Sigma Journal of Engineering and Natural Sciences
Web page info: https://sigma.yildiz.edu.tr
DOI: 10.14744/sigma.2022.00107

Research Article

A comparative analysis of software-defined network controllers in terms of network forensics processes and capabilities

Altuğ ÇİL[1,2,*], Mehmet DEMİRCİ[3,1]

[1]*Informatics Institute, Gazi University, Ankara, 06500, Türkiye*
[2]*The Scientific and Technological Research Council of Turkey (TÜBİTAK), Ankara, 06500, Türkiye*
[3]*Department of Computer Engineering, Faculty of Engineering, Gazi University, Ankara, 06570, Türkiye*

ARTICLE INFO

Article history
Received: 06 July 2022
Revised: 19 August 2022
Accepted: 17 December 2022

Keywords:
Computer Networks; Cyber Security; Forensics; Software-Defined Networks; OpenFlow; Southbound Interface; Ryu; ONOS; OpenDaylight; POX

ABSTRACT

The proliferation of software-defined networks (SDN) increases the necessity of security and forensic research in this field. Network forensics is of particular importance considering the ever-increasing traffic density and variety of devices, and SDN has great potential for improved forensic processes thanks to its ability to provide a centralized view and control of the network. This article's motivation is the lack of a standard forensic process in SDN. The main objective of this study is to examine the differences in the forensic processes of different SDN controllers, whether the southbound interface data is sufficient for the forensic processes, and whether it is possible to choose the best controller in terms of forensics. Four of the most widely used controllers have been selected and tested under seven different scenarios to observe how the results were obtained in terms of forensics. During the tests, in addition to the routine data accesses, attack preparation tools and denial-of-service attack tools were used to expand the scope. Experiments in which each scenario was applied for four different controllers demonstrated that different controllers have different characteristics in network forensics parameters, such as attack type detection, attacker information, service interruptions, packet size, and the number of packets. Experiments proved that southbound interface data is sufficient for forensic processes, different controllers have different characteristics in forensic processes, none of the most used controllers is the best to cover all forensic processes, and a standard forensic method is required for software-defined network forensics.

Cite this article as: Çil A, Demirci M. A comparative analysis of software-defined network controllers in terms of network forensics processes and capabilities. Sigma J Eng Nat Sci 2024;42(2):425–437.

INTRODUCTION

Software-defined network (SDN) is an innovative network architecture with lower cost, higher flexibility, better agility, and simpler management compared to traditional networks [1]. It is hardware-independent and any network device from any manufacturer can be controlled with SDN. As shown in Figure 1, the basic structure of SDN consists

***Corresponding author.**
*E-mail address: altug.cil@tubitak.gov.tr

This paper was recommended for publication in revised form by Editor-in-Chief Ahmet Selim Dalkılıç of three different layers: The application, Control, and Data (Infrastructure) layer. The data layer contains physical or virtual forwarding devices, while the control layer is responsible for making the decisions that dictate the behavior in the network. These two layers communicate via the southbound interface. The application layer hosts various functions such as firewall, monitoring, load balancing, etc., and communicates with the control layer via the northbound interface. Keeping in mind the increasing prevalence of intelligent programmable devices in the network, SDN is able to provide new network management solutions to improve overall network performance [2]. With this new architecture, SDN makes it easier to focus on network management processes and offers network administrators great opportunities in terms of traffic monitoring and management [3]. Additionally, SDN provides increased security, easier policy management, and less energy usage [4].

OpenFlow is the most common southbound interface protocol used to provide a communication channel between the control and data layers in SDN. It facilitates innovation and enables efficient service orchestration and automation [5], and allows for the communication of devices in the data layer with the SDN environment [6]. Different tasks such as network management, security applications, and load balancing are realized in SDN by the flow rules generated by the controller and sent to the data layer using OpenFlow messages [7]. OpenFlow-enabled data layer devices contain flow tables to store rules with information on how to forward packets. For OpenFlow to be enabled in hybrid networks during the transition from traditional networks, certain sub-domains must be defined, and communication settings must be made for traditional networks [8].

SDN ensures the security of SDN infrastructures and leverages SDN-based approaches to improve security [5]. Research on SDN security is relatively new, thus requiring further analysis. The innovations provided by SDN allow

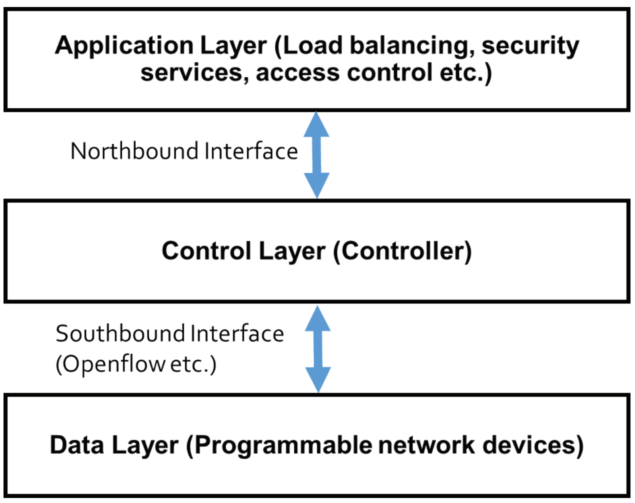
Application Layer (Load balancing, security services, access control etc.)
Northbound Interface
Control Layer (Controller)
Southbound Interface (Openflow etc.)
Data Layer (Programmable network devices)

Figure 1. Software-defined networking (SDN) architecture.

for improved security via security solutions produced with SDN itself, allowing for increased ability to follow up on attacks and facilitate attack detection [9]. These capabilities may be used to strengthen SDN forensics. Despite these advantages, a downfall is that SDN technology may be open to modern attacks [2]. SDN security research has focused on many different types of attacks and protocols. For instance, the attacker can create a new flow table with the help of a controller and forward the new rules to the network, mislead the network or create location spoofing by exploiting the weaknesses of the users through LLDP (Link Layer Discovery Protocol) packets [10]. In a solid infrastructure, a well-established authentication service between the switches and the controllers should be present. Otherwise, it will allow the attacker to examine the traffic [11]. An attacker can also perform a denial-of-service attack in a network environment using SDN by changing the flow tables [9]. This type of attack can be mixed with other attack types such as smurf attack, fraggle attack, and DNS amplification flooding attack. New methods have been employed to reduce the possibility of the controller being disabled [12]. The ARP (Address Resolution Protocol) poisoning attack may also be used to compromise the whole SDN topology by altering the identity of the controller, which is the most central and important part of SDN [13].

While many security issues and forensic processes in traditional networks are also applicable to SDN, new opportunities and new challenges have arisen in this young domain. This study considers the types of attacks emphasized in the literature in cyber attack scenarios within the scope of forensic experiments. Network forensics is of great importance in combating cyber attacks as it can enable both a better response during an attack and a more thorough forensic analysis after the attack. SDN offers a new and effective way to manage networks through programmability and increased flexibility [2]. As in all forensic analysis studies, the time record (stamp) must be kept in a reliable manner to comply with the legal ground. Network analysis is generally the examination of the data flowing through the network, focusing on the activities of the attacker [14]. The major questions to be answered by the analysis are “What”, “Why”, “Who”, “How,” and “When”. Network analysis techniques are composed of examination, report, analysis, investigation, presentation, identity, protection, decision-making, and collection and are grouped into proactive and reactive methods [15]. The general concepts in datasets that must be fulfilled are Findability, Accessibility, Interoperability, and Reusability (FAIR) [16].

