A Forensic Methodology for Software-Defined Network Switches

Tommy Chin, Kaiqi Xiong

Chapter 6

A FORENSIC METHODOLOGY FOR SOFTWARE-DEFINED NETWORK SWITCHES

Tommy Chin and Kaiqi Xiong

Abstract This chapter presents a forensic methodology for computing systems in a software-defined networking environment that consists of an application plane, control plane and data plane. The methodology involves a forensic examination of the software-defined networking infrastructure from the perspective of a switch. Memory images of a live switch and southbound communications are leveraged to enable forensic investigators to identify and locate potential evidence for triage in real time. The methodology is evaluated using a real-world testbed exposed to network attacks. The experimental results demonstrate the effectiveness of the methodology for forensic investigations of software-defined networking infrastructures.

Keywords: Software-defined networks, incident response, forensics, switches

1. Introduction

Software-defined networking (SDN) is a popular enterprise technology that employs a number of security mechanisms [4, 5, 8]. However, attackers often erase log files and historical data in targeted systems to mask their malicious activities. This requires the application of forensic techniques to investigate the compromised systems. Several researchers have studied the forensic aspects of software-defined networking, including data centers [2], traceback techniques [6] and the management layer [10]. However, little, if any, research has focused on triage techniques for software-defined networking infrastructures, specifically for switching devices.

Several attacks have been developed that target software-defined networks [7, 9, 11]. Defensive mechanisms for combating these attacks

heavily depend on sophisticated detection techniques based on signatures and heuristic patterns [5, 23]. Switching software such as Open vSwitch (OVS) [16] is widely deployed to forward traffic in software-defined network infrastructures (Open vSwitch traditionally executes in an operating system of choice). However, in the case of a data breach or other compromise, it may be necessary to conduct forensic analyses of all the devices in a software-defined network infrastructure.

This chapter presents a forensic methodology for analyzing switching devices in software-defined networking environments. The methodology leverages memory images of a live switch with traffic capture from southbound communications, where a southbound interface enables a network component to communicate with a lower-level component. Note that southbound communications refers to the exchange of software-defined networking messages between a switch and controller. On the other hand, northbound communications refers to traffic between an application-oriented system interface and controller that involves procedural calls. Northbound communications ties applications with controllers, but this traffic is out of scope because the focus is on switching devices. The forensic methodology was evaluated using the Global Environment for Network Innovation (GENI), a heterogeneous testbed that provides extensive capabilities for software-defined networking research [3]. Specifically, a series of network attacks was used to thoroughly examine areas of interest in a forensic investigation.

2. Background

Network forensic approaches are applied to network devices that transport data of relevance to investigations [2, 10]. However, traditional approaches provide limited results due to the proprietary, vendor-specific nature of networking devices. Statistical information about networking devices can be derived from Simple Network Management Protocol (SNMP) traffic and logging services such as syslog. Other sources of information include configuration files and settings that are backed up remotely or locally depending on administrative needs. However, analyzing such information using available forensic techniques (e.g., [2, 6, 10]) often provides limited network event and system notification data.

Software-defined network switching devices, which come in hardware and software variants, contain potentially valuable forensic information. Software-based switching devices commonly reside in Linux-based computing systems [5, 19] and can therefore be investigated using traditional host-based forensic methods. Although valuable information may be obtained about the software-defined networking infrastructure, limited in-

formation is obtained if only local files are considered. System memory also contains useful information about running services and programs and can be forensically analyzed using the Volatility Framework [18]. However, there is limited research related to these aspects in software-defined networks.

A switching device in a software-defined network forwards traffic from one location to other destinations. This traffic originates from a variety of users, some of whom may be malicious. The switching device, which has no way of divining user intent, simply forwards the traffic to its destination. After a security incident, a forensic analyst can use network device statistics to discern the point-of-entry and other information about a malicious actor. However, a switching device is also a valuable source of evidence and, as such, should be considered during a forensic investigation.

The proposed methodology addresses the two main challenges involved in forensic analyses of networking devices. The first challenge deals with the timeline of events following a compromise, which is critical to an investigation. In a traditional network, memory contents may be erased due to normal operations, leading to the loss of valuable event information. Fortunately, software-based switches traditionally reside in virtualized systems that provide real-time snapshots of memory. The experimental evaluation conducted in this research examined the amount of time that an incident response team requires to collect event information from memory before the normal operations of a virtual machine (VM) clear the memory contents.

