FlowVisor Vulnerability Analysis

Ying Qian
Department of Computer Science
East China Normal University
Shanghai, China
yqian@cs.ecnu.edu.cn

Wanqing You, Kai Qian,
Department of Computer Science
Kennedaw State University
Marietta, GA, USA
{wyou, kqian}@kennesaw.edu

***Abstract*—This paper explored possible vulnerabilities in FlowVisor, SDN virtualized tool, and analyzed the potential isolation issue between multiple virtual slices by FlowVisor. We discovered a security vulnerability of interference between virtual slices in SDN, which leaves holes for potential malicious attacks. We proposed an event handling mechanism to be implemented in FlowVisor to avoid flow space overlapping.**

Keywords—OpenFlow; virtualization; security

I. Introduction

FlowVisor is an OpenFlow virtualization controller, based on Java, which plays a role of transparent proxy between OpenFlow switches and multiple OpenFlow controllers. With FlowVisor, we can create multiple isolated virtual logical networks ("slices") with different addressing and flow forwarding mechanisms on same physical infrastructure where same network resources such as OpenFlow switches/ports can be shared by different controllers in different slices. Slices can be defined in layer 1-4 by any combination. The layer model is the same as the definition in network. When sliced by switch ports, the policy is implemented in layer 1. When the Ethernet address or type are specified, slicing is implemented in layer 2, while src/dst IP address or type are enforced in layer 3. src/dst TCP/UDP port or ICMP code/type are used in layer 4. FlowVisor has responsibility to enforce isolation policy within each slice so that one slice should not control packet traffic flow from any other slices.

However, threats exist if the slicing policy is not implemented appropriately. In Sherwood's work [1], FlowVisor as a network virtualization layer has been discussed in detail. Victor explored three potential security issues related to header fields of flow spaces [2], which was based on a VLAN slicing policy; other threats can involve multiple flow spaces to cause forwarding loops or black hole in network [3].

This paper was investigating vulnerabilities of FlowVisor's isolation mechanisms in a virtualized SDN environment. We explored the vulnerabilities on the port based slice definition. We identified the configurations, which may result in a security hole and allow a malicious controller interfere other slices and break isolation principle for network virtualization.

II. Interference Scenario

We discover an interference scenario in FlowVisor, which caused by flow space overlapping. For example, we have two FlowVisor slices (slice1 and slice2), shown in the Fig. 1. Each slice is managed by a separate controller (C1 or C2) to control all the traffic in its slice.

We illustrate the interference scenario where two flow spaces added by administrator intersect with each other in the match fields, which could cause C1 intervening the traffic of C2. In this case, the matching fields specified by flow space 1 include that in flow space 2.

Suppose two Flowspace rules exist in flowvisor:

Rule1: dpid1 x=1, y=2, priority=10, Slice:slice1=…, Action1

Rule2: dpid1 x=1, y=2, z=3, priority=10, Slice:slice2=…, Action2

where x, y and z are three different matching fields.

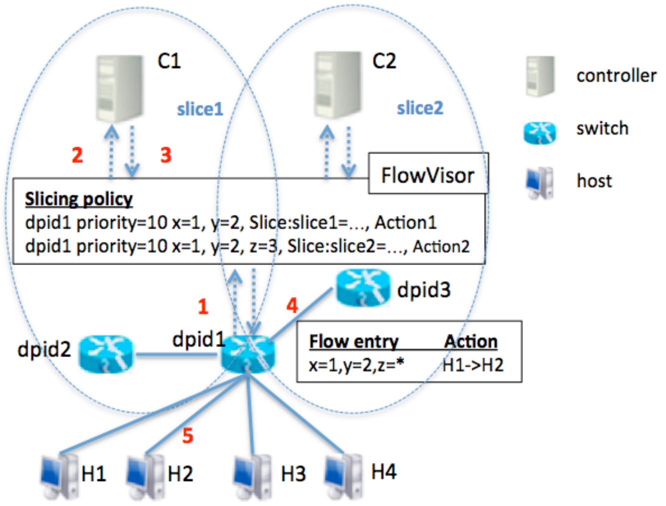

Fig. 1. Overlap of Flow spaces.

