be used, although there are none in the current stage.

In the FITS testbed, FlowVisor is used to slice the OpenFlow network (represented by the Open vSwitch switches) into different virtual networks. Each slice is defined by a specific VLAN ID tag (Virtual Local Area Network Identifier, IEEE 802.1Q standard). In other words, packets with a given VLAN ID tag belong to a specific slice, and packets with other VLAN ID tags belong to other slices or no slice at all. In this paper, we identify problems and vulnerabilities in the FlowVisor's isolation mechanisms, discovered through our experience using FlowVisor in the FITS testbed. The vulnerabilities are directly connected to the addressing space isolation mechanism, which does not isolate the VLAN ID field. In addition, the lack of control of which actions are allowed by each controller enables packet manipulation and injection attacks. Our proposal addresses and neutralizes these vulnerabilities.

4.2 Experiments and results

In order to validate the proposed mechanism, working alongside FlowVisor, we have developed two experiments to prove that our mechanism neutralizes the previously discussed vulnerabilities. For the test environment, we have used the Open vSwitch 1.10 software switch (OpenFlow protocol 1.0 compatible), FlowVisor 1.4 and the POX 0.1.0 OpenFlow controller. As the traffic receiver and sender, we have used notebooks Sony VGN-Z870A. As OpenFlow controllers, we have used two notebooks Sony VGN-Z870A, responsible for sending commands to the OpenFlow switch (Open vSwitch). The test environment is shown in Fig. 3. Both controllers are connected to the FlowVisor, and the FlowVisor is connected to the OpenFlow switch, both PCs with a 3.2 GHz Intel Core i7 processor and 8 GB of RAM memory. The operational system used was Debian Wheezy. For the traffic generation, we have use the IPERF [10] tool, and the identification of traffic of each controller was made by using VLAN ID tags. The tests were repeated 100 times, with a confidence interval of 95 %.

Table 2 Permission data base example

Virtual network	OpenFlow Action 1 (send to controller)	OF Action 2 (discard)	...	OF Action N
Controller A	Allowed	Denied	...	Denied
Controller B	Denied	Denied	...	Allowed
Controller C	Allowed	Allowed	...	Allowed

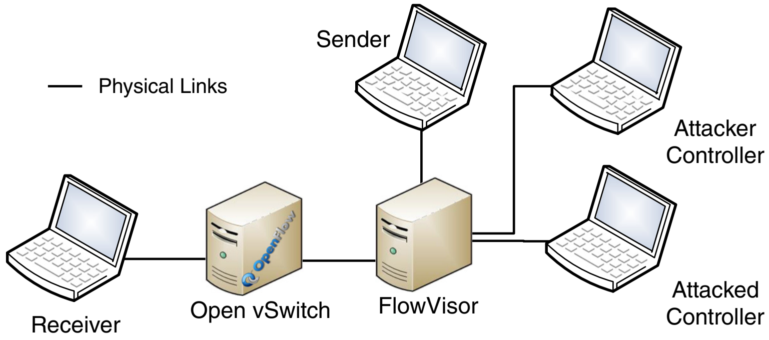

Fig. 3 Test environment for the proposed experiments

In the first experiment, that we call Traffic Discard Attack, the Attacker Controller tries to modify a flow rule of the Attacked Controller, so that the Attacked Controller's traffic is discarded. In this scenario, the Attacked Controller sends a constant flow of 10 Mb/s (UDP) data for the receiver. The Attacker controller then sends a command to the OpenFlow switch, so that all traffic from the Attacked Controller is discarded. The results of the experiment can be seen in the Fig. 4, where the discard command from the Attacker Controller is sent in the 50 seconds mark. We can see that, without the use of our mechanism, the packets from the Attacked Controller begin to be discarded as soon as the discard command is sent. On the other hand, the use of our proposed mechanism resulted in the rejection of the discard command, since the Attacker Controller has no permission over that flow, and the data flow goes unaffected.

In the second experiment, that we call Traffic Injection Attack, the Attacker Controller tries to create a flow rule that rewrites the VLAN ID tag of its own packets, so that its traffic is perceived as being of the Attacked Controller. In this scenario, the Attacked Controller sends a constant data flow of 3 Mb/s (UDP) to the receiver. The Attacking Controller sends a constant data flow of 7 Mb/s (UDP) to the receiver. The Attacking Controller then sends a create flow command to the OpenFlow switch, so that the new flow rewrites the VLAN ID tag of the Attacking Controller's packets to the Attacked Controller's VLAN ID tag value. The results of the experiment are shown in Figs. 5 and 6. For the case in which our proposed mechanism is not used (Fig. 5), we can see that the Attacker Controller is able to create the flow rule and rewrite its own VLAN ID tag, thus making its own traffic look like the Attacked Controller's. The throughput of the Attacked Controller then appears to be around 10 Mb/s. On the other hand, by using our proposed Action Slicing mechanism (Fig. 6), the Attacker Controller's command is not accepted, since it affects traffic from other controller, and the traffic of both controllers go unaffected.

