

SOFTWARE DEFINED NETWORK (SDN) BASED PACKET FORWARDING VERIFICATION

Field of the Invention

This disclosure relates to a method and system for verifying the veracity of data packet being forwarded in a network. Particularly, this disclosure relates to a method and system for verifying forwarding path inconsistency or malicious tampering of a data packet in a network.

Summary of the Prior Art

Software-Defined Networking (SDN) is a new network system. Compared with traditional network system, SDN physically separates the control layer and data layer of a network. The control layer is responsible for computing routes for packets, while the data layer is responsible for forwarding packets. In a traditional network, distributed algorithms are adopted to compute routes, and a router or switch comprises both control layer and data layer. This makes network configuration and management very complex. In contrast, SDN separates these two layers physically. Particularly, the control layer is implemented in a centralized server named SDN controller while the data layer remains at switches (the term router is no longer used in SDN). SDN makes network configuration easy. A network administrator only needs to configure SDN controller to manage the network.

Figure 1 shows the architecture of SDN 100. Essentially, SDN has three layers, namely, data layer 130, control layer 120 and application layer 110. The data layer 130 includes devices for forwarding packets, i.e. switches. The control layer 120 provides interfaces to control data layer. The application layer 110 is responsible for high-level logic. The interface between control layer and data layer is called southbound interface 150, while the interface between control layer and application layer is called northbound interface 140.

Through the southbound interface 150 of SDN, control layer 120 controls the forwarding in data layer 130, and data layer 130 reports its status to control layer 120. There are several southbound interface 150 protocols such as OpenFlow, ForCES, PCE-P, etc. Among them, OpenFlow is the most mature and most used.

Figure 2 illustrates a process based on the OpenFlow protocol. OpenFlow standard regulates that a switch should have one or more flow tables as well as one group table. Each flow table can have multiple flow table entries. An entry defines a specific network flow and

describes actions on the flow. A network flow is a group of packets sharing one or more packet fields or sharing some bits. For example, an entry may define the packets with a given destination IP (say 10.0.1.2) as a flow. The actions may include any of the following: outputting packets to a given output port, outputting packets to a flow table, outputting packets to a group table, sending packets to controller, modifying the header of each packet in the flow, and recording the number of processed packets in the flow. A group table includes multiple action buckets. The same as a flow table entry, a bucket defines one or more actions on packets. There are multiple types of group tables: all, select, and fast failover. The indirect type is very similar as flow table entry, but it can apply to multiple flow table entries. The all type will execute the actions in all buckets. Based on the configured weights of all buckets, the select type will randomly select a bucket's actions to execute. The probability of selecting a bucket is in priority to the bucket's weight. The fast failover type will execute the actions of the first executable bucket. Each group table or even each bucket in the group table can count a quantity of data packets or bytes forwarded by the group table or bucket. Each flow entry may be attached to a group table. After a data packet enters a switch, the data packet is matched by a flow entry in the first flow table. After a corresponding action is performed, the flow entry may directly output the data packet to a group table, a switch port, or a subsequent flow table. If the data packet is output to the group table, the data packet is forwarded according to logic of the group table. If the data packet is output to the output port, the switch directly outputs the data packet. If the data packet is output to the subsequent flow table, the data packet continues to be forwarded according to the original forwarding logic.

In a current network, misrouting attack and data tempering attack are two kinds of attacks faced by network administrators. These attacks are all related to the trustworthiness of data routing. Misrouting attack can cause a router/switch to forward a data packet to an incorrect port. In other words, data packets are not forwarded in an expected path when misrouting attack occurs. Data tempering attack can cause a router/switch to temper the data it forwards. In other words, the original data packets can be tempered when the data packets are forwarded in the expected path. These attacks bring a serious security risk to a user such as privacy data leakage. An attacker may forward a network flow to a machine controlled by the attacker, so as to listen to user traffic, or directly tamper with a user's data packet, so as to inject malicious code.

Hence, those skilled in the art are striving to provide a method and system of periodically verifying the veracity of flow of data packet forwarding in a network.

Summary of the Invention

The above and other problems are solved and an advance in the state of art is made by systems and methods provided by embodiments in accordance with the disclosure. A first advantage of embodiments of systems and methods in accordance with the disclosure is that the forwarding path or integrity of the data packet is being verified periodically. This can effectively reduce performance problems such as network unreachability and network delay increase due to a network fault. A second advantage of embodiments of systems and methods in accordance with the disclosure is that by detecting whether a data packet is forwarded along a correct forwarding path, the systems and method in accordance with the disclosure effectively reduces risks that a user traffic is listened to and stolen, thereby protecting a user's privacy. In addition, by detecting whether a data packet is maliciously tampered with in the network, the systems and method in accordance with the disclosure can effectively prevent some man-in-the-middle attacks and propagation of some viruses, which can provide better security assurance for the network. A third advantage of embodiments of systems and methods in accordance with the disclosure is that the systems and methods impose little impact on the network. This is because the systems and methods in accordance with the disclosure are dependent on only lightweight operations, for example, adding a flow label to a data packet, removing a flow label, counting a quantity of data packets, and sending a data packet to a control plane/layer. No encryption and decryption operations are required.

A first aspect of the disclosure describes a system for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller, a first switch, a second switch, and a third switch. The SDN controller is configured to: generate and transmit orders to the first switch and the second switch to return a first detection data and a second detection data of a network flow; receive the first detection data and the second detection data, the first detection data being a number of data packets from the network flow with a first label reported by the first and second switches, and the second detection data being a content associated to the data packets from the network flow with the first label reported by the first and second switches; perform an analysis on the first and second detection data; and raise an action ticket if the analysis on the first and second detection data fails.

According to an embodiment of the first aspect, the first switch and the second switch are two switches in an expected forwarding path of the network flow, and the second switch

is a downstream switch with respect to the first switch. According to an embodiment of this embodiment, the first switch is an ingress switch and the second switch is an egress switch.

According to an embodiment of the first aspect, the SDN controller is specially configured to: determine whether the number of data packets received from the first switch is equal to the number of data packets received from the second switch; and determine whether the content associated to the data packets received from the first and second switches are the same.

According to an embodiment of the first aspect, the SDN controller is configured to: generate and transmit orders to the third switch, the third switch being an intermediate switch randomly selected between the first and second switches in the forwarding path of the network flow.

According to an embodiment of the first aspect, the SDN controller is configured to: compare the number of data packets received from each of the first, second and third switches are the same; and compare the content associated to the data packets received from each of the first, second and third switches are the same.

According to an embodiment of the first aspect, the first switch is configured to: update flow tables and group tables according to the orders received from the SDN controller; receive and sample data packets from the selected network flow; tag sampled data packets with the first label; and forward the data packets to the next hop and a content associated to the data packets with the first label to the SDN controller and report the number of sampled data packets to the SDN controller.

According to an embodiment of the first aspect, the first switch is further configured to: tag non-sampled data packets with the second label.

According to an embodiment of the first aspect, the first switch is further configured to: compute a message digest of all the sampled data packets to determine the content associated to the data packets with the first label.

According to an embodiment of the first aspect, the second switch is configured to: update flow tables and group tables according to the orders received from the SDN controller; receive data packets from the selected network flow; determine whether the data packets have been tagged with the first label; in response to determining the data packets having been tagged with the first label, remove the first label from the data packets; and

forward the number of data packets with the first label and the content associated to the data packets with the first label to the SDN controller.

According to an embodiment of the first aspect, the second switch is further configured to: remove the second label from the data packets in response to determining the data packets not having been tagged with the first label.

According to an embodiment of the first aspect, the second switch is further configured to: compute a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

According to an embodiment of the first aspect, the third switch is configured to: update flow tables and group tables according to the orders received from the SDN controller; receive data packets from the selected network flow; count the number of data packets with the first label; and forward the number of data packets with the first label to the SDN controller.

According to an embodiment of the first aspect, the third switch is configured to: forward the content associated to the data packets with the first label to the SDN controller.

According to an embodiment of the first aspect, the third switch is further configured to: compute a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

A second aspect of the disclosure describes a method for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) having a SDN controller and multiple switches, the method comprising: generating and transmitting orders to a first switch and a second switch to return a first detection data and a second detection data of a network flow; receiving the first detection data and the second detection data, the first detection data being the number data packets from the network flow with a first label reported by the first and second switches, and the second detection data being a content associated to the data packets from the network flow with the first label reported by the first and second switches; performing an analysis on the first and second detection data; and raising an action ticket if the analysis on the first and second detection data fails.

According to an embodiment of the second aspect, the step of performing an analysis on the first and second detection data comprises: determining whether the number of data packets received from the first switch is equal to the number of data packets received

from the second switch; and determining whether the contents associated to data packets received from the first and second switches are the same.

According to an embodiment of the second aspect, the method further comprises generating and transmitting orders to a third switch, the third switch being an intermediate switch randomly selected between the first and second switches in the forwarding path of the network flow.

According to an embodiment of the second aspect, the step of performing an analysis on the first and second detection data comprises: determining whether the number of data packets received from each of the first, second and third switches are the same; and determining whether the content associated to the data packets received from each of the first and second switches are the same.

According to an embodiment of the second aspect, the step of performing an analysis on the first and second detection data comprises: determining whether the number of data packets received from each of the first, second and third switches are the same; and determining whether the content associated to the data packets received from each of the first, second and third switches are the same.

A third aspect of the disclosure describes a controller for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN), the SDN controller having a non-transitory memory, a processor and instructions stored on the memory executable by the processor to: generate and transmit orders to a first switch and a second switch to return a first detection data and a second detection data of a network flow; receive the first detection data and the second detection data, the first detection data being the number data packets from the network flow with a first label reported by the first and second switches, and the second detection data being a content associated to the data packets from the network flow with the first label reported by the first and second switches; perform an analysis on the first and second detection data; and raise an action ticket if the analysis on the first and second detection data fails.

According to an embodiment of the third aspect, the instruction to perform an analysis on the first and second detection data comprises instructions stored on the memory executable by the processor to: determine whether the number of data packets received from the first switch is equal to the number of data packets received from the second switch; and determine whether the contents associated to data packets received from the first and second switches are the same.