To ensure information security in institutions, security systems like firewalls and intrusion detection systems (IDS) are used, and network analysis studies have a significant place in the development of these devices [17]. Although security studies have been carried out on SDN, they are poor in terms of forensic analysis [18]. Finding the root cause of attacks against SDN by applying forensic methods employed in traditional networks is insufficient and requires new approaches [19]. The innovations brought by SDN, such as data size, traffic monitoring, and the control layer being at the center of events, will allow for improved security [20].

Researchers have recently employed advanced methods such as deep learning to enable tracing the source of an attack [21]. This, in turn, will facilitate specifically targeting the source and identify the type of denial-of-service attack. Some researchers have gone beyond security analysis and developed modules that can solve system requirements. A module developed in the Python-based POX controller analyzes web server accesses to extract web statistics allowing to distinguish between web traffic and non-web traffic flows in the SDN infrastructure [22]. Ping flood attack packets on the POX controller can be prevented by detecting anomalies using machine learning [23]. Balarezo et al. carried out an experiment on denial-of-service attacks using TCP's retransmission timeout to the southbound interface via OpenFlow, and they proposed a durable model by revealing the vulnerabilities of the system [24].

All layers of SDN can be used to obtain evidence against suspicious activity. The controller is a vital evidence source as it is placed in the center of the structure, but it also makes it a target for possible attacks, casting doubt on the reliability of the data. The northbound and southbound interfaces can also be used for forensic analysis as they connect the controller to applications and infrastructure [25]. Pandya et al. tried different ways to obtain data from three different SDN controllers (OpenDaylight, Project Floodlight, and Ryu), and examined them in the Wireshark environment. The study concluded that the southbound interface was the best location to obtain data. A Python-based tool was used to conclude that the OpenDaylight controller gave the most successful results in terms of forensics [18]. Khan et al. proposed a forensic layer where the northbound interface is excluded, and the data produced by the network devices is examined by placing a central layer outside the southbound interface [26]. In another study, some varying attack types were tested in an environment where only a single controller was used; Ryu. The study proposed that previous studies do not adequately focus on the data obtained from the controllers. Thus is the records created in the controller are vital in the detection of DoS attacks. However, the difficulty in keeping topology poisoning attacks in controllers arose due to the temporary nature of the flow table [27].

A new method titled "Virtual Network Forensic Process" was proposed but only focused on packets flowing over the network, and the records from the controller were ignored [28]. Network policies built on OpenFlow rules can be used by a malicious user for various attacks but can also be prevented [29]. In a study, forensic applications at different levels of SDN were carried out by considering the security perspectives called "SDNForensics". In the framework where runtime logs, memory, and network packets were taken through three different data layers, memory information and logs were also obtained from users suggesting the integration of sources [30]. Another study proposed a new tool seeking the cause of the problem by searching the control and data layers for deficiencies in connection and packet transmission. This tool is beneficial in SDN as it is designed to diagnose actual control layer attacks by placing itself on the floodlight controller located between the control and data layers [31].

SDN controllers, despite their similarities, also have differences between them in terms of speed, security, infrastructure, programming language, environment, etc. The objectives of this study are to

- compare SDN controllers in terms of forensic informatics, taking into account the fundamentals of forensic information,
- determine whether the southbound interface data is sufficient for forensics,
- determine whether the best controller can be selected in terms of forensics by comparing different controllers during routine communication and under cyber attacks.

Beyond these objectives, the determination of whether the forensic processes can be carried out with minimum data size and amount was emphasized in all processes of this study. The variable parameter in our study, which occurs in the data layer and the southbound interface, is the change of controllers. This study is different from similar articles in that it investigates the differences in the data that can be obtained from the southbound interface during routine communication and attacks for four popular controllers, which were evaluated and compared in terms of forensics for the first time in the literature. The test steps allow for the visualization of differences between the information obtained from the southbound interface and the virtual switches. Important parameters of forensics such as accessed information, access speed, packet size and percentage, and access time have been tested during routine communication and with attacks that are considered important in terms of cyber security. With the absence of similar studies, this article may inspire other studies comparing different SDN controllers in detail under various network attack scenarios and evaluating these issues in terms of forensic computing.

METHODS AND EXPERIMENTS

Based on the problem of the lack of a standard forensic process in software-defined networks, the main idea of this study is the following research questions:

- Are there differences in forensic processes in different controllers?
- Is southbound interface data sufficient for the forensic process?
- Is the behavior of the controllers and the evidence obtained different during routine communication and attack?
- Is it possible to choose the best controller in terms of SDN forensics?

Within the scope of these objectives, seven different scenarios detailed below were tested with four different controllers (Ryu, POX, ONOS, and OpenDaylight). As shown in Figure 2, the test environment for each experiment had a tree topology with 16 hosts with various tools in a virtual infrastructure. The tests were performed in the same way by changing only the controllers of each experimental environment. While performing the tests, the information created on the network could be watched live and stored for later analyses. Since the controllers were the only variable during the tests, it was possible to see what kind of differences the controller change made on the data layer and the southbound interface from the forensic point of view. Tests' approaches were not only what data can be obtained, but also from the point of view of what size data can be accessed for the necessary information. In the experimental environment, the details given below, the primary focus is on the information that can be obtained in the data layer and southbound interface of software-defined networks during routine communication. Then, whether the message content detection can be done from these sections and whether it can be done in the case of which controllers are used were tested. Observations were made on how the controller changes differed in line with the purpose of port scanning and web server scans used in preparation for attacks. Finally, the results were obtained by applying different types of denial-of-service attacks separately for each controller to see how the correct information was obtained and the integrity of communication during the attack. The reason for choosing routine communication, message content detection, and denial-of-service attacks as different and independent from each other is to cover the entire forensic processes. As a result of varying controllers, different results from each scenario were obtained where attackers and victims were selected from 16 users. The details of the network observations and experiments carried out are shared in detail under each heading.

Infrastructure Information

In the SDN topology, Mininet [32] was used in a virtual server environment. In the relevant environment Ryu [33], ONOS [34], OpenDaylight [35], and POX [36] controllers were used. Ubuntu Desktop [37] was used as the environment infrastructure and as virtualization infrastructure, Virtualbox [38] were used on a laptop with 8 GB RAM, four virtual processors, and 150 GB HDD. The laptop had an Intel Core i7-10510U CPU 1.8GHz, 16GB DDR4 RAM, and 500GB SSD Disk capacity. OpenvSwitch [39] was used as a virtual switch. Packet inspections were carried out with Wireshark [40], and for forensic computing in network topology, listed programs were used; Yersinia [41], Nikto [42], Iperf [43], Nmap [44], Netcat and Hping3 [45]. Figure 2 shows the SDN environment, which is established with the help of Mininet, tree topology was used, and the experiments were carried out by creating a virtual environment consisting of 16 users (10.1.1.1-10.10.1.16), 15 virtual switches, and 1 controller. While Ryu, ONOS, and POX controllers were installed on Ubuntu 20.04 Desktop, the OpenDaylight controller was running on Ubuntu 14.04 Desktop. It was assumed that controllers have no rootkits and control channel is trusty.