Regarding overhead introduced by our proposal, in all our experiments the processing overhead experienced was

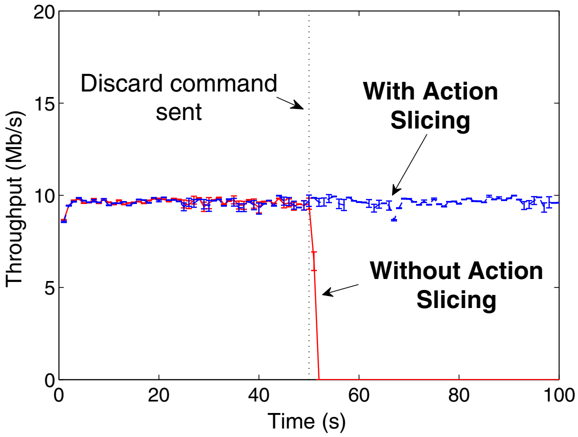

Fig. 4 Traffic discard attack test

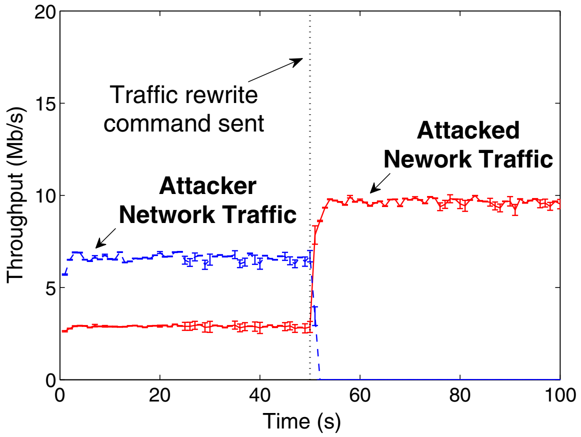

Fig. 5 Traffic injection attack test without action slicing

under a tenth of microsecond, negligible when compared with the main processing cost of matching the packet with the flowspace. This could be achieved due to the closely connected implementation and high-performance data structures. The memory overhead is also negligible since the number of possible actions is fixed under a few dozens, making the permission database lightweight. There is no communication overhead introduced by our proposal.

5 Trade-offs of action slicing

The Action Slicing mechanism neutralizes the vulnerabilities discussed in this paper by controlling which actions a virtual network controller is able to use in its flow rules. However, it is necessary to clarify the potential impact of such control policies on the behavior of virtual networks and its users.

By limiting the set of actions available to a virtual network controller, we also limit the behavior of the network and its flexibility to deal and process packets. It is necessary to evaluate the needs of each virtual network and how to respond to security incidents appropriately. Limiting the use of a field rewrite action can prevent attacks by a malicious controller, but may also prevent a normal controller from manipulating its own packets freely. Regarding this we can take the FITS testbed as an example. FITS uses the VLAN ID field to define a virtual network, so virtual networks should never be able to remove or change this field and thus such actions are not allowed. In this sense, the Action Slicing mechanism can be used to restrict the use of actions that would only bring negative effects to other virtual networks, and which are not necessary during normal operation.

6 Related work

Resource isolation among virtual networks and attack prevention are challenges common to various network

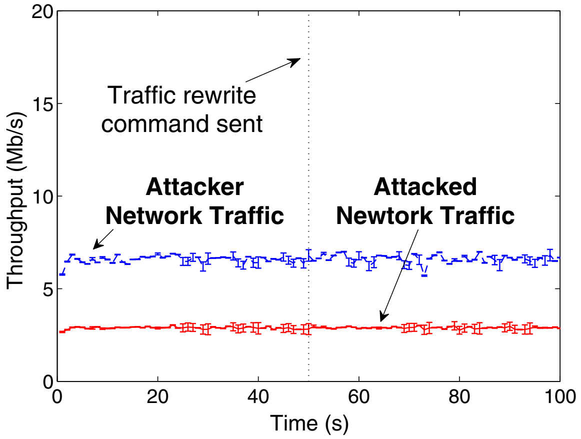

Fig. 6 Traffic injection attack test with action slicing

virtualization platforms. In OpenFlow networks, threats may rise from users, controllers, or even the OpenFlow protocol itself.

Benton *et al.* [5] make a vulnerability study of the OpenFlow protocol. The main problem pointed out by the authors is the lack of mandatory security mechanisms in the protocol, such as TLS (Transport Layer Security). As a consequence, different attacks and undesirable actions are possible. For instance, given its homogeneous interface to all devices through the OpenFlow protocol, the lack of appropriate security mechanisms facilitates man-in-the-middle attacks, allowing a malicious user to steal information or take control of network elements. In the present paper, we have extended the security analysis to FlowVisor-based OpenFlow virtualized environments.

