Computer Science and Information Systems 18(3):867–891 https://doi.org/10.2298/CSIS200206049G

# Intrusion Prevention with Attack Traceback and Software-defined Control Plane for Campus Networks

Guangfeng Guo[1,2], Junxing Zhang[1,⋆], and Zhanfei Ma[2]

[1] College of Computer Science, Inner Mongolia University
010021 Hohhot, China
guoguangfeng@163.com, junxing@imu.edu.cn (#Corresponding author)
[2] Baotou Teachers' College, Inner Mongolia University of Science & Technology
014030 Baotou, China
mazhanfei@163.com

**Abstract.** As traditional networks, the software-defined campus network also suffers from intrusion attacks. Current solutions for intrusion prevention cannot meet the requirements of the campus network. Existing methods of attack traceback are either limited to specific protocols or incur high overhead. To protect the data center (DC) of the campus network from internal and external attacks, we propose an Intrusion Prevention System (IPS) based on the coordinated control between the detection engine, the attack traceback agent, and the software-defined control plane. Our solution includes a novel algorithm to infer the best switch port for defending different attacks of varied scales based on the inverse HSA (Header Space Analysis) and the global view of the software-defined controller. The proposed scheme can effectively and timely block the malicious traffic not only protecting victim hosts from attacks but also preventing the whole network from suffering unwanted transmission burden. The proposed IPS is deployed on the bypass of the DC switch and collects network traffic by port mirroring. Compared with the traditional serial deployment, the new design helps defend the DC internal attacks, reduce the probability of network congestion, and avoid the single point of failure. The experimental results show that the overhead of our IPS is very low, which enables it to meet the real-time requirements. The average defense time is between 10 and 14 ms for the data center internal attacks of different scales. For external attacks, the maximum defense time is about 76 ms for a large-scale network with 100 switches.

**Keywords:** IPS, Intrusion Prevention System, SDN, Software-defined Network, Attack Traceback, Inverse Forwarding Function, HSA, Header Space Analysis, Campus Networks, DC, Data Center.

## 1. Introduction

The overall situation of network security is not optimistic in recent years. Intrusions from the Internet have brought serious consequences, which pose a great threat to information security of networked systems. In the first quarter of the year, DDoS attacks rose more than 278 percent compared to Q1 2019 and more than 542 percent compared to the last quarter, according to Nexusguards Q1 2020 Threat Report [16]. In addition, attacks from the internal hosts infected by computer viruses have exhibited great destructiveness in

⋆ Corresponding author

several security incidents. Worms play an important role in internal attacks. The worm installs itself in the memory of the computer but it has the capability to transfer itself to other hosts automatically even without human intervention, thus making it much more serious than a virus. In recent two years, some enterprises have been attacked by the worm GandCrab, which encrypts victims' files and demands ransom payment in order to regain access to their data. Since launching in January 2018, GandCrabs authors claimed to have brought in over $2 billion in illicit ransom payments [15].

Software-defined networking (SDN) [26] has its roots anchored deeply in education and drives the evolution of the campus network. It demonstrates that the campus network can do more than serving universities, and they are also capable of helping a diverse set of users with varying needs. Therefore, the software-defined campus network is becoming increasingly important. The data center (DC) of the campus network holds the most critical data assets of a university, college, or institute, so it is the key protection unit of the network and also the focus of this paper. Unfortunately, the security of software-defined campus networks is worrying. As traditional networks, software-defined networks also suffer from attacks such as DoS (Denial of Service), U2R (User to Root), R2L (Remote to Local), Probe, etc. Given that SDN is famous for its contributions to the network architecture, we consider taking advantage of its dynamic flow control, network-wide visibility, and network programmability to improve its security, especially its intrusion prevention capability.

The Intrusion Prevention System (IPS) has been widely adopted to enhance network security. Traditionally, it is deployed between the core switch of the campus network and the DC switch in series. Once external malicious traffic attacks DC hosts through the IPS, it can effectively protect them from these attacks. However, before entering the IPS, a large amount of malicious traffic is often forwarded by other switches inside the campus network, which brings additional burden to the network. Moreover, if an internal malicious host attacks other DC hosts, the IPS can no longer protect them because the internal malicious traffic is forwarded by the DC switch without through the IPS. Therefore, we want to build an innovative intrusion prevention system that protects DC hosts from both internal and external attacks. The proposed system is deployed on the bypass of the DC switch and it collects network traffic with port mirroring. The new design prevents the IPS-incurred single point of failure from happening, avoids the network congestion caused by the serial deployment, and defends the DC internal attacks. Further, the new system makes use of the inverse forwarding functions derived by expanding the Header Space Analysis (HSA) [20] framework to accurately trace attack traffic back and find the best switch port for blocking it. Finally, the system harnesses the synergy of the intrusion detection engine, the attack traceback agent, and the software-defined control plane to block intrusion attacks from the source in real time using the OpenFlow protocol and prevents the malicious traffic from soaring at the beginning of attacks.

The contributions of this paper are as follows:

- To protect the data center of the campus network from internal and external attacks, we propose an intrusion prevention system based on the coordinated control between the intrusion detection engine, the attack traceback agent, and the software-defined control plane.

- According to the inverse HSA, we design a novel real-time protocol-independent algorithm to infer the best switch port for preventing different intrusion attacks of varied scales using the forwarding model of the entire network.
- We implement a prototype of the proposed IPS and evaluate its performance in the software-defined campus networks of various scales under the intrusion attacks such as NULL scanning and FIN scanning.

The rest of the paper is organized as follows. Section 2 summarises related work to the IPS in SDN and Attack Traceback. In Section 3 we describe our design principles and system architecture. Section 4 presents our algorithm of finding the best defense switch port. Section 5 presents the implementation details of the proposed IPS. Section 6 details our performance evaluation experiments. In Section 7 we discuss the advantages and drawbacks of various attack traceback methods. Finally, conclusions are drawn in Section 8.

