TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 06/09 pp400–412
Volume 22, Number 4, August 2017

RouteGuardian: Constructing Secure Routing Paths in Software-Defined Networking

Mengmeng Wang, Jianwei Liu, Jian Mao*, Haosu Cheng, Jie Chen, and Chan Qi

Abstract: Software-Defined Networking (SDN) decouples the control plane and the data plane in network switches and routers, which enables the rapid innovation and optimization of routing and switching configurations. However, traditional routing mechanisms in SDN, based on the Dijkstra shortest path, do not take the capacity of nodes into account, which may lead to network congestion. Moreover, security resource utilization in SDN is inefficient and is not addressed by existing routing algorithms. In this paper, we propose RouteGuardian, a reliable security-oriented SDN routing mechanism, which considers the capabilities of SDN switch nodes combined with a Network Security Virtualization framework. Our scheme employs the distributed network security devices effectively to ensure analysis of abnormal traffic and malicious node isolation. Furthermore, RouteGuardian supports dynamic routing reconfiguration according to the latest network status. We prototyped RouteGuardian and conducted theoretical analysis and performance evaluation. Our results demonstrate that this approach can effectively use the existing security devices and mechanisms in SDN.

Key words: Software-Defined Networking (SDN); network security virtualization; capacity-based routing; security-oriented routing; dynamic routing reconfiguration

1 Introduction

Software-Defined Networking (SDN) is a typical centralized network architecture for managing and operating networks. It facilitates network management and eases the burden of solving networking problems via the logically centralized control offered by a controller[1–7]. SDN decouples the control layer from the data layer and provides new ways for the dynamic control and management of packet forwarding and processing in switches. In SDN, a centralized controller defines network behaviors and configures network devices via a set of policies, which control where network traffic flows, e.g., whether or when network traffic should go through a particular security device. Therefore, the network intelligence in SDN is logically centralized in the controllers, while the devices in the infrastructure layer are simple packet-forwarding devices.

Many security modules, devices, and middle-boxes are employed to improve the security of SDN networks[8–16]. Although these security resources can provide many security benefits to SDN networks, they may not be deployed in the physical locations that can best meet the diverse and increasing security demands of different users. SDN offers the opportunity to use security resources in a network flexibly. For example, Shin et al.[17] presented the concept of Network Security Virtualization (NSV), which uses SDN technology to virtualize security functions and resources to network administrators/users, and thus improve the utilization of existing security devices. However, NSV does not consider network capacity when virtualizing security

• Mengmeng Wang, Jianwei Liu, Jian Mao, Haosu Cheng, Jie Chen, and Chan Qi are with School of Electronic and Information Engineering, Beihang University, Beijing 100191, China. E-mail: jane10603@126.com; liujianwei@buaa.edu.cn; maojian@buaa.edu.cn.

∗ To whom correspondence should be addressed.

Manuscript received: 2016-11-20; revised: 2016-12-28; accepted: 2017-01-10

resources in the network, which introduces unexpected network loads. If the load exceeds the network capacity near a security device, this results in congestion and denial of service.

In this paper, we propose RouteGuardian, a reliable security-oriented routing mechanism, which enables the SDN controller to make full use of security resources and ensures the reliability of established routing paths. RouteGuardian provides a weighted shortest-path routing algorithm, in which the weighting is derived from the network nodes' capabilities, including the network and security capabilities. RouteGuardian supports adaptive routing path reconfiguration when the controller perceives practical congestion caused by attack events or other network accidents in the established paths.

We prototyped our approach and deployed RouteGuardian on a POX controller[18]. We extended the existing Application Layer and POX controller with RouteGuardian modules. We evaluated the effectiveness and performance of RouteGuardian and demonstrated its effectiveness.

Our Contributions. In summary, we make following contributions in this paper:

- We propose a reliable security-oriented SDN routing mechanism, RouteGuardian, according to the capabilities of SDN switch nodes combined with an NSV framework. Our scheme effectively employed the distributed network security devices to ensure analysis of abnormal traffic and malicious node isolation. Furthermore, RouteGuardian supports dynamic routing reconfiguration according to the latest network status.
- We develop a reliable security-oriented routing algorithm. The proposed algorithm takes the network and security capabilities of network switches as inputs, and makes use of the k-shortest path algorithm to ensure minimum cost to the network when establishing a routing path. Our algorithm offers a good balance of efficiency, availability, reliability, and security.
- Finally, we implement a prototype of RouteGuardian, and evaluate its performance. The results demonstrate that our approach can optimally use the existing security devices and mechanisms in SDN, and effectively ensure the abnormal flow isolation with dynamic routing path reconfiguration .

Paper Organization. The rest of this paper is organized as follows: Section 2 illustrates the background to our work. Section 3 illustrates the proposed reliable security-oriented routing scheme. Section 4 describes the detailed system design of RouteGuardian. Section 5 presents the prototype implementation of RouteGuardian and analyzes the performance of the system. Section 6 discusses related work and Section 7 concludes the paper.

2 Background

We present a novel model for constructing secure routing paths through security entities and nodes with high capabilities in SDN. This will improve the security of data delivery in SDN and prevent adversaries from launching attacks through malicious or non-trusted node selection.

The method of constructing secure routing paths is significantly different in SDN compared with traditional networks. As illustrated in Fig. 1a, the security policies in traditional networks are enforced by physically forcing traffic to flow through a certain device (e.g., an intrusion detection system, or a firewall). However, SDN topology is virtual. As illustrated in Fig. 1b, the logically centralized controller in SDN provides a high-level view of the whole network to control programs. This means that the controller has a strong ability to control network flow and can deploy security policies generated by corresponding applications to switches.

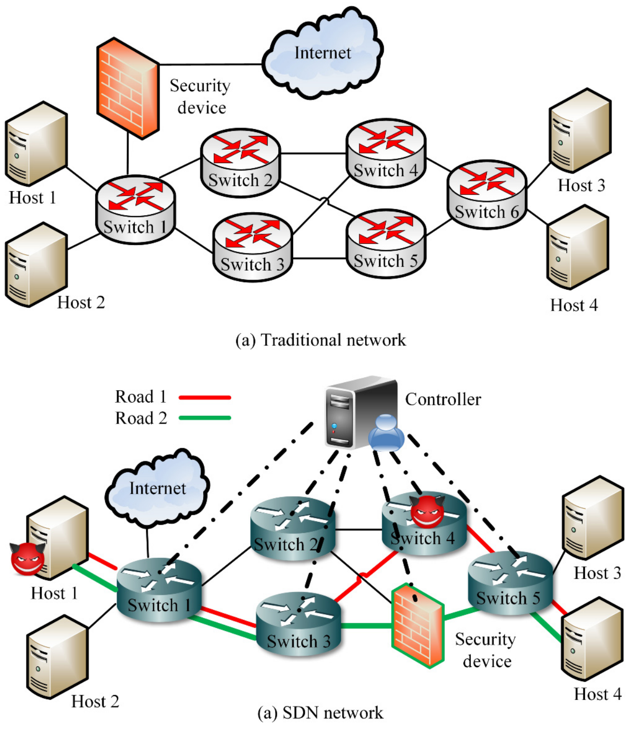


Fig. 1 Routing path construction in SDN and traditional networks.