According to an embodiment of the third aspect, the instruction further comprises instructions stored on the memory executable by the processor to: generate and transmit orders to a third switch, the third switch being an intermediate switch randomly selected between the first and second switches in the forwarding path of the network flow.

According to an embodiment of the third aspect, the instruction to perform an analysis on the first and second detection data comprises instructions stored on the memory executable by the processor to: determine whether the number of data packets received from each of the first, second and third switches are the same; and determine whether the content associated to the data packets received from each of the first and second switches are the same.

According to an embodiment of the third aspect, the instruction to perform an analysis on the first and second detection data comprises instructions stored on the memory executable by the processor to: determine whether the number of data packets received from each of the first, second and third switches are the same; and determine whether the content associated to the data packets received from each of the first, second and third switches are the same.

A fourth aspect of the disclosure describes a method for a switch in verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller and the switch. The method comprises updating flow tables and group tables according to orders received from the SDN controller; receiving and sampling data packets from the selected network flow; tagging sampled data packets with the first label; and forwarding the data packets to the next hop and a content associated to the data packets with the first label to the SDN controller and report the number of sampled data packets to the SDN controller.

According to an embodiment of the fourth aspect, the switch is in an expected forwarding path of the network flow.

According to an embodiment of the fourth aspect, the method further comprises tagging non-sampled data packets with the second label.

According to an embodiment of the fourth aspect, the method further comprises computing a message digest of all the sampled data packets to determine the content associated to the data packets with the first label.

A fifth aspect of the disclosure describes a switch for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller and the switch; the switch having a non-transitory memory, a processor and instructions stored on the memory executable by the processor to: update flow tables and group tables according to orders received from the SDN controller; receive and sample data packets from the selected network flow; tag sampled data packets with the first label; and forward the data packets to the next hop and a content associated to the data packets with the first label to the SDN controller and report the number of sampled data packets to the SDN controller.

According to an embodiment of the fifth aspect, the switch is in an expected forwarding path of the network flow.

According to an embodiment of the fifth aspect, the instruction further comprises instructions stored on the memory executable by the processor to: tag non-sampled data packets with the second label.

According to an embodiment of the fifth aspect, the instruction further comprises instructions stored on the memory executable by the processor to: compute a message digest of all the sampled data packets to determine the content associated to the data packets with the first label.

A sixth aspect of the disclosure describes a method for a first switch and a second switch in verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller, the first switch and the second switch. The method comprising: updating flow tables and group tables according to the orders received from the SDN controller; receiving data packets from the selected network flow; determining whether the data packets have been tagged with a first label; in response to determining the data packets having been tagged with the first label, removing the first label from the data packets; and forwarding the number of data packets with the first label and the content associated to the data packets with the first label to the SDN controller.

According to an embodiment of the sixth aspect, the first switch and the second switch are two switches in an expected forwarding path of the network flow, and the second switch is a downstream switch with respect to the first switch.

According to an embodiment of the sixth aspect, the method further comprising: removing, by the second switch, a second label from the data packets in response to determining the data packets not having been tagged with the first label; wherein the second label is marked by the first switch.

According to an embodiment of the sixth aspect, the method further comprising: computing a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

A seventh aspect of the disclosure describes a second switch for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller a first switch and the second switch; the second switch having a non-transitory memory, a processor and instructions stored on the memory executable by the processor to: update flow tables and group tables according to the orders received from the SDN controller; receive data packets from the selected network flow; determine whether the data packets have been tagged with a first label; in response to determining the data packets having been tagged with the first label, removing the first label from the data packets; and forward the number of data packets with the first label and the content associated to the data packets with the first label to the SDN controller.

According to an embodiment of the seventh aspect, the instruction further comprises instructions stored on the memory executable by the processor to: remove a second label from the data packets in response to determining the data packets not having been tagged with the first label; wherein the second label is marked by the first switch.

According to an embodiment of the seventh aspect, the instruction further comprises instructions stored on the memory executable by the processor to: compute a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

An eighth aspect of the disclosure describes a method for a third switch in verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller, a first switch, a second switch and the third switch. The method comprising: receiving data packets from a selected network flow; counting the number of data packets with a first label; and forwarding the number of data packets with the first label to the SDN controller.

According to an embodiment of the eighth aspect, the method further comprising: forwarding a content associated to the data packets with the first label to the SDN controller.

According to an embodiment of the eighth aspect, the method further comprising: computing a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

A ninth aspect of the disclosure describes a third switch for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller, a first switch a second switch and the third switch; wherein the first and the second switch are two switches in an expected forwarding path of the network flow, the first switch is an ingress switch and the second switch is an egress switch; the third switch is located between the first switch and the second switch; the third switch having a non-transitory memory, a processor and instructions stored on the memory executable by the processor to: update flow tables and group tables according to orders received from the SDN controller; receive and sample data packets from the selected network flow; tag sampled data packets with the first label; and forward the data packets to the next hop and a content associated to the data packets with the first label to the SDN controller and report the number of sampled data packets to the SDN controller.

According to an embodiment of the ninth aspect, the instruction further comprises instructions stored on the memory executable by the processor to: forward a content associated to the data packets with the first label to the SDN controller.

According to an embodiment of the ninth aspect, the instruction further comprises instructions stored on the memory executable by the processor to: compute a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

Brief Description of the Drawings

The above advantages and features in accordance with this disclosure are described in the following detailed description and are shown in the following drawings:

Figure 1 illustrating the architecture of the Software-Defined Networking (SDN);

Figure 2 illustrating an example of a process based on the OpenFlow protocol;

Figure 3 illustrating a system for verifying a veracity of data packets being forwarded in a Software Defined Network in accordance with an embodiment of this disclosure;

Figure 4 illustrating an exemplary processing system representing the processing systems in an SDN controller that execute instructions to perform the processes in accordance with embodiments of this disclosure;

Figure 5 illustrating a process performed by the SDN controller to verify a veracity of data packets being forwarded in a Software Defined Network devices in accordance with an embodiment of this disclosure;

Figure 6 illustrating a process performed by the first switch after receiving orders from the SDN controller in accordance with an embodiment of this disclosure;

Figure 7 illustrating a process performed by the second switch after receiving orders from the SDN controller in accordance with an embodiment of this disclosure;

Figure 8 illustrating a process performed by the third switch after receiving orders from the SDN controller in accordance with an embodiment of this disclosure;

Figure 9 illustrating an internal structural diagram of core devices in accordance with an embodiment of this disclosure;

Figure 10 illustrating flow entries and group tables generated on an ingress switch in accordance with an embodiment of this disclosure;

Figure 11 illustrating flow entries and group tables generated on an intermediate switch in accordance with an embodiment of this disclosure;

Figure 12 illustrating flow entries and group tables generated on an egress switch in accordance with an embodiment of this disclosure;

Figure 13 illustrating further flow entries and group tables generated on the egress switch in accordance with an embodiment of this disclosure;

Figure 14 illustrating a selection of an ingress switch, an egress switch, and an intermediate switch in accordance with an embodiment of this disclosure;

Figure 15 illustrating an interaction process among the modules in the control layer and the data layer in accordance with an embodiment of this disclosure;

Figure 16 illustrating a process of forwarding a data packet by an ingress switch in accordance with an embodiment of this disclosure;

Figure 17 illustrating a process of forwarding a data packet by an egress switch in accordance with an embodiment of this disclosure;

Figure 18 illustrating a process of forwarding a data packet by an ingress switch in batches in accordance with an embodiment of this disclosure;

Figure 19 illustrating a process of forwarding a data packet by an egress switch in batches in accordance with an embodiment of this disclosure;

Figure 20 illustrating a process performed by the detection module in accordance with an embodiment of this disclosure;

Figure 21 illustrates a process performed by the detection process control module in accordance with an embodiment of this disclosure; and

Figure 22 illustrates a process performed by the table entry generation module in accordance with an embodiment of this disclosure.

Detailed Description

This disclosure relates to a method and system for verifying the veracity of data packet being forwarded in a network. Particularly, this disclosure relates to a method and system for verifying forwarding path inconsistency or malicious tampering of a data packet in a network.

In a current network, two problems faced by network administrators are device failures and network attacks. A device failure causes a device to forward a data packet to an incorrect port. This increases a network delay and brings a serious security risk to a user. A network attack even directly steals a user's privacy data. An attacker may forward a network flow to a machine controlled by the attacker, so as to listen to user traffic, or directly tamper with a user's data packet, so as to inject malicious code.

In these failure or attack scenarios, a data packet is usually forwarded to an incorrect path or is directly tampered with. In the present disclosure, data packet forwarding verification includes two aspects: (1) path inconsistency detection and (2) data packet anti-tampering. Path inconsistency means that a data packet is forwarded to an incorrect path, and anti-tampering is to prevent a data packet from being tampered with by a malicious attacker during forwarding. Based on this, for the two abnormal behaviours, the system and method of the present disclosure proposes a new detection module residing in the control layer. Specifically, key elements of the system and method of the present disclosure are as follows:

1. With reference to an OpenFlow group table and a flow label, the present disclosure proposes a solution in which flow sampling is performed and then

traffic statistics is collected on a sampled flow to detect forwarding path inconsistency of a data packet.

2. Based on a sampling manner that is the same as the sampling manner in "1", the present disclosure further proposes a data packet anti-tampering mechanism.

A current network architecture provides a so-called "end-to-end" service. The service only ensures that a data packet can be forwarded from a host to another host, but does not ensure correctness of a forwarding path or integrity of the data packet. However, ensuring the correctness of the forwarding path of the data packet and the integrity of the data packet plays a vital role in a network, especially from a perspective of security.

Main objectives of the system and method of the present disclosure are to detect possible forwarding path inconsistency or malicious tampering of a data packet in a network. The forwarding path inconsistency means that the data packet is not forwarded according to a "correct" path, but is forwarded to an incorrect node and then forwarded to a destination; or is directly discarded in a forwarding graph. The malicious tampering of the data packet means that a malicious attacker injects malicious code, adds other data, or deletes data content in the data packet during forwarding.