## 2. Related Work

### 2.1. IPS in SDN

There is some existing work that leverages SDN for intrusion prevention. Based on the deployment modes of security components, the existing work can be classified into two categories: (1) security applications built on the SDN controller [35,8], (2) security devices that work in cooperation with the SDN control plane [34,9]. Changhoon Yoon et al. [35] implement four types of security functions with SDN in Floodlight applications and evaluate their Floodlight [1] application in real testbeds. For the NIPS (Network Intrusion Prevention System) application, the payload delivery from the data plane to the control plane would incur substantial overhead. Pin-Jui Chen and Yen-Wen Chen [8] propose a defense mechanism, which can find attack packets previously identified through the Sniffer function, and once the abnormal flow is found, the protection mechanism of the Firewall function will be activated. But its evaluation method is simple, and it is difficult to meet the security requirements of campus networks; its real-time performance is not evaluated; the problem of tracing the source of the attacker is not resolved. Xing et al. [34] presented an implementation of Snort IPS for protecting the cloud platform using Snort IDS [10] while sending the blocking action to the SDN controller. The authors get the benefit of snort as open source IDS and adjusted it for integration. But tracing the source of the attacker isn't considered. Yaping Chi et al. [9] propose a scheme for the cloud platform intrusion prevention, and the result shows that the efficiency of the intrusion detection in the new scheme can be improved by two times compared with the traditional intrusion prevention scheme. The solution applies to the cloud environment only; it is difficult to adopt this solution to meet the diverse security needs of the campus network, however.

### 2.2. Attack Traceback

Attack traceback is not a goal, but a means to defend against great harmful attacks (such as Dos). Identifying the origins of attack packets is the first step in making attackers accountable. Besides, after figuring out the network path which the attack traffic follows,

the victim under the attack can apply defense measures such as packet filtering further from the victim and closer to the source.

Some scholars have started research on the attack traceback. IP traceback is a technique for tracing the paths of IP datagrams back toward their origins and also serves as the main technique for attack traceback. its methods can be divided into 5 categories:

1) Link Testing

Link testing [24] is an approach which determines the upstream of attacking traffic hop-by-hop while the attack is in progress. It is compatible with the existing protocols and the network infrastructure, such as routers. However, it is only suitable for tracking the attacks that last for long times.

2) ICMP Trace

This scheme is for each router to come up with an ICMP traceback message [3] or reach directed to the identical destination. The trace message itself consists of consequent and previous hop data and a time stamp. It utilizing the explicitly generated ICMP Traceback message were proposed in [17,31]. It incurs higher overhead in computation, storage and bandwidth.

3) Logging

This solution involves storing packet digests or signatures at intermediate routers and using data-mining techniques to see the trail that the packets traversed [32]. The drawbacks of this technique include significant amount of resources have to be reserved at intermediate routers and hence large overhead on the network, complexity, centralized management.

4) Overlay Networks

CenterTrack [33] is an overlay network, consisting of IP tunnels or other connections, that is used to selectively reroute interesting datagrams directly from edge routers to special tracking routers. The tracking routers, or associated sniffers, can easily determine the ingress edge router by observing from which tunnel the datagrams arrive. The datagrams can be examined, then dropped or forwarded to the appropriate egress point. It need to add special tracking routers, and is easy to find by attackers due to rerouting interesting datagrams.

5) Packet Marking

Packet-marking methods [7,30,4] are characterized by inserting traceback data into the IP packet to be traced, thus marking the packet on its way through the various routers on the network to the destination host. The method is subdivided into Probabilistic Packet Marking (PPM) [29] and Deterministic Packet Marking (DPM) [2]. Each router marks the packet with some probability in the PPM scheme, while every packet passing through the first ingress edge router is only marked with the IP address of the router in the DPM scheme. It requires modifications to the protocol, and cannot handle fragmentation and does not work with IPv6 and is not compatible with IPSec.

To be brief, each type has its advantages and drawbacks. The first type is only suitable for tracking the attacks that last for long times; the second and third type usually incur extra network burdens; the other types of methods require particular routers in the data path.

In this paper, we propose an intrusion prevention system that leverages the coordinated control between the detection engine, the attack traceback agent, and the software-defined control plane. In terms of the SDN design, our solution belongs to the second type of

the system layouts described in Section 2.1. We assume that all the forwarding devices in the campus network are stateless and SDN-enabled, and they are controlled by the SDN Controller; the initial network topology is known in advance, and the attacks can be discovered by the detection engine of the IPS.

Current methods of attack traceback in Section 2.2 are either limited by specific protocols or incur high overhead. In this paper, we construct the inverse forwarding functions by expanding HSA framework, and design a novel real-time and protocol-independent algorithm of finding the best switch port for defense. In the later section, we will demonstrate the proposed solution can accurately and timely prevent attacks of varied types and scales.

## 3. Design Principles and System Architecture

The data center of the campus networks holds the most critical data assets, so it requires an efficient IPS solution for protecting its servers and other hosts from internal and external attacks. In the section, we discuss the design principles of a common intrusion prevention system for software-defined campus networks, propose an innovative IPS architecture to meet these principles, and introduce the intrusion prevention process of our proposal IPS.

### 3.1. Threat Model

The software-defined campus network not only suffers from the traditional attack (such as Dos, U2R, R2L, and Probe) but also sustains attack for the SDN control plane. Defending the latter attack is our future work, and is out of scope in this paper.

We assume the threats against the data center (DC) server of the campus, except attack for the SDN control plane. The victim may be a DC internal host, an external host or even an Internet host.

### 3.2. Design Principles

The design principles of a common intrusion prevention system for software-defined campus networks are based on the following key properties:

- **Needing an efficient solution for mainly focusing on protection data center servers of campus networks**: The data centers of campus networks hold the most critical data assets. So the IPS mainly protection data center of campus networks avoiding interior and external attacks in the campus network, and avoids single-point failure and reduces the probability of network congestion.
- **Detecting different types of attack**: the victim host of the data center can suffer from data center internal hosts, other campus network hosts(such as hosts of the office network) and Internet hosts.
- **Finding the best defense switch port for different types of attack**: The IPS can pinpoint the attack source and find the best defense switch port for different attacks and directly block the malicious traffic from injecting switches, which can eliminate dispensable transmission burdens which are raised by malicious attacks.

– **Compatibility with any OF-enabled device and protocol-independent**: The algorithm of finding the best defense switch port can be valid with any OF-enabled device (such as a Layer 2 switch or a router, etc.), regardless of which protocol it belongs to.
– **Real-time**: The IPS can intercept the malicious traffic at the beginning of the attack avoiding the malicious traffic soaring late.

### 3.3. Overall Architecture

The campus network provides network access services for students and faculties, divided into multiple types of subnets (such as the data center LAN and the office LAN). A minimum software-defined campus network consists of an SDN controller, three switches, several servers and dozens of workstations, as shown in Fig. 1. The border switch links the campus network to the Internet, and the other two switches connect servers and workstations. The DC (data center) switch and application servers comprise the data center LAN of the campus. Similarly, the ON (office network) switch and users' workstations form an office LAN for the campus.