Therefore, the architecture of SDN makes it more flexible to control if and when the network traffic goes through a security device.

For example, in an SDN network, as illustrated in Fig. 1b, if Host 1, which is controlled by an attacker, wants to visit Host 4 for a malicious service, Host 1 should first send its request to the nearest switch. As switch 1 is unable to find a flow rule/policy to respond to the request that requires rerouting, it reports this request to the SDN controller as a *Packet-In* message. With centralized control of the SDN controller, the routing application that communicates with it can build a global view of the topology of all the switches connected to the controller. Then, the controller runs a routing algorithm, based on the current topology information, to compute a new route from the source to the destination, and pushes a route update to the involved switches for future communication between Hosts 1 and 4. Then, if the green road (Road 2) in Fig. 1b is pushed to these involved switches, once the compromised Host 1 is detected by the security device deployed in this road, Host 1 is immediately isolated. However, if the red road (Road 1) in Fig. 1b is pushed, as there is no filtering or security protection in this road, Host 4 would be attacked.

Routing rules/policies in SDN, which are assigned by the controller to switches, control where and when traffic flow goes through a certain device. If the controller does not consider the network and security capabilities of the nodes in the SDN network, it cannot find the optimal routing paths that match the reliability and security requirements of the users.

With increasing security demands, more and more nodes need to be deployed in the already complicated SDN networks. Thus, the centralized controller is required to push more security polices when constructing routing paths, which makes constructing secure routing paths in SDN more and more error-prone and challenging.

3 Reliable Security-Oriented Routing Scheme

In this section, we present our reliable security-oriented routing scheme. The SDN network topology is shown in Fig. 2 and consists of three types of entities: *Controller*, *Hosts*, and *Switches*. Hosts 1, 3, and 4 are benign hosts and Host 2 is a malicious client. The nodes $S1, \cdots, S11$ are switch nodes. Switch nodes

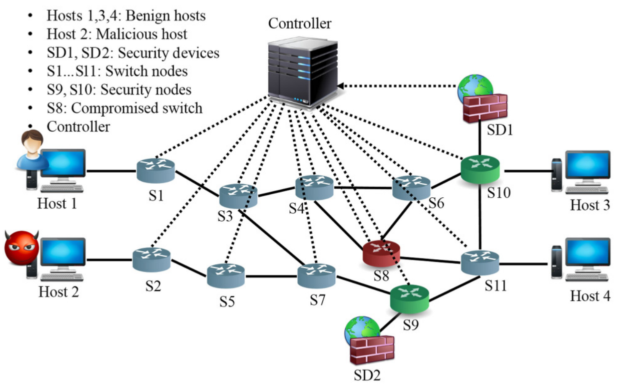


Fig. 2 Overall topology.

equipped with security resources (e.g., firewall or IDS) are called security nodes, e.g., S9 and S10 in Fig. 2, which are equipped with security devices SD1 and SD2, respectively.

3.1 Problem definition

We first define the problem addressed in this paper. The network is represented as a graph $G = (V, E)$. V is a set of nodes, where each node represents a switch in the SDN network. E is a set of edges, which represents the connections between the SDN switches. A switch $v_i \in V$ has several properties that will be discussed in Section 3.2. With this notion of network, our problem is formulated as follows.

Given a source node $v_{\rm s}$, a destination node $v_{\rm d}$, and a set of security requirements $R = \{r_1, r_2, \cdots, r_n\}$, find the max-capacity path from $v_{\rm s}$ to $v_{\rm d}$ that passes nodes satisfying the requirement set R.

To achieve this, in Section 3.2, we will present our basic routing algorithm that ensures minimum network cost and maximum reliability when selecting the routing path. Taking the security requirements into account, we will propose our security-oriented routing path algorithm in Section 3.3.

3.2 Basic algorithm—Network capability based routing

3.2.1 Network capability

The capability of each node in an SDN network is attached to its unique identity, which is used in making routing decisions. We represent the network capability of a switch node in SDN by a multi-attribute vector, where each attribute indicates the tendency of the switch node to conduct a specific action[19]. The capability of a switch node v_i is defined as a k-

dimensional vector of T_i, shown as follows:

$$T_i = \{t_i^{(1)}, t_i^{(2)}, \cdots, t_i^{(k)}\} \tag{1}$$

where $t_i^{(j)}$ stands for the j-th dimension of the capability of node v_i, $j \in \{1, 2, \cdots, k\}$, and each dimension $t_i^{(j)} \in [0, 1]$ corresponds to one action $\text{Action}_i^{(j)}$.

The network capability of a node is calculated based on its past behavior and is defined as the probability that the node will behave the same in the next period. We consider the following three actions of a network node, similar to the model proposed by Mahmoud et al.[19]

$t_i^{(1)}$: **The probability of successfully relaying a packet.** Let N_{relay} be the number of packets that are relayed in the last λ sessions and N_{total} be the total number of incoming packets in the last λ sessions, then, $t_i^{(1)}$ depicts the probability that v_i will relay a packet successfully.

$$t_i^{(1)} = \frac{N_{\text{relay}}}{N_{\text{total}}} \tag{2}$$

$t_i^{(2)}$: **The probability of not breaking a route.** Let N_{broken} be the number of sessions broken by v_i in the last λ sessions, then, the capability value $t_i^{(2)}$ depicts the probability that v_i will not break a route in the last λ sessions.

$$t_i^{(2)} = 1 - \frac{N_{\text{broken}}}{\lambda} \tag{3}$$

$t_i^{(3)}$: **The probability of relaying at least ϵ packets in a session.** Let $N_{s-\text{relay}}^{\epsilon}$ be the number of sessions that v_i relayed in at least ϵ packets, then, $t_i^{(3)}$ depicts the percentage that v_i relayed at least ϵ packets in the last λ sessions.

$$t_i^{(3)} = \frac{N_{s-\text{relay}}^{\epsilon}}{\lambda} \tag{4}$$

3.2.2 Reliability of a routing path

To ensure the reliability of a routing path, we take two types of network capability into account: *node capability* and *link capability*. Link capability is usually measured from the link's bandwidth between two nodes. The bandwidth requirements (specified by the users) of the network links should be satisfied, before we evaluate the reliability of the routing path based on the nodes' capabilities. As the SDN controller obtains the status of the network resources periodically, it can create a refined network topology, in which the links satisfy the bandwidth requirements. In the rest of this paper, we take the refined topology as the input for our algorithm, which means the link bandwidth is already satisfied, and our algorithms should evaluate the reliability of the routing path according to node capabilities only.