By detecting these problems, the system and method of the present disclosure can effectively reduce performance problems such as network unreachability and network delay increase due to a network fault. More importantly, by detecting whether a data packet is forwarded along a correct forwarding path, the system and method of the present disclosure effectively reduces risks that user traffic is listened to and stolen, thereby protecting a user's privacy and property security. In addition, by detecting whether a data packet is maliciously tampered with in the network, the system and method of the present disclosure can effectively prevent some man-in-the-middle attacks and propagation of some viruses, which can provide better security assurance for the network.

From a perspective of practicability, the system and method of the present disclosure imposes little impact on the network. Firstly, the system and method of the present disclosure is dependent on only lightweight operations, for example, adding a flow label to a data packet, removing a flow label, counting a quantity of data packets, and sending a data packet to a control plane/layer. The system and method of the present disclosure does not relate to lots of encryption and decryption operations. Therefore, the system and method of the present disclosure imposes little impact on performance of a switch. Secondly, the system and method of the present disclosure can be implemented by using some existing

hardware of the switch, for example, calculating an ID value of a data packet. This can improve performance of the present disclosure, and also facilitate wide deployment and implementation of the system and method of present disclosure. Finally, the impact of the system and method of the present disclosure can be controlled on the network by means of parameter adjustment. Therefore, the system and method of the present disclosure can basically be applicable to a network of any amount of traffic.

The system and method of the present disclosure detects forwarding path consistency of a data packet by collecting statistics on traffic data of a network flow, and verifies integrity of the data packet by using the data packet or a unique ID value of the data packet. Herein, the unique ID value of the data packet may be a hash value of the data packet, for example, an MD5 or SHA1 value. To ensure that the detection in the system and method of the present disclosure does not cause serious impact on network performance, a group table mechanism in OpenFlow is used to sample a network flow and add a unique flow entry to a sampled part, and then detect only a sampled data packet. Briefly, the system and method according to the present disclosure includes four key elements as follows:

1. The system and method according to the present disclosure uses a group table mechanism to sample each network flow, and then add a unique distinguishable label to a sampled data packet and add a random label that is different from the unique distinguishable label to another data packet. In the system and method according to the present disclosure, only the data packet added with a unique distinguishable label is detected.

2. For data packets added with a unique distinguishable label in a flow, a quantity of data packets with a unique distinguishable label is separately counted on the first-hop switch, a randomly selected intermediate switch, and the last-hop switch of the flow. Then the quantities are sent to a control plane by using a flow entry traffic statistics message (Flow Stats) in OpenFlow. The control plane compares the three groups of statistical data. If the three groups of statistical data are inconsistent or are greatly different, it is considered that forwarding path inconsistency of the network flow occurs. That is, the network flow is forwarded along an incorrect path. Alternatively, only the first-hop and last-hop switches are implemented and the control plane compares the two groups of statistical data.

3. For data packets added with a unique distinguishable label in a flow, the data packets are simultaneously forwarded to a correct output port and the control plane on the first-hop switch and the last-hop switch of the flow. After detection, the control plane compares these data packets. Whether the data packets are maliciously tampered with is determined according to whether all data packets appearing on the first-hop switch also

appear on the last-hop switch. If there is a data packet that appears on the first-hop switch but does not appear on the last-hop switch, it is considered that the flow is maliciously tampered with during forwarding. Alternatively, the randomly selected intermediate switch also forwards the data packet with the unique distinguishable label in the flow and the control plane compares the data packets from the last-hop, intermediate-hop and last-hop.

4. Many optional mechanisms are available in the system and method according to the present disclosure. These mechanisms can help the system and method according to the present disclosure fulfil its role and improve working efficiency.

System Architecture Diagram

Application of the present disclosure depends on SDN system. From a perspective of a network scale, the system and method according to the present disclosure is not restricted by the network scale. A target network of the system and method according to the present disclosure may be a local area network, a corporate network, or even a large-scale ISP network. In these networks, operators may deploy the system and method according to the present disclosure on a control plane/layer. Then events of data packet forwarding path inconsistency and malicious data packet anti-tampering in the networks may be detected.

Figure 3 illustrates the system 300 for verifying a veracity of data packets being forwarded in a Software Defined Network having a SDN controller 305 and multiple switches 310.

The switches 310 are network elements in a typical SDN system. In the present disclosure, three switches are being ordered by the SDN controller to perform certain processes to verify the veracity of data packets being forwarded in a forwarding path. Briefly, a forwarding path indicates path required to be traverse from one switch to another. Hence, to ensure that the data packets are traverse in the correct forwarding path, the SDN controller instructs the three switches, via updating the flow tables and group tables of the three switches, to perform the required processes. The three switches are ingress switch, egress switch and intermediate switch being one of the switches between the ingress and egress switches. In one embodiment, only ingress and egress switches are required for the verification. In another embodiment, in addition to the ingress and egress switches, a third switch, i.e. the intermediate switch is implemented. Essentially, the switches selected are not adjacent switches and are typically apart from each other. Further details of the instructions to the three switches will be described below.

The SDN controller 310 may be a computing system such as a server computer or a virtual machine running on a server computer.

Figure 4 illustrates an example of a processing system 400 in a server computer representing the processing systems in the SDN controller 310 that execute instructions to perform the processes described below in accordance with embodiments of this disclosure. The processing system 400 may also be applicable to represent the processing system in a network element such as the switches in accordance with this disclosure. One skilled in the art will recognize that the instructions may be stored and/or performed as hardware, firmware, or software without departing from this invention. Further, one skilled in the art will recognize that the exact configuration of each processing system may be different and the exact configuration of the processing system executing processes in accordance with this invention may vary and processing system 400 shown in figure 4 is provided by way of example only.

Processing system 400 includes a processor 410, a display 420, a keypad 430, a memory 440, network device 450 and an I/O device 460.

The display 420, keypad 430, memory 440, network device 45 and I/O device 460 and any number of other peripheral devices connect to processor 410 to exchange data with processor 410 for use in applications being executed by processor 410.

The display 420 receives display data from processor 410 and display images on a screen for a user to see. The display 440 may be a liquid crystal display (LCD) or organic light-emitting diode (OLED) display. The keypad 420 receives user input and transmits the input to processor 410. In some embodiments, the display 440 may be a touch sensitive surface that functions as a keypad to receive user input.

The memory 440 is a device that transmits and receives data to and from processor 410 for storing data to a memory. The data being stored include instructions for performing the processes as will be described below and the flow tables and group tables. The memory may comprise non-volatile memory such as a Read Only Memory (ROM) or volatile memory, such as Random Access Memory (RAM). Non-volatile memory stores instructions and data needed to operate various sub-systems of processing system 400. Volatile memory stores the instructions and data needed by processor 410 to perform the processes for providing a system in accordance with this invention. One skilled in the art will recognize that any non-transitory memory may be used and the exact type used is left as a design choice to those skilled in the art.

The network device 450 connects processor 410 to the SDN system for transmission of data to and from other network elements within the SDN system.

The processor 410 is a processor, microprocessor, or any combination of processors and microprocessors that execute instructions to perform the processes in accordance with the present disclosure. The processor has the capability to execute various application programs that are stored in the memory 440. These application programs can receive inputs from the user via the keypad 430.

Figure 5 illustrates a process 500 performed by the SDN controller to verify a veracity of data packets being forwarded in a Software Defined Network. Process 500 begins with step 505 by determining and selecting network flows that need to be detected for verification on the veracity of network flows in the SDN system.

In step 510, process 500 generates orders or instructions to a first switch and a second switch. The first and second switches are two switches in an expected forwarding path of the network flow. Particularly, the first switch is an upstream switch while the second switch is a downstream switch with respect to the first switch. For example, the first switch may be an ingress switch while the second switch may be an egress switch. In one embodiment, in addition to the first and second switches, process 500 also generates orders or instructions to a third switch which is one of the switches between the first and second switches. Particularly, process 500 generates orders to the switches by based on OpenFlow standard to update the relevant switches' flow tables and group tables. The orders and instructions to the switches are briefly discussed as follows:

The orders to the first switch include: 1) sampling data packets received for the selected network flow at a predetermined sampling rate, 2) tagging the sampled data packets with a first label, 3) counting the number of data packets being sampled, and 4) forwarding the data packets to the next hop and the SDN controller and reporting the number of sampled data packets to the SDN controller. The first label is a unique label. Further details of the unique label will be described below. In another embodiment, a second label is introduced in step 2. Particularly, the second label is tagged to non-sampled data packets. Essentially, the second label acts as a confusion label. In yet another embodiment, the number of sampled data packets may be reported to the SDN controller after a predetermined time period such as after a hard timeout. Alternatively, the number of sampled data packets may be reported to the SDN controller when the number of sampled data packets reaches a predetermined threshold.

The orders to the second switch include: 1) counting the number of data packets with first label received; 2) removing the first label, 3) forwarding data packets to the next hop and the SDN controller, and reporting the number of data packets with the first label to the SDN controller. In another embodiment where confusion label is introduced, step 3 is modified such that the second label is also removed. In yet another embodiment, the number of sampled data packets may be reported to the SDN controller after a predetermined time period such as after a hard timeout. Alternatively, the number of sampled data packets may be reported to the SDN controller when the number of sampled data packets reaches a predetermined threshold.

The orders to the third switch include: 1) counting the number of data packets with the first label receive; 2) forwarding data packets to the next hop; and 3) reporting the number of data packets with the first label to the SDN controller. In another embodiment, the third switch may be required to forward the data packet with the first label to the SDN controller in step 3 together when reporting the number of data packets with the first label to the SDN controller. In yet another embodiment, the number of data packets with the first label may be reported to the SDN controller by way of the SDN controller 305 using a FlowRemoved message.

In step 515, process 500 transmits the orders to the switches. Particularly, process 500 transmits the orders to the switches by sending OpenFlow messages to the relevant switches to respective flow tables and group tables. Steps 505-515 is the setting up of the switches to verify a veracity of data packets being forwarded in a certain network flow in a Software Defined Network.

In step 520, process 500 receives detection data from the two switches. In particular, two types of detection data are received from the switches, namely, quantities of data packets forwarded by the first and second switches, and data packets forwarded by first and second switches. In the embodiment where a third switch is implemented, process 500 also receives the quantities of data packets and data packets from the third switch.