Before starting the experiments, the bandwidth was measured with the help of iperf in the topologies to which all controller types were connected. There were changes in bandwidth between the hosts depending on the controller type, but the lowest one had 1.5Gbits/s bandwidth in the normal operating mode. These values are sufficient to carry out the planned measurements and studies within the scope of this study carried out in the local network environment. Additionally, this study also has an experiment about

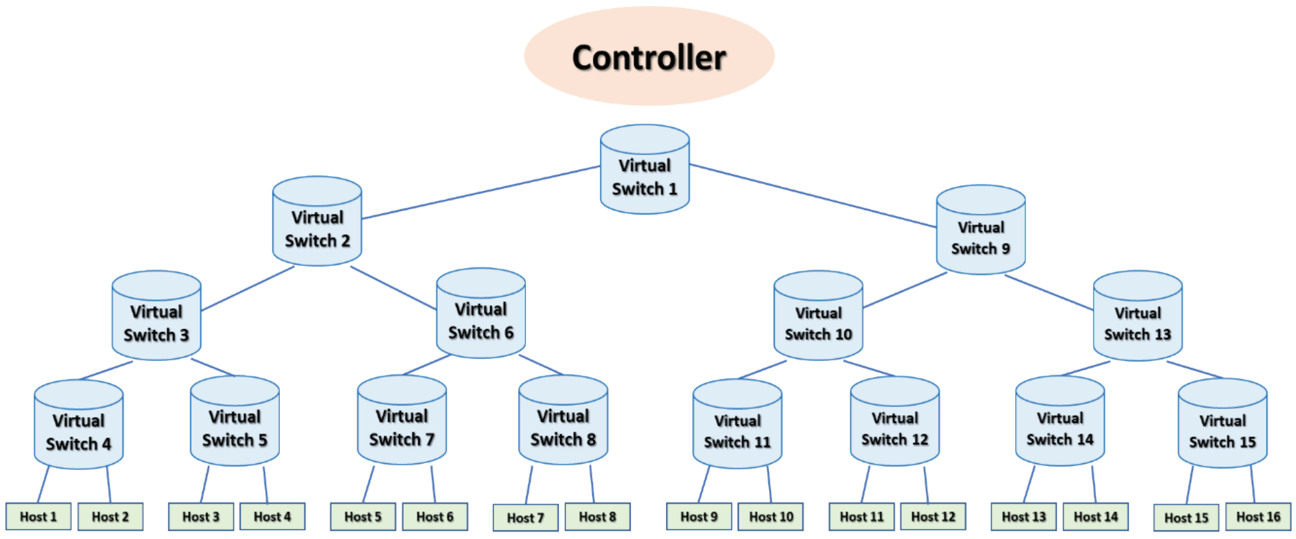

Figure 2. Experimental environment topology.

whether the controllers changed the delay values between the hosts, and the results show us that the delay times were between 0.1ms and 0.25ms on average for each controller. Also, the experiments showed that the lowest latency values occur in the Ryu controller with an average of 0.1 ms.

Forensic Process Start - Routine Communication

In the routine communication experiment, simultaneous ICMP packets were transmitted from all hosts to each other on Mininet (with the pingall command), and all data for all controllers were captured and filtered. Figure 3 shows the processes performed in this experiment and re-run for all four controllers, with details given below. Then an investigation was made to observe what kind of differences occurred between topologies in which the controllers differ in the network communication, which can be considered the simplest level, the information to be obtained from the data layer, and the packet sizes. In the same scenario, all data such as time, source, and destination through the files of southbound interface network captures in topologies were accessible where the controllers were different. IP address and MAC address information were also available in all captures. Packets were obtained in environments where Ryu, ONOS, and OpenDaylight controllers have OpenFlow 1.3 packet, and the relevant information can be seen under the "data" heading as additional information. In the POX controller, source and destination address information can be seen directly in pcap files since OpenFlow 1.0 was used.

To perform the same function, the highest percentage of southbound interface packets compared to all traffic (all interfaces and southbound interface network) occurred on the POX controller at 68.3%. ONOS followed the POX controller with 54.4%, while Ryu ranked third with 42.6%. With a big difference, the OpenDaylight controller covered only 6.3% of all packet numbers numerically, as well as obtaining all the information. Table 1 specifies the percentage information about the packets obtained in this experimental environment for four different controllers.

After interpretation of this scope as the ratio of the total data size to the southbound interface, the highest percentage of data size occured in the POX controller with 76.9%, while the least data size occured in the OpenDaylight controller with 14.7%. The main reason why the percentages differ with the number of packets is that the packet sizes on the southbound interface are different from each other when the controllers are different. In the average packet size, the smallest average packet size in the relevant scenario was 137 bytes in the Ryu controller.

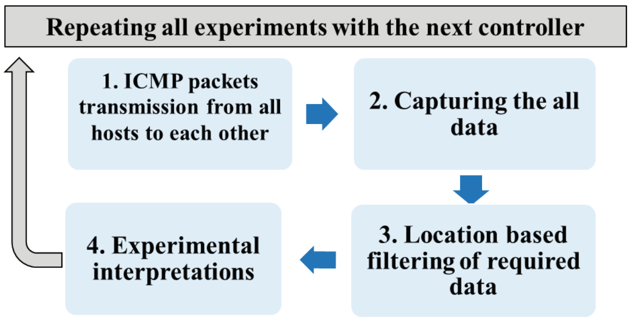
Repeating all experiments with the next controller
1. ICMP packets transmission from all hosts to each other
2. Capturing the all data
3. Location based filtering of required data
4. Experimental interpretations

Figure 3. Routine communication experiment process.

Table 1. Routine communication forensics experiments packet count and size information

Controller	Southbound Interface Packets (%)	Southbound Interface Packet Size (%)
Ryu	42.6	61.6
ONOS	54.4	72
OpenDaylight	6.3	14.7
POX	68.3	76.9

Compared to the packets passing through the southbound interface, the above situations stand out, but the situation differs on the host side. The largest number of packets and the largest size of data were on the interfaces of OpenDaylight, which produced the least number of southbound interface packets for the same process. The least number of packets and data creation on the host side was realized in the scenario with the POX controller. In the OpenDaylight controller's scenario, in which the largest data flowed over the hosts, the data size passing over the hosts was 5.9 times that of the POX controller, where the data size passing over the hosts was the least.

It would also be useful to establish an additional perspective on virtual switches. For this reason, various comparisons were performed, where data flow is expected over all virtual switches. In the experimental topology shown in Figure 2, the switches at a higher level (i.e. virtual switches 2&9) have a marginally higher (approximately 5%) packet number and size than switches located below the topology (i.e. virtual switch 4&5). In addition, the number and size of packets between the switches at the same level are nearly identical.

In this experiment, in which the southbound interface logs are seen to be sufficient, the OpenDaylight controller had the least number of packets and the smallest packet size. In the host-oriented log collection, the least number of packets and the smallest packet size occurred while using the POX controller.