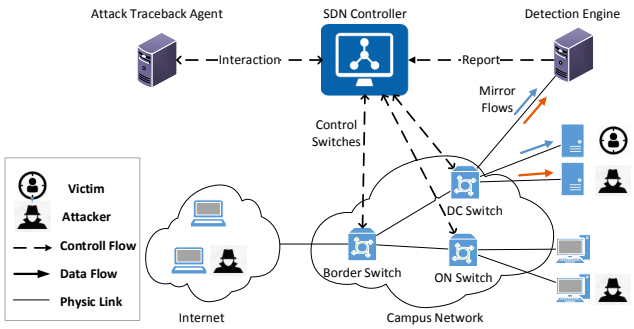


**Fig. 1.** Intrusion Prevention Architecture on an SDN-based Campus Network

The intrusion prevention architecture for the software-defined campus network is composed of an intrusion detection engine, an attack traceback agent, and the software-defined control plane, which includes all the switches and the controller mentioned above. It realizes the coordinated control between them to detect intrusions as early as possible and prevent attacks as soon as possible. The DC switch regularly mirrors the ingress traffic of the DC servers and sends it to the detection engine. The engine captures and analyses the traffic, and sends alarm messages to the controller if intrusion attempts are detected. In our design, the detection engine is deployed on the bypass of the DC switch. The engine has two NICs, with one interface connected the DC switch and the other joined up to the same LAN with the SDN controller and the trackback agent. The SDN controller transmits network states, such as flow table entries of all switches and topology changes of the total network, to the traceback agent also on a regular basis. According to the received

network states, the traceback agent establishes a forwarding model of the entire network and uses it as the global view to infer the best switch port for defending intrusion attacks.

### 3.4. Intrusion Prevention Process

We have divided the process of detecting intrusion attempts and preventing attacks into four steps, as shown in Fig. 2.

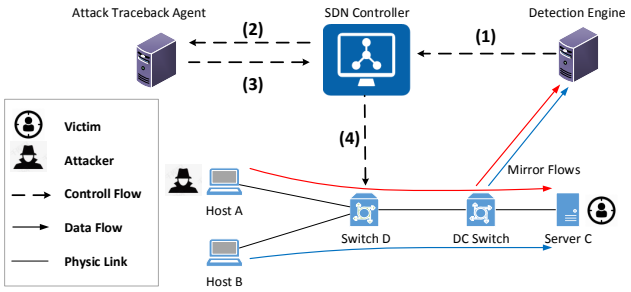


**Fig. 2.** Attack Detection and Defense Process

**1) The detection engine discovers intrusion attempts and sends alarm messages to the controller.** The detection engine constantly monitors the mirrored traffic. When it detects an intrusion attempt, it immediately sends an alarm message, which contains the header field of malicious packets, to the controller via the TCP connection.

**2) The controller sends a request to the traceback agent asking for the best switch port for defense.** When the controller receives the alarm message, it needs to find the best switch port to prevent the intrusion and take the appropriate measures. However, it is hard for the controller to pinpoint the attack source and then determine the best defense port based on the source and the overall situation of the network. The attack source might be a host in the data center network, office network, or Internet. Therefore, the controller queries the traceback agent for the best switch port for defense via the TCP connection.

**3) The traceback agent determines the best switch port for defense and responds to the controller.** The traceback agent runs the inverse HSA algorithm (details given later in Section 4) to trace attack packets back to their origins and infer the best switch port for defense according to the previously established forwarding model of the whole network, and finally returns the identified switch port to the controller.

**4) The controller generates and sends the OpenFlow message to the switch where the port is located to block the malicious traffic.** Subsequently, the controller generates a Flow-Mod or Port-Mod message [28] according to the returned switch port and the corresponding defense strategies, and then sends the OpenFlow message to the switch where the port is located. Once the switch receives the message, it updates its flow tables or changes the link state of the port to down preventing the malicious traffic from injecting the network again. As shown in Fig. 2, both Host A and Host B locate outside the data center, their traffic follows Switch D, the DC switch to Server C; the malicious traffic

from Host A to Server C is blocked on Switch D preventing it from being forwarded to Server C through the DC switch while the normal traffic from Host B reaches Server C unobstructed.

## 4. Algorithm Design

In the IPS architecture described in the previous section, it is essential to pinpoint the attack source and infer the best switch port for defense. In this section, we first expand the HSA [20] framework to construct inverse forwarding functions of all switch ports in the network. Then, we design a protocol-independent algorithm to find the best switch port for defense using the inverse forwarding functions and the backtrack technique [5].

### 4.1. Header Space Analysis

The theory was first proposed by Kazemian Peyman [20], and used for network verification and debugging. The definition of this theory is as follows:

**Header Space**, $H$ : A packet header is represented as a flat sequence of ones and zeros. Formally, a header is a point, and a flow is a region in the $\{0,1\}^L$ space, where L is an upper bound on the header length.

**Network Space**, $N$ : The network is modeled as a set of boxes called switches with external interfaces called ports, each of which is modeled as having a unique identifier. If we take the cross-product of the switch-port space (the space of all ports in the network, S) with $H$, we can represent a packet traversing on a link as a point in $\{0,1\}^L * \{1, ..., P\}$ space, where $\{1, ..., P\}$ is the list of ports in the network.

**Switch Transfer Function**, $T()$ : As a packet traverses the network, it is transformed from one point in Network Space to another point(s) in Network Space. A node can be modeled using its transfer function $T$ that maps header $h$ arriving on port $p$:

$$T(h,p) : (h,p) \rightarrow \{(h1,p1),(h2,p2),\cdots\} \tag{1}$$

**Network Transfer Function**, $\Psi()$ : Given that switch ports are numbered uniquely, all the switch transfer functions are combined into a composite transfer function describing the overall behavior of the network. Formally, if a network consists of $n$ boxes with transfer functions $T_1(.), \ldots, T_n(.)$, then

$$\Psi(h,p) = \begin{cases} T_1(h,p) & \text{if} \quad p \in switch_1 \\ \cdots & \cdots \\ T_n(h,p) & \text{if} \quad p \in switch_n \end{cases} \tag{2}$$

**Topology Transfer Function**, $\Gamma()$ : A unidirectional link connects a source port $P_{src}$ to a destination port $P_{dst}$ and delivers packets from $P_{src}$ to $P_{dst}$. The topology of a network is defined by the set of links in the network, each represented by its source and destination ports. We can model the network topology using a topology transfer function, $\Gamma()$, defined as:

$$\Gamma(h,p) = \begin{cases} \{(h,p^*\} & \text{if } p\, connected\, to\, p^* \\ \{\} & \text{if } p\, is\, not\, connected \end{cases} \tag{3}$$

**Inverse of Switch Transfer Function**, $T^{-1}()$ : For a given switch, the finding application requires working backward from an output header to determine what input (header, port) pairs could have produced it. We define the inverse of switch transfer function as:

$$T^{-1}(h,p) = \{(h',p')|(h,p) \in T(h',p')\} \tag{4}$$

### 4.2. Inverse Forwarding Functions

To inverse a forwarding function, we need to work backward from a pair of output packet header and output port to infer which pair of input header and input port might have produced it. We expand the HSA theory to add the following two inverse functions.