In step 525, process 500 performs an analysis on the two types of detection data. Particularly, process 500 performs the following comparison on detection data of each network flow:

1. Whether the number of data packets forwarded by the first switch is equal to that forwarded by the second switch; and

2. Whether the data packets forwarded by the first and second switches are the same.

In step 530, if the number of data packets forwarded by the first switch is equal to that forwarded by the second switch, and the data packets forwarded by each of the first and second switches are the same, it indicates that the network flow is correctly forwarded and no further action is required and process 500 ends thereafter. If the number of data packets forwarded by the first switch are not equal to that forwarded by the second switch, and the data packets forwarded by each of the first and second switches are not the same, it indicates that the network flow is incorrectly forwarded and further action is required and process 500 proceeds to step 535 to raise an action ticket. The action ticket may be an alert to the user to check the selected network flow.

In the embodiment where the third switch is implemented, step 525 is modified such that the first comparison determines whether the number of data packets received from each of the first, second and third switches are the same. Further, in the embodiment where the third switch is required to forward the data packet with the first label to the SDN controller when reporting the number of data packets with the first label to the SDN controller, step 525 is modified such that the second comparison determines whether the data packets received from each of the first, second and third switches are the same.

In another embodiment, message digest(s) may be received by the SDN controller instead of the data packets. Hence, step 525 may be modified such that the second comparison determines whether the message digests received from each of the first, second and third switches are the same.

Process 500 ends after step 530 or 535.

In one embodiment, process 500 is initiated periodically based on a predetermined parameter. The parameter may be to perform a detection cycle once after a period of time, or employ a period of sleep time after a detection cycle is complete. Each detection cycle may be performed at a fixed or random time. One skilled in the art will recognise that other methods of periodically initiating process 500 may be implemented without departing from the disclosure.

Figure 6 illustrates a process 600 performed by the first switch after receiving the orders from the SDN controller. Process 600 begins with step 605 by updating the flow tables and group tables.

In step 610, process 600 receives and samples data packets from the selected network flow. In step 615, the sampled data packets are tagged with the first label. In another embodiment where a second label is introduced, the non-sampled data packets are tagged with the second label in step 620.

In step 630, process 600 forwards the number of sampled data packets and at least one sampled data packet to the SDN controller. The sampled data packet that is forwarded to the SDN controller may be the first sampled data packet.

In one embodiment, instead of sending the first sampled data packet to the SDN controller, process 600 computes a message digest for each of the sampled data packets, buffers these message digests when necessary, and sends the message digests to the SDN controller only when a cache is full or when the detection ends. Particularly, step 630 is modified as follows: 1) calculate a message digest for the sampled data packet; 2) storing the message digest in a cache; 3) if the quantity of buffered message digests in the cache reaches a predetermined threshold, sending all the message digests in the cache to the SDN controller; and 4) clearing cache before it proceeds to step 635. Alternatively, if a switch does not use cache, step 630 may be modified such that process 600 sends each message digest to the SDN controller immediately after generating it in step 1. Alternatively, step 630 may be further modified such that in step 3, instead of sending each sampled data packet's message digest to the SDN controller, the process calculates a combined message digest of all sampled multiple data packets and sends the combined message digest to the SDN controller. There are many ways to compute the combined message digest. For example, let $D1$ be a function to compute a message digest from a binary string in this document, and $D2$ be another function to compute a message digest from a binary string. The combined message digest may be computed as $D2(D1(\text{packet1}) \text{ XOR } D1(\text{packet2}) \text{ XOR } \dots)$. Alternatively, the combined message digest may be computed as $D1(\text{packet1} \text{ XOR } \text{packet2} \text{ XOR } \dots)$. The well known Merkle tree could be used to compute the combined message digest too, and the root of Merkle tree is the combined message digest.

In step 635, process 600 forwards the data packets to the next hop, i.e. next switch in the network flow.

Process 600 ends after step 635.

Figure 7 illustrates a process 700 performed by the second switch after receiving the orders from the SDN controller. Process 700 begins with step 705 by updating the flow tables and group tables.

In step 710, process 700 receives data packets from the selected network flow. In step 715, process 700 determines whether the data packets have been tagged with the first label. For data packets with the first label, process 700 proceeds to step 720. For data packets without the first label, process 700 proceeds to step 730.

In step 720, process 700 forwards the number of data packets with the first label and at least one data packet with the first label to the SDN controller, and removes the first label from the data packet. The data packet with the first label that is forwarded to the SDN controller may be the first determined data packet with the first label. After step 720, process 700 proceeds to step 730.

In one embodiment, instead of sending the first data packet with the first label to the SDN controller, process 700 computes a message digest of all the data packet with the first label, buffers these message digests when necessary, and sends the message digests to the SDN controller only when a cache is full or when the detection ends. Particularly, step 720 is modified as follows: 1) calculating a message digest for each of the data packet with the first label; 2) storing the message digests in a cache; 3a) if the quantity of buffered message digests in the cache reaches a predetermined threshold, forwarding all the message digests in the cache to the SDN controller, clearing cache and removing the first label before it proceeds to step 730; 3b) if the quantity of buffered message digests in the cache has not reached the predetermined threshold, proceeding to step 730. Alternatively, if the second switch does not use cache, step 720 may be modified such that process 700 sends each message digest to the SDN controller immediately after generating it in step 1. Alternatively, step 720 may be further modified such that in step 3a, instead of forwarding all the message digests in the cache to the SDN controller, the process calculates a combined message digest based on all the message digests in the cache and forwards the combined message digest to the SDN controller.

In step 730, process 700 forwards the data packets to the next hop. In another embodiment where a second label is introduced, the second label in data packets are removed before forwarding to the next hop.

Figure 8 illustrates a process 800 performed by the third switch after receiving the orders from the SDN controller. This process is applicable where the third switch is implemented. Process 800 begins with step 805 by updating the flow tables and group tables.

In step 810, process 700 receives data packets from the selected network flow. In step 815, process 800 determines whether the data packets have been tagged with the first label.

For data packets with the first label, process 800 proceeds to step 820. For data packets without the first label, process 800 proceeds to step 830.

In step 820, process 800 forwards the number of data packets with the first label to the SDN controller after predetermined time period such as a hard timeout. Step 820 is essentially counting the number of data packets with the first label and forwarding the final number after a predetermined time period. After step 820, process 800 proceeds to step 830. In step 830, process 800 forwards the data packets to the next hop. Process 800 ends after step 830.

In the embodiment where the third switch is required to forward the data packet with the first label to the SDN controller when reporting the number of data packets with the first label to the SDN controller, step 820 is modified such that in addition to forwarding the number of data packets with the first label to the SDN controller, step 820 further forwards at least one data packet with the first label is forwarded to the SDN controller. The data packet with the first label that is forwarded to the SDN controller may be the first determined data packet with the first label. Alternatively, instead of sending the first data packet with the first label to the SDN controller, process 800 computes a message digest of all the data packet with the first label, buffers these message digests when necessary, and sends the message digests to the SDN controller only when a cache is full or when the detection ends. Particularly, step 820 is modified as follows: 1) calculating a message digest for each of the data packet with the first label; 2) storing the message digests in a cache; 3a) if the quantity of buffered message digests in the cache reaches a predetermined threshold, forwarding all the message digests in the cache to the SDN controller and clearing cache before it proceeds to step 830; 3b) if the quantity of buffered message digests in the cache has not reached the predetermined threshold, proceeding to step 830. Alternatively, if the third switch does not use cache, step 820 may be modified such that process 800 sends each message digest to the SDN controller immediately after generating it in step 1. Alternatively, step 820 may be further modified such that in step 3a, instead of forwarding all the message digests in the cache to the SDN controller, the process calculates a combined message digest based on all the message digests in the cache and forwards the combined message digest to the SDN controller.

Process 800 ends after step 830.

The processes as illustrated above with reference to figures 5-8 may be performed by the SDN controller 305 using software modules stored on the memory 440 executable by the processor 410. Figure 9 illustrates an internal structural diagram 900 modules involved for performing the processes according to the present disclosure. As shown in figure 9, there are essentially two parts, namely a control plane 301 and a data plane 302. A detection module 905 in the memory 440 comprises eight modules: (1) a forwarding graph module 910, (2) a detection process control module 920, (3) a table entry generation module 930, (4) a switch selection policy 940, (5) a label pool 950, (6) a detection data storage module 960, (7) a data analysis module 970, and (8) an abnormality handling module 980. On the data plane 302, a flow entry and a group table of a switch interact with the modules in the control plane. We shall first describe the eight modules before describing the processes involved.

Forwarding graph module 910

The forwarding graph module 910 maintains all network flows and their forwarding paths in the network. A forwarding graph is a directed graph. The forwarding graph module 910 uses flow entries or group tables as nodes, uses forwarding dependencies between the flow entries and between the group tables as edges, and further abstracts flow entries and group tables by which a network flow is forwarded in the network. In the present disclosure, the forwarding graph module 910 is used to maintain a flow forwarding path, and a flow entry and a group table that forward a flow. If a request for flow entries and group tables for forwarding paths is received from the detection process control module 910, the forwarding graph module 910 retrieves the flow entries and group tables for the requested forwarding paths and transmits the same to the detection process control module 910. The forwarding graph module 910 may be constructed by analyzing dependencies between flow entries. For example, a switch s_1 and a switch s_2 are connected, and a data packet forwarded by a flow entry r_1 on the switch s_1 is subsequently processed by a flow entry r_2 on the switch s_2 . In the forwarding graph, there is an edge between the flow entries r_1 and r_2 . This indicates that a network flow is continuously processed by $\{r_1, r_2\}$ in sequence. The forwarding graph is a variable directed graph, and is updated as a flow entry or a group table is updated. When a flow entry or a group table is updated, only a dependency relationship between the flow entry or the group table and a flow entry or a group table in a previous-hop switch of the flow entry or the group table needs to be recalculated. That is, the forwarding graph can be efficiently maintained in an incremental manner, without requiring lots of overheads.

Forwarding graph is well known in the art. Since in this disclosure, the forwarding graph module is only used to maintain a network flow forwarding path, a detailed description of the forwarding graph is omitted for brevity.