Messaging Content Forensics Process

In addition to the details of time, source and destination, etc., information of the packets, in terms of forensic computing, the issue of whether some information in the packet content can be obtained from the southbound interface can be considered an important step. The process of obtaining the packet content in the transmitted messages and detailed

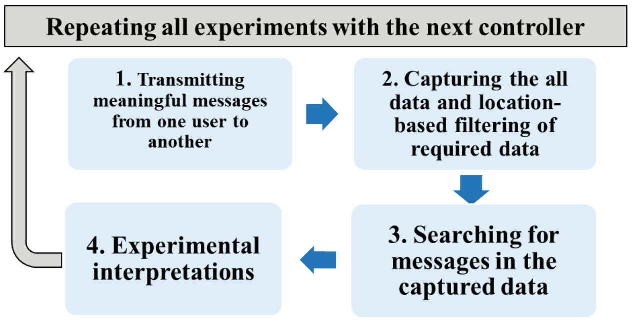
Repeating all experiments with the next controller
1. Transmitting meaningful messages from one user to another
2. Capturing the all data and location-based filtering of required data
3. Searching for messages in the captured data
4. Experimental interpretations

Figure 4. Messaging content forensics experiment process.

examinations, shown in Figure 4, has an important place in terms of network security and management.

Figure 4 shows the main headings of the processes, detailed below, which were performed few times for all four controllers for this experiment, which aims to obtain the message content. In the related experiment, messaging was carried out between two hosts over a port determined with the help of “netcat”. Host1 was put into listening mode over a specified port, and meaningful messages were transmitted to host1 via host16. Accordingly, it was possible to access all transmitted message contents as well as source, destination, and time information in the examination made on packets flowing on virtual switches. A different result in the examinations made on the southbound interface side of this situation were observed.

During the related messaging process in the test environment, the content of any message could not be accessed through the relevant packet in the environment where the Ryu controller was running. In ONOS and POX controllers, the content of the first message about the messaging was seen, while the content of the messages following could not be found in the network flow. Finally, in the test environment where the OpenDaylight controller was running, all messaging information was included on the southbound interface, and it gave the most useful results in terms of reaching the message contents.

Table 2 shows that the percentage of the number of packets compared to the whole scenario is mostly in the southbound interface in the scenario where the OpenDaylight controller is used. In a forensic method, which is aimed to be created by focusing only on the southbound interface, it can be evaluated that the size of 70% of the packets will not provide a great advantage. However, it is a fact that a 30% packet size can also be beneficial for forensic investigations in terms of speed and storage in structures with very large data sizes. On the other hand, considering that the main purpose of a forensic process is primarily to obtain the desired data, the OpenDaylight controller gave the most successful results even though the data size and the number of packets were high in this experimental environment.

Table 2. Message content forensics experiments packet count and size information

Controller	Southbound Interface Packets (%)	Southbound Interface Packet Size (%)
Ryu	57.7	60
ONOS	64.3	84.4
OpenDaylight	77.3	70
POX	49	62

Port Scanning Forensics Process

Nmap is an open-source tool for network scanning and vulnerability detection. Nmap is used in an environment to discover devices and the services they offer, find open ports, and scan for security risks. Nmap is also used by malicious users who want to explore the network before attacks. In the same test environment, all devices on the entire IP block (10.1.1.0/24) were scanned with Nmap, including version information and delay information. Then all the data was captured, and port scan results were examined for all controllers.

Figure 5 shows the experiment in which the port scanning and interpretation process is repeated for all controllers. All four controllers were run in the same topology, and the lowest latency as a result of Nmap scans was determined in the environment where the OpenDaylight controller was running. In the same experiment, all necessary data, such as time, source, and destination, were accessible by the southbound interface in all controller scenarios. In addition to accessing this data, time information were present in all controllers’ southbound interface packets. With the help of Wireshark, filters can be based on many parameters such as MAC address, IP address, and interface. In this way, no detail is overlooked. Although no results were obtained in the scans made with the h1 in IP address over the controllers, Wireshark allowed for detailed searches with the MAC address yielding the relevant IP address information.

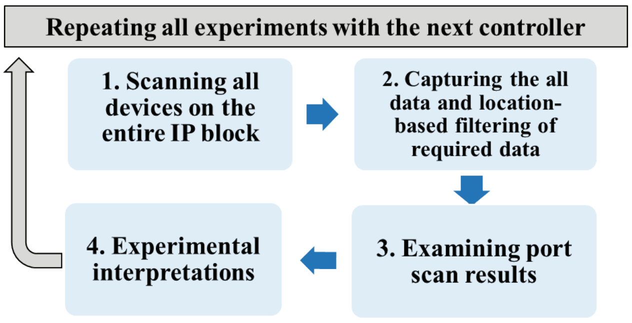
Repeating all experiments with the next controller
1. Scanning all devices on the entire IP block
2. Capturing the all data and location-based filtering of required data
3. Examining port scan results
4. Experimental interpretations

Figure 5. Port scanning forensics experiment process.

Table 3. Port scanning forensic experiments packet count and size information

Controller	Southbound Interface Packets (%)	Southbound Interface Packet Size (%)
Ryu	56.6	80.2
ONOS	40.7	75.3
OpenDaylight	33.25	63
POX	44.6	69.7

As shown in Table 3, the highest number of packets and the percentage of all traffic of the related test was in the southbound interface of the environment where the Ryu controller was running. Also, the least number and percentage of packets occurred in the OpenDaylight controller.

In the test environment using the Ryu controller, the southbound interface packet size was 80% of the packet size of all flowing traffic. In addition, OpenDaylight controller resulted in 63% where the lowest percentage occurred.

Although the scan was for the hosts based on the IP address, for this scenario, the environment in which the average lowest number of packets occurs, especially for the switch interfaces to which the hosts are connected, stood out as the environment where the Ryu controller is located. The difference in the number of packets accumulated on the hosts for the same scan in the environment where Ryu was running was almost three-fold. (157 packets host2 vs. 479 packets host9). The difference in the number of packets is striking. Although the rates close to this situation are seen in the environment where POX and OpenDaylight controllers are running, in the environment where the ONOS controller is running the number of packets passing through the interfaces of the hosts was almost the same and around 525 packets. This result reveals how much of a difference the same network functions can make with only a change of controller.

In order to establish a forensic system with only the data to be obtained by the host, the Ryu controller is more advantageous in terms of the number and size of the packets. In this experiment, where all the necessary information is accessed through the switches and the controller, the OpenDaylight controller can provide an advantage if southbound interface data is desired to be retrieved because OpenDaylight performed this analysis with the lowest number of packets and the lowest size, with the lowest delay against the scans.

Web Server Scan Forensics Process

A scan was performed from host16 for the web server running on host1 using an open-source security vulnerability scanner named "Nikto", which scans for basic security vulnerabilities on the basic HTTP server installed on a host in the topology. Then, all the data was captured and web server scan results were examined. Figure 6 shows the test processes for all controllers, including the web server scan and the examination of the packets resulting from the scan. In terms of the result obtained from the application, it showed the same result in the topology with four controllers, and the scanning results were the same. The scan results in all scenarios had the necessary information on the switches to reveal the scans as well.