**Inverse of Network Transfer Function**,$\Psi^{-1}()$ :For a given header at an output port $(h,p)$, $\Psi^{-1}(h,p)$ is the set of all input headers at input port, $(h_i,p_i)$,such that (h,p) $\in$ $\Psi(h_i,p_i)$:

$$\Psi^{-1}(h,p) = \{(h',p')|(h,p) \in \Psi(h',p')\} \tag{5}$$

A transfer function maps each (h,p) pair to a set of other pairs. By following the mapping backward, we can invert a transfer function.

$$\Psi^{-1}(h,p) = \begin{cases} {T_1}^{-1}(h,p) & \text{if} \quad p \in switch_1 \\ \quad \ldots & \quad \ldots \\ {T_n}^{-1}(h,p) & \text{if} \quad p \in switch_n \end{cases} \tag{6}$$

**Inverse of Topology Transfer Function**, $\Gamma^{-1}()$ : For an arbitral head space h and a given port p, a port $p'$ is connected to $p$, and the packet reaches from port $p'$ to $p$.

$$\Gamma^{-1}(h,p) = \{(h,p')|(h,p) \in \Gamma(h,p')\} \tag{7}$$

### 4.3. Algorithm

In a campus network, there are $m$ OF-enabled switches: $S$=$\{s_1,...,s_{m-1},s_m\}$, and the switch $s_i$ has $n_i$ physics ports. In the total campus network, there are $n$ switch ports: $P$=$\{p_1,...,p_{n-1},p_n\}$, $n = \sum_{i=1}^{m} n_i$. The IPS captures an attack packet from the mirror traffic of the DC switch $j$ and gets its header space $h_{tar}$; according to Switch $j$'s Mac-Port table, the port $p$ attached to the victim host is found. According to the principle of network reachability, there is an attack path from the attack source switch port $q$ to $p$. The path is denoted by:

$$q \to p_1 \to ... \to p_{k-1} \to p_k \to p \tag{8}$$

the malicious traffic is injected through Port $q$, so it is identified as the best switch port for defense. Port $q$ is calculated by:

$$(h,q) = \Psi^{-1}(\Gamma^{-1}(...(\Psi^{-1}(h_{tar},p)))) \tag{9}$$

According to the flow tables of the total switches, Switch Transfer Function $T()$ is calculated, and Inverse of Switch Transfer Function $T^{-1}()$ is calculated by Equation (4), then

Inverse of Network Transfer Function $\Psi^{-1}()$ is calculated by Equation (6). According to the network topology, Inverse of Topology Transfer Function $\Gamma^{-1}()$ is calculated. Finally, we trace the pair (header, port) backward (using the inverse of transfer functions at each step) to find the attack source port $q$ as the best switch port for defense.

To pinpoint the attack source and infer the best switch port for defense, we have designed the algorithm (see Fig. 3). The algorithm makes use of the following three functions: 1) $mac\_src(h_{tar})$ refers to the source MAC address of the attack packet;
2) $mac\_dst(h_{tar})$ refers to the destination MAC address of the attack packet;
3)$find(T, mac\_src(h_{tar}))$ queries the source MAC address of the attack packet from $T$, which is the set of ($port$,$mac$) pairs maintained by the DC switch, and returns the corresponding switch port.

The algorithm exploits inverse forwarding functions defined above to infer possible input ports that can forward packets to the target port, then uses the backtrack method to traverse all inferred ports according to the depth-first search strategy, and eventually finds the attack source. The attack source may be either an internal host of the campus network or an external Internet host. In the former case, the algorithm returns the switch port attached to the internal malicious host as the best port for defense. In the latter, the algorithm returns a WAN port of border switches or routers as the best port for defense.

The complexity of our proposed algorithm is O($P \cdot N$), where $P$ is the total number of activating ports which connect to other hosts or switches and $N$ is the maximum number of flow entries in an arbitral switch. For a given header at an output port $(h_0, p_0)$, $\Psi^{-1}(h, p)$ is the set of all input headers at the input port, so the returning (header,port) pairs count of $\Psi^{-1}(h, p)$ is greater than or equal to 1. If the returning (header,port) pairs count of each $\Psi^{-1}(h, p)$ equal to 1, the search path length will less than the diameter of networks. If the returning (header,port) pairs count of each $\Psi^{-1}(h, p)$ is greater than 1, the algorithm needs to traverse the each returning (header,port) pairs in proper order, the search process didn't be terminated until finding a first feasible solution adopting the depth-first search strategy.

## 5. Implementation

We implement a prototype of the proposed IPS at the central SDN controller using the open-source Ryu SDN Framework [11] and the analysis engine using the open-source IDS Snort [10]. The analysis engine inspects the mirror traffic, and send an alarm to the controller detecting an intrusion attempt. We develop two components: the attack traceback agent, the data forwarding and attack mitigation APP built on the SDN controller.

**The attack traceback agent** is developed by Python Language, communicate with the SDN controller over a TCP socket, and is a socket-based server that 1) receives flow entries varieties of all switches and topology changes of the entire network, 2) and returns the best defense switch for an attack event. The first function is capturing flow entries varieties and topology changes to establish the forwarding model of the entire network; The second function is tracing the attack packet back to its origin based on the above forwarding model, and calculating the best defense switch port by the algorithm(see Fig. 3). Based on a base class library of the open-source project Header Space Library [19], we add critical codes to support inverse forwarding functions and implement the agent's total functions.

**Require:**
The header space of a captured attack packet: $h_{tar}$;
The set of ($port$,$mac$) pairs maintained by the DCN switch: $T$;
The set of switch ports that attached to Internet: $P_{wan}$;
The set of switch ports that attached to hosts or Internet: $P_{end}$.
**Ensure:** The switch port that the attack packet is injected into: $q$.