The second challenge deals with local storage in a software-defined networking device. A software-defined switching device traditionally has minimal secondary memory (hard drive) space and maximizes RAM and processing power to obtain adequate network transmission performance. Open vSwitch, a common distributed virtual multilayer switch used in software-defined networks, has numerous logging mechanisms that enable a variety of event information to be stored locally. If the virtual machine has limited storage space during the operation of the switch, older historical data is overwritten with new events in a continuous cycle. Remote storage of system logs can be implemented; however, due to the design of a software-defined network, this storage would have to reside in the control plane. A design limitation of software-defined networks prevents access from the data plane to the control plane; this requires a potentially costly solution to be implemented to obtain information. The experimental evaluation conducted in this research examines some problems related to the local storage of software-defined switches.

3. Related Work

A software-defined network is interesting from the security perspective because its controller provides an overall view of the managed network [15] and because the network is programmable [16, 17]. Security research related to software-defined networks has primarily concentrated on the detection and mitigation of link flooding [12], denial-of-service, man-in-middle and other attacks [8, 19]. However, most studies have employed Mininet [14] to conduct simulations for performance and security evaluations; however, these experiments are often not very realistic.

While traditional network forensics is a mature area, limited research has focused on forensic analysis techniques for software-defined networks. The work of Bates et al. [2] stands out in that it employs software-defined networking as a tool for digital forensics. However, the approach requires a middlebox system to collect traffic information in a software-defined network and save it to local storage.

This approach has two limitations. First, while a middlebox enables a full network traffic capture for analysis of a variety of events, the capture does not include the southbound communications from a controller, which provide critical information about events leading up to the incident and post incident. The second limitation is that adequate memory is essential to the functioning of a software-defined switch, but the preservation of memory contents is vital to forensic analysis because the memory contains crucial information about switch operation. The use of a middlebox is reasonable, but the memory contents may not be accessed without root privileges, which poses a major security risk.

Thus, the approach of Bates et al. [2] may not provide a forensic investigator with adequate information about a security incident. On the other hand, the proposed forensic methodology enables an investigator to examine and catalog incident information.

4. Proposed Forensic Methodology

A software-defined networking environment is impacted by security risks to switching devices, controllers and network peripherals. As more devices and peripherals are incorporated in a network, the number of vulnerabilities increase and, therefore, additional risks are introduced. This work assumes that a threat actor launches an attack from outside the network topology (wide-area network). It also assumes that the threat actor has compromised an internal computing system and has wiped all the content of the local hard drive. Finally, it is assumed the threat actor cannot access the switching device operating system (Open vSwitch) and controller (Floodlight) [17].

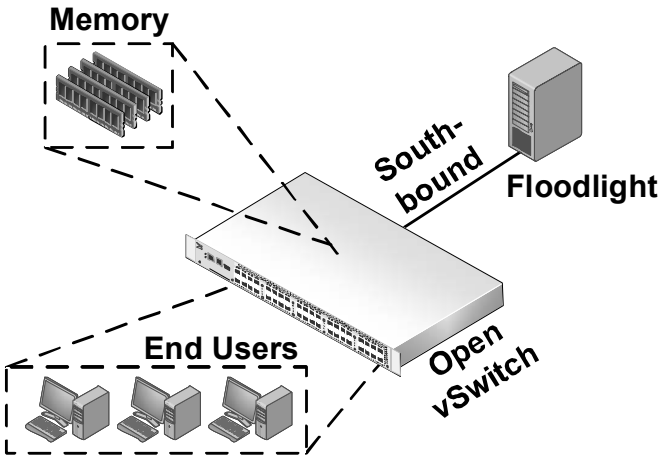

Figure 1. Threat model.

Figure 1 presents the threat model. The critical areas of investigation are the memory and the southbound communications of a software-defined switch captured by local logging mechanisms as determined by the software configuration. The figure shows a general switch configuration in which multiple computing devices are attached to the switch via physical or virtual configurations.

It is assumed that a threat actor resides in the group of end users. End users are targeted by the threat actor, but the software-defined network design ensures that Open vSwitch and Floodlight are untouched. Additionally, the activities of the threat actor leave forensic artifacts in memory and southbound communications traffic.