Detection process control module 920

The detection process control module 920 coordinates the entire detection process. First, the detection process control module 920 selects network flows that need to be detected. Particularly, the network flows that need to be detected may be selected randomly according to a predetermined set of policy. For example, setting a sampling rate or the number of selected flows. Alternatively, the network flows that need to be detected may be selected based on preselected set of filter rules. For example, all flows destination to an IP or IP prefix. One skilled in the art will recognise that other types of selection may be implemented without departing from the disclosure. After selecting the network flows that need to be detected, the detection process control module 920 obtains the forwarding paths of the selected network flows from the forwarding graph module. The forwarding paths contain flow entries and group tables of the selected network flows. Thereafter, the detection process control module 920 transmits the selected network flows that need to be detected to the table entry generation module 930. In response to receiving the selected network flows that need to be detected, the table entry generation module 930 generates detection-purpose flow entries and group tables for these to-be-detected network flows and forwards it back to the detection process control module 920. In response to receiving the flow entries and group tables for the selected network flow that need to be detected, the detection process control module 920 then transmits the same flow entries and group tables to the data plane 302. Finally, the detection process control module 920 notifies the data analysis module 970 to analyse all detected data after the current detection is complete.

To ensure normal running of the entire network, the detection process must be executed periodically. One possible manner is to perform detection once after a period of time, or employ a period of sleep time after one detection is complete. Each detection may be performed at a fixed or random time. Parameters related to detection process control are listed in Table 1 below.

Parameter Name	Parameter Description
t	Detection time, that is, time interval from delivery of a detection-purpose

	flow entry and group table to collection of detection data.
d	Detection interval, that is, time from completion of a detection to start of a next detection.

Table 1

Table entry generation module 930

Based on the information from the switch selection policy module and the label pool module, the table entry generation module 930 generates a corresponding detection-purpose flow entry and group table according to a network flow and a forwarding path of the network flow of the selected network flow received from the detection process control module 920. Particularly, the table entry generation module 930 receives a request from the detection process control module 920 containing the network flows that need to be detected. The table entry generation module 930 then requests a unique label and a confusion label for the network flow from the label pool module 950, and obtains, from the switch selection policy module 940, a switch selected for detection.

During the sampling of the to-be-detected network flow, only a data packet in a sampled part is processed, and therefore the table entry generation module 930 needs to determine the sampling ratio (s). Based on the information, namely, labels (i.e. unique and confusion labels), detection switch, and sampling ratio (s), the table entry generation module 930 generates a detection-purpose flow entry and group table for each detected network flow. Hard timeout (Hard Timeout) is set for these flow entries. After the hard timeout, these flow entries are automatically deleted, and the detection is complete.

One skilled in the art will recognise that various network flow sampling schemes may be implemented without departing from the disclosure. Particularly, while this disclosure describes how to use a group table to cooperate with a SDN controller 305 to perform periodic sampling, the actual implementation only requires that a detected flow can be forwarded to at least two switches on a forwarding path of the flow for sampling where a first switch is a switch in the upstream of the forwarding path where a second switch is a switch in the downstream of the forwarding path. In one embodiment, in addition to the first and second switches, a third switch may be further required. The third switch is one of the switches between the first and second switches. For example, the three switches may be ingress switch, randomly selected intermediate switch, and egress switch.

Optionally, some possible network flow sampling schemes that may be implemented according to this disclosure may be as follows:

Periodic sampling - That is, some flows are detected according to a rule each time a detection starts, and each flow is detected once after a predetermined time period.

Random detection - That is, a detected flow is randomly selected in each detection. A detection time and a detection period may be fixed or random.

Each switch may use a different sampling function, but calculation needs to be performed in advance to ensure that some data packets in each detected flow are sampled by at least three switches on a forwarding path of the flow.

The logic of the table entry generation module 930 for generating a flow entry and group table is described below with reference to figures 10-13. It is assumed herein that an original match field of a flow entry is m and an output port is p . Related symbols are shown in table 2 below.

Parameter Name	Parameter Description
m	Match field of a flow when no detection is performed
p	Output port of a network flow when no detection is performed
s	Sampling ratio, that is, probability that a data packet in a detected network flow is sampled for detection.

Table 2

Figure 10 illustrates flow entries and group tables generated on an ingress switch. The table entry generation module modifies the original flow entry forwarding the flow, and adds two group tables. Modifying the flow entry aims to forward the network flow 1005 to the group tables. The first group table 1010 is used to perform flow sampling. A flow that is not sampled is added with a label randomly selected from R , and then is forwarded to a next hop 1050. A flow that is sampled is added with a label l_f , and then is forwarded to a next group table 1020 for processing. In the next group table 1020, these data packets are simultaneously forwarded to the SDN controller 305 and a next hop 1050. These data packets are stored by the detection data storage module 1060 as detection data. At the end of detection, these data packets are compared with data packets received from the last hop.

Forwarding the data packet to the controller aims to compare the data packet with a data packet that is forwarded by the egress switch to the controller, so as to discover a data packet tampering behaviour. At the end of detection, the controller changes the flow entry back to a state before the detection. Each group table has a timeout associated with it and the ingress switch will delete the group tables upon timeout.

Figure 11 illustrates flow entries and group tables generated on an intermediate switch. A detection module 905 delivers in advance a flow entry 1105 that matches both the original match field and the label l_f , so that a data packet that is sampled and added with the label l_f is matched by the flow entry. The flow entry is only used to count a quantity of data packets with the label l_f , that is, a quantity of sampled data packets. After the hard timeout, these flow entries are automatically deleted, and the detection is terminated. The SDN controller 305 can obtain, by using a FlowRemoved message, quantities of data packets forwarded by these flow entries.

In another embodiment, the intermediate switch may be required to forward the data packet with the unique label l_f , to the SDN controller when reporting the number of data packets with the unique label l_f , to the SDN controller. In such circumstances, the flow entry 1105 is modified such that the action includes “output:CONTROLLER”.

Figure 12 illustrates flow entries and group tables generated on an egress switch. A detection module 905 also delivers in advance a flow entry 1205 that matches both the original match field and the label l_f . The flow entry 1205 forwards a sampled data packet to a group table 1210. The group table 1210 sends the data to the SDN controller 305, and sends the data to a network egress (which is the next hop 1250) after removing the label. In addition, on the egress switch, the detection module 905 also modifies the original flow entry forwarding the flow, so that a label randomly generated on a data packet that is not sampled can be removed as shown in figure 13. Particularly, the flow entry 1305 forwards a data packet that matches both the original match field and the label R to the group table 1310. In turn, the group table 1310 removes the label R and sends the data packet to the network egress 1250.

Switch selection policy 940

The switch selection policy 940 selects a group of switches for a network flow and forwarding path. In the present disclosure, flow entries and group tables are delivered on the group of switches, and then are used to complete detection. In the present disclosure, three switches need to be selected for each network flow: an ingress switch 1410, an egress

switch 1430, and an intermediate switch 1420 randomly selected between the ingress switch and the egress switch, as shown in figure 14. In one embodiment, only the ingress switch 1410 and egress switch 1420 are required.

In this disclosure, an OpenFlow switch is not required to be modified. In this way, the present disclosure can work normally when OpenFlow is not modified. However, if the switch can provide some extended functions, performance of the present disclosure will be improved. The following two options are provided:

1. The switch forwards an entire data packet to the control layer. Actually, the switch needs to forward only an ID value of the data packet to a controller. The ID value may be a hash value of the data packet, for example, an MD5 value or an SHA1 value. Using an ID value of a data packet effectively reduces overheads of communication between the control layer and the data layer.
2. The switch forwards ID values of data packets to the control layer in batches. The switch first locally calculates an ID value of a sampled data packet, and then buffers the ID value. When a given quantity of ID values is buffered or the detection ends, the switch forwards the buffered ID values to the control layer in batches.

The switch selection policy 940 selects a group of switches from a network flow and forwarding path upon receiving a request from the table entry generation module 930. The request includes the selected network flow and forwarding path of the selected network flow. Based on the request, the switch selection policy 940 selects a group of switches to perform the deliver the data packet according to the selected network flow. The group of switches include an ingress switch 1410, an egress switch 1430, and an intermediate switch 1420 randomly selected between the ingress switch and the egress switch. The selected group of switches is then transmitted back to the table entry generation module 930.

Label pool 950

The label pool maintains two groups of labels. Labels in the first group are used to uniquely identify a network flow that is being detected, and a sampled data packet in the network flow is added with the unique label. Labels in the second group are used to confuse an attacker, and are added to data packets in network flows that are not detected and data packets that are not sampled in the detected network flow. In a detection, any label in the first group of labels can be used only once, and the confusion label can be used multiple times.

For convenience, in the subsequent description, the first label set is expressed as L and the second label set is expressed as R . A unique label associated with a detected network flow f is expressed as l_f . Related symbols are shown in table 2 below.

Parameter Name	Parameter Description
l_f	Unique identifiable label that is generated for a flow f , and is unique in all flow labels.
L	Set of unique identifiable labels generated for all detected flows in a detection ($l_f \in L$).
R	Set of identifiable labels that are not unique and are generated randomly ($R \cap L = \phi$).

In the present disclosure, labels are mainly used to achieve the following two objectives: using each label to uniquely identify some sampled data packets in a detected flow; and adding labels to flows that are not detected and some data packets that are not sampled in the detected flow, to confuse an attacker. In this way, each detected flow requires a unique label. A label for confusion may be used repeatedly, and therefore a quantity of labels for confusion may be fixed. It is assumed that there are $|F|$ flows in a network, and in the present disclosure, N flows are detected at a time and C labels are used to confuse an attacker. Therefore, there are a total of $N + C$ labels required.

The present disclosure imposes no requirement on a label, as long as the label can distinguish a network flow, for example, a label for a differentiated service (TOS), a VLAN label, or an MPLS label. If there is a small quantity of available labels, detection on the entire network may be completed by means of multiple detections, and only some network flows are detected in each detection. To describe a label selection method, the TOS, VLAN, and MPLS labels are compared in the following.

TOS label: A length of the TOS label is 1 bit. 256 different TOS labels can be used. The TOS label can support small networks only. Labels of a given quantity, for example, 240 labels, may be used to identify detected network flows, and other labels are used as confusion labels.