**if** $mac\_src(h_{tar}) \in T.mac$ **then**
  $q \leftarrow find(T, mac\_src(h_{tar}))$
  **return** $q$
**else**
  $p_{tar} \leftarrow find(T, mac\_dst(h_{tar}))$
  $history \leftarrow p_{tar}$
  $Stack \leftarrow [header : h_{tar}, port : p_{tar}]$
  **while** $Stack.size() > 0$ **do**
    $r \leftarrow Stack.pop()$
    $history.append(r.port)$
    $temp \leftarrow \Psi^{-1}(r.h, r.p)$
    **for** $(h, p)$ in $temp$ **do**
      **if** $p \notin history$ **then**
        **if** $p \in P_{wan}$ **then**
          $Deque.addLast(p)$
        **else**
          $Deque.addFirst(p)$
        **end if**
      **end if**
    **end for**
    **while** $Deque.size() > 0$ **do**
      $q \leftarrow Deque.removeFirst()$
      **if** $q \in P_{end}$ **then**
        **return** $q$
      **else**
        $(h, p') \leftarrow \Gamma^{-1}(h, q)$
        **if** $p' \notin history$ **then**
          $Stack.push(h, p')$
          break
        **end if**
      **end if**
    **end while**
  **end while**
**end if**

**Fig. 3.** Find the Best Defense Switch Port Algorithm

**The data forwarding and attack mitigation APP** built on the SDN controller is developed by Python Language, and has the following modules:

**1) Forward Data**. Because RYU's demo applications (such as $simple\_switch\_13.py$ and $rest\_router.py$ etc.) don't support to construct mixed networks which comprise Layer 2 and Layer 3 forwarding devices, in order to emulate the topology of the software-defined campus network, we complete the forwarding APP to support the mixed network architecture.

**2) Send Network State Messages**. The module sends a series of network state messages to the attack traceback agent via a TCP socket regularly. The network state messages mainly include three types: flow entries varieties messages by each Flow-Mod message, topology changes messages by each related Ryu event, and the MAC address changes messages of servers for updating the set of ($port$,$mac$) pairs $T$. We embed the corresponding code into the above forwarding APP. For example, in term of the MAC address changes messages, we capture ARP messages of the DCN switch to get the MAC address change states and send the state change messages to the attack traceback agent.

**3) Receive Alert Messages**. The module uses an existing library, namely $snort.lib$, which enables the SDN controller as the server to receive Snort alert messages from the analysis engine by a TCP Socket. On the Snort machine, the application Pigrelay [21] running is configured, to enable that the alarm messages generated are sent to the Ryu controller using a TCP socket. Once the controller receives an alarm message from the analysis engine, it then generates an intrusion event.

**4) Response to Intrusion Events**. Once it monitors an intrusion event, it sends a quest of looking up the best defense switch port to the attack traceback agent via a TCP socket. When the attack traceback agent returns the best defense switch port, according to its position and the appropriate defensive strategy, the App automatically generates a Flow-Mod or Port-Mod message and sends it to the defensive switch. After the switch receives the OpenFlow message, it updates its flow tables or changes the returning port link state to down, blocking the malicious traffic injecting it again.

## 6. Evaluation

To assess the IPS prototype system depicted above, we choose port scan attacks, i.e. NULL scanning and FIN scanning. We utilize Nmap [23] to conduct malicious scans, Hping3 [14] to generate the background traffic with the constant flow rate, and sflowtool [25] to capture and analyze data traffic.

We used two Lenovo ThinkServer RD440s, each of which has one Inter Xeon CPU E5-2407 and thirty-two GB memory, to install VMWare Exsi 5.1 for instantiating virtual machines and building two testbeds (see Table 1, 2). We conducted four types of assessments. The first type evaluates the effectiveness of our IPS to prevent intrusion attacks, the second type assesses the effectiveness of the attack traceback algorithm, the third type assesses the timeliness of the proposed system in intrusion prevention, and the last type of evaluation compares the system performance of the campus network when equipped with our IPS, existing IPS, or without any IPS. The first three types of assessment are carried out on Testbed I, while the last type is conducted on Testbed II.

**Table 1.** Testbed I Configuration

| No. | Hardware | Software |
|---|---|---|
| 1 | 2 CPUs/6GB RAM | Snort |
| 2 | 4 CPUs/2GB RAM | Ryu/Beacon/Tracing Attack Agent |
| 3 | 2 CPUs/2GB RAM | sFlow |
| 4 | 2 CPUs/6GB RAM | Mininet |

**Table 2.** Testbed II Configuration

| No. | Hardware | Software |
|---|---|---|
| 1 | 2 CPUs/6GB RAM | Snort |
| 2 | 4 CPUs/2GB RAM | Ryu/Tracing Attack Agent |
| 3 | 2 CPUs/2GB RAM | sFlow |
| 4 | 4 CPUs/4GB RAM | LXC/OpenvSwitch |
| 5 | 4 CPUs/4GB RAM | LXC/OpenvSwitch |
| 6 | 2 CPUs/2GB RAM | LXC/Linux Bridge |
| 7 | 4 CPUs/4GB RAM | Linux Bridge |

### 6.1. Effectiveness of Intrusion Prevention

We use Mininet to emulate a small-scale software-defined campus network with a typical topology illustrated in Fig. 4. In the topology, we use two hosts H5, H6, and a router R1 to emulate the Internet, and the OpenFlow switch S2 acts as a border router between the campus network and Internet, and the switches S1, S2, S3 are OpenFlow switches controlled by a Ryu controller.

We carried out three experiments to evaluate the effectiveness of our IPS. The emulated host H3 in the data center network acts as the victim in each experiment. The different experiments assess the IPS effectiveness under various attacks (see Table 3). The first experiment demonstrates the attack from a DC internal host, i.e. H4 acts as the malicious host, which is named Attack I. The second experiment reveals the attack from an ON host, i.e. H1 acts as the attack source, which is called Attack II. The third one focuses on the attack from an Internet host, i.e. H5 acts as the intrusion traffic origin, which is termed Attack III. In each of the three experiments, we assess the effectiveness of the proposed IPS under two scenarios. In the first scenario, our IPS is not configured with attack traceback, while in the second one our IPS has the fully functional attack traceback mechanism.