Given a routing path $p = \{v_1, \cdots, v_n\}$, the corresponding reliability attributes of p are depicted as $T_p^{(j)}$, which depicts the probability that the j-th action will be conducted in all nodes on the path p, where $j \in \{1, 2, 3\}$. $T_p^{(j)}$ can be calculated according to Eq. (5).

$$T_{(p)}^{(j)} = \prod_{v_i \in \{v_1, \cdots, v_n\}} t_{v_i}^{(j)} \tag{5}$$

We aggregate the reliability properties of a routing path and compute its reliability by following Eq. (6).

$$T(p) = \sum_{j=1}^{k} T_{(p)}^{(j)} \omega_j \tag{6}$$

where ω_j is a weight corresponding to the reliability attributes $T_{(p)}^{(j)}$, where $j \in \{1, 2, 3\}$. ω_j reflects the sensitivities of each attribute specified by users in the SDN network. Given a specified sensitivity vector $(\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)})$, the weight value ω_j ($j \in \{1, 2, 3\}$) is normalized as

$$\omega_j = \frac{\alpha^{(j)}}{\sum_{k=1}^{3} \alpha^{(k)}} \tag{7}$$

We illustrate the detailed process of our Network Capability based Routing algorithm in Algorithm 1. If

Algorithm 1 Network Capability based Routing Algorithm

```
Input: start node: $v_s$
       destination node: $v_d$
       bandwidth requirements: br
Output: the path with the max-capacity from $v_s$ to $v_d$
1: BEGIN
2: The SDN controller detects topology of the network based on the bandwidth requirements
3: Compute a set of possible paths $P_{\text{paths}} = \{P_1, \cdots, P_K\}$ with the $K$-shortest path algorithm, which consists $K$ different paths from $v_s$ to $v_d$
4: The SDN controller collects the latest status of all the nodes in $P_{\text{paths}}$
5: for each path $P_K \in P_{\text{paths}}$ do
6:     $T(\text{route})_{P_K}^{(j)} = \prod_{\text{Node}_i \in \{\text{Node}_1, \cdots, \text{Node}_n\}} t_{\text{Node}_i}^{(j)}$
       $T(\text{route})_{P_K} = \sum_{j=1}^{K} T(\text{route})_{P_K}^{(j)} \omega_j$
7: Select the path with the max-capacity from $P_{\text{paths}}$ as the routing path
8: Prepare the routing path information and distribute them to the corresponding switches
9: END
```

the bandwidth requirement from a user is not given, the controller detects the topology of the network without considering the link bandwidth. Therefore, depending on these various parameters in real time, nodes with low capabilities will not have a chance to take part in routes as they significantly degrade route reliability.

Figure 3 shows an example clarifying the process of using our Network Capability based Routing algorithm. As illustrated in Fig. 3, there are six possible routing paths from the source node S1 to the destination node S7, and the first column in Table 1 illustrates the detailed information on each path. In this example, the parameter K is set as $K = 6$, so the K-shortest path algorithm outputs six routing paths.

Then, the SDN controller collects the latest status of all the nodes along these six routing paths, as shown in Table 2. Suppose $\omega_1 = \omega_2 = \omega_3 = \frac{1}{3}$, the third column in Table 1 shows the capability value of each possible routing path. Therefore, route2 will be output as the last routing path by the Network Capability based Routing algorithm, because it is the path with the max-capacity and a relatively low communication overhead (it takes 4 hops) from S1 to S7.

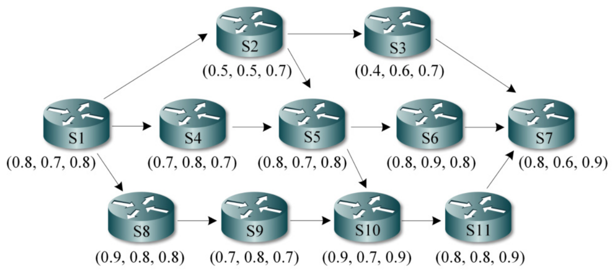


Fig. 3 Possible routing paths and nodes status.

Table 1 Capability of each possible routing path.

Case	Route path	Capability
route1	(S1→ S2→ S3→ S7)	0.20
route2	(S1→ S4→ S5→ S6→ S7)	0.27
route3	(S1→ S8→ S9→ S10→ S11→ S7)	0.26
route4	(S1→ S2→ S5→ S6→ S7)	0.22
route5	(S1→ S2→ S5→ S10→ S11→ S7)	0.20
route6	(S1→ S4→ S5→ S10→ S11→ S7)	0.24

Table 2 The latest status of all the nodes in P_{paths}.

Node	$t^{(1)}$	$t^{(2)}$	$t^{(3)}$
S1	0.8	0.7	0.8
S2	0.5	0.5	0.7
S3	0.4	0.6	0.7
S4	0.7	0.8	0.7
S5	0.8	0.7	0.8
S6	0.8	0.9	0.8
S7	0.8	0.6	0.9
S8	0.9	0.8	0.8
S9	0.7	0.8	0.7
S10	0.9	0.7	0.9
S11	0.8	0.8	0.9

3.3 Security-oriented routing and dynamic reconfiguration

As different users in SDN usually specify or adjust their security requirements according to the current network status, we present the Security-oriented Routing and Dynamic Reconfiguration Algorithm based on the capacity of each network node and the realtime security requirements of users. As illustrated in Algorithm 2, this algorithm considers how network packets should pass through specific security devices to meet the security requirements from different users. Meanwhile, it also considers the various security devices and functions for routing path construction, and aims to help the controller choose reasonable security devices based on the security demands from different users.

Figure 4a is the original network topology, which contains eleven SDN-enabled switches (denoted as S1–S11). Among these switches, S1 is the start node, and S7 is the destination node. Figure 4b shows traditional packet delivery, which is based on shortest path routing without considering the capability of nodes or the

Algorithm 2 Security-oriented Routing and Dynamic Reconfiguration Algorithm

```
Input: start node: $v_s$
       destination node: $v_d$
       bandwidth requirements: br
       security requirements: $R = \{r_1, r_2, \cdots, r_n\}$
Output: the max-capacity path meets $R$ from $v_s$ to $v_d$
1: BEGIN
2: The SDN controller detects the security devices in the network
3: Analyse $R$ and get the set of security devices $D = \{sd_1, sd_2, \cdots, sd_n\}$ that meets $R$
4: Find the max-capacity path $P(v_s, sd_1)$ from $v_s$ to $sd_1$ with Algorithm 1
5: for each $sd_\iota \in D$ do
6:    if $\iota > 1$
7:       find the max-capacity path $P_{(sd_{\iota-1}, sd_\iota)}$ from $sd_{\iota-1}$ to $sd_\iota$ with the Algorithm 1
8: Find the max-capacity path $P(sd_n, v_d)$ from $sd_n$ to $v_d$ with Algorithm 1
9: The max-capacity path from $v_s$ to $v_d$ that meets $R$ is:
   $\{P(v_s, sd_1), P(sd_1, sd_2), \cdots, P(sd_{n-1}, sd_n), P(sd_n, v_d)\}$
10: Prepare the routing path information and distribute them to the corresponding security devices and switches
11: END
```

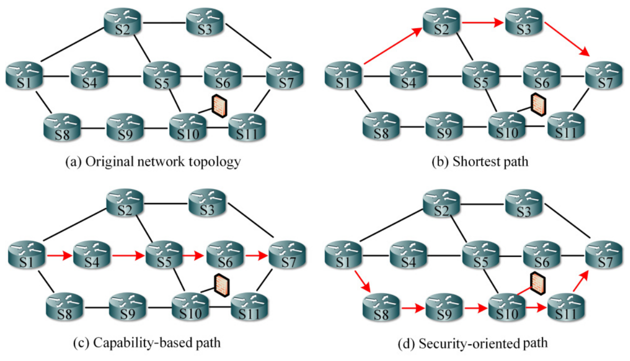

S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
(a) Original network topology
(b) Shortest path
(c) Capability-based path
(d) Security-oriented path

Fig. 4 Example scenario for our security-oriented routing algorithm.

security requirements from users. Thus, packets from the start node S1 are simply delivered to S7 through the path (S1→ S2→ S3 → S7).