VLAN label: A length of the VLAN label is 32 bits. A total of 4096 VLAN labels may be used, which is sufficient for a common network.

MPLS label: A length of the MPLS label is 32 bits. 1048576 MPLS labels can be supported, which is sufficient for any network.

In short, the MPLS label is an optimal choice as the length of the MPLS label is in an acceptable range and there are enough labels to support.

While the foregoing describes a manner in which a unique label is allocated to a detected data packet and another label is randomly selected for another data packet, one skilled in the art will recognise that a diversified label allocation schemes may be implemented without departing from the disclosure. Importantly, a label is used to label a detected data packet and a corresponding flow. If detected traffic and traffic that is not detected do not need to be confused, only one unique label needs to be selected for each detected flow, and then a corresponding label is added to a sampled data packet. Some possible label allocation schemes that may be implemented are as follows:

1. A unique label is allocated to each flow. In each detection, a sampled data packet is added with a unique label of a corresponding flow, and another data packet is added with another random label that is different from any unique label.
2. Two unique labels are generated for each flow. One unique label is used to indicate a sampled data packet, and the other unique label is used to indicate a data packet that is not sampled. Detection is performed only on sampled data packets in some flows.

Detection data storage module 960

The detection data storage module 960 stores detection data from the data layer, mainly including two types of data: quantities of data packets forwarded by detection-purpose flow entries, and data packets forwarded by detection-purpose flow entries on the first-hop (ingress) and last-hop (egress) switches. The data packet forwarded by the detection-purpose flow entries on the egress switch may be obtained by using a FlowRemoved message when the detection-purpose flow entries automatically expire. The data packet forwarded by the detection-purpose flow entries on the ingress switch is displayed in a group table and forwarded to the control layer 301, that is, output:CONTROLLER in a generated group table.

Detection data analysis module 970

The detection data analysis module 970 analyses a detection result. After successfully delivering a detection-purpose flow entry and group table, the detection process control module 920 notifies the detection data analysis module 970 of a completion time of the current detection. When the completion time is reached, the detection data analysis module 970 first obtains detection data of each detected flow from the detection data storage module 960, including quantities of data packets forwarded by detection-purpose flow entries, and data packets forwarded by detection-purpose flow entries on the first-hop and last-hop switches. Then the detection data analysis module 970 notifies the label pool module that unique labels corresponding to these detected flows are no longer used. Finally, the detection data analysis module 970 performs the following comparison on detection data of each network flow:

1. Whether the number of data packets forwarded by the detection-purpose flow entries reported by the first-hop (i.e. ingress switch) is equal to that reported by the last-hop; and
2. Whether the data packets forwarded by the detection-purpose flow entries on the first-hop (i.e. ingress switch) and last-hop (i.e. egress switch) switches are equal are the same.

If the number of data packets forwarded by the detection-purpose flow entries reported by the first-hop (i.e. ingress switch) is equal to that reported by the last-hop (i.e. egress switch), and the data packets forwarded by the detection-purpose flow entries on first-hop (i.e. ingress switch) and last-hop (i.e. egress switch) switches are equal, it indicates that the network flow is correctly forwarded. Otherwise, the network flow is abnormally forwarded.

In the embodiment where the intermediate switch is implemented the first comparison is modified such that the detection data analysis module 970 determines whether the number of data packets received from each of the first-hop, last-hop and intermediate-hop are the same. Further, in the embodiment where the third switch is required to forward the data packet with the first label to the SDN controller, the second comparison is modified such that the detection data analysis module 970 determines whether the data packets received from each of the first-hop, last-hop and intermediate-hop are the same.

Abnormality handling module 980

The abnormality handling module is responsible for specific abnormality handling. When the detection data analysis module 970 detects a forwarding abnormality of a network flow, the abnormality handling module 980 sends an alarm to an administrator, and calculates a new forwarding path. The new forwarding path keeps away from all nodes on an original forwarding path.

The detection module 905 periodically initiates detection. In particular, the detection process control module 920 periodically initiates detection. Figure 15 illustrates an interaction process 1500 among the eight modules in the memory 440 and the data plane 302. The modules are based on the software implementation.

The detection process flow control module 920 controls the entire detection process, and is responsible for scheduling periodic detection, including selecting to-be-detected flows, generating flow entries and group tables for the to-be-detected flows, delivering the flow entries and group tables, and instructing the data analysis module 970 to complete data analysis.

The detection process control module 920 first decides and selects network flows that need to be detected. The detection process control module 920 then obtains a forwarding path of the network flows that need to be detected from the forwarding graph module 910 in step 1505. Subsequently, the detection process control module 920 transfers the forwarding path to the table entry generation module 930 in step 1510. The forwarding path includes flow table entries and group tables for detection.

The table entry generation module 930 in response to receiving the forwarding path from the detection process control module 920, calls the switch selection policy module 340 to select a detection switch for the network flow, and requests a unique label and a confusion label from the label pool 950 in step 1515. According to a set sampling ratio, the table entry generation module 930 finally generates several flow entries and a group table for the network flow and forwards the generated flow entries and a group table for the network flow to the detection process control module 920. In response to receiving the generated flow entries and a group table for the network flow from the table entry generation module 930, the detection process control module 920 transmits the flow entries and group table to the data plane 302 and then notifies the data analysis module 970 of analysing detection data after the detection is completed in step 1520.

For the detection-purpose flow entries, "hard timeout" (hard timeout) is set. After the hard timeout, these flow entries are automatically deleted, and the detection is terminated.

The SDN controller 305 can obtain, by using a FlowRemoved message, quantities of data packets forwarded by these flow entries in step 1530. After the detection is terminated, the data analysis module 970 analyses the detection data received by the SDN controller 305 in step 1535, and releases a unique label associated with the detected network flow in step 1540. If the detection data analysis module finds an abnormality in the forwarding path, a forwarding abnormality is sent to the abnormality handling module 980 for further processing.

Two embodiments will be described below, which respectively describe a scenario in which a switch cannot be sent to the controller in batches (via the use of a function of "sending MD5 values to a controller") and a scenario in which the switch can be sent to the SDN controller in batches.

We shall now describe how a data packet is forwarded on the data plane, that is, on a switch. In the first embodiment, the switch does not have any special processing behaviour. To detail a data packet forwarding process, only one flow is used as an example to describe how an ingress switch, a randomly selected intermediate switch, and an egress switch of the flow forward a data packet.

Figure 16 illustrates a process 1600 of an ingress switch forwarding a data packet. Process 1600 begins with step 1605 where data packets of the network flow reach the ingress switch and the data packet is first matched by a flow entry. The flow entry forwards the data packet to group table 1 for sampling in step 1610. The data packet is sampled according to the determined sampling ratio s and is used as a detection packet. In step 1615, if a data packet is sampled, the data packet is added with a unique identifiable label l_f , and then continues to be transferred to group table 2 for processing in step 1630. If a data packet is not sampled, the data packet is added with a label randomly selected from a set R , and is directly forwarded to a next hop. In group table 2, the data packet is simultaneously forwarded to a controller and a next hop in step 1635. Forwarding the data packet to the SDN controller aims to compare the data packet with a data packet that is forwarded by the egress switch to the SDN controller, so as to discover a data packet tampering behaviour. At the end of detection (i.e. hard timeout), the controller changes the flow entry back to a state before the detection, and deletes the two group tables. One skilled in the art will recognise that at least one sampled data packet is required to be transmitted to the SDN controller for purposes of discovering a data packet tampering behaviour and the exact number of sampled data packets and the selection of sampled data packets to be transmitted to the SDN controller is left to those skilled in the art.

The controller delivers a flow entry to the randomly selected intermediate switch. The flow entry counts a quantity of data packets whose label is l_f in the flow. When the detection is complete, the flow entry is deleted. The SDN controller 305 can obtain, by using a FlowRemoved message, quantities of data packets forwarded by these flow entries. In another embodiment where the intermediate switch is required to forward the data packet with the unique label l_f , to the SDN controller, the flow entry is modified such that the data packet with label l_f , is transmitted to the SDN controller. One skilled in the art will recognise that at least one data packet with label l_f , is required to be transmitted to the SDN controller for purposes of discovering a data packet tampering behaviour and the exact number of data packets with label l_f , and the selection of data packets with label l_f , to be transmitted to the SDN controller is left to those skilled in the art.

Figure 17 illustrates a process 1700 of an egress switch forwarding a data packet. Process 1700 begins with step 1705 where data packets of the network flow reach the egress switch and process 1700 determines whether the data packet contains the unique label l_f . If the data packet contains the label l_f , process 1700 proceeds to step 1715 where the data packet is forwarded to the group table. If the data packet does not contain the unique label l_f , process 1700 proceeds to step 1710 where the random label is removed and subsequently forwarded to the next hop.

In step 1720, the data packet containing the unique label l_f is forwarded to the controller, the label l_f is removed from the data packet and subsequently forwarded to the next hop.

Embodiment 2 emphasizes how to use a switch to decrease overheads of the present disclosure. In this embodiment, a mechanism of the switch is used to calculate a message digest for a data packet that is originally forwarded to a controller, and then buffer the message digest, and send the message digest to the controller at a proper time. The message digest can be computed with a message digest algorithm such as MD5, SHA-1, SHA-2 and SHA-3.

In this embodiment, interaction between the modules remains the same as that shown in figure 15. The forwarding graph module 910 still works in the preceding manner. The detection module works roughly in the preceding manner, and the only difference is that a detection module 905 in this embodiment only knows message digests of data packets that are forwarded by an ingress switch and an egress switch to the SDN controller. The

detection module 905 discovers possible malicious data packet tampering by comparing whether the two groups of message digests are the same.

In the first embodiment, the ingress switch and the egress switch directly forward sampled data packets to the controller. In this embodiment, however, the switch locally calculates message digests for these data packets, buffers these message digests when necessary, and sends the message digests to the controller only when a cache is full or when the detection ends.