**Table 3.** Different Attack Types

| Attack Type | Victim | Attacker |
|---|---|---|
| I | a DC server | a DC internal host |
| II | a DC server | an ON host |
| III | a DC server | an Internet host |

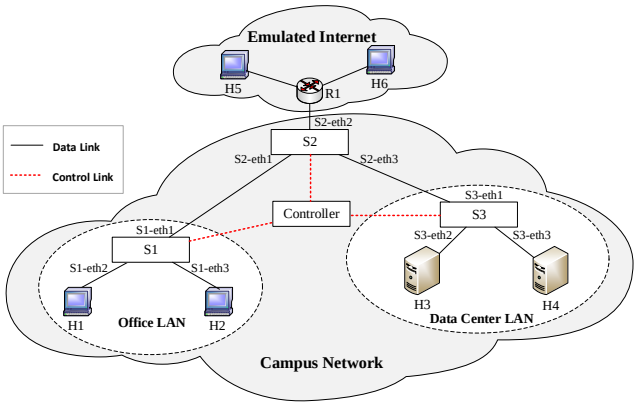


**Fig. 4.** Emulated Topology of the Software-defined Campus Network

Fig.5a and Fig.5b illustrate the packet rate changes during the process the victim host H3 is attacked by an internal malicious host H4, which is also located in the data center. As depicted by the red solid line, the ingress traffic of the victim H3 increases with the egress traffic of the malicious H4 at the beginning of the attack. However, the ingress traffic of H3 quickly falls to 0 pps at the 13th second and the 33rd second respectively in the two figures, which indicates that the proposed IPS is capable of detecting and stemming the internal attack traffic under both scenarios.

As depicted by the solid red line in both Fig.5c and Fig.5d, the victim host H3 starts to receive the ingress background traffic (from the host H2) at the constant rate of 5 pps since the 5th second. After about 20 seconds, an ON host H1 starts to attack H3. In the "Without Attack Traceback" scenario shown in Fig.5c above, the ingress traffic of the victim H3 begins to decrease at the 23rd second because the IPS has detected and blocked the attack traffic. However, the ON switch S1 continues to forward the attack traffic to its egress port eth1, as indicated by the dotted blue line, which changes with the dash-dotted green line, i.e. the malicious traffic. It reveals that the defense port chosen by the IPS without attack trackback is not on the switch nearest to the attack origin, i.e. S1, so the malicious traffic continues to consume network resources after being detected. In the "With Attack Traceback" scenario in Fig.5d below, after the IPS stems the malicious traffic at the 26th second, both the ingress traffic of the victim and the egress traffic of the ON switch S1 drop to the normal level containing only the background traffic. It indicates that the switch attached to the malicious host, i.e. S1, is employed to directly block the attack.

The experimental results in Fig.5e and Fig.5f exhibit the traffic changes when the victim H3 is attacked by an Internet host H5. Similar to Fig. 5c, the red solid line in Fig.5e shows the ingress traffic of H3 rises at the beginning of the attack and goes down after the IPS detects and blocks the attack traffic, but the dotted blue line exposes the egress traffic of the ON switch S2 still involves the attack traffic. In contrast, Fig.5f demonstrates both the ingress traffic of H3 and the egress traffic of S2 descend after the IPS takes action.

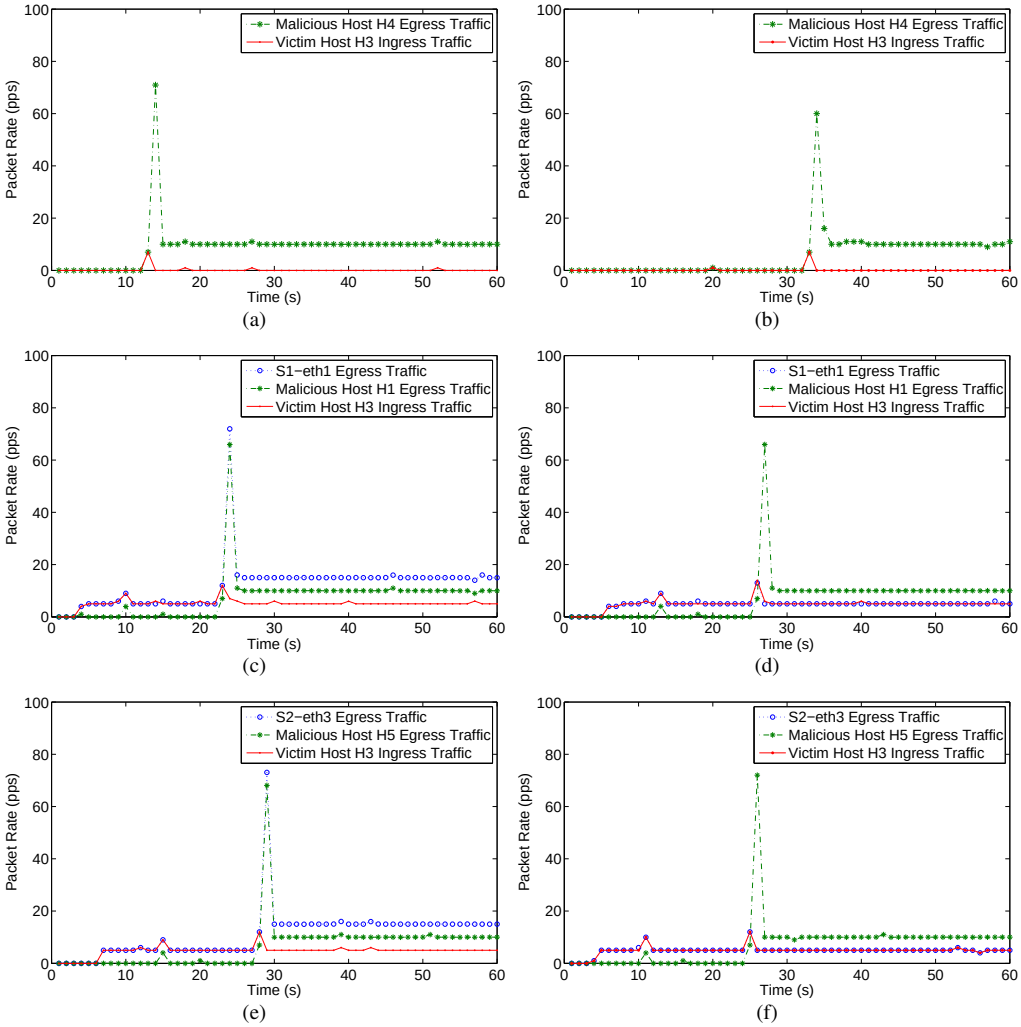


**Fig. 5.** Traffic Changes during Three Types of Attacks Using the IPS with or without Attack Traceback: (a) Without Traceback For Attack I; (b) With Traceback For Attack I; (c) Without Traceback For Attack II; (d) With Traceback For Attack II; (e) Without Traceback For Attack III; (f) With Traceback For Attack III

These experiments bring us the following enlightenment. For Attack I, the proposed IPS can avoid malicious traffic from soaring with or without attack traceback. For the other attacks, there are fundamental differences between the two IPS configurations. If the proposed IPS cannot trace attack packets back, malicious traffic can still bring transmission burden to the network even if it is blocked from injecting the victim host. On the contrary, our complete IPS solution can accurately find the best switch port for blocking malicious traffic, so it is able to prevent intrusion attacks more effectively.