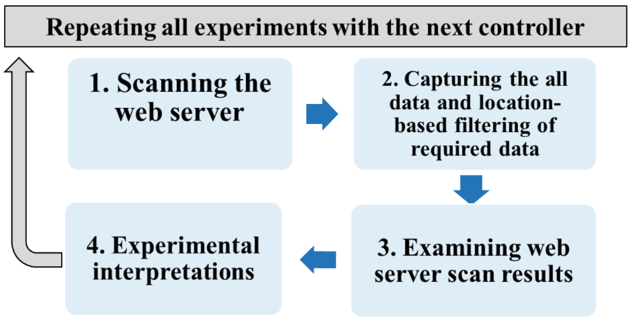
Repeating all experiments with the next controller
1. Scanning the web server
2. Capturing the all data and location-based filtering of required data
3. Examining web server scan results
4. Experimental interpretations

Figure 6. Web server scan forensics experiment process.

In the scenario with the Ryu controller, the southbound interface had only 0.1% of all packets, but no information about the vulnerability scan could be detected. However in the OpenDaylight controller, packets that were considered suspicious due to their structure were detected, but the source and destination addresses could not be reached in the packet detail.

On the other hand, in scenarios with ONOS and POX, it was possible to access necessary information such as source and destination address, suspicious packet detection, and time information. However, the ONOS controller detected this detection with 0.42% of the total number of packets in the scenario in the southbound interface, while the POX controller made this detection with a ratio of 35.4%. This shows that ONOS is the most suitable controller selection in terms of forensics in the relevant scenario.

It was a remarkable result in the experiments that the web server scan took 8-10 times longer in the POX controller than in other experiments. In addition, in the experimental environment where the POX controller is used, the number of packets circulating on the network is more than three times as many as the number of packets in the experimental environment where the Ryu controller is used, and more than twice as many as the experimental environment where the ONOS controller is used.

Denial-of-Service Attacks Forensics Process

TCP SYN Flood Attack

A simulation of denial-of-service (DoS) attacks, was performed with the open-source hping3 tool. In this attack, packets with a fake source IP address were sent to a host in flood, and whether the communication from another host to the related host and peripheral units was interrupted was

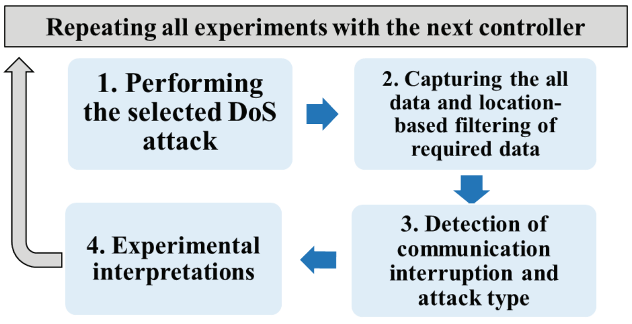
Repeating all experiments with the next controller
1. Performing the selected DoS attack
2. Capturing the all data and location-based filtering of required data
3. Detection of communication interruption and attack type
4. Experimental interpretations

Figure 7. DoS attacks forensics experiment process.

analyzed. The attack is a DoS attack based on the attacker's transmission of SYN packets and no SYN-ACK response. The network flow experienced during the attack was subjected to packet inspection as in all scenarios and tests. Figure 7 shows the general information about the process of DoS attacks carried out with three different methods detailed under three headings in this study. Additionaly, the process of detecting communication interruption is carried out in experimental environments with different controllers.

Scenarios for all controllers were completed and communication was not interrupted in any scenario. The lowest bandwidth, which was 1.5 Gbits/s between the hosts, was effective in not interrupting the communication. Details such as the source, destination address, and time information of the attack were detected together with the fake IP address of the attacker in the southbound interface section in all controller scenarios. The "ACK Flag is not set" warning was seen among the packets and the attack type appeared with this information.

The Ryu controller also managed to gather the necessary information with 0.02% of the total number of packets in its scenario. Ryu also managed to provide the lowest values in this attack in terms of average packet size and total packet size. While the average southbound interface packet size was 119Bytes in the Ryu scenario, this value was the highest at 285Bytes in the POX scenario.

Spanning Tree Protocol (STP) Attack

After this attack, denial-of-service attacks on controllers continued with different methods other than packet flooding, and the tool "yersinia" was again subjected to forensic examination for four controllers in two different scenarios. The first of these attacks was performed by a BPDU (Bridge Protocol Data Unit) attack prepared against the Spanning Tree Protocol (STP) of the Yersinia tool. STP protocol shares port states via BPDU to avoid loops on Layer 2 devices. As soon as the attack started, approximately 25.000 BPDU packets per second were sent to the target for decommissioning.

Although it was the same attack in the same topology used in the OpenDaylight controller, a dramatically decreased number of packets were created on the southbound interface compared to other controllers. Suspicious packet information occurred in all scenarios with fake source addresses, but although many STP packets appeared in all other controllers in the southbound interface, they were not visible in the OpenDaylight controller, providing missing information for detecting the attack.

In the ONOS scenario, the ONOS controller provided the necessary information with 6.8% of the number of packets in all traffic and 11.6% of the total size. In the scenarios where other controllers were used, although the communication was disabled during the attack, there was no failure in the communication in the scenario where the ONOS controller was used.

Cisco Discovery Protocol (CDP) Table Attack

Finally, the CDP table attack was performed, which was prepared by the yersinia tool. CDP is used to share information about other directly connected network equipment that supports it, such as operating system versions and IP addresses. The main purpose of the attack is to reach a large number of neighbor requests in a short time to the victim's device and to run out of resources, halt tasks, and ultimately cause the system to be out of service.

Similar to the STP attack scenario, there was no downtime when the ONOS controller was. However, no trace of attack was found in southbound interface packets in the ONOS controller scenario. While this was also true for OpenDaylight, it was obvious that CDP packets were flooded on the southbound interfaces of Ryu and POX controllers. Meticulous investigations revealed that, in addition to suspicious packets from many different sources, there were packets that were directly called CDP packets that could easily guide the forensic analysis.

Throughout the attacks, communication was disabled in the POX scenario and never came back. In the Ryu scenario, communication was lost and after the attack was terminated, the communication between the hosts was still cut off and partially restored. In the OpenDaylight scenario, communication was lost during the attack but returned after the attack was over.

In summary, Ryu was the only controller that stood out in terms of forensic analysis. No controller fully provided both the criterion of continuing to serve, which is one of the important criteria in terms of forensic analysis, and the supply of valuable data in terms of the accumulation of data in the southbound interface at the same time.

EVALUATION AND RECOMMENDATIONS

While forensic computing is continuing to gain importance and with the difficulty of storing the expanding data size in the environment of the most popular four software-defined network controllers, the experiments done with these four controllers show results that support some examples in the literature with more details regarding forensic computing. The process of testing the durability

Table 4. In the tests applied, whether the relevant scenario could be detected in the southbound interface of each controller (+ Detected, ± Partially Detected, - Undetected)

Detection Status in Scenario (Southbound Interface)	Ryu	ONOS	OpenDaylight	POX
Routine Communication	+	+	+	+
Message Content Reading	-	±	+	±
Port Scanning	+	+	+	+
Web Server Scan	-	+	±	+
TCP SYN Flood	+	+	+	+
STP Attack	+	+	±	+
CDP Attack	+	-	-	+

of the controllers during routine forensic operations and attacks and revealing their differences by examining them in terms of forensic information was done for the first time.