Figure 18 illustrates a process 1800 of the ingress switch forwarding a data packet. Steps 1805-1830 correspond to steps 1605-1630. The difference between process 1800 and process 1600 is that step 1635 of process 1600 is changed to steps 1835-1850, where batches of switch are implemented. In particular, in step 1835, Group table 2 calculates a message digest for the sampled data packet, and then stores the message digest to a cache. In step 1840, if the quantity of buffered message digests in the cache reaches a predetermined threshold, process 1800 proceeds to step 1845 and the ingress switch forwards all the message digests in the cache to the controller, forwards the data packet to the next hop and clear cache. Alternatively, process 1800 computes a combined message digest based on all the message digests in the cache and forwards the combined message digest to the controller and clear cache. If the cache is not full, process 1800 forwards the data packet to the next hop in step 1850. Process 1800 ends after step 1820, 1845 or 1850.

Figure 19 illustrates a process 1900 of the egress switch forwarding a data packet. Process 1900 begins step 1905 where a data packet reaches the egress switch. If a label of the data packet is not l_f , process 1900 proceeds to step 1910 to remove the label from the data packet, and forward the data packet is directly to an egress of the network. If the label of the data packet is l_f , process 1900 proceeds to step 1915 and forwards the data packet to group table 1. In step 1920, group table 1 calculates a message digest for the data packet, and then stores the message digest to a cache. In step 1925, if a quantity of buffered message digests reaches a predetermined threshold, process 1900 proceeds to step 1930 and the egress switch forwards message digests in the cache to the controller. After step 1930 or if the cache is not full, the label of the data packet is removed and forwarded to the next hop (which is normally to an egress of the network) in step 1935. At the end of detection, the egress switch should send all message digests in the cache to the SDN controller 305, and clear the cache.

In another embodiment, the intermediate switch is also required to forward the message digests of data packets to be forwarded to the SDN controller for verifying any

possible malicious data packet tampering by comparing message digests from the three switches.

Figure 20 illustrates a process 2000 performed by the detection module 305 in accordance with this disclosure. Process 2000 begins with step 2005 by determining and selecting network flows that need to be detected for verification on the veracity of network flows in the SDN system. This step is undertaken by the detection process control module 910 and further details would be described below with reference to figure 21.

In step 2010, process 2000 obtains the forwarding paths of the selected network flows from the forwarding graph module 910 and generates detection-purpose flow entries and group tables for these to-be-detected network flows. The detection-purpose flow entries and group tables for these to-be-detected network flows is generated by the table entry generation module 930. This step of generating detection-purpose flow entries and group tables is undertaken by the table entry generation module 930 and further details would be described below with reference to figure 22.

In step 2015, process 2000 transmits the detection-purpose flow entries and group tables for the selected network flows to the data plane 302; and notifies the data analysis module 970 to analyse all detected data after the current detection is complete.

In step 2020, process 2000 stores two types of detection data on the detection data storage module 960, namely, quantities of data packets forward by detection-purpose flow entries, and data packets forwarded by detection-purpose flow entries on the first-hop and last-hop switches.

In step 2025, process 2000 performs an analysis on the two types of detection data and releases the labels. Particularly, process 2000 performs the following comparison on detection data of each network flow:

1. Whether the number of data packets forwarded by the detection-purpose flow entries reported by the ingress switch is equal to that reported by the egress switch; and
2. Whether the data packets forwarded by the detection-purpose flow entries on the ingress switch and egress switch are the same.

In step 2030, if the number of data packets forwarded by the detection-purpose flow entries reported by the ingress switch is equal to that reported by the egress switch, and the data packets forwarded by the detection-purpose flow entries on the ingress switch and egress switch are the same, it indicates that the network flow is correctly forwarded and no

further action is required and process 2000 ends thereafter. If the number of data packets forwarded by the detection-purpose flow entries reported by the ingress switch is not equal to that reported by the egress switch, and the data packets forwarded by the detection-purpose flow entries on the ingress switch and egress switch are not the same, it indicates that the network flow is incorrectly forwarded and further action is required and process 2000 proceeds to step 2035 and raised an action ticket to the abnormality handling module 980.

In the embodiment where the intermediate switch is implemented the first comparison is modified such that step 2025 of process 2000 determines whether the number of data packets received from each of the ingress switch, egress switch and intermediate switch are the same. Further, in the embodiment where the third switch is required to forward the data packet with the first label to the SDN controller, step 2025 is modified such that the second comparison determines whether the data packets received from each of the ingress switch, egress switch and intermediate switch are the same.

In another embodiment where message digests are received from each of the ingress switch, intermediate switch and egress switch, step 2025 is modified such that the second comparison determines whether the message digests received from the ingress switch, intermediate switch and egress switch are equal. In yet another embodiment, the multiple message digests of each of the ingress switch, intermediate switch and egress switch, are combined by calculating a combined message digest for each of the ingress switch, intermediate switch and egress switch.

Process 2000 ends after step 2030 or 2035.

Figure 21 illustrates a process 2100 performed by the detection process control module 910. Process 2100 is initiated periodically based on a predetermined parameter. The parameter may be to perform a detection cycle once after a period of time, or employ a period of sleep time after a detection cycle is complete. Each detection cycle may be performed at a fixed or random time.

Process 2100 begins with step 2105 by determining and selecting network flows that need to be detected. Particularly, the network flows that need to be detected may be selected randomly according to a predetermined set of policy. For example, the predetermined set of policy may be setting a sampling rate or the number of selected network flows. Alternatively, the network flows that need to be detected may be selected based on preselected set of filter rules. For example, all network flows destination directed to an IP or IP prefix. One skilled in the art will recognise that other types of selection may be

implemented without departing from the disclosure. The predetermined set of policy, preselected set of filter rules or other predetermined method of determining network flows for verification may be stored as a data structure in a memory in the SDN system. Step 2105 is required to retrieve the data structure containing the information to determine the network flows for verification.

In step 2110, process 2100 obtains the forwarding paths of the selected network flows from the forwarding graph module 910. The forwarding paths contain flow entries and group tables of the selected network flows.

In step 2115, process 2100 transmits the flow entries and group tables of the selected network flows that need to be detected to the table entry generation module 930.

In step 2120, process 2100 receives detection-purpose flow entries and group tables for the selected network flows from the table entry generation module 930.

In step 2125, process 2100 transmits the detection-purpose flow entries and group tables for the selected network flows to the data plane 302; and notifies the data analysis module 970 to analyse all detected data after the current detection is complete. The notification to the data analysis module 970 contains the completion time.

Process 2100 ends after step 2125.

Figure 22 illustrates a process 2200 performed by the table entry generation module 930 in accordance with this disclosure. Process 2200 begins with step 2205 by receiving a request from the detection process control module 920 containing the network flows that need to be detected.

In step 2210, in response to receiving the request from the detection process control module, process 2200 requests a unique label (l_f) and a confusion label (R) from the label pool module 950, and obtains detection switches (i.e. ingress switch, randomly selected intermediate switch, and egress switch) selected for detection from the switch selection policy module 940.

In step 2215, process 2200 determines the sampling ratio (s). It is noted that during the sampling of the to-be-detected network flow, only a data packet in a sampled part is processed. Hence, a sampling ratio needs to be determined.

In step 2220, based on the information, namely, labels (i.e. unique and confusion labels), detection switch, and sampling ratio (s), process 2200 generates a corresponding

detection-purpose flow entries and group tables according to a network flow and a forwarding path of the network flow of the selected network flow contained in the request received from the detection process control module 920 in step 2205.

In particular, for the ingress switch, process 2200 modifies the original flow entry forwarding the flow, and adds two group tables. The first group table 1010 is used to perform flow sampling. A data packet that is not sampled is added with a label randomly selected from R , and then forwarded to a next hop 1050. A data packet that is sampled is added with a label l_f , and then is forwarded to a next group table 1020 for processing. In the second group table 1020, the data packets are simultaneously forwarded to the SDN controller and a next hop 1050.

For the egress switch, process 2200 modifies the original flow entry forwarding the flow, and adds one group table 1210. The original flow entry forwarding the flow is modified such that a sampled data packet in a flow entry 1205 that matches both the original match field and the unique label l_f is forwarded to a group table 1210. The group table 1210 sends the data to the control plane 301, and sends the data to a network egress (which is the next hop 1250) after removing the label. In addition, on the egress switch, process 2200 also modifies the original flow entry forwarding the flow, so that the random label R on a data packet that is not sampled is removed from the data packet.

For the intermediate switch, process 2200 modifies the original flow entry forwarding the flow such that a number of sampled data packets in a flow entry that matches both the original match field and the unique label l_f is determined and the data packets are then forwarded to the next hop. In another embodiment where the intermediate switch is required to forward the data packet with the unique label to the SDN controller, process 2200 further modifies the original flow entry such that sampled data packets in a flow entry that matches both the original match field and the unique label l_f are sent to the SDN controller.

Hard timeout (Hard Timeout) is set for these flow entries. After the hard timeout, these detection-purpose flow entries and group tables are automatically deleted, and the detection is complete. The logic of the table entry generation module 930 for generating a flow entry and group table is described above with reference to figures 10-13.

In step 2225, process 2200 transmits the detection-purpose flow entries and group tables to the detection process control module 910.

Process 2200 ends after step 2225.

While the processes described with reference to figures 15-22 mentioned the use of two switches, namely, ingress switch and egress switch, one skilled in the art will recognise that the ingress switch may be one of the switches upstream of the forwarding path of the network flow while the egress switch may be one of the switches downstream of the forwarding path of the network flow. Further, where a third switch is implemented, namely the intermediate switch, one skilled in the art will recognise that the third switch is any one of the switches between the first and second switches.

The above is a description of embodiments of a method and system based on SDN system to verify the veracity of data packet being forwarded in a network. It is foreseeable that those skilled in the art can and will design alternative method and system based on this disclosure that infringe upon this disclosure as set forth in the following claims.

CLAIMS:

1. A system for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller, a first switch, a second switch, and a third switch;

the SDN controller is configured to:

generate and transmit orders to the first switch and the second switch to return a first detection data and a second detection data of a network flow;

receive the first detection data and the second detection data, the first detection data being a number of data packets from the network flow with a first label reported by the first and second switches, and the second detection data being a content associated to the data packets from the network flow with the first label reported by the first and second switches;

perform an analysis on the first and second detection data; and

raise an action ticket if the analysis on the first and second detection data fails.