### 6.2. Effectiveness of Attack Traceback

To evaluate the effectiveness of the attack traceback algorithm for large scale campus networks, we use Mininet [27] to replicate the Stanford backbone network, which is a population of more than 15,000 students, 2,000 faculty, and five /16 IPv4 subnets. According to the literatures [36,18], we use Open vSwitch (OVS) [13] to emulate the routers, and install the reserved equivalent OpenFlow rules in the OVS switches with the SDN Controller Beacon [12]. we implement a bundle of the Beacon which can send a series of network state messages to the attack traceback agent regularly. We use emulated hosts to attack the victim host, and perform the attack traceback algorithm to pinpoint the attack source and infer the best switch port for defense. Figure 6 shows the part of the network that is used for experiments in this section. In the entire topology, there are 26 OF-enabled switches and 240 hosts (Due to the limited space, hosts is omitted in Figure 6). we use the two core switches S1, S2 to connect Internet, the two access switches S15, S16 act as the data center LAN switches, and the other access switch S3, S4,..., S14 act as the office LAN switches.

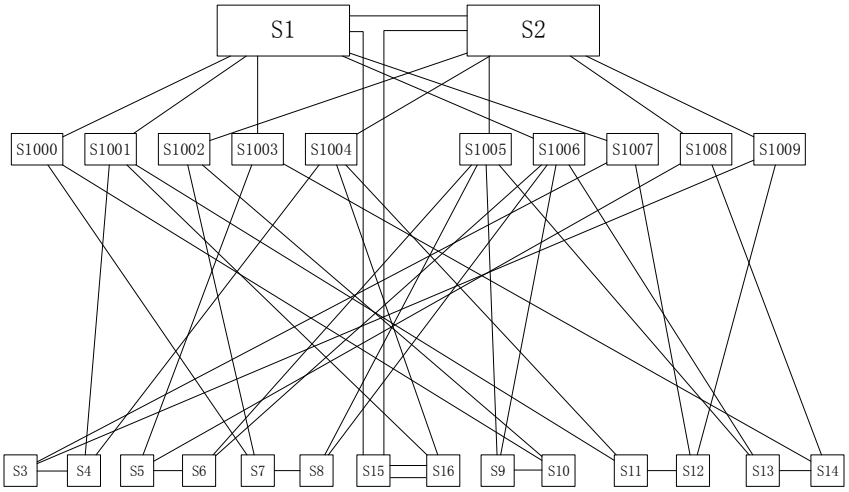


**Fig. 6.** Topology of the Backbone Network of Stanford University

We carried out three experiments to evaluate the effectiveness of our attack traceback algorithm. The emulated host H227 connected with Switch S15 in the data center network

acts as the victim in each experiment. The different experiments assess the algorithm effectiveness under various attacks (see Table 3). The first experiment demonstrates the attack from two DC internal hosts, i.e. H208 and H210 connected with Switch S15 acts as the malicious host, which is named Attack I. The second experiment reveals the attack from ON two hosts, i.e. H77 connected with Switch S4 and H197 connected with Switch S13 act as the attack source, which is called Attack II. The third one focuses on the attack from two Internet hosts, i.e. H17 and H18 connected with Switch H1 act as the intrusion traffic origin, which is termed Attack III. In each of the three experiments, we perform our attack traceback algorithm 10 times for each attack, record the traceback path, the returned switch port and its execution time.

The above experimental results show our algorithm can pinpoint the attack source for three kinds of attacks and infer the best switch port for defense. As depicted by the red dash-dotted line in Fig.7, we use our trackback algorithm to find the attack source host H18 and infer the switch port S1-eth4 connected Internet, as the best switch port for defense, following the trackback path S15 $\rightarrow$ S1. Also, as depicted by the yellow dash-dotted line, we use the algorithm to find the attacker H197 and infer Port S13-eth9 connected H197, as the blocking port, along the path S15 $\rightarrow$ S1 $\rightarrow$ S1006 $\rightarrow$ S13.

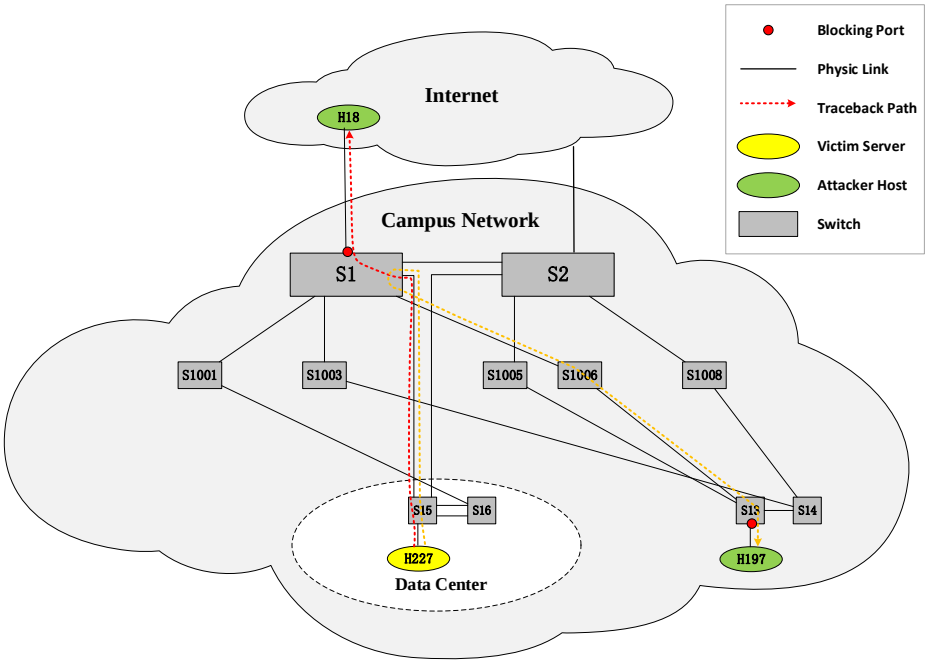


**Fig. 7.** Traceback Paths and Blocking Ports under Various Attacks

Fig.8 shows the execution time of our trackback algorithm under three types of attacks defined previously in the Stanford campus network. As depicted by the green boxes, it takes about 1 ms to infer the best switch port for defending Attack I. Because the attack is