To interpret the whole process, data collection was on the main focus, which is an important early stage of forensic processes. Table 4 shows how the information and attacks in experiments can be obtained in the southbound interface packets of four different controllers directly or by interpreting the obtained packet information. The information in the table shows that in routine communication and attack scenarios, the highest detection success in southbound interface-oriented packets is achieved in experiments using the POX controller. The Ryu controller did not achieve detection using southbound interface packets in two of the scenarios, while OpenDaylight and ONOS were unable to detect CDP attacks. In experiments, routine communication could be detected on the southbound interface in every scenario using each of the different controllers, whereas the CDP attack was the most difficult experiment to detect in southbound interface packets.

Beyond the most obvious parameters of forensic processes, such as attack detection, source, and target, it is also important that these detections can be made as quickly as possible and with as little data as possible. Although experiments were carried out on the same topology and under the same conditions, there were significant differences in the average packet size as well as the number of packets and packet sizes that differ from each other in the southbound interface, as shown in Figure 8 & Figure 9.

It was observed that the same situation even in the number of packets and packet size accumulated on the switches, especially in the port scanning scenario. Another remarkable result is that the completion time of the scenario with the POX controller in the web server vulnerability scanning experiment is 8-10 times longer than the other experiments.

The fact that critical information cannot be seen on the controller, which is seen as the most successful in other attack types in some attacks, is proof that it would not be right to prefer a single controller for routine operations and attacks. The delay times were also different when the controllers were changed in routine communication, and lower delay values were obtained in different controllers in the scans. For example, when there was no operation on the network, the Ryu controller provided the lowest latency, while the lowest latency in Nmap scanning was provided in the environment where the OpenDaylight controller was running.

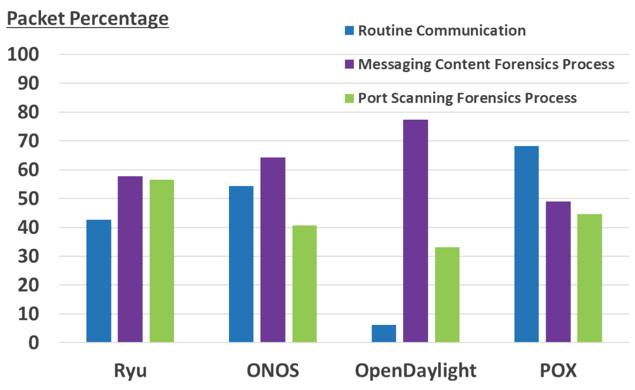

Figure 8. Southbound interface packets' percentages in all test traffic.

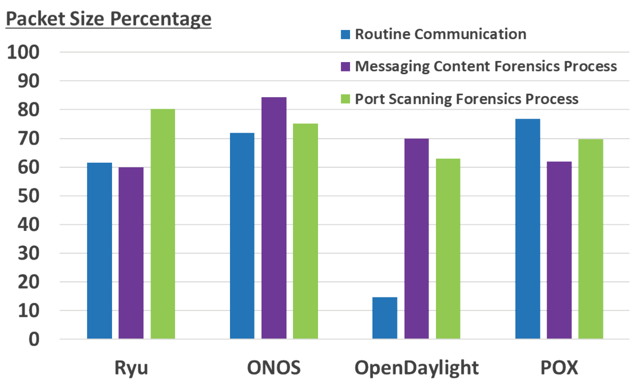

Figure 9. Southbound interface packet sizes' percentages in all test traffic.

While there was a difference of up to 3 times in one controller in the same experiment between the number of packets of the switch interfaces to which the hosts are directly connected, there was almost no difference in the same experiment, while another controller was operating.

As shown in this study, which also focuses on packet size and number, the small number of packets or the small size of the packet does not mean much in terms of forensic analysis. For example, the necessary determinations could not be made in the Ryu controller scenario with the lowest packet size and number in the web server scan. Despite this situation, it is an important part of the forensic method in software-defined networks.

Table 5 shows an observational comparison where the most valuable results were obtained. It focuses on the controller with the lowest packet number, size, and percentage, the controllers which have the communication and attack information, time information, source-destination information, and where the system is disabled in the attack experiments and the southbound interface analysis. There are serious differences between the results from the experiments, and these conclusions can be a useful guide for researchers who will work in this context to choose the controller that best fits their usage ideas. Below, is a brief discussion of the results presented in Table 5.

- The lowest percentages of packet count and size obtained from the southbound interface are in the OpenDaylight controller, and Ryu follows the OpenDaylight controller.
- Obtaining time information is clearly critical in forensic computing. The information about the experiments performed on the packets obtained from the southbound interface can be accessed a large majority of the time. The inaccessibility of time information as a parameter related to the inability to detect the experiment. The results show that there will be no lack of time information in packets from the southbound interface. Reaching the source and target packets can be considered as a parameter depending on the experiment not being detected and the time information is not being reached. As an exception to all experiments, in the scenario where the OpenDaylight controller was used for web server scanning, the source and destination addresses could not be reached in the southbound interface packets.
- It is also important that the system from which the data will be retrieved is not out of service as a result of the attacks. Although the data is obtained not directly from the system itself but from the data flowing on the network, the deactivation of a centralized system such as a controller will lead to incomplete data. Some controllers went out of service shortly after some denial-of-service attacks started. The out-of-service controllers started to serve again after the attack ended, and in some of them, the communication was partially or completely restored. This shows that availability is a priority and an important issue for forensic investigations.
- Overall, controllers that produced the best results in experiments considering all the forensic features are mentioned. The OpenDaylight controller was the most successful in three of the seven different experiments, the ONOS controller in two experiments, and the Ryu controller in one experiment. In the last experiment, no controller was successful in terms of forensic computing. According to the focus of new studies, it will be possible to determine which controller will be more useful with the help of this table as it summarizes attacks carried out for the first time with different controllers.

Another issue that can be addressed beyond examining the results of all these experiments is why there are differences in these results. When these reasons are handled in

Table 5. The general result of applied experiments and attacks

Detection Status in Scenario/ Southbound Interface	Lowest Packet Count Percentage	Lowest Packet Size Percentage	Information About the Scenario	Time Information Obtainable	Source-Target Information Obtainable	Non-disabled Controllers	Best Controller for Southbound Forensics
Routine Communication	OpenDaylight	OpenDaylight	Ryu, ONOS, OpenDaylight, POX	Ryu, ONOS, OpenDaylight, POX	Ryu, ONOS, OpenDaylight, POX	N/A	OpenDaylight
Message Content Reading	POX	Ryu	OpenDaylight, ONOS (partial), POX (partial)	OpenDaylight, ONOS (partial), POX (partial)	OpenDaylight, ONOS (partial), POX (partial)	N/A	OpenDaylight
Port Scanning	OpenDaylight	OpenDaylight	Ryu, ONOS, OpenDaylight, POX	Ryu, ONOS, OpenDaylight, POX	Ryu, ONOS, OpenDaylight, POX	N/A	OpenDaylight
Web Server Scan	Ryu	Ryu	ONOS, POX, OpenDaylight	ONOS, POX, OpenDaylight	ONOS, POX	N/A	ONOS
TCP SYN Flood	Ryu	Ryu	Ryu, ONOS, OpenDaylight, POX	Ryu, ONOS, OpenDaylight, POX	Ryu, ONOS, OpenDaylight, POX	Ryu, ONOS, OpenDaylight, POX	Ryu
STP Attack	OpenDaylight	OpenDaylight	Ryu, ONOS, POX, OpenDaylight (partial)	Ryu, ONOS, OpenDaylight (partial), POX	Ryu, ONOS, OpenDaylight, POX	ONOS	ONOS
CDP Attack	OpenDaylight	OpenDaylight	Ryu, POX	Ryu, POX	Ryu, POX	ONOS	-