Then, we describe how our new algorithms work and illustrate them using the same network structure. Figure 4c shows an example of constructing routing paths while considering the capability of nodes and routes in an SDN network. Based on the status information collected by the SDN controller along the possible K routing paths, the Network Capability based Routing Algorithm computes the capability of each path, and selects the max-capacity path as the last routing path. Therefore, in this case, all the packets from S1 to S7 would be forwarded through the path (S1 → S4 → S5 → S6 → S7) by running our Capability-based Routing Algorithm, as illustrated in Algorithm 1.

Figure 4d shows our example when the security requirement from a user is specified, which demands that all packets from the start node S1 to the destination node S7 should be inspected by a security device (e.g., an IDS or a firewall). In this scenario, when the SDN controller detects the security devices in the network, it could find that switch S10 is attached to a firewall, which could meet the security requirement of this user. Therefore, our Security-oriented Routing Algorithm would first find the max-capacity path from S1 to S10 with Algorithm 1, then, find the max-capacity path from S10 to S7. Finally, it outputs (S1 → S8 → S9 → S10 → S11) as the last routing path.

Therefore, our Security-oriented Routing Algorithm will help the SDN controller deliver and redirect the flow traffic in SDN to different security nodes, and deploy security entities into reasonable places, thus providing more secure routing path for users and improving the security resource utilization in SDN.

4 RouteGuardian

In this section, we present the system design of the security-oriented routing path mechanism, RouteGuardian, which resides in the SDN controller. In our system, RouteGuardian timely monitors the status of network resources in the SDN, thereby perceiving the network and security capabilities of the network nodes. RouteGuardian calculates the security-oriented routing paths based on the transmission and security requirements specified by users and the network nodes' capabilities. Benefiting from the network resource virtualization infrastructure, RouteGuardian ensures that the established routing path satisfies the network security and reliability requirements effectively. If there is any abnormal traffic detected by the network security resources deployed in the selected routing path, RouteGuardian will react in a timely manner according to the security policies and dynamically reconfigure the routing paths.

4.1 Overall architecture

As shown in Fig. 5, RouteGuardian extends regular controllers with four additional modules: (1) *Policy Parser*, (2) *Resource Status Monitor*, (3) *Routing Rule Generator*, and (4) *Incident Reactor*.

Policy Parser. This module is an interface that mediates a set of high-level security requirements into the corresponding security policies. For example, if the security requirement of a user in SDN specifies that all network packets from Host 1 to port 80 of Host 2 should be blocked, then the Policy Parser module translates this security requirement into a corresponding security policy that the Routing Rule Generator module can accept when constructing routing paths. As illustrated in Fig. 5, if SD2 (security device) just has a rule to block

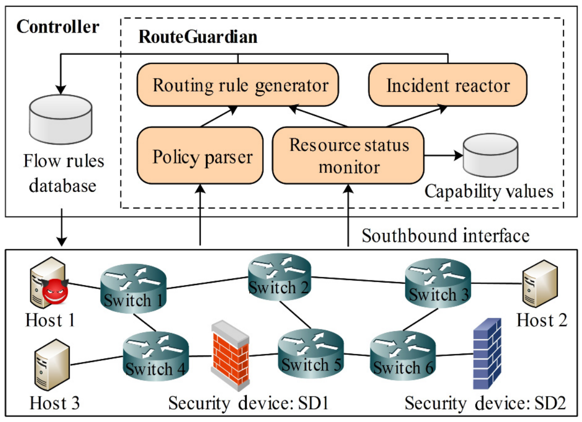


Fig. 5 Conceptual architecture of RouteGuardian.

network packets from Host 1 to port 80 of Host 2, then SD2 would be selected as the necessary passing security device when the Routing Rule Generator constructs the secure routing paths from Host 1 to Host 2.

Resource Status Monitor. As the status of nodes and the bandwidth in SDN is dynamic and time-sensitive, the Resource Status Monitor module periodically collects the status of switch nodes, bandwidth, and security devices in the network, and stores this information in the Capability Value database. The information collected by this module is critical metrics/input data for the Routing Rule Generator module to construct an optimal routing path that meets the security requirements of a user.

Routing Rule Generator. The Network Capability based Routing Algorithm and the Security-oriented Routing and Dynamic Reconfiguration Algorithm run in this module. This module constructs routing paths based on the latest status of the nodes and the various security needs of users in SDN. Specifically, based on the realtime security requirements of a user in SDN, this module outputs the max-capacity path from the source node to the destination node satisfying the security requirements.

Incident Reactor. This module creates response strategies corresponding to the security policies, e.g., isolating the malicious node/host that creates abnormal traffic, dropping the malicious packets, etc. Meanwhile, the Resource Status Monitor module updates the capability status of the network resource and the SDN controller reconfigures the routing paths according to the outputs of the Routing Rule Generator module.

4.2 Typical operations of RouteGuardian

In this subsection, we use Fig. 6 to illustrate typical operations of RouteGuardian.

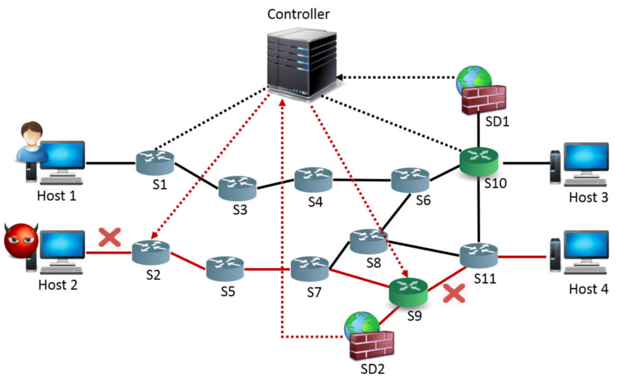


Fig. 6 Secure routing path establishment and security response.

Security-oriented routing path construction. When a user applies for a secure routing path from its host node Host 1 to a destination node Host 3, Host 1 should first send its request to the nearest switch S1. Then, S1 sends a routing establishment request to the controller. The Policy Parser module creates the corresponding reliability and security policies according to the capability and security requirements included in the routing request sent by Host 1. The Resource Status Monitor module timely collects the capability status of the network resources. The Routing Rule Generator module calculates and outputs routing/flow rules according to the polices and capability status of the network resources using the Security-oriented Routing and Dynamic Reconfiguration Algorithm. The controller assigns the secure routing flow rules to the switch nodes included in the routing path, and establishes the required route ensuring the security and reliability requirements simultaneously.