defended within the DC LAN, it only needs to find the Port-Mac table of the DC switch (as line 2 in Fig. 3). As depicted by the blue boxes, their execution time varies greatly for the attackers H77 and H197 although they are belong to the same Attack II. The reason for this difference is that it spent different time to traverse the list of flow entries for the trackbacking switches due to the different number of flow entries for the different switches. As shown in the red boxes, it takes about 2 ms to infer the best switch port for defending Attack III. For Attack I, Attack II, and Attack III, its execution time is 1.1ms, 3.5ms, and 2.2ms, respectively. And the average execution time is about 2.27 ms. In general, the more switches that traceback, the total number of traversal flow entries also increases, and the corresponding execution time will also be longer. Overall, the execution time in the experiments demonstrates that our attack traceback algorithm can work in real-time.

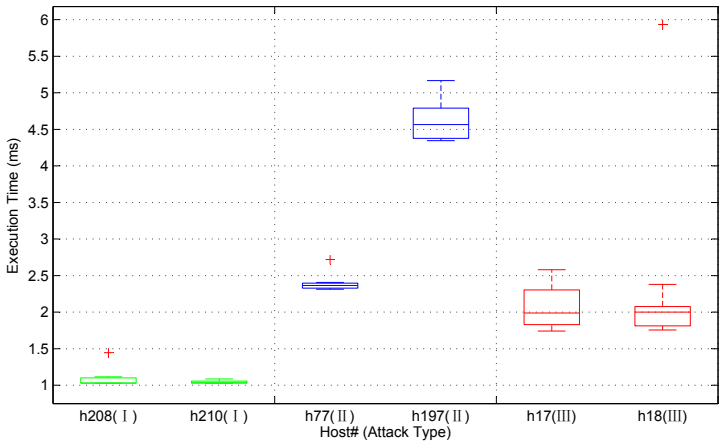


**Fig. 8.** Execution Time of the Traceback Algorithm under Various Attacks

### 6.3. Timeliness of Intrusion Prevention

To assess how timely the proposed system can prevent intrusion attacks, we use Mininet to emulate three different scales as manifested in Table 4. In each scale campus network, we attempt to launch three different types of attacks (see Table 3) and measure the defense time spent from receiving the alarm until sending out the OpenFlow message, which is comprised of the computing time of the algorithm (see Fig. 3) and the other time.

Fig.6 shows the defense time under three types of attacks defined previously in networks of different scales. It takes 10-14 ms for the proposed IPS to defend Attack I. Because the attack is defended within the DC LAN, the time varies little with the scale of the campus network. For Attack II and Attack III, with the growth of the network scale, the defense time increases linearly, mainly because the computing time of the attack traceback algorithm, depicted by the lower portion of the bar with horizontal strips, increases when it has to recursively search more switch ports. Thus, the longest defense time, about

**Table 4.** Three Scale Campus Networks Topologies

| Scale Type | Switch Count | LAN Count | Host Count |
|---|---|---|---|
| Small | 5 | 4 | 40 |
| Medium | 50 | 49 | 490 |
| Large | 100 | 99 | 990 |

76 ms, happens in the large-scale network with 100 switches when it experiences attacks from the Internet. Moreover, the other time, depicted by the upper portion of the bar with diagonal strips, varies litter with an average of 12.2 ms, because time spent on generating OpenFlow messages and exchanging them and other messages is not affected by the network scale and attack type. The defense time in the experiments demonstrates that our IPS and the attack traceback algorithm work in real-time.

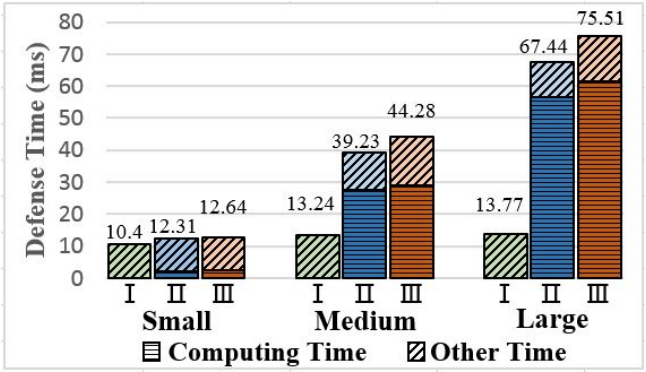


**Fig. 9.** Defense Time in Each Experiment

### 6.4. Overhead of Intrusion Prevention

We use LXC [22] to emulate a small-scale software-defined campus network with a typical topology portrayed in Fig. 4. As Table 2 shown, we use three LXC hosts (VM4, VM5, VM6) and VM7 (running Linux Bridge Application as a router in the Internet zone) to emulate the campus network. VM4 has created two containers (H1 and H2) and two OpenvSwitch switches (S1 and S2). VM5 has created two containers (H3 and H4) and one OpenvSwitch switch (S3). VM6 has created two containers (H5, H6). Switch S1, S2 and S3 are controlled by a Ryu controller (VM2). The network is connected to the Internet through the border switch S2 and the router R1 that is emulated with Linux Bridge [6]. To assess the overhead of the proposed IPS, we compare the performance of the network when it is configured without IPS, with the existing IPS, and with our IPS.

When the existing IPS is used, Snort is configured in the inline mode to work as the IPS, and it is deployed between the border switch S2 and DC switch S3 in series. When the proposed IPS is used, as delineated in Fig. 1, the detection engine is deployed on the bypass of the DC switch S3 and Snort is configured in the passive mode. In each case, we evaluate the throughput and RTT of the two paths: one is from an Internet host to a DC host, and another is from an ON host to the same DC host.

As shown in Fig.10a and Fig.10b, the throughput of the network with our proposed IPS is very close to the throughput when the network has no IPS, as revealed by the green dashed line and red solid line. On the contrary, as indicated by the blue dash-dotted line, the throughput with the existing IPS degrades most 