the focus of this study, even though they are in the same topology in cases such as detecting, packet size, number of packets, and not being out of service, the big differences experienced only with the controller change can also create new study areas. To determine the reasons for the differences that come with the controller change, it will be useful to focus on the development of the controller, which is the only parameter that changes in the processes. In addition to OpenDaylight and ONOS controllers using the Java infrastructure, the development process of the related software is at the forefront of the differences that occur in the scenarios where the Ryu and POX controllers developed with the python language are used. This situation may cause differentiation of the logs to be obtained from the software directly on the controllers themselves. Even though the data obtained from the network environment contains major differences, this can be considered a threat in terms of the development of forensic processes in software-defined networks. In addition, although the communication between the control layer and the data layer is realized through OpenFlow, additional information, routing method differences, and other added rules specific to the controller also differentiate the forensic data that can be detected beyond the packet size and number.

The results of this study, which are shown for the first time in the literature, are also likely to be guiding in the field of software-defined network forensics. For example, examining the topology type change in terms of software-defined network forensics and conducting studies on the change of forensic view capabilities as a result of the change of OpenFlow versions while using the same controller will increase forensic knowledge in the literature. In addition, it is possible to make improvements by transferring the good aspects of controllers in terms of forensic computing to other weak controllers in modules. Considering the differences in the experiments, it is extremely important to progress starting from the controller development stage and to consider a minimum forensic meeting expectation for all controllers as a standard to strengthen forensics in software-defined networks. This will prevent the differences in forensic detection shown in Tables 4 and 5, as well as the differences in the number and size of packets in Figures 8 and 9.

CONCLUSION

Network forensics has great potential in the SDN realm, but to fully exploit this potential, it is necessary to know the limitations and capabilities of SDN components in forensics. The controller is the most critical component in SDN as it maintains a central view of the entire network and makes forwarding decisions. In this study, for the first time in the literature, seven different experimental attack scenarios were applied with four different controllers, and the packets captured over the data layer and southbound interface were examined in detail. In doing so, the important issues in terms of forensics, such as packet size, source-target detection, attack and attacker information, and the effects of denial-of-service attacks, took into account.

This study focuses on the ability to obtain the most meaningful communication data from the smallest possible size and number of packets. The speed of obtaining this data is also extremely important for live attack detection and affects the integrity of the entire network in communication. The results of experiments that were done with these points in mind show that different controllers show advantages and disadvantages in terms of event detection, packet size or service delivery in different tests, and differences between controllers in southbound interface-oriented forensic processes. The ONOS controller, one of the four different controllers, was used for the first time in this study, where data such as packet size, number and percentage, and information about it are shared. Another novel contribution of this study is an examination of the kind of information that can be accessed during routine operations and during cyber attacks and where this information can be obtained from in the network. This study also shows the new result about the service status of the four controllers during cyber attacks and reveals that the controllers also exhibit differences in terms of service continuity and quality. It would be beneficial to ensure that controllers meet certain standards to be dependable in forensic investigations while also being adaptable according to the laws and regulations of different states.

The experiments show that there may be information deficiencies in some controllers and scenarios if the data is taken, collected, and stored only from the southbound interface, ignoring other data sources. However, as a common result, focusing on the southbound interface in terms of forensics in SDN will be the right choice due to its advantages, such as service delivery, detection, and log size. This will enable the necessary inspections to be carried out from a single point, with high quality, low overhead, and in a vendor-independent manner. On the other hand, contrary to the claims in certain previous studies, it is not correct to accept any single controller as the most successful option in terms of forensics.

It is essential to examine the reasons for these differences in terms of forensic information and to design an SDN controller that provides optimum benefit in terms of forensic information. The implementation of a forensic management system as a general protocol will be of great benefit, considering the rapidly increasing use of SDN infrastructures. In this context, it is planned that these studies will continue, taking into account the specific situations of the application layer.

AUTHORSHIP CONTRIBUTIONS

Altuğ Çil: Conceptualization, Methodology, Resources Software, Formal analysis, Investigation, Data curation, Writing – Original Draft, Visualization. **Mehmet Demirci:** Validation, Writing – Review & Editing, Supervision, Project administration.

DATA AVAILABILITY STATEMENT

The authors confirm that the data that supports the findings of this study are available within the article. Raw data that support the finding of this study are available from the corresponding author, upon reasonable request.

CONFLICT OF INTEREST

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

ETHICS

There are no ethical issues with the publication of this manuscript.

REFERENCES

[1] Abdelaziz A, Fong AT, Gani A, Garba U, Khan S, Akhunzada A, et al. Distributed controller clustering in software defined networks. PLoS One 2017;12. [CrossRef]

[2] Rawat DB, Reddy SR. Software defined networking architecture, security and energy efficiency: A survey. IEEE Commun Surv Tutorials 2017;19:325−346. [CrossRef]

[3] Van Adrichem NLM, Doerr C, Kuipers FA. OpenNetMon: Network monitoring in OpenFlow software-defined networks. IEEE/IFIP NOMS 2014 - IEEE/IFIP Netw. Oper. Manag. Symp. Manag. a Softw. Defin. World, IEEE Computer Society; 2014. [CrossRef]

[4] ONF White Paper, Software-Defined Networking: The New Norm for Networks, OPEN NETWORKING FOUNDATION, 2012. Available at: https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/ Accessed on Sep 5, 2022.

[5] Yan Z, Zhang P, Vasilakos A V. A security and trust framework for virtualized networks and software-defined networking. Secur Commun Networks 2016;9:3059−3069. [CrossRef]

[6] Chourishi D, Miri A, Milic M, Ismaeel S. Role-based multiple controllers for load balancing and security in SDN. 2015 IEEE Canada Int. Humanit. Technol. Conf. IHTC 2015, Institute of Electrical and Electronics Engineers Inc.; 2015. [CrossRef]

[7] Al-Najjar A, Layeghy S, Portmann M. Pushing SDN to the end-host, network load balancing using OpenFlow. 2016 IEEE Int. Conf. Pervasive Comput. Commun. Work. PerCom Work. 2016, Institute of Electrical and Electronics Engineers Inc.; 2016. [CrossRef]

[8] Kreutz D, Ramos FMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: A comprehensive survey. Proc IEEE 2015;103:14−76. [CrossRef]

[9] Shin S, Xu L, Hong S, Gu G. Enhancing Network Security through Software Defined Networking (SDN). 2016 25th Int. Conf. Comput. Commun. Networks, ICCCN 2016, Institute of Electrical and Electronics Engineers Inc.; 2016. [CrossRef]

[10] Nguyen T, Yoo M. Analysis of link discovery service attacks in SDN controller. 2017 International Conference on Information Networking (ICOIN);2017. p. 259−261.

