Analysis and Evaluation of OpenFlow Message Usage for Security Applications

Sebastian Seeber, Gabi Dreo Rodosek, Gaëtan Hurel, Rémi Badonnel

Analysis and Evaluation of OpenFlow Message Usage for Security Applications

Sebastian Seeber[1], Gabi Dreo Rodosek[1], Gaëtan Hurel[2], and Rémi Badonnel[2]

[1] Universität der Bundeswehr München
Department of Computer Science
85577 Neubiberg, Germany
sebastian.seeber, gabi.dreo@unibw.de
[2] Université de Lorraine - Inria Nancy Grand-Est
Campus Scientifique
54600 Villers-les-Nancy, France
gaetan.hurel@inria.fr, remi.badonnel@loria.fr

Abstract. With the advances in cloud computing and virtualization technologies, Software-Defined Networking (SDN) has become a fertile ground for building network applications regarding management and security using the OpenFlow protocol giving access to the forwarding plane. This paper presents an analysis and evaluation of OpenFlow message usage for supporting network security applications. After describing the considered security attacks, we present mitigation and defence strategies that are currently used in SDN environments to tackle them. We then analyze the dependencies of these mechanisms to OpenFlow messages that support their instantiation. Finally, we conduct series of experiments on software and hardware OpenFlow switches in order to validate our analysis and quantify the limits of current security mechanisms with different OpenFlow implementations.

1 Introduction

Software-defined networking (SDN) has become a major paradigm for network programmability with the large-scale deployment of cloud infrastructures and the virtualization of network functions. It currently provides a convenient support to the design and implementation of different services, including security mitigation mechanisms, through the abstraction of higher-level functionality. In particular, it is often perceived or expected as a potential solution for enabling fast reconfiguration operations in order to address the growing complexity of networking environments. Indeed, decision making processes can be facilitated at the SDN controllers level, e.g. about forwarding paths, based on the logical global view of the network that is abstracted and given to applications. Moreover, the close relationship between network intelligence and the forwarding plane enables a faster reply and a more flexible way to react to security incidents, in comparison to other traditional solutions.

The abstraction induced by software-defined networking poses also important security issues with respect to the reliability and dependencies of solutions that are built on top of them. This statement is even more critical when these applications using network programmability facilities are intended to detect or prevent security attacks. Typically, these solutions are based on the OpenFlow standardized protocol, which is one of the most prominent software-defined solutions for supporting communications between network controllers and programmable switches. It therefore plays a central role in the effective reliability of applications. However, the various implementations of this protocol react in different ways. For instance, the timing and count of OpenFlow messages may differ for hardware and software implementations and among multiple vendors, which may have a direct impact on the overall performances of software-defined applications. In that context, a major challenge is to analyze the dependencies of security solutions to software-defined networking protocols, such as the exploitation of OpenFlow messages. It is also important to evaluate their performance impact on different hardware implementations to draw conclusions about the effectiveness of security approaches based on OpenFlow messages. Otherwise, vendors of OpenFlow-based security applications are bound to specific hardware and thus dashes the expectations of software-defined networking with respect to open vendor independent and standardized interfaces.

The rest of this paper is organized as follows: Section 2 gives an overview of security attacks that have been considered in this analysis and describes SDN-based security mitigation currently available to address them. In Section 3 we analyze the dependencies of these security solutions in terms of OpenFlow message usage through a dedicated mapping. We then evaluate in Section 4 the performance of different OpenFlow implementations and the induced impact on security applications. Section 5 details related work in the area of software-defined security. Section 6 concludes the paper and points out several research perspectives.

2 Network attacks and SDN-based defences

Considering the traditional taxonomy of security attacks published in [12], our analysis has focused on SDN mitigation mechanisms for tackling two major categories of security attacks, namely overloading attacks and information gathering attacks. We remind in this section each of these categories and detail defence strategies designed in traditional and software-defined environments. These strategies will then serve as a basis for analyzing and quantifying the dependencies of security mechanisms to OpenFlow messages.