Security Response and Reconfiguration. Once there is malicious traffic (such as the red-line shown in Fig. 6) inspected by a security node (e.g., S9), the security node will inform the controller. The Incident Reactor module creates the response strategies corresponding to the security policies, e.g., isolating the malicious node/host (e.g., Host 2) that creates the abnormal traffic, dropping the malicious packets, etc. Meanwhile, the Resource Status Monitor module updates the capability status of the network resources and the controller reconfigures the routing paths according to the output of the Routing Rule Generator module.

5 Evaluation

We prototyped our approach, RouteGuardian, and its functionality and performance overhead are evaluated in this section. We select Mininet[20], which is popularly used for emulating OpenFlow network environments, to emulate our network topologies. We also extended the POX controller with four additional modules as illustrated in Fig. 5 to support the main functionality of RouteGuardian. We set up the experiment on a 3.40 GHz Intel Core(TM) I3-2130 platform with 6 GB RAM, running 32-bit Ubuntu 12.10 (Kernel version 3.8.0).

We compared RouteGuardian with the shortest path algorithm from two aspects: (1) the method of routing path construction, and (2) the performance overhead introduced by the routing algorithm.

5.1 Routing path comparison

Figure 7 illustrates the network topology used in our experiment. Our goal is to show the main differences in the method of routing path construction between RouteGuardian and the basic shortest path algorithm. In our experiment, Host 1 requests to send a packet to port 80 of Host 2, while Host 2 has specified a security requirement that all network traffic from Host 1 to port 80 should be audited by a security device. Meanwhile, Switch 3 (with IP 192.168.1.3) is a malicious node, which usually breaks a route. The ping test from Switch 2 (with IP 192.168.1.2) in this network is shown in Fig. 8.

While Host 1 requests to send a packet to port 80 of Host 2, Switch 1 (the nearest switch) reports it to the controller. In a common SDN controller, as there are no special modules for security demands, it just runs the shortest path algorithm (Dijkstra's algorithm) based on the topology information, outputs a path without any security policy, and pushes this routing path update to the involved switch(es). In this case, the path (Switch 2 → Switch 3 → Switch 5) will be selected as the optimal path. However, as Switch 3 (with IP 192.168.1.3) is a malicious node, it usually captures the network scenario, rewrites packets, and then breaks routes by replaying packets with *tcpreplay*, as shown in Fig. 9. Therefore, the routing path (Switch 2 → Switch 3 → Switch 5) will not be a secure routing path.

With RouteGuardian deployed on the controller, things are different. After resolving the network topology information, RouteGuardian accommodates the high-level security requirement from Host 2 into the corresponding security policy. As Switch 3 usually breaks a route with low capability, then it cannot be selected by RouteGuardian for routing path construction. Finally, RouteGuardian outputs the path (Switch 2 → Switch 1 → Switch 4 → Switch 5) as the optimal path, which meets the security requirement from Host 2. Meanwhile, RouteGuardian can also give a security response to these malicious events and isolate a certain malicious node. The ping test from Switch 2 (with IP 192.168.1.2) is shown in Fig. 10. We can see that Switch 3 is isolated by RouteGuardian.

5.2 Time complexity analysis

For an SDN network $G = (V, E)$, V is a set of nodes, E is a set of edges. Let n be the number of nodes in V, and m be the number of edges in E. Then, for the shortest path from the source node v_s to the destination node v_d, the time complexity of the Dijkstra's algorithm using a Fibonacci heap is $O(m + n \log n)$, and the K-shortest path algorithm[21] used in RouteGuardian has a time complexity of $O(K(m + n \log n))$.

Figure 11 illustrates the overhead between the shortest path algorithms and the K-shortest path with different K values. The X-axis corresponds to the number of nodes in an SDN network, and the Y-axis corresponds to the time consumed when constructing

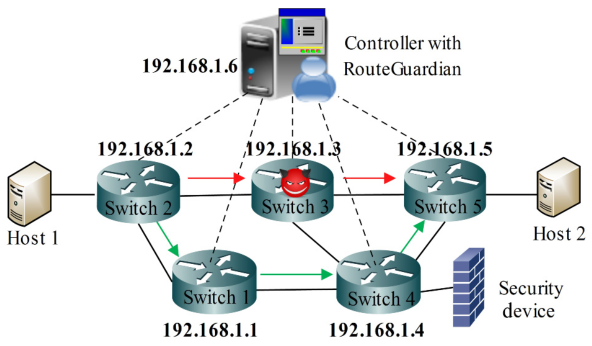
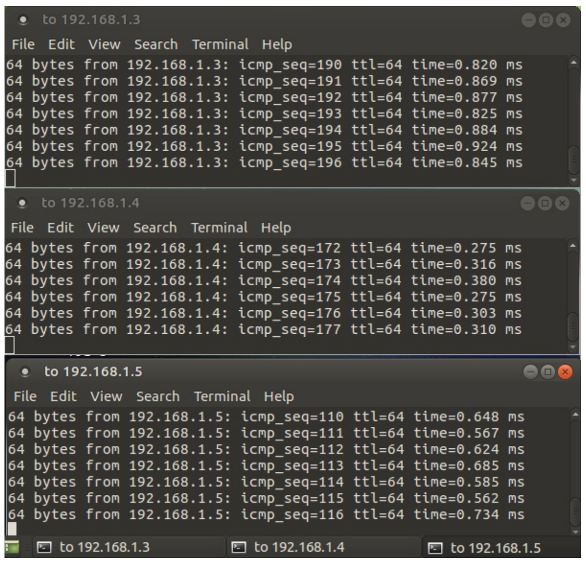


Fig. 7 Routing paths construction comparison.

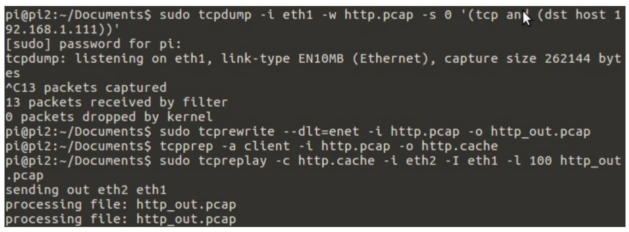

to 192.168.1.3
File Edit View Search Terminal Help
to 192.168.1.4
File Edit View Search Terminal Help
to 192.168.1.5
File Edit View Search Terminal Help
to 192.168.1.3
to 192.168.1.4
to 192.168.1.5

```
64 bytes from 192.168.1.3: icmp_seq=190 ttl=64 time=0.820 ms
64 bytes from 192.168.1.3: icmp_seq=191 ttl=64 time=0.869 ms
64 bytes from 192.168.1.3: icmp_seq=192 ttl=64 time=0.877 ms
64 bytes from 192.168.1.3: icmp_seq=193 ttl=64 time=0.825 ms
64 bytes from 192.168.1.3: icmp_seq=194 ttl=64 time=0.884 ms
64 bytes from 192.168.1.3: icmp_seq=195 ttl=64 time=0.924 ms
64 bytes from 192.168.1.3: icmp_seq=196 ttl=64 time=0.845 ms
```

```
64 bytes from 192.168.1.4: icmp_seq=172 ttl=64 time=0.275 ms
64 bytes from 192.168.1.4: icmp_seq=173 ttl=64 time=0.316 ms
64 bytes from 192.168.1.4: icmp_seq=174 ttl=64 time=0.380 ms
64 bytes from 192.168.1.4: icmp_seq=175 ttl=64 time=0.275 ms
64 bytes from 192.168.1.4: icmp_seq=176 ttl=64 time=0.303 ms
64 bytes from 192.168.1.4: icmp_seq=177 ttl=64 time=0.310 ms
```