In virtualizing OpenFlow switches, an alternative approach to FlowVisor is FSFW (FlowSpace Firewall) [11]. Instead of attempting to interpret and modify a rule, like FlowVisor, FSFW either allows a rule to pass or rejects it, sending an error to the controller. The main difference between FSFW and FlowVisor is in the scope. FSFW implements its virtualization by using different VLAN tags for different controllers in different interfaces of the switch, while FlowVisor have a broader virtual network definition. Since FSFW does not include which actions are allowed for each controller in the network, some vulnerabilities described in this paper are still possible, especially the Field Rewrite Problem. The proposed Action Slicing Mechanism could be easily modified to support FSFW.

Another approach, proposed as a substitute for FlowVisor, is OpenVirteX [12]. Instead of dividing the flowspace in slices, like FlowVisor, OpenVirteX provides each virtual network with its own flowspace, making use of internal address translation mechanisms. It is important to understand the differences between both tools. By using internal address translation mechanisms, OpenVirteX eliminates the need of dividing the addressing space (flowspace) among the virtual networks, however also imposes limitations on the addressing space itself. Due to its design, OpenVirteX does not allow flow actions that change the MAC addresses of a packet, nor allow flow rules to be created with wildcarded MAC address fields. In addition, the number of IP addresses available to be used for a network is also reduced. Such limitations are not present in FlowVisor, and it is easy to see how each tool could be more suitable for specific environments, as the authors themselves suggest [13]. In the case of the FITS testbed, it is important for the virtual networks to be able to manipulate the MAC and IP address fields of their packets freely, thus the choice to use FlowVisor. It is also important to note that, since the vulnerabilities discussed in this paper are inherent to address space sharing, they do not apply to OpenVirteX.

In this paper we have focused the addressing space resource isolation mechanism, but other primitive network resources still need their own isolation mechanisms. Mattos and Duarte [14] propose QFlow. QFlow implements a system for bandwidth isolation for OpenFlow networks, using separate queues in the Open vSwitch software switch. In QFlow, it is possible to specify a guaranteed minimum and an allowed maximum value for the bandwidth of each queue, and then forward the desired traffic through a given queue. QFlow also introduces memory and device CPU control mechanisms for the Xen platform [15], but does not define such mechanisms for OpenFlow. Min *et al.* [6] propose changes to FlowVisor, integrating a similar bandwidth control mechanism for virtual networks and an admission control mechanism for new virtual networks, based on bandwidth requirements.

In an OpenFlow controller, many user applications or modules are executed in parallel, and a malicious application can disrupt network operation. Porras *et al.* [16] proposed FortNOX, an extension to the NOX controller that controls which commands are allowed for each application, based on a role-based access control (RBAC) mechanism. To allow users to create their own security modules and policies, Shin *et al.* [17] proposed FRESCO, an extension of FortNOX. FRESCO is a framework for development and execution of user-defined security modules. The main difference between FortNOX/FRESCO and FlowVisor is that the former acts inside a single controller and FlowVisor concerns isolation among multiple controllers. Inside a controller, many applications may want to access the same resource, thus justifying the use of the RBAC mechanism. FlowVisor provides network virtualization by providing every controller a slice, and no controller should be able to access another controller's slice. Thus, by extending FlowVisor's flowspace, the proposed Action Slicing mechanism can be deployed easily and with minimum overhead.

7 Conclusion and future work

In this work, we have evaluated the addressing space isolation mechanisms of FlowVisor. From the vulnerability analysis of the FlowVisor's addressing space isolation, we have discovered serious vulnerabilities and proposed a solution, the Action Slicing mechanism.

From our experience with the FITS testbed, we have discovered vulnerabilities in FlowVisor's addressing space isolation mechanism. Such vulnerabilities allow a malicious controller to disturb and manipulate traffic of other virtual networks. In addition, FlowVisor does not provide control over which types of actions a virtual network control may define in its flows. This is a problem, since actions that modify packet header fields may interfere with traffic from other virtual networks. It is important to note that such problems affect not only the FITS testbed, but any system that uses FlowVisor, such as the OFELIA [18] testbed. To address these problems, we have proposed the modification of FlowVisor and the creation of an Action Slicing mechanism, allowing FlowVisor to control which types of actions are allowed to each virtual network controller. The developed mechanism was implemented and incorporated to the current version of the FITS testbed, and the evaluation results show that the attacks are no longer possible.