When a host (H1) intends to send a packet to another host (H2), it looks for corresponding matching rule in switch's flow table to find the direction for the packet. When no rule is matched, a Package_In event is thrown from switch to controller to ask what to do. Since we have FlowVisor sitting between switch and controller, the request from switch reaches at FlowVisor first, and then FlowVisor forwards the request to its controlling controller. After controller makes the decision to deal with the request, a Packet_Out event is thrown to

FlowVisor and a new rule in generated and inserted into the switch.

The red labels from 1 to 5 in Fig. 1 indicate five steps happening when a new Packet_In message with x=1, y=2, z=4 comes.

1. A Packet_In message with x=1, y=2, z=4 comes.

2. No matching rules found. The message is forwarded to C1 by FlowVisor to ask how to deal with this packet.

3. Controller C1 makes the decision to add new rule.

4. A new rule, with match fields specifying "x=1, y=2, z=*, Slice:slice1=…", is inserted into the switch's flow table.

5. A new flow with x=1, y=2, z=3 comes, which should belong to slice2. The packet will take the actions specified by the newly added rule (x=1, y=2, z=*, Slice:slice1=…). It means slice1 controls the traffic of slice2. This is one of the possible interferences that could happen if flow space configuration is not implemented appropriately.

III. Threats and Recommendations

The scenario discussed above illustrates some of the potential vulnerabilities when FlowVisor is introduced into OpenFlow. FlowVisor is implemented to achieve network virtualization in SDN, so that production networks and testing networks 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 brought 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 issue is vital in order to achieve a secure SDN network.

We recommend a new strategy that an additional event handling mechanism should be added into FlowVisor. The basic idea is to create a new_insert event to handle the case of adding a new flow space. When a new flow space is configured, a new_insert event takes place; then the new_insert event handler should take care of this kind of event. When receiving the event, the event handler will check whether there is overlap of flow spaces.

The possible strategy to check the overlap of flow spaces mentioned above is shown as Fig. 2. It means every time when a new flow space is added, the script below will be triggered to go through the existing flow space 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 decision either rewriting the existing flow space or giving up the new flow space.

IV. Implementation

We implement and test flow space overlapping scenario in this section. We use Mininet as the network simulation tool to build a virtual OpenFlow network, and interact with it via command line interface or APIs. We implement diamond topology in our test, and it is sliced based on switch's ports into an upper slice and a lower slice with two switches shared by different slices. After the slicing policy is done by "*fvctl*" commands, we can "*pingall*" to test the reachability between all pairs of hosts. After the slicing policy of this experimentation is done, only hosts in the same slice are reachable from one to another.

Now we start to add flow spaces to create flow space overlapping. In our case, the newly added flow spaces are discussed in previous section. As result, first controller C1 would take care of the requests that should have been responded by controller C2. Therefore we demonstrate that the security vulnerability exists.

```
//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.
function checkOverlap (Array <FlowSpace> flows, FlowSpace
newFlow)
START:
foreach flow in flows:
  if (newFlow ∩ flow ≠ Φ ):
      prompt: rewrite or quit
      break;
  end if
end foreach
END
```

Fig. 2. Function for checking flow space overlap.

V. Conclusion and Future Works

In the paper, we explored the slicing policy in details, and found the potential interference vulnerability while implementing FlowVisor in SDN network. We proposed an event handling mechanism in FlowVisor to avoid flow space overlapping. In the future, we will dedicate to employing the strategy we proposed. Moreover, we will try to implement other slicing policy based on VLAN ID and other slicing mechanism and try to explore more potential vulnerabilities to make SND network more secure.

References

[1] R. Sherwood, G. Gibb, K. K. Yap, G. Appenzeller, M. Casado, N. Mckeown, and G. Parulkar, "FlowVisor: A Network Virtualization Layer", Techinical Report, 2009.

[2] V. T. Costa, L. H. M. K. Costa, "Vulnerability Study of FlowVisor-based Virtualized Network Environments", Techinical Report, 2013.

[3] A. Khurshid, X. Zou, W. X. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time", 10th USENIX Symposium on Networked Systems Design and Implementation, Lombard, Illinois, USA, April, 2013.