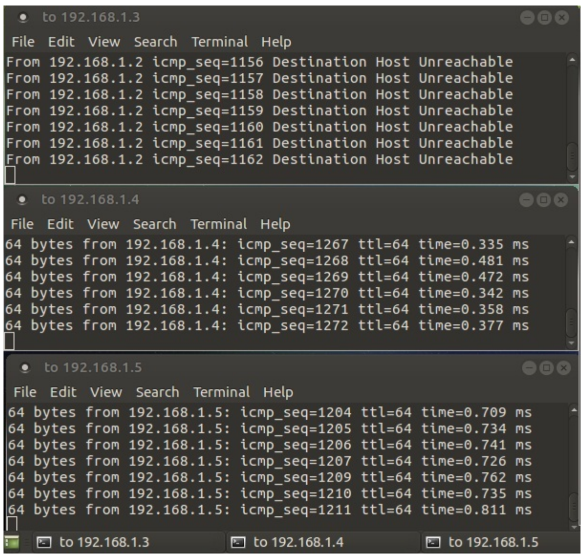
```
64 bytes from 192.168.1.5: icmp_seq=110 ttl=64 time=0.648 ms
64 bytes from 192.168.1.5: icmp_seq=111 ttl=64 time=0.567 ms
64 bytes from 192.168.1.5: icmp_seq=112 ttl=64 time=0.624 ms
64 bytes from 192.168.1.5: icmp_seq=113 ttl=64 time=0.685 ms
64 bytes from 192.168.1.5: icmp_seq=114 ttl=64 time=0.585 ms
64 bytes from 192.168.1.5: icmp_seq=115 ttl=64 time=0.562 ms
64 bytes from 192.168.1.5: icmp_seq=116 ttl=64 time=0.734 ms
```

Fig. 8 Ping test (IP 192.168.1.2).



```
pi@pi2:~/Documents$ sudo tcpdump -i eth1 -w http.pcap -s 0 '(tcp and (dst host 1
92.168.1.111))'
[sudo] password for pi:
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 byt
es
^C13 packets captured
13 packets received by filter
0 packets dropped by kernel
pi@pi2:~/Documents$ sudo tcprewrite --dlt=enet -i http.pcap -o http_out.pcap
pi@pi2:~/Documents$ tcpprep -a client -i http.pcap -o http.cache
pi@pi2:~/Documents$ sudo tcpreplay -c http.cache -i eth2 -I eth1 -l 100 http_out
.pcap
sending out eth2 eth1
processing file: http_out.pcap
processing file: http_out.pcap
```

Fig. 9 Switch 3 captures, rewrites, and replays packets.



```
to 192.168.1.3
File  Edit  View  Search  Terminal  Help
From 192.168.1.2 icmp_seq=1156 Destination Host Unreachable
From 192.168.1.2 icmp_seq=1157 Destination Host Unreachable
From 192.168.1.2 icmp_seq=1158 Destination Host Unreachable
From 192.168.1.2 icmp_seq=1159 Destination Host Unreachable
From 192.168.1.2 icmp_seq=1160 Destination Host Unreachable
From 192.168.1.2 icmp_seq=1161 Destination Host Unreachable
From 192.168.1.2 icmp_seq=1162 Destination Host Unreachable

to 192.168.1.4
File  Edit  View  Search  Terminal  Help
64 bytes from 192.168.1.4: icmp_seq=1267 ttl=64 time=0.335 ms
64 bytes from 192.168.1.4: icmp_seq=1268 ttl=64 time=0.481 ms
64 bytes from 192.168.1.4: icmp_seq=1269 ttl=64 time=0.472 ms
64 bytes from 192.168.1.4: icmp_seq=1270 ttl=64 time=0.342 ms
64 bytes from 192.168.1.4: icmp_seq=1271 ttl=64 time=0.358 ms
64 bytes from 192.168.1.4: icmp_seq=1272 ttl=64 time=0.377 ms

to 192.168.1.5
File  Edit  View  Search  Terminal  Help
64 bytes from 192.168.1.5: icmp_seq=1204 ttl=64 time=0.709 ms
64 bytes from 192.168.1.5: icmp_seq=1205 ttl=64 time=0.734 ms
64 bytes from 192.168.1.5: icmp_seq=1206 ttl=64 time=0.741 ms
64 bytes from 192.168.1.5: icmp_seq=1207 ttl=64 time=0.726 ms
64 bytes from 192.168.1.5: icmp_seq=1209 ttl=64 time=0.762 ms
64 bytes from 192.168.1.5: icmp_seq=1210 ttl=64 time=0.735 ms
64 bytes from 192.168.1.5: icmp_seq=1211 ttl=64 time=0.811 ms

to 192.168.1.3    to 192.168.1.4    to 192.168.1.5
```

Fig. 10 Security response from RouteGuardian.

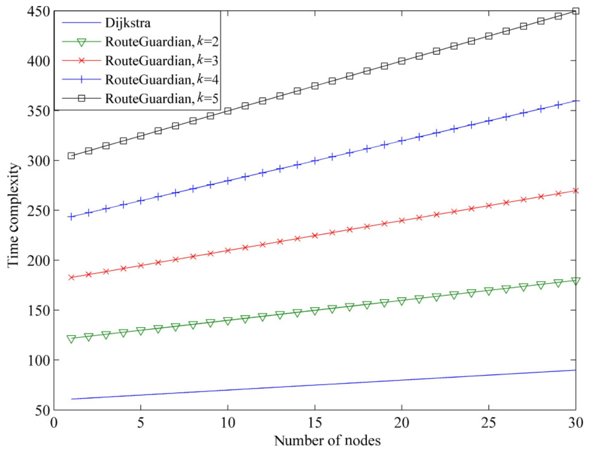


Fig. 11 Time complexity comparison.

the routing paths. Then, when the value of k is small, RouteGuardian is efficient.

5.3 Impact of network size on the performance of algorithms

The performance of RouteGuardian changes with the size of the SDN network. To investigate the impact of network size on the performance of RouteGuardian, we consider SDN networks with 20, 30, 40, 50, and 100 nodes. The detailed configuration of each network is shown in Table 3, and Fig. 12 illustrates an example of an SDN network with 50 nodes. In Fig. 12, the switch nodes with color markers, e.g., S9 and S19, are security nodes equipped with security resources (e.g., firewall, IDS, etc.). Meanwhile, the capacity of each node is uniformly chosen in the range [0.01, 1.00].

For each network size, we evaluated the performance

Table 3 Network configuration.