2.1 Mitigation of overloading attacks

These last years have seen an increase of overloading attacks, with in particular distributed denial-of-service (DDoS) [2] whose growth has been evaluated to 90 % in the last 12 months by a recent report from Akamai [1]. The main methods used in networking and software-defined networking are based on flooding techniques, where the attacker generates a very high amount of packets to overload the target environment. A typical example is given by smurf attacks which generate ICMP echo/reply packets, where the source IP address is spoofed, with broadcast networks to multiply traffic. Following this approach, a couple of low-bandwidth sources can easily kill high-bandwidth connections. The overloading attacks may also rely on amplification techniques. In that case, the approach consists in turning a small amount of bandwidth coming from a few machines into huge attacks targeted on a specific device. For instance, in the case of NTP (Network Time Protocol) amplification attacks, this is made possible by the fact that no authentication is required in order to obtain a response. Therefore, the attacker is capable of forging their address so that the generated request looks like it originated from the intended victims machine. The attacker sends forged requests to a large distributed number of servers across the network. Since the response is up to 200 times bigger than the request, a large attack can be initiated by simply a single machine, once amplified through a number of distributed NTP servers. Such type of response is possible due to the `monlist` command, which is available in NTP servers. This command can return the addresses of up to the last 600 machines that the NTP server has interacted with. The amplification factor of domain name service (DNS) is much lower and ranges of around 40 to 100 depending on the effort that the attacker puts into the preparation of its attack. An overview of typical bandwidth amplification factors is presented in [3]. There exist many similar overloading attacks that are initiated via network, such as Teardrop, Bonk, Boink and Ping of Death. The impact is always to severely impair or disable an host or at least its IP stack, but through packet fragmentation techniques or vulnerability reassembling. However, these attacks require of course an host IP stack in order to receive packets from the attacker.

Mitigation strategies against overloading attacks within traditional networks are experiencing difficulties regarding their deployment, because most of them induce high network complexity and prohibitive operational cost [26]. In the meantime, SDN-based networks prove to be much more flexible due to their programmable nature. In an SDN-based network, the centralized view of the network state by the controller(s), as well as the capacity of the network to be dynamically reprogrammed, significantly ease the deployment of mitigation strategies, such as DDoS mitigation, initially designed for traditional networks. In particular, Moving Target Defense (MTD) is an intrusion prevention mechanism used to periodically change a deterministic attribute (typically the IP address) of a chosen host, in order to confuse attackers and thus protect the host. Usually, deploying MTD mechanisms within traditional networks is difficult and costly, because it involves the usage of dedicated hardware facilities for

hosting the MTD intelligence [14]. In an SDN-based network, the MTD intelligence is hosted at the controller level, which is able of dynamically reconfiguring the network according to its dedicated algorithm [15]. Traffic analysis for intrusion and anomaly detection within a given network is another example of mitigation strategies simplified by the SDN paradigm [18]. In such a context, the controller can simply query the switches of the network in order to gather statistical information about the network traffic, and then detect potential intrusion or anomaly according to its detection algorithm(s). Once an attack has been detected, blocking the source of the attack or redirecting the associated malicious flows to security middleboxes - i.e. providing an intrusion response - can be done at the controller level by reprogramming the whole network [20]. Such reprogramming steps are important for implementing efficient intrusion tolerance mechanisms.

2.2 Mitigation of information gathering attacks

Another important category of attacks corresponds to information gathering. The one refers to the process of determining the characteristics of one or more remote hosts (and/or networks). Information gathering can be used to construct a model of the target host, and to facilitate future penetration attempts. There exist several and complementary methods to perform a remote information gathering in the literature at various levels:

- **Host detection**: this method tries to identify if a host is available. In most of the cases this is done by a *ping* or *fping* which elicits e.g. an ICMP ECHO_REPLY from a victim.
- **Service detection**: service detection is typically performed based on port scanning. The objective is to detect the availability of UDP, RPC or TCP services, e.g. HTTP, DNS, through the execution of SYN or FIN scanning or slight variations like fragmentation scanning.
- **Network topology detection**: to get more information about a network, methods like TTL modulation, e.g. with *traceroute* or record route, e.g. *ping -R* can be performed. Another non-invasive method to learn more about a network is by network sniffing.
- **Operating system detection**: since the implementations of TCP/IP stacks of operating systems are different, the behavior of such an implementation can give information about the concrete operating system. This could be an interesting information to get access to the victim system, because the attacker can determine which vulnerabilities are present and exploitable. An additional name for this method is TCP/IP stack fingerprinting.