By leveraging the two main components of a switch, memory and network traffic, along with the service log files, a forensic investigator can identify and analyze artifacts related to a network compromise:

- **Memory Artifacts:** A network switch has memory components in a variety of specifications and sizes. The memory components provide rapid-access storage locations for network switching. From the perspective of forensic analysis, remnants of software variables and other artifacts stored in memory can provide useful information about the operations of the switch. Forensic tools such as Volatility, `edb` and Strings may be used for memory analysis. The experimental evaluation analyzed the amount of time various artifacts remain in the memory of an Open vSwitch.

- **Southbound Traffic:** During the operation of a software-defined network switch, a software-defined network controller provides in-

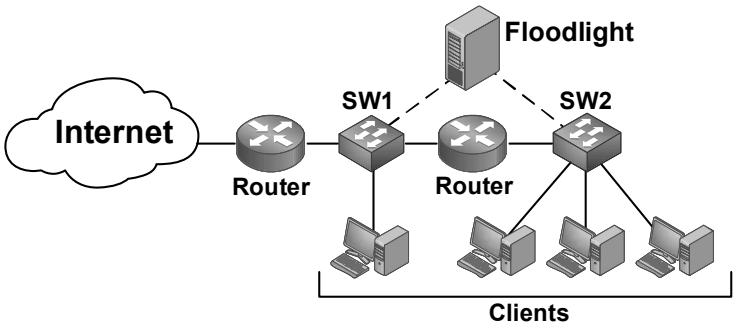

Figure 2. Experimental network topology.

structions and flow data to support network traffic to and from network devices. The communications between a controller and switch is known as southbound traffic, which mainly comprises OpenFlow messages [15]. The communications can contain a variety of artifacts that are valuable to a forensic investigation. The experimental evaluation used the `tcpdump` tool to capture and analyze southbound communications.

- **Service Log Files:** An Open vSwitch generates several log files that exclude operating system events. The log files reside in various locations of local storage and can be collected and analyzed for events of interest in a forensic investigation. The experimental evaluation considered some critical files for analysis and identified the information that can be utilized for forensic purposes. The events of interest include flow insertions, performance alerts and fault errors, among others.

5. Experimental Evaluation

This section describes the application of the proposed forensic analysis methodology on a software-defined network that uses an Open vSwitch. The experiments identify the valuable information that can be recovered and help provide a timeline for the incident under investigation.

5.1 Experimental Setup

The experimental evaluation of the proposed forensic methodology used GENI [3], which virtualized all the network nodes. Figure 2 presents the topology of the experimental network. Two switch placements were configured: SW1 positioned between two routers and SW2 positioned behind a router. SW1 and SW2 were positioned to emulate a demilita-

Table 1. Attack timeline.

Time	Actions
00:00	Start (Authenticate)
00:15	Plant Meterpreter Shell
00:20	Exfiltrate Data
08:53	Wipe Target Drive
08:54	End (Exit Network)

rized zone (DMZ) and an internal network, respectively. All the nodes in the GENI topology had the same hardware specifications: 2.10 GHz Intel Xeon CPU E5-2450 with 1 GB RAM and 16 GB hard drive running Ubuntu 14.04. All the network links were set to 100 mbps and the routers were Linux nodes with routing functionality. No latency, loss or link degradation occurred in the experimental configuration.

5.2 Attack Scenario

A threat actor could be an insider or an external entity. The attack scenario considered in the evaluation assumed that the attacker resides in the Internet cloud. It was also assumed that the actor could conduct reconnaissance to obtain network topology information.

Clients in the GENI topology were configured to have compromised SSH accounts. The threat actor targeted a client in the demilitarized zone and internal network. A remote shell was set up on the targeted machine. The Metasploit Framework [13] was then used to plant a Meterpreter shell in the targeted machine and various configuration files were subsequently exfiltrated to the Internet cloud.

Table 1 presents the timeline of the events involved in the attack on the software-defined network infrastructure. The threat actor gained access to the client using a compromised administrative access account, planted the shell, exfiltrated data, wiped the target drive and exited the network. Note that the time required to exfiltrate the data and wipe the target hard drive depends on the system resources and the quantity of data involved. However, in this scenario, after exfiltrating the data, the threat actor simply initiated the process to wipe the drive and exited the network.