Case	Number of nodes	Number of links
Network 1	20	23
Network 2	30	34
Network 3	40	49
Network 4	50	63
Network 5	100	127

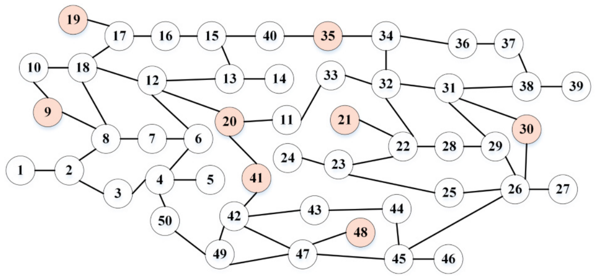


Fig. 12 Topology example for network with 50 nodes.

of our proposed algorithms. Figure 13 illustrates the time cost of RouteGuardian with different k and network sizes. Each simulation result is averaged over 50 samples. The X-axis corresponds to the SDN network size, and the Y-axis corresponds to the average time cost for RouteGuardian to compute routing paths during the processes of routing path construction. The time cost is also affected by the network topology. We observe that the performance overhead introduced by RouteGuardian for routing computation is acceptable.

6 Related Work

The field of routing in SDN has been actively researched. Here we focus on work related to our central topic, i.e., constructing secure routing paths in SDN. Two types of prior work are particularly relevant: routing paths construction in SDN, and secure routing scheme analysis in traditional networks, which

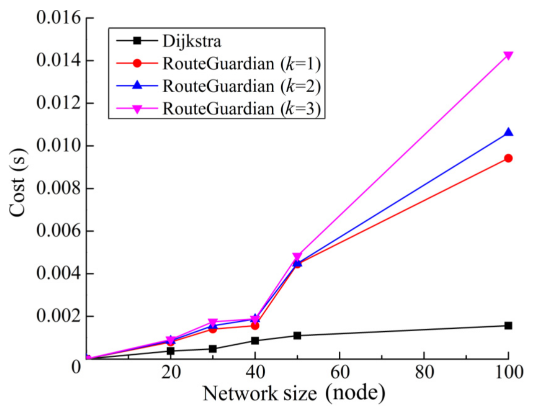


Fig. 13 Time cost with different network sizes.

considers the parameters (i.e., bandwidth, trust, node, and link capability) for routing generation.

6.1 Secure routing path construction in SDN

Recently, several studies have been proposed to address the SDN routing construction issue. To maximize the security resource utilization in SDN, Shin et al.[17] presented the concept of NSV and four basic routing algorithms, which virtualize security resources and provide security response functions from network devices when necessary. However, current routing algorithms in NSV do not consider the capability of each network node when routing network packets. Huang et al.[22] proposed a novel cost model, which simulates the usage costs of nodes and link resource, to maximize the network throughput under both critical network resources and user bandwidth demand constraints. Shen et al.[23] proposed a new reliable multicast tree for reliable multicast routing in SDN to minimize both tree and recovery costs. However, solutions[22, 23] consider many more multicast routing problems and costs in SDNs. Compared with them, we focus more on the security of routing.

Lee and Sheu[24] proposed a routing algorithm for SDN with segment routing, which considers the balance of traffic load and reduces the extra cost of packet header size in a network. Agarwal et al.[25] considered the problem of traffic engineering in the case where an SDN controller controls only a few forwarding elements, and tried to optimize the network utilization and decrease the packet drop rate. To balance the usage of link bandwidth and flow table when routing in SDN, Lee et al.[26] proposed a novel resource preference aware routing algorithm. Meanwhile, Li et al.[6] focused on the problem of routing under middlebox sequence constraints and designed a fast recovery mechanism by exploiting the remaining link and middlebox resources locally. Huang et al.[27] and Wan et al.[28] also developed an evaluation system that considered routing engineering in SDN.

In addition, Yoon et al.[29] explored the attack surface of SDN by actually attacking each layer in an SDN stack. Park et al.[30] considered how to enrich security functions/features in software-defined environments and proposed a new software switch architecture, which enables software switches to easily provide security services without additional security devices or applications.

Compared with these previous works, the primary focus of our work is security and network capability-based routing path construction. We take the network capability of these security devices and switches as a unique concern.

6.2 Secure routing path construction in traditional networks

Previous work on secure routing in traditional networks has extensively discussed trust, vulnerability, and reliability issues. These techniques are complementary to our design of RouteGuardian.

Mahmoud et al.[19] presented a trust-based routing protocol that directs network traffic to those highly-trusted nodes having sufficient energy, to minimize the probability of breaking a route. Johnson et al.[31] considered the various security concerns for route selection in anonymity networks. Chen et al.[32] proposed a dynamic trust model to optimize the secure routing protocol in DTN environments. Kang and Gligor[33] introduced the notion of routing bottlenecks, and presented their key characteristics, including size, link type, and distance from host destinations. Chen et al.[34] used symmetric cryptography for data forwarding and presented a highly-scalable anonymity system that leverages next-generation Internet architecture design.

Meanwhile, some schemes regarding mobility-aware routing and multicast routing[35–38] can also be combined with our dynamic routing path construction.

7 Conclusion

In this paper, we propose RouteGuardian, a new mechanism based on NSV, which dynamically establishes reliable secure-oriented routing paths in SDN, and aggregates switch node capabilities. RouteGuardian takes the nodes' network and security capabilities as critical metrics, thus ensuring the establishment of a reliable routing path and enforcing malicious traffic detection, isolation, and dynamic routing reconfiguration. We prototyped our approach and the experiment results demonstrate that RouteGuardian supports robust secure routing path establishment and effectively utilizes the existing security devices distributed in SDN. We will improve the performance of RouteGuardian in our future work, and aim to deploy it to distributed controllers to improve control plane scalability.

Acknowledgment

We thank the anonymous reviewers and editors for their helpful and insightful comments. This work was supported in part by the National Natural Science Foundation of China (Nos. 61402029, 61370190, and 61379002) and the National Key Basic Research Program (973) of China (No. 2012CB315905).

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, Openflow: Enabling innovation in campus networks, *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[2] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, Ethane: Taking control of the enterprise, in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM07*, Kyoto, Japan, 2007, pp. 1–12.

[3] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, SANE: A protection architecture for enterprise networks, in *15th USENIX Security Symposium*, USENIX Association, 2006, pp. 1–15.

[4] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, Software-defined networking: A comprehensive survey, *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[6] X. Li, H. Wu, D. Gruenbacher, C. Scoglio, and T. Anjali, Efficient routing for middlebox policy enforcement in software-defined networking, *Computer Networks*, vol. 110, pp. 243–252, 2016.

[7] S. Shin, L. Xu, S. Hong, and G. Gu, Enhancing network security through Software Defined Networking (SDN), in *Proceedings of the 25th International Conference on Computer Communication and Networks (ICCCN'16)*, Waikoloa, HI, USA, 2016, pp. 1–9.

[8] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, Avant-guard: Scalable and vigilant switch flow management in software-defined networks, in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS13*, Berlin, Germany, 2013, pp. 413–424.

[9] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, Securing the software-defined network control layer, in *Proceedings of 2015 Annual Network and Distributed System Security Symposium, NDSS15*, San Diego, CA, USA, 2015, pp. 1–15.

[10] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, A security enforcement kernel for openflow networks, in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN12*, Helsinki, Finland, 2012, pp. 121–126.