In the past, information gathering was performed with a one to one or one to many model; i.e. an attacker performs techniques linear against either one target host or a logical group of targets (e.g. a subnet). These methods were often optimized for speed and executed in parallel (e.g. nmap). Newer types of information gathering methods use distributed methods following the many to one

or many to many model. Therefore, an attacker tries to use multiple hosts to execute some information gathering methods in random and non-linear ways. The aim of the distribution is to avoid detection either by human analysis or network intrusion detection systems.

Mitigation strategies for information gathering attacks within SDN-based networks prove to be quite similar to the ones mentioned for traditional networks. The main difference resides in the holistic view of the controller(s), which may ease both the statistic aggregation and the correlation steps, as well as blocking (after detection) by reprogramming the network. For instance, in the case of Moving Target Defence (MTD) solutions mentioned above, the mechanisms make the attacker task harder, since the information obtained from a scanning attack at a period p may not be correct anymore at the period $p+1$. Some other advanced MTD mechanisms have been designed in SDN-based networks to add network noise, such as dynamic fake servers and fake open ports [13], as well as to prevent OS fingerprinting and service version/banner grabbing. However, this last feature may induce overhead at the controller and/or the switch layer, since it requires to look and modify information in upper layers (e.g. httpd version in HTTP header), which might seem contrary to the SDN paradigm principles.

3 Analysis of OpenFlow Messages used for Network Security Applications

Mitigation strategies take benefits from facilities offered by software-defined networking, even when they rely on similar models and methods well-known in traditional networks. These solutions often built on top of the software-defined layer introduce however dependencies of security mechanisms to these facilities, in particular to the OpenFlow protocol, that we analyze in the section. We typically consider three main deployment categories in software-defined infrastructures: reactive, proactive and hybrid deployments. In all of them, a flow-table lookup is performed when a network flow reaches a switch. Depending on the implementation, e.g. software vSwitch or hardware switch (ASIC (Application-Specific Integrated Circuit)) flow tables are accessed. In case no matching flow is found a request to the controller is sent for further instructions.

In a reactive approach, the controller acts upon these requests through the creation and installation of a rule in the switch's flow-table for the corresponding packet. In a proactive approach, the controller populates flow-table entries for all possible traffic matches possible for this switch in advance. This mode is comparable with typical traditional routing entries today, where all static entries are installed ahead in time. Following this proactive implementation, no request needs to be sent to the controller, since all incoming flows should find a matching entry. The major advantage of proactive deployments is due to the fact that all packets are forwarded in line rate (considering flow-table entries are stored in TCAM (Ternary Content-Addressable Memory) and no delay is added. In addi-

OpenFlow Message	Security Functionality
PACKET_IN	Monitoring of e.g number of new flows (detection)
OFPFlowMod	Traffic redirection and queuing (mitigation)
OFPMeterMod	Rate-limiting (detection)
OFP*StatsRequest	Detection based on statistics collection

Table 1: Mapping of OpenFlow messages to security functionality addressing overloading attacks

tion, hybrid environments exist where the flexibility of a reactive environment for a set of traffic is used, while the low-latency forwarding (proactive) is used for the rest of the traffic.

Our analysis with respect to OpenFlow message usage for network security considers a reactive environment. Indeed, software-defined networking and in particular the OpenFlow protocol is typically leveraged for a dynamic reconfiguration and setup of the network. In addition, proactive deployments are quite inflexible. Therefore, proactive scenarios are often based on hybrid environments with a reactive part that is not necessarily activated. Our approach is applicable to all OpenFlow-enabled SDN environments that include a reactive part. This behavior allows us to gather and measure useful OpenFlow-related information, such as PACKET_IN messages, which are necessary for several types of security related applications. Based on this consideration, and within a security context, we specified for each attack category, a mapping of the OpenFlow message types that are used for serving security functionalities, such as detection, mitigation and reconfiguration purposes. These security mechanisms therefore rely on the reliability of these messages and the information they carry (e.g. counters). We considered the following OpenFlow message types in this security-oriented analysis and mapping:

- **PACKET_IN messages**: these are sent from the OpenFlow-enabled switch to the controller in case a new flow arrives at the switch and no matching flow-table entry is found. This behavior is useful for detection and mitigation approaches, like e.g. blacklisting or firewalling. In this case the number of new flows (e.g. IP addresses) can be counted (gathering stats on-the-fly) and if too many new IP addresses arrive, whether they are allowed or not this could be an indication for e.g. a DDoS attack or anomalous behavior in the monitored network. To gather these kinds of statistical data there, exist dedicated OpenFlow messages (e.g. MULTIPART_REQUEST). Compared to those messages, gathering PACKET_IN-based statistics is done on-the-fly for reactive environments inducing no additional OpenFlow communication

between the controller and the switch(es). In proactive environments MULTIPART_REQUEST messages can assume this task.

- **OFPFlowMod, OFPFlowStatsRequest / OFPFlowAggregateStatsRequest messages**: these are suitable for redirection and traffic mirroring. These messages can be useful to mirror traffic to different types of intrusion detection middleboxes or security appliances. In addition, there exist specific flags within these messages to reset packet and byte counters (OFPFF_RESET_COUNT) in the switch or modify the configuration of a switch in a sense that it sends a message once a flow rule has expired. As a last use-case these messages can be used to mitigate an attack by dropping malicious packets.
- **OFPMeterMod / OFPMeterStatsRequest messages**: these allow a rate-limiting configuration, which was originally designed for quality of service purposes. Nevertheless, this functionality can be used in the area of detection for sampling packets.
- **OFPQueueStatsRequest messages**: these can be used to gather statistics from existing queues. In the area of security applications an option is to set up two queues: one queue for legitimate traffic with high bandwidth and one queue for suspicious and malicious traffic with a limited throughput. The process of setting up such a queue states part of prevention or mitigating an attack. The statistics collection part is relevant for detection purposes. Using the set-queue attribute an application can set up a defined action (e.g. OUTPUT, DROP) for a specific queue.
- **Multipart messages**: these provide plenty of options useful for detection purposes. Using the OFP*Stats[Request—Reply] messages, statistics about flows and rules can be gathered from the switches. These methods are useful for applications to detect, e.g. possible anomalies in the traffic flows. In addition, considering pre-installed rule sets for security applications the statistics collection methods are necessary to derive possible security events. To reduce false-positives in such a detection approach correlation methods need to be developed.
- **PacketOut messages**: these enable forging packets and send them to security devices / middleboxes in order to reconfigure the network according to already detected incidents or to change configuration options to improve detection capabilities.
- **OFPFlowRemoved messages**: these are suitable for security diagnosis and testing purposes. With associated counters involved they could also be useful to improve detection capabilities.

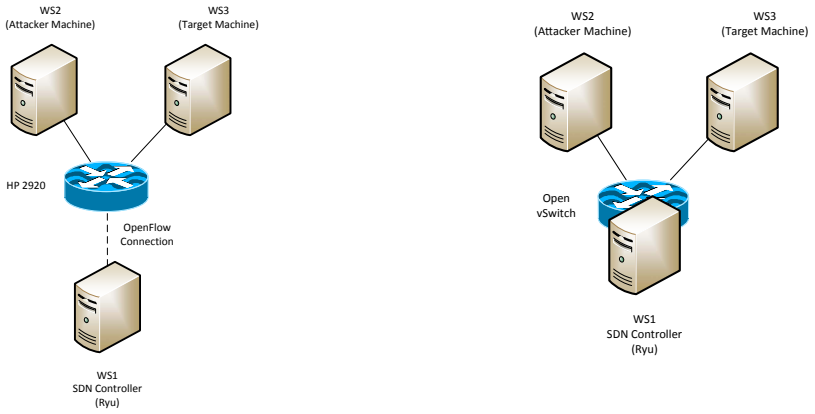
The different results of this security-oriented analysis of OpenFlow message usage are summarized in Table 1 corresponding to overloading mitigation strategies, and in Table 2 focusing on information gathering mitigation strategies. Keeping in mind these dependencies between security application goals (e.g. detection, mitigation, reconfiguration) and OpenFlow messages, there is a need to evaluate the implementation of OpenFlow messages in existing software-defined devices (software and hardware), in order to quantify the potential impact on these mitigation mechanisms.