As future work, we plan to integrate our proposal to the official FlowVisor repository. We also plan to contribute to OFELIA by demonstrating the discovered vulnerabilities in the OFELIA testbed. We are also interested in extending the vulnerability analysis to the remaining isolation mechanisms for OpenFlow networks, such as the FlowVisor's topology isolation mechanism and the queue-based bandwidth isolation mechanism. In the study of isolation mechanisms, there is a interest in isolation mechanisms for resources that still do not have an established control mechanism for OpenFlow networks, such as forwarding tables and device CPU.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

In this research work, LMKC provided the test environment, access to the FITS testbed and the initial course of action to be taken in the research process. VTC investigated the validity of the hypothesis, carried out the implementation, testing and analysis of the data. Both authors participated in the evaluation and organization of the results. Both authors read and approved the final manuscript.

Acknowledgements

This work was partially funded by CAPES, CNPq SECFUNET Project (proc. 590047/2011-6), FAPERJ, and FINEP.

Received: 13 October 2014 Accepted: 29 July 2015
Published online: 18 August 2015

References

1. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J (2008) OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 38(2):69–74
2. Sherwood R, Gibb G, Yap KK, Appenzeller G, Casado M, McKeown N, Parulkar G (2009) Flowvisor: A network virtualization layer. OpenFlow Switch Consortium, Tech. Rep. http://sb.tmit.bme.hu/mediawiki/images/c/c0/FlowVisor.pdf
3. Sezer S, Scott-Hayward S, Chouhan PK, Fraser B, Lake D, Finnegan J, Viljoen N, Miller M, Rao N (2013) Are we ready for sdn? implementation challenges for software-defined networks. Commun Mag IEEE 51(7):36–43
4. GTA/UFRJ (2011) Future Internet Testbed with Security. http://www.gta.ufrj.br/fits. [Online; Accessed in December-2013]
5. Benton K, Camp LJ, Small C (2013) OpenFlow vulnerability assessment. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, Hong Kong, China. pp 151–152
6. Min S, Kim S, Lee J, Kim B, Hong W, Kong J (2012) Implementation of an OpenFlow network virtualization for multi-controller environment. In: Advanced Communication Technology (ICACT), 2012 14th International Conference On. IEEE, PyeongChang, Korea (South). pp 589–592
7. Moraes IM, Mattos DMF, Ferraz LHG, Campista MEM, Rubinstein MG, Costa LHMK, MD deAmorim, Velloso PB, Duarte OCMB, Pujolle G (2014) Fits: A flexible virtual network testbed architecture. Comput Netw 63:221–237
8. Pisa PS, Couto RS, Carvalho HE, Neto DJ, Fernandes NC, Campista MEM, Costa LHM, Duarte OCM, Pujolle G (2011) VNEXT: Virtual network management for xen-based testbeds. In: International Conference on the Network of the Future (NOF). IEEE, Amsterdam, The Netherlands. pp 41–45
9. Pfaff B, Pettit J, Amidon K, Casado M, Koponen T, Shenker S (2009) Extending networking into the virtualization layer. In: Eighth ACM Workshop on Hot Topics in Networks (Hotnets). HotNets-VIII, New York City, NY, USA
10. IPERF- Traffic Measurement Tool. http://iperf.fr/. [Online; Acessado em 02-Fevereiro-2014]
11. GlobalNOC (2013) FSFW: FlowSpace Firewall. http://globalnoc.iu.edu/sdn/fsfw.html/. [Online; Accessed in February-2015]
12. Al-Shabibi A, De Leenheer M, Gerola M, Koshibe A, Parulkar G, Salvadori E, Snow B (2014) Openvirtex: make your virtual sdns programmable. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, Chicago, Illinois, USA. pp 25–30
13. ONLab (2013) OVS FAQ. http://ovx.onlab.us/documentation/faq/. [Online; Accessed in June-2015]
14. Mattos DMF, Duarte OCMB (2012) QFlow: Um Sistema com Garantia de Isolamento e Oferta de Qualidade de Serviço para Redes Virtualizadas. In: XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC' 2012, Ouro Preto, MG, Brazil, May 2012. pp 536–549
15. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A (2003) Xen and the art of virtualization. ACM SIGOPS Oper Syst Rev 37(5):164–177
16. Porras P, Shin S, Yegneswaran V, Fong M, Tyson M, Gu G (2012) A security enforcement kernel for OpenFlow networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. ACM, Helsinki, Finland. pp 121–126
17. Shin S, Porras P, Yegneswaran V, Fong M, Gu G, Tyson M (2013) Fresco: Modular composable security services for software-defined networks. In: Proceedings of Network and Distributed Security Symposium. NDSS Symposium 2013, San Diego, CA, USA
18. Köpsel A, Woesner H (2011) OFELIA: Pan-european test facility for OpenFlow experimentation. In: Towards a Service-Based Internet vol. 6994, ServiceWave 2011, Poznan, Poland, October 26–28, 2011.Proceedings. Springer, Springer Berlin Heidelberg. pp 311–312