5.3 Memory Analysis

Open vSwitch executes on a computing platform whose memory potentially contains valuable artifacts. Figure 3 presents the three-step

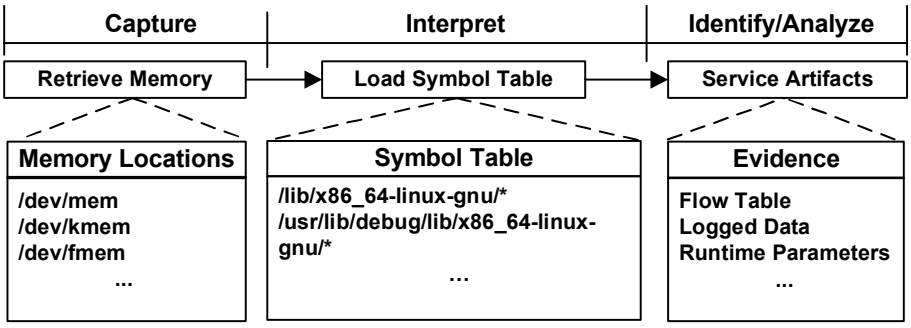

Figure 3. Memory analysis.

approach for conducting a memory analysis of an Open vSwitch. The three-step approach includes: (i) capture; (ii) interpretation; and (iii) identification and analysis.

Memory artifacts can be obtained via root-level access from several locations as shown in Figure 3. The following artifacts were discovered during the experiments:

- **Flow Table:** The flow table is used by the Open vSwitch to forward traffic to the correct destinations. The flow table is stored in memory and can be utilized to identify the origin of an attack and the path traversed within the network by the threat actor. The information collected included MAC addresses, switch port interfaces through which traffic traversed and port descriptors such as link states, speed and the number of packets sent. This information can be used by a forensic investigator to reconstruct the software-defined network topology up to one neighboring device.

- **Logged Data:** Open vSwitch has several logging mechanisms that can be leveraged in a forensic investigation. Although the log entries are written to local storage, the memory analysis of Open vSwitch revealed that logged events were stored for well over 20 minutes. Network activity and memory capacity determine when the events are erased and replaced. Note that network activity refers to the number of network connections between hosts. A large data transfer between two clients generates just one event whereas microtransactions between clients generate many events.

- **Runtime Parameters:** An administrator has to configure Open vSwitch to tailor it to the network requirements. The parameters and configurations stored in memory are of value in a forensic

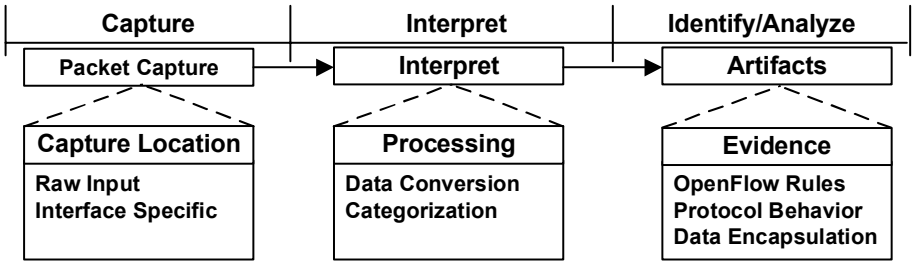

Figure 4. Southbound communications analysis.

investigation. These provide information about the logging mechanisms, controller and Open vSwitch plugins.

5.4 Southbound Traffic Analysis

Southbound communications are vital to the operation of a software-defined network because the controller utilizes this channel to send a variety of commands to a switch. In a forensic investigation, southbound traffic may be captured and analyzed to discover traces of the threat actor depending on how much time has elapsed.

Figure 4 shows the three-step approach for analyzing southbound communications. Southbound communications traffic can be captured using a variety of networking tools. Analysis of the captured communications during the experimental evaluation revealed OpenFlow traffic, but this quickly disappeared after the software-defined network flow timed out. Note that a software-defined network flow is inserted into a switch for data communications between a source and destination. The flow information has a timeout period in order to maintain the size of the flow table. If communications traffic stops after a period of time set by the administrator, then the flow disappears and the controller is informed. The timing of these communications depends heavily on the configuration of the Open vSwitch and the attached controller. The experiments used a default setting of five seconds and several runs were performed to analyze this issue. Note that a software-defined network uses several protocols to implement the desired network functionality, and this traffic can be analyzed in a forensic investigation.

A software-defined network controller has numerous other mechanisms to implement the network topology. The mechanisms can be identified by the behavioral aspects and data encapsulation techniques used to maintain and operate the network topology. For example, OpenFlow has two protocol-specific flags, OF Type 13 and OF Type 10, that enable data plane traffic to be transmitted and received by the controller,

respectively. Specially-crafted packets that leverage these flags may be configured and sent between the software-defined network planes to establish network links and collect statistical information.