OpenFlow Message	Security Functionality
PACKET_IN	Collection of new flows/packets (detection)
FlowMod	Traffic redirection and queuing (mitigation)
PacketOut	Confuse scanning by sending forged packets to the attacker

Table 2: Mapping of OpenFlow messages to security functionality addressing information gathering attacks

4 Performance Evaluation

Based on this analysis, we have performed a series of experiments in order to evaluate the accuracy and reliability of OpenFlow messages. The objective of this quantification is then to infer the potential impact of this performances on security applications developed on software-defined networking infrastructures. In that context, we have built dedicated testbeds based on hardware and software SDN solutions and have focused on two different types of messages, namely PACKET_IN and OFPQueueStatsRequest messages. However, this approach is generic and can be easily applied to the other messages identified in the previous section. The main reason of this focus was to gather statistics comparable to NetFlow / sFlow data from an OpenFlow-enabled switch.



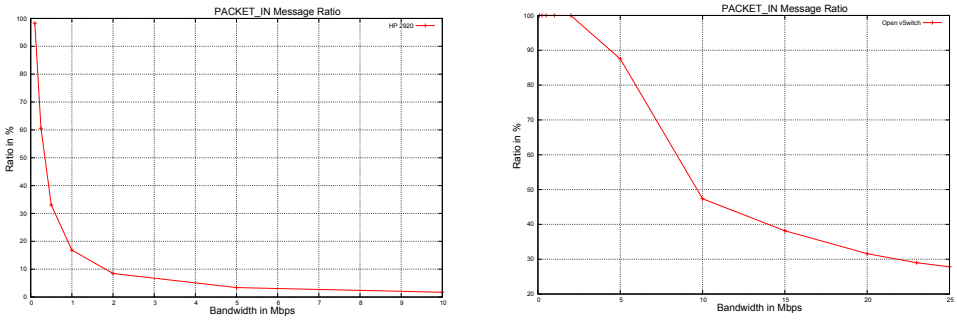
(a) Experimental Setup with an HP 2920 switch

(b) Experimental Setup with an Open vSwitch [4] on WS1

Fig. 1: Experimentation with SDN-enabled Software and Hardware Switches

We have considered the following experimental setup for performing our evaluation, with two different testbeds. Our first testbed consists of three workstations (4xCore i3 2.93GHz and 4 GB RAM) with the Debian 7 (kernel 3.2.0) operating system. All workstations have a gigabit network card installed that is connected directly to an HP2920-24G OpenFlow-enabled switch running the WB.15.16.005 firmware and OpenFlow version 1.3 enabled. On the switch, two workstations (W2 and W3) are connected within the same VLAN which is managed via the OpenFlow protocol. The other workstation(W1) directly connected to a switch port which is dedicated for OpenFlow controller messages (OpenFlow Management VLAN). For this purpose, the machine W1 is running an SDN controller. The Ryu framework in version 3.18 [5] is chosen, because it supports OpenFlow version 1.3 and is well maintained. The motivation of having two workstations (W2 and W3) connected to each other is to replay pcap files containing attacks on one machine (W2) and receive attacks from the pcap on the other machine (W3). Thus, the machine W2 can be seen as an attacker and the machine W3 as the target. Figure 1a corresponds to the first testbed. A modification of the experimental setup was done to verify the behavior of a software switch. For this second testbed, we installed Open vSwitch [4] on the machine W1 where the SDN controller with the Ryu framework is located and is connected to the Open vSwitch [4] locally. Moreover, we installed an additional 2 port gigabit network card and bridged the ports via the Open vSwitch [4] to connect the attacker (represented by the machine W2) and the target machine W3 (see Figure 1b corresponding to the second testbed). During our experiments, we replayed network traces derived from the DEFCON 22 hacking conference [7]. We used *tcpprep* in order to change IP addresses and simulate the traffic flow from the attacker workstation (W2) to the target workstation (W3). Furthermore, we deduced the statistics from the traces to assess the results with respect to OFPQueueStatsRequest messages. We modified the existing Ryu [5] controller code so that no new flow rule is pushed to the switch and make sure that all PACKET_IN messages are counted in time when they arrive at the controller. In addition, a flow-mod message is introduced during the initialization phase of the controller in order to insert a flow-rule that matches any packet (relevant for the OFPQueueStatsRequest messages evaluation).