[11] S. Scott-Hayward, C. Kane, and S. Sezer, Operationcheckpoint: Sdn application control, in *Proceedings of the 22nd International Conference on Network Protocols, ICNP14*, 2014, pp. 618–623.

[12] S. Hong, L. Xu, H. Wang, and G. Gu, Poisoning network visibility in software-defined networks: New attacks and countermeasures, in *Proceedings of 2015 Annual Network and Distributed System Security Symposium, NDSS15*, San Diego, CA, USA, 2015.

[13] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, Towards a secure controller platform for openflow applications, in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN13*, Hong Kong, China, 2013, pp. 171–172.

[14] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, Rosemary: A robust, secure, and high-performance network operating system, in *ACM SIGSAC Conference on Computer and Communications Security, CCS14*, Scottsdale, AZ, USA, 2014, pp. 78–89.

[15] H. Wang, L. Xu, and G. Gu, Floodguard: A dos attack prevention extension in software-defined networks, in *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN15*, 2015, pp. 239–250.

[16] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith, Enabling practical software-defined networking security applications with OFX, in *Proceedings of the Network and Distributed System Security Symposium 2016, NDSS'16*, San Diego, CA, USA, 2016, pp. 1–15.

[17] S. Shin, H. Wang, and G. Gu, A first step toward network security virtualization: From concept to prototype, *Information Forensics and Security, IEEE Transactions on*, vol. 10, no. 10, pp. 2236–2249, 2015.

[18] POX, http://www.noxrepo.org/pox/about-pox/, Accessed on May 9, 2016.

[19] M. M. E. A. Mahmoud, X. Lin, and X. S. Shen, Secure and reliable routing protocols for heterogeneous multihop wireless networks, *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1140–1153, 2015.

[20] Mininet, http://mininet.org/, Accessed on May 11, 2016.

[21] N. Katoh, T. Ibaraki, and H. Mine, An efficient algorithm for k shortest simple paths, *Networks*, vol. 12, no. 4, pp. 411–427, 1982.

[22] M. Huang, W. Liang, Z. Xu, W. Xuz, S. Guo, and Y. Xu, Dynamic routing for network throughput maximization in software-defined networks, in *the 35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016*, 2016, pp. 1–9.

[23] S. Shen, L. Huang, D. Yang, and W. Chen, Reliable multicast routing for software-defined networks, in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 181–189.

[24] M. Lee and J. Sheu, An efficient routing algorithm based on segment routing in software-defined networking, *Computer Networks*, vol. 103, pp. 44–55, 2016.

[25] S. Agarwal, M. Kodialam, and T. V. Lakshman, Traffic engineering in software defined networks, in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 2211–2219.

[26] D. Lee, P. Hong, and J. Li, RPA-RA: A resource preference aware routing algorithm in software defined network, in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[27] H. Huang, S. Guo, P. Li, B. Ye, and I. Stojmenovic, Joint optimization of rule placement and traffic engineering for QoS provisioning in software defined network, *IEEE Transactions on Computers*, vol. 64, no.12, pp. 3488–3499, 2015.

[28] K. Wan, X-F. Luo, Y. Jiang, and K. Xu, The flow-oriented scheduling algorithms in SDN system, (in Chinese), *Chinese Journal of Computers*, vol. 39, no. 6, pp. 1208–1223, 2016.

[29] C. Yoon and S. Lee, Attacking SDN infrastructures: Are we ready for the next-gen networking? in *Proceedings of Black Hat USA 2016*, Las Vegas, NV, USA, 2016.

[30] T. Park, Y. Kim, and S. Shin, UNISAFE: A union of security actions for software switches, in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, SDN-NFV Security'16*, New York, NY, USA, 2016, pp. 13–18.

[31] A. M. Johnson, P. Syverson, R. Dingledine, and N. Mathewson, Trust-based anonymous communication: Adversary models and routing algorithms, in *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS11*, New York, NY, USA, 2011, pp. 175–186.

[32] I. Chen, F. Bao, M. Chang, and J. Cho, Dynamic trust management for delay tolerant networks and its application to secure routing, *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, pp. 1200–1210, 2014.

[33] M. S. Kang and V. D. Gligor, Routing bottlenecks in the Internet: Causes, exploits, and countermeasures, in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS'14*, New York, NY, USA, 2014, pp. 321–333.

[34] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig, HORNET: High-speed onion routing at the network layer, in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS'15*, New York, NY, USA, 2015, pp. 1441–1454.

[35] L. Zhang, Z. Cai, J. Lu, and X. Wang, Mobility-aware routing in delay tolerant networks, *Personal and Ubiquitous Computing*, vol. 19, no. 7, pp. 1111–1123, 2015.

[36] Z. Cai, R. Goebel, and G. Lin, Size-constrained tree partitioning: Approximating the multicast k-tree routing problem, *Theoretical Computer Science*, vol. 412, no. 3, pp. 240–245, 2011.

[37] Z. Cai, G. Lin, and G. Xue, Improved approximation algorithms for the capacitated multicast routing problem, in *Computing and Combinatorics: 11th Annual International Conference*, Springer, 2005, pp. 136–145.

[38] Z. Cai, Z. Chen, and G. Lin, A 3.4713-approximation algorithm for the capacitated multicast tree routing problem, *Theoretical Computer Science*, vol. 410, no. 52, pp. 5415–5424, 2009.

Mengmeng Wang received the BS and MS degrees in computer science and technology from North China University of Water Resources and Electric Power, Henan, China in 2010 and 2013, respectively. She is currently a PhD candidate in Lab of Information and Network Security, School of Electronic and Information Engineering, Beihang University, Beijing, China. Her current research interests include security of software-defined networking and network and information security.

Jianwei Liu received the BS and MS degrees in electronic and information from Shandong University, China in 1985 and 1988, respectively. He received the PhD degree in communication and electronic system from Xidian University, China in 1998. Now, he is a professor with School of Electronic and Information Engineering, Beihang University, Beijing, China. His current research interests include wireless communication network, cryptography, and network and information security.

Jian Mao received the BS degree and PhD degree from Xidian University, China. She is an assistant professor with School of Electronic and Information Engineering, Beihang University, Beijing, China. Her research interests include cloud security, web security, and mobile security.

Haosu Cheng received the BS degree in information technology from Fontys University, Eindhoven, the Netherlands in 2007 and MS degree in software engineering of distributed systems from Royal Institute of Technology, Stockholm, Sweden in 2010. He is currently a PhD candidate in Lab of Information and Network Security, School of Electronic and Information Engineering, Beihang University, Beijing, China. His current research interests include Internet of Things, security of Software-defined Networking, and network and information security.

Jie Chen received the MS degree in electronic engineering from the Northwest Polytechnic University, Shaanxi, China, in 2010. He is currently a PhD candidate in Lab of Information and Network Security, School of Electronic and Information Engineering, Beihang University, Beijing, China. His current research interests include software define network security, Ad hoc network security, and future network design.

Chan Qi received the BS degree from Beihang University, Beijing, China, in 2015. She is currently a master student in Lab of Information and Network Security, School of Electronic and Information Engineering, Beihang University. Her current research interests include software define network security and network and information security.