In the attack scenario, the threat actor launched a simple attack and exfiltrated data. The associated communications are visible when a flow is inserted into a switching device, when a flow is updated and when link discovery protocols are used to maintain statistical information about a network path. Southbound communications can also reveal the transfer of exfiltrated data via delay analysis techniques that identify link latency. An investigator can use this information to carefully verify the location of the threat actor and the path traversed to the targeted machine.

5.5 Service-Level Event Logging

Although memory and southbound communications analyses can provide detailed information about a software switch, local logging mechanisms can also be employed by default on Open vSwitch to capture relevant events pertaining to software-based networking services. Open vSwitch provides three key log files:

- **ovs-ctl.log:** This log file contains information about the start and stop times of Open vSwitch. The artifacts can be used to verify the time sequence of network events and to show that Open vSwitch was running during an incident and that it was not killed or restarted during analysis.

- **ovsdb-server.log:** This log file provides event information related to a running database that maintains the Open vSwitch flow table. While this log file provides minimal details about network characteristics, the logged information is useful for investigating denial-of-service attacks that use spoofed IP addresses.

- **ovs-vswitchd.log:** This log file provides the most valuable artifacts related to traffic flows and controller communications. Analysis of the log file in the attack scenario revealed the threat actor's initial communications and network exit times. While limited details are provided about southbound communications, information in the log file provides a useful high-level overview of the channel. Indeed, using the logged information in conjunction with a southbound traffic capture can help verify the integrity of the southbound link.

5.6 Discussion

The attack scenario considered in the experimental evaluation involved a threat actor who entered the configured GENI network, compromised a targeted client, exfiltrated data and erased all the information related to the attack. However, the software switch could not be tampered with by the threat actor due to access control limitations imposed by the software-defined network configuration.

Three components of interest in forensic investigations of software-defined networks are memory, southbound traffic and service-level log files. Memory analysis revealed several artifacts that can be used to identify a threat actor's point-of-entry into the network along with a timeline. This information can be used along with residual information in southbound communications to verify the timing of key events, depending, of course, on how long after the compromise the incident response is conducted. While southbound communications might provide limited information due to a delay in incident response, service-level local log files can provide adequate information to correlate events and their times. An event timeline is an important aspect of a forensic investigation and can be very helpful in identifying the threat actor.

The following configurations are recommended to enhance forensic investigations of software-defined networks:

- **Memory Snapshots:** Analysis of the memory of a switching device is vital to a forensic investigation. The software switch considered in this study was merely a virtual machine running Open vSwitch on a hypervisor. Due to the nature of virtualization, several hypervisor platforms provide memory snapshot functionality. Periodic memory snapshots should be taken to record a history. However, since a memory image can be very large depending on the amount of memory allocated to a virtual machine, it is advisable to focus the snapshots on important areas of memory. Additionally, compression or historical data rotation could be used to ensure that data is not lost.

- **Centralized Logging Service:** In a software-defined network, limited amount of hard drive space may be allocated to a software switch due to hardware constraints or space conservation. Therefore, it is recommended to use a centralized logging service to collect information stored in service-level log files related to a switch in order to offload the hard drive space and facilitate data querying and analysis. The centralized logging service should be located in the data plane, but this can present a risk to the software-defined network infrastructure because this device would have access to

the data and control planes. Consequently, the centralized logging service should be positioned in the control plane or application to ensure the secure collection of relevant information without interactions with data plane users.

6. Conclusions

Software-defined network controllers are widely used to manage network traffic, allocate computing resources [1, 20–22], control network policies and detect and mitigate security attacks. However, limited research has focused on forensic analysis techniques for software-defined network controllers and devices. This chapter has attempted to address the gap by proposing an approach for forensically analyzing a software-defined network infrastructure from the perspective of a switch. The chapter also identifies the important artifacts that can be found in the memory image of a software switch and in southbound communications traffic. While several researchers have used Mininet to provide simulation results pertaining to their approaches, this research has employed real-time attacks on a real-world software-defined network to demonstrate the efficacy of the proposed approach.

Future research will focus on re-targeting the proposed approach to forensically analyze software-defined network controllers. Efforts will also concentrate on extending the approach to conducting forensic analyses of large-scale networks and applications.