[11] Chi PW, Kuo CT, Guo JW, Lei CL. How to detect a compromised SDN switch. 1st IEEE Conf. Netw. Softwarization Software-Defined Infrastructures Networks, Clouds, IoT Serv. NETSOFT 2015, Institute of Electrical and Electronics Engineers Inc.; 2015.

[12] Bawany NZ, Shamsi JA, Salah K. DDoS attack detection and mitigation using SDN: Methods, practices, and solutions. Arab J Sci Eng 2017;42:425−441. [CrossRef]

[13] Hong S, Xu L, Wang H, Gu G. Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures, Internet Society; 2015. [CrossRef]

[14] Shrivastava G. Network forensics: Methodical literature review. 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom);2016. p. 2203−2208.

[15] Shrivastava G, Sharma K, Kumari R. Network forensics: Today and tomorrow. 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom); 2016. p. 2234−2238.

[16] Ring M, Wunderlich S, Scheuring D, Landes D, Hotho A. A survey of network-based intrusion detection data sets. Comput Secur 2019;86:147−167. [CrossRef]

[17] DFRWS Technical Report, A Road Map for Digital Forensic Research, The Digital Forensic Research Conference, DFRWS 2001, Page 27. https://dfrws.org/wp-content/uploads/2019/06/2001_USA_a_road_map_for_digital_forensic_research.pdf (Accessed on Jul 2, 2022.

[18] Pandya MK, Homayoun S, Dehghantanha A. Forensics Investigation of OpenFlow-Based SDN Platforms. In: Dehghantanha A., Conti M., Dargahi T. (Eds) Cyber Threat Intelligence 2018. Advances in Information Security, vol 70. Springer, Cham.;2018. p. 281−297. [CrossRef]

[19] Akbari I, Tahoun E, Salahuddin MA, Limam N, Boutaba R. ATMoS: Autonomous Threat Mitigation in SDN using Reinforcement Learning. NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium; 2020. [CrossRef]

[20] Waseem Q, Alshamrani SS, Nisar K, Din WISW, Alghamdi AS. Future technology: Software-defined network (SDN) forensic. Symmetry (Basel) 2021;13. [CrossRef]

[21] Revathi M, Ramalingam VV, Amutha B. A Machine Learning Based Detection and Mitigation of the DDOS Attack by Using SDN Controller Framework. Wirel Pers Commun 2021;127:2417−2441. [CrossRef]

[22] Muraga, W H, Seman K, Marhusin M F. A POX Controller Module to Collect Web Traffic Statistics in SDN Environment. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 2017;10:2105−2110.

[23] Bouba Mahamat S, Çeken C. Anomaly detection in software-defined networking using machine learning. Düzce Üniversitesi Bilim ve Teknoloji Dergisi 2019;7:748−756. [CrossRef]

[24] Balarezo JF, Wang S, Chavez KG, Al-Hourani A, Fu J, Sithamparanathan K. Low-rate TCP DDoS Attack Model in the Southbound Channel of Software Defined Networks. 2020 14th Int. Conf. Signal Process. Commun. Syst. ICSPCS 2020 - Proc., Institute of Electrical and Electronics Engineers Inc.; 2020. [CrossRef]

[25] Khan S, Gani A, Wahab AWA, Abdelaziz A, Ko K, Khan MK, et al. Software-defined network forensics: Motivation, potential locations, requirements, and challenges. IEEE Netw 2016;30:6−13. [CrossRef]

[26] Khan S, Gani A, Wahab AWA, Abdelaziz A, Bagiwa MA. FML: A novel forensics management layer for software defined networks. Proc. 2016 6th Int. Conf. - Cloud Syst. Big Data Eng. Conflu. 2016, Institute of Electrical and Electronics Engineers Inc.; 2016, p. 619−623. [CrossRef]

[27] Mugitama SA, Dwi N, Cahyani W, Sukarno P. An Evidence-Based Technical Process for OpenFlow-Based SDN Forensics; An Evidence-Based Technical Process for OpenFlow-Based SDN Forensics 2020;1−6. [CrossRef]

[28] Spiekermann D, Keller J, Eggendorfer T. Network forensic investigation in OpenFlow networks with ForCon. DFRWS 2017 EU - Proc. 4th Annu. DFRWS Eur., Digital Forensic Research Workshop; 2017, p. S66−S74. [CrossRef]

[29] Achleitner S, La Porta T, Jaeger T, McDaniel P. Adversarial network forensics in software defined networking. SOSR 2017 - Proc. 2017 Symp. SDN Res., Association for Computing Machinery, Inc; 2017, p. 8−20. [CrossRef]

[30] Zhang S, Meng X, Wang L. SDNForensics: A Comprehensive Forensics Framework for Software Defined Network; 2016.

[31] Wang H, Yang G, Chinprutthiwong P, Xu L, Zhang Y, Gu G. Towards fine-grained network security forensics and diagnosis in the SDN era. Proc. ACM Conf. Comput. Commun. Secur., Association for Computing Machinery; 2018, p. 3−16. [CrossRef]

[32] Mininet Project Contributors. Mininet, v2.3.0 (version 2.3.0), An Instant Virtual Network on your laptop, 2021. Available at: http://mininet.org/ Accessed on Jul 2, 2022.

[33] Ryu SDN Framework Community. Ryu Controller, v4.34 (version 4.34), 2017. Available at: https://Ryu-sdn.org/, Accessed on Jul 2, 2022.

[34] Open Networking Foundation. ONOS Controller, v2.5.1 (version 2.5.1), 2021. Available at: https://opennetworking.org/ONOS/ Accessed on Jul 2, 2022.

[35] OpenDaylight Project The Linux Foundation. Opendaygliht Controller, Beryllium release, 2016. Available at: https://www.OpenDaylight.org/ Accessed on Jul 2, 2022.

[36] McCauley et al. POX controller. v0.7.0 (version 0.7.0), 2015. Available at: https://noxrepo.github.io/POX-doc/html/ Accessed on Jul 2, 2022.

[37] Ubuntu, Desktop version 20.04&14.04, 2021. Available at: https://ubuntu.com/ Accessed on Jul 2, 2022.

[38] Oracle, VirtualBox v6.1 (version 6.1), 2021. https://www.virtualbox.org/ Accessed on Jul 2, 2022.

[39] Linux Foundation Collaborative Project, Openvswitch v2.13.1, (version 2.13.1), 2016. Available at: https://www.openvswitch.org/ Accessed on Jul 2, 2022.

[40] Wireshark, v3.2.3 (version 3.2.3), 2021. Available at: https://www.wireshark.org/ Accessed on Jul 2, 2022.

[41] Yersinia, v0.8.2, (version 0.8.2), 2021. Available at: https://www.kali.org/tools/yersinia/ Accessed on Jul 2, 2022.

[42] Nikto, v2.1.5 (version 2.1.5), 2012. Available at: https://tools.kali.org/information-gathering/nikto Accessed on Jul 2, 2022.

[43] Iperf, v2.0.13 (version 2.0.13), 2019. Available at: https://iperf.fr/ Accessed on Jul 2, 2022.

[44] Nmap, v7.80 (version 7.80), 2019. Available at: https://nmap.org/ Accessed on Jul 2, 2022.

[45] Hping3, v3.0.0, (version 3.0.0-alpha-2), 2021. Available at: https://tools.kali.org/information-gathering/hping3 Accessed on Jul 2, 2022.