In order to quantify the capacity of sending PACKET_IN messages from the switch to the controller, we replayed traces at different speeds from the attacker workstation W2. We considered respectively the following bandwidth speeds: 0.1Mbps, 0.25Mbps, 0.5Mbps, 1.0Mbps, 2.0Mbps, 5.0Mbps, and 10.0Mbps. In order to verify the speed and number of packets on the attacker and target workstations, we used common Linux tools, namely *ip -s link*, *iftop* or *nload*. In parallel, we counted on our modified Ryu [5] controller the number of OpenFlow PACKET_IN messages. In order to test our modified Ryu [5] controller script, we also counted the number of packets via the interface statistics on the workstation W1 on which the SDN controller is running. Based on these experiments, we evaluated the difference between the replayed packets and the effective received PACKET_IN messages. We then calculated the percentage of received



(a) PACKET_IN Message Ratio on HP 2920

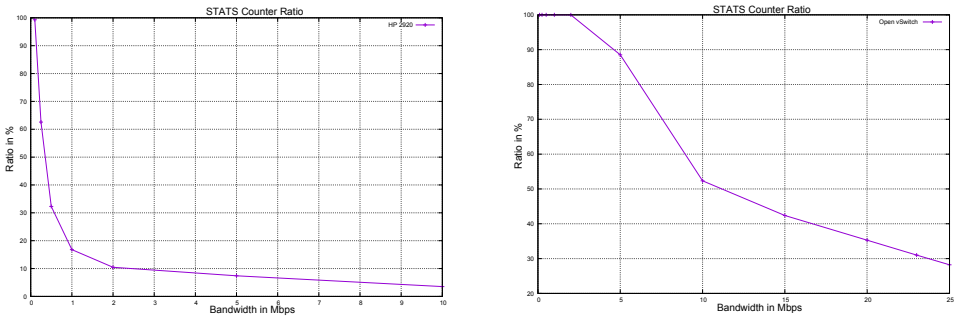
(b) PACKET_IN Message Ratio on Open vSwitch [4]

Fig. 2: PACKET_IN Message Ratio on SDN-enabled Switches

PACKET_IN messages with respect to the replayed packets. This value permits to quantify how much traffic in respect of bandwidth we can utilize from the switch without losing PACKET_IN messages, and therefore distorting statistic values collected from the SDN switch, this distorsion having a direct impact on the security application performance.

The results are given in Figure 2a and Figure 2b where we plotted the ratio of lost PACKET_IN messages while varying the bandwidth used when replaying traces for respectively the HP 2920 hardware SDN switch and the Open vSwitch software SDN switch. The results clearly show that the distorsion can be quite important in both cases, even with the bandwidth dedicated to the generated traffic is low. When we compare the two figures, it appears that the phenomenon is even more important with the first testbed, corresponding to the hardware SDN switch in our case. In a second serie of experiments, we quantified the performance with respect to the OFPQueueStatsRequest messages. We assessed the counters for the installed flow rules. Figure 3a and Figure 3b illustrate the relationship between the bandwidth and the ratio of correct packet counters for the earlier mentioned HP switch and the Open vSwitch. We can observe on the figures a similar trend as the one obtained with the experiments with the PACKET_IN messages. These results are particularly interesting, when we know that counters from matching flow-rules are preferably used to implement detection solutions, such as Defense4All [8], a module for the commonly used SDN Controller OpenDaylight [19].

These results raise important concerns about the implementation of OpenFlow in hardware as well as software solutions, and the implication that may directly have these differences in the context of security applications. It highlights severe differences in sending OpenFlow messages from the SDN switch to the controller, which may significantly degrade the performance of security mitigation mechanisms implemented based on software-defined networks.



(a) Correct Packet Counter Ratio on HP 2920

(b) Correct Packet Counter Ratio on Open vSwitch [4]

Fig. 3: Correct Packet Counter Ratio on SDN-enabled Switches

5 Related Work

Security aspects related to software-defined networking and its deployments have already been discussed by Schehlmann et al. in [21]. Several approaches facing security using SDN concepts have also been proposed recently. Kreutz et al. [17] argue for building dependable and secure SDN applications. Therefore, they identify and describe current threat vectors in SDN environments that could be exploited. They then propose a general design to overcome the identified threats. Complementarily, Scott et al. [22] investigate possible new security issues introduced through SDN and identify the affected layers. Focusing on network security approaches using SDN capabilities, François et al. [10] reviews recent research efforts and provides a qualitative comparison, complementary to our analytical and empirical evaluation.

Furthermore, existing work has been focusing on more specific attacks and their mitigation. Shishira et al. summarizes several types of distributed denial-of-service (DDoS) attacks and recently developed mitigation approaches in [23]. Vizváry et al. have analyzed the detection and mitigation of DDoS attacks using an OpenFlow enabled SDN environment in [24]. Using self-organizing maps, the authors of [6] propose a method to detect DDoS attacks based on flow analysis. Feamster et al. [9] investigated possibilities to detect botnet traffic by using distributed monitoring approaches. Jafarian et al. presented an approach to hide the real IP addresses by introducing a virtual IP address to hide real hosts from unauthorized scanners. A similar approach was introduced by Kampanakis et al. [16] to obfuscate the attack surface. Combining traditional network features (sFlow) and OpenFlow, Giotis et al. [11] proposed a mechanism to detect anomalies and mitigate attacks by modifying flow tables. A different architecture for monitoring and SDN control was proposed by Zaalouk et al. [25] to enhance the development of security applications by separating control and monitoring functions. In addition, the architecture supports a controller-agnostic application development by decoupling application development from the SDN controller. Our work rather aims at highlighting the limits of current software-defined solutions for implementing and supporting these security solutions.

6 Conclusion

The increasing interest for software-defined networking has contributed to the development of dedicated security solutions. However, these solutions typically built on top of these infrastructures may suffer from the performance of supporting protocols, such as the OpenFlow protocol, and their different implementations. In that context, we have proposed in the paper an analysis and evaluation of OpenFlow message usage by network security applications, in order to quantify these dependencies and their impact.

We have first describe two categories of security attacks, namely overloading attacks and information gathering attacks, that are quite common in these environments, and have detailed regular and SDN-based mitigation mechanisms that have been designed for tackling them. We have then analyzed for each category the dependencies of these mechanisms to the OpenFlow protocol commonly supporting the communications between SDN controllers and switches. These dependencies have been identified through the mapping of OpenFlow messages to security functionalities in that context. Based on this analyzis, we performed series of experiments for comparing and evaluating the accuracy and reliability that can be expected with respect to these messages based on two different testbeds. We first considered OpenFlow PACKET_IN messages that are typically generated when a new flow arrives to an SDN switch and no matching rule is found in the existing rule-set. We observed that the number of PACKET_IN messages sent to the controller strongly depends on the line speed of flows sent to the switch. For a higher line speed, the switch was not able to send PACKET_IN messages at the same speed when new packets arrived. This is particularly impacting, because this directly influences the statistics gathered from the switch, which are used by security solutions as a starting point for several detection approaches. We then performed experiments with respect to OFPQueueStatsRequest messages that are used to provide statistics on existing queues, and observed a similar degradation of performance. When the line speed is high, the precision of counters per flow-rule can significantly decrease.

As future work, we are interested in performing complementary experiments, in order to extend our methodology to additional OpenFlow message types. This analysis will permit to further investigate the dependencies of security applications and their limits regarding SDN implementations. This could directly influence the design of these security mechanisms, and allow us to infer and specify guidelines and patterns with that respect, in order to maximize security performance.

Acknowledgment

The authors wish to thank the member of the Chair for Communication Systems and Internet Services at the Universität der Bundeswehr München, headed by Prof. Dr. Gabi Dreo Rodosek, for helpful discussions and valuable comments for this paper. This work was partly funded by FLAMINGO, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

References

1. Akamai - Q4 2014 State of the Internet – Security Report, http://www.stateoftheinternet.com/resources-web-security-2014-q4-internet-security-report.html, last visited on 2016-02-04
2. Arbor Networks - Worldwide Infrastructure Security Report 2014, http://pages.arbornetworks.com/rs/arbor/images/WISR2014.pdf
3. US-CERT Alert (TA14-017A) UDP-Based Amplification Attacks, https://www.us-cert.gov/ncas/alerts/TA14-017A, last visited on 2016-02-04
4. Open vSwitch Community: Open vswitch, http://openvswitch.org/, last visited on 2016-02-04
5. Ryu SDN Framework Community: Ryu sdn controller, http://osrg.github.io/ryu/, last visited on 2016-02-04
6. Braga, R., Mota, E., Passito, A.: Lightweight ddos flooding attack detection using nox/openflow. In: Local Computer Networks (LCN), 2010 IEEE 35th Conference on. pp. 408–415. IEEE (2010)
7. DEF CON Communications, Inc.: Defcon pcap traces, https://www.defcon.org/html/links/dc-torrent.html, last visited on 2016-02-04
8. Defense4All: Defense4all module, https://wiki.opendaylight.org/view/Project_Proposals:Defense4All, last visited on 2016-02-04
9. Feamster, N.: Outsourcing home network security. In: Proceedings of the 2010 ACM SIGCOMM workshop on Home networks. pp. 37–42. ACM (2010)
10. François, J., Dolberg, L., Festor, O., Engel, T.: Network security through software defined networking: a survey. In: IIT Real-Time Communications (RTC) Conference-Principles, Systems and Applications of IP Telecommunications (IPTComm). ACM (2014)
11. Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V.: Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. Computer Networks 62, 122–136 (2014)
12. Hansman, S., Hunt, R.: A taxonomy of network and computer attacks. Computers & Security 24(1), 31–43 (2005)
13. Jafarian, J.H., Al-Shaer, E., Duan, Q.: Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the first workshop on Hot topics in software defined networks. pp. 127–132. ACM (2012)
14. Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S.: Moving target defense: creating asymmetric uncertainty for cyber threats, vol. 54. Springer Science & Business Media (2011)
15. Kampanakis, P., Perros, H., Beyene, T.: Sdn-based solutions for moving target defense network protection. In: A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on. pp. 1–6 (June 2014)
16. Kampanakis, P., Perros, H., Beyene, T.: Sdn-based solutions for moving target defense network protection. In: A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on. pp. 1–6 (June 2014)
17. Kreutz, D., Ramos, F., Verissimo, P.: Towards secure and dependable software-defined networks. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. pp. 55–60. ACM (2013)

18. Lara, A., Kolasani, A., Ramamurthy, B.: Network innovation using openflow: A survey. Communications Surveys Tutorials, IEEE 16(1), 493–512 (First 2014)
19. OpenDaylight: Sdn controller opendaylight, `https://www.opendaylight.org/`, last visited on 2016-02-04
20. Sahay, R., Blanc, G., Zhang, Z., Debar, H.: Towards autonomic ddos mitigation using software-defined networking. In: 2015 Network and Distributed System Security Symposium (NDSS'15). pp. 1–6 (February 2015)
21. Schehlmann, L., Abt, S., Baier, H.: 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 (2014)
22. Scott-Hayward, S., O'Callaghan, G., Sezer, S.: Sdn security: A survey. In: Future Networks and Services (SDN4FNS), 2013 IEEE SDN for. pp. 1–7. IEEE (2013)
23. Shishira, S., Pai, V., Manamohan, K.: Current trends in detection and mitigation of denial of service attacks-a survey. International Journal of Computer Applications (2014)
24. Vizváry, M., Vykopal, J.: Future of ddos attacks mitigation in software defined networks. In: Monitoring and Securing Virtualized Networks and Services, pp. 123–127. Springer (2014)
25. Zaalouk, A., Khondoker, R., Marx, R., Bayarou, K.: Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions. In: Network Operations and Management Symposium (NOMS), 2014 IEEE. pp. 1–9. IEEE (2014)
26. Zargar, S., Joshi, J., Tipper, D.: A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. Communications Surveys Tutorials, IEEE 15(4), 2046–2069 (Fourth 2013)