Acknowledgement

This research was partially supported by the National Science Foundation under Grant Nos. CNS 1620871, CNS 1633978 and CNS 1636622; by the BBN/GPO Project 1936 under Grant No. CNS 1346688; and by a seed grant from the Florida Center for Cybersecurity (FC^2).

References

[1] A. Akella and K. Xiong, Quality of service (QoS) guaranteed network resource allocation via software-defined networking (SDN), *Proceedings of the Twelfth International Conference on Dependable, Autonomic and Secure Computing*, pp. 7–13, 2014.

[2] A. Bates, K. Butler, A. Haeberlen, M. Sherr and W. Zhou, Let SDN be your eyes: Secure forensics in data center networks, *Proceedings of the Network and Distributed System Security Workshop on Security of Emerging Network Technologies*, 2014.

[3] M. Berman, J. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci and I. Seskar, GENI: A federated testbed for innovative network experiments, *Journal of Computer Networks*, vol. 61, pp. 5–24, 2014.

[4] T. Chin, X. Mountrouidou, X. Li and K. Xiong, An SDN-supported collaborative approach for DDoS flooding detection and containment, *Proceedings of the IEEE Military Communications Conference*, pp. 659–664, 2015.

[5] T. Chin, X. Mountrouidou, X. Li and K. Xiong, Selective packet inspection to detect DoS flooding using software-defined networking, *Proceedings of the Thirty-Fifth IEEE International Conference on Distributed Computing Systems Workshops*, pp. 95–99, 2015.

[6] J. Francois and O. Festor, Anomaly traceback using software-defined networking, *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 203–208, 2014.

[7] S. Hong, L. Xu, H. Wang and G. Gu, Poisoning network visibility in software-defined networks: New attacks and countermeasures, *Proceedings of the Twenty-Second Annual Network and Distributed System Security Symposium*, 2015.

[8] H. Hu, W. Han, G. Ahn and Z. Zhao, FlowGuard: Building robust firewalls for software-defined networks, *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, pp. 97–102, 2014.

[9] M. Kang, S. Lee and V. Gligor, The crossfire attack, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 127–141, 2013.

[10] S. Khan, A. Gani, A. Wahab, A. Abdelaziz and M. Bagiwa, FML: A novel forensic management layer for software-defined networks, *Proceedings of the Sixth IEEE International Conference on Cloud System and Big Data Engineering (Confluence)*, pp. 619–623, 2016.

[11] D. Kreutz, F. Ramos and P. Verissimo, Towards secure and dependable software-defined networks, *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software-Defined Networking*, pp. 55–60, 2013.

[12] C. Liaskos, V. Kotronis and X. Dimitropoulos, A novel framework for modeling and mitigating distributed link flooding attacks, *Proceedings of the Thirty-Fifth IEEE International Conference on Computer Communications*, 2016.

[13] D. Maynor, K. Mookhey, J. Cervini, F. Roslan and K. Beaver, *Metasploit Toolkit for Penetration Testing, Exploit Development and Vulnerability Research*, Syngress, Burlington, Massachusetts, 2007.

[14] Mininet, Mininet (`mininet.org`), 2017.

[15] Open Networking Foundation, OpenFlow Switch Specification, Version 1.5.1 (Protocol Version `0x06`), ONF TS-025, Menlo Park, California, 2015.

[16] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon and M. Casado, The design and implementation of Open vSwitch, *Proceedings of the Twelfth USENIX Symposium on Networked Systems Design and Implementation*, pp. 117–130, 2015.

[17] Project Floodlight, Floodlight (`www.projectfloodlight.org/floodlight`), 2017.

[18] Volatility Foundation, Volatility Framework (`www.volatilityfoundation.org`), 2017.

[19] H. Wang, L. Xu and G. Gu, FloodGuard: A DoS attack prevention extension in software-defined networks, *Proceedings of the Forty-Fifth IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 239–250, 2015.

[20] K. Xiong, Web services performance modeling and analysis, *Proceedings of the International Symposium on High Capacity Optical Networks and Enabling Technologies*, 2006.

[21] K. Xiong, Multiple priority customer service guarantees in cluster computing, *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, 2009.

[22] K. Xiong, *Resource Optimization and Security for Cloud Services*, John Wiley and Sons, Hoboken, New Jersey, 2014.

[23] A. Zaalouk, R. Khondoker, R. Marx and K. Bayarou, OrchSec: An orchestrator-based architecture for enhancing network security using network monitoring and SDN control functions, *Proceedings of the Twenty-Sixth Network Operations and Management Symposium*, 2014.