2. The system according to claim 1 wherein the first switch and the second switch are two switches in an expected forwarding path of the network flow, and the second switch is a downstream switch with respect to the first switch.

3. The system according to claim 1 or 2 wherein the first switch is an ingress switch and the second switch is an egress switch.

4. The system according to anyone of claims 1-3 wherein the SDN controller is specially configured to:

determine whether the number of data packets received from the first switch is equal to the number of data packets received from the second switch; and

determine whether the content associated to the data packets received from the first and second switches are the same.

5. The system according to any one of claims 1-4, wherein the SDN controller is configured to:

generate and transmit orders to the third switch, the third switch being an intermediate switch randomly selected between the first and second switches in the forwarding path of the network flow.

6. The system according to claim 5 wherein the SDN controller is configured to:
 - compare the number of data packets received from each of the first, second and third switches are the same; and
 - compare the content associated to the data packets received from each of the first, second and third switches are the same.
7. The system according to anyone of claims 1-6 wherein the first switch is configured to:
 - update flow tables and group tables according to the orders received from the SDN controller;
 - receive and sample data packets from the selected network flow;
 - tag sampled data packets with the first label; and
 - forward the data packets to the next hop and a content associated to the data packets with the first label to the SDN controller and report the number of sampled data packets to the SDN controller.
8. The system according to claim 7 wherein the first switch is further configured to:
 - tag non-sampled data packets with the second label.
9. The system according to claims 7 or 8 wherein the first switch is further configured to:
 - compute a message digest of all the sampled data packets to determine the content associated to the data packets with the first label.
10. The system according to anyone of claims 1-9 wherein the second switch is configured to:
 - update flow tables and group tables according to the orders received from the SDN controller;
 - receive data packets from the selected network flow;
 - determine whether the data packets have been tagged with the first label;
 - in response to determining the data packets having been tagged with the first label, remove the first label from the data packets; and
 - forward the number of data packets with the first label and the content associated to the data packets with the first label to the SDN controller.
11. The system according to claim 10 wherein the second switch is further configured to:
 - remove the second label from the data packets in response to determining the data

packets not having been tagged with the first label.

12. The system according to claim 11 wherein the second switch is further configured to:
 compute a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

13. The system according to claim 5 or 6 wherein the third switch is configured to:
 update flow tables and group tables according to the orders received from the SDN controller;
 receive data packets from the selected network flow;
 count the number of data packets with the first label; and
 forward the number of data packets with the first label to the SDN controller.

14. The system according to claim 13 wherein the third switch is configured to:
 forward the content associated to the data packets with the first label to the SDN controller.

15. The system according to claim 13 or 14 wherein the third switch is further configured to:
 compute a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

16. A method for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) having a SDN controller and multiple switches, the method comprising:

generating and transmitting orders to a first switch and a second switch to return a first detection data and a second detection data of a network flow;

receiving the first detection data and the second detection data, the first detection data being the number data packets from the network flow with a first label reported by the first and second switches, and the second detection data being a content associated to the data packets from the network flow with the first label reported by the first and second switches;

performing an analysis on the first and second detection data; and

raising an action ticket if the analysis on the first and second detection data fails.

17. The method according to claim 16 wherein the step of performing an analysis on the first and second detection data comprises:

determining whether the number of data packets received from the first switch is equal to the number of data packets received from the second switch; and

determining whether the contents associated to data packets received from the first and second switches are the same.

18. The method according to claim 16 or 17 further comprising:

generating and transmitting orders to a third switch, the third switch being an intermediate switch randomly selected between the first and second switches in the forwarding path of the network flow.

19. The method according to claim 18 wherein the step of performing an analysis on the first and second detection data comprises:

determining whether the number of data packets received from each of the first, second and third switches are the same; and

determining whether the content associated to the data packets received from each of the first and second switches are the same.

20. The method according to claims 18 wherein the step of performing an analysis on the first and second detection data comprises:

determining whether the number of data packets received from each of the first, second and third switches are the same; and

determining whether the content associated to the data packets received from each of the first, second and third switches are the same.

21. A controller for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN), the SDN controller having a non-transitory memory, a processor and instructions stored on the memory executable by the processor to:

generate and transmit orders to a first switch and a second switch to return a first detection data and a second detection data of a network flow;

receive the first detection data and the second detection data, the first detection data being the number data packets from the network flow with a first label reported by the first and second switches, and the second detection data being a content associated to the data packets from the network flow with the first label reported by the first and second switches;

perform an analysis on the first and second detection data; and
 raise an action ticket if the analysis on the first and second detection data fails.

22. The controller according to claim 21 wherein the instruction to perform an analysis on the first and second detection data comprises instructions stored on the memory executable by the processor to:

determine whether the number of data packets received from the first switch is equal to the number of data packets received from the second switch; and

determine whether the contents associated to data packets received from the first and second switches are the same.

23. The controller according to claim 21 or 22 wherein the instruction further comprises instructions stored on the memory executable by the processor to:

generate and transmit orders to a third switch, the third switch being an intermediate switch randomly selected between the first and second switches in the forwarding path of the network flow.

24. The controller according to claim 23 wherein the instruction to perform an analysis on the first and second detection data comprises instructions stored on the memory executable by the processor to:

determine whether the number of data packets received from each of the first, second and third switches are the same; and

determine whether the content associated to the data packets received from each of the first and second switches are the same.

25. The controller according to claims 23 wherein the instruction to perform an analysis on the first and second detection data comprises instructions stored on the memory executable by the processor to:

determine whether the number of data packets received from each of the first, second and third switches are the same; and

determine whether the content associated to the data packets received from each of the first, second and third switches are the same.

26. A method for a switch in verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller and the switch, the method comprising:

updating flow tables and group tables according to orders received from the SDN controller;

receiving and sampling data packets from the selected network flow;

tagging sampled data packets with the first label; and

forwarding the data packets to the next hop and a content associated to the data packets with the first label to the SDN controller and report the number of sampled data packets to the SDN controller.

27. The method according to claim 26 wherein the switch is in an expected forwarding path of the network flow.

28. The method according to claims 26 or 27 wherein the method further comprising:
tagging non-sampled data packets with the second label.

29. The method according to anyone of claims 26-28 wherein the method further comprising:

computing a message digest of all the sampled data packets to determine the content associated to the data packets with the first label.

30. A switch for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller and the switch; the switch having a non-transitory memory, a processor and instructions stored on the memory executable by the processor to:

update flow tables and group tables according to orders received from the SDN controller;

receive and sample data packets from the selected network flow;

tag sampled data packets with the first label; and

forward the data packets to the next hop and a content associated to the data packets with the first label to the SDN controller and report the number of sampled data packets to the SDN controller.

31. The switch according to claim 30 wherein the switch is in an expected forwarding path of the network flow.

32. The switch according to claims 30 or 31 wherein the instruction further comprises instructions stored on the memory executable by the processor to:

tag non-sampled data packets with the second label.

33. The switch according to anyone of claims 30-32 wherein the instruction further comprises instructions stored on the memory executable by the processor to:

compute a message digest of all the sampled data packets to determine the content associated to the data packets with the first label.

34. A method for a first switch and a second switch in verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller, the first switch and the second switch; the method comprising:

updating flow tables and group tables according to the orders received from the SDN controller;

receiving data packets from the selected network flow;

determining whether the data packets have been tagged with a first label;

in response to determining the data packets having been tagged with the first label, removing the first label from the data packets; and

forwarding the number of data packets with the first label and the content associated to the data packets with the first label to the SDN controller.

35. The method according to claim 34 wherein the first switch and the second switch are two switches in an expected forwarding path of the network flow, and the second switch is a downstream switch with respect to the first switch.

36. The method according to claims 34 or 35 wherein the method further comprising:

Removing, by the second switch, a second label from the data packets in response to determining the data packets not having been tagged with the first label; wherein the second label is marked by the first switch.

37. The method according to anyone of claims 34-36 wherein the method further comprising:

computing a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

38. A second switch for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller a first switch and the second switch; the second switch having a non-transitory memory, a processor and

instructions stored on the memory executable by the processor to:

- update flow tables and group tables according to the orders received from the SDN controller;

- receive data packets from the selected network flow;

- determine whether the data packets have been tagged with a first label;

- in response to determining the data packets having been tagged with the first label, removing the first label from the data packets; and

- forward the number of data packets with the first label and the content associated to the data packets with the first label to the SDN controller.

39. The second switch according to claims 38 wherein the instruction further comprises instructions stored on the memory executable by the processor to:

- remove a second label from the data packets in response to determining the data packets not having been tagged with the first label; wherein the second label is marked by the first switch.

40. The second switch according to claims 38 or 39 wherein the instruction further comprises instructions stored on the memory executable by the processor to:

- compute a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

41. A method for a third switch in verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller, a first switch, a second switch and the third switch, the method comprising:

- receiving data packets from a selected network flow;

- counting the number of data packets with a first label; and

- forwarding the number of data packets with the first label to the SDN controller.

42. The method according to claim 41 wherein the method further comprising:

- forwarding a content associated to the data packets with the first label to the SDN controller.

43. The method according to claim 41 or 42 wherein the method further comprising:

- computing a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.

44. A third switch for verifying a veracity of data packets being forwarded in a Software Defined Network (SDN) comprising a SDN controller, a first switch a second switch and the third switch; wherein the first and the second switch are two switches in an expected forwarding path of the network flow, the first switch is an ingress switch and the second switch is an egress switch; the third switch is located between the first switch and the second switch;

the third switch having a non-transitory memory, a processor and instructions stored on the memory executable by the processor to:

update flow tables and group tables according to orders received from the SDN controller;

receive and sample data packets from the selected network flow;

tag sampled data packets with the first label; and

forward the data packets to the next hop and a content associated to the data packets with the first label to the SDN controller and report the number of sampled data packets to the SDN controller.

45. The third switch according to claim 46 wherein the instruction further comprises instructions stored on the memory executable by the processor to:

forward a content associated to the data packets with the first label to the SDN controller.

46. The third switch according to claims 44 or 45 wherein the instruction further comprises instructions stored on the memory executable by the processor to:

compute a message digest of all the data packets with the first label to determine the content associated to the data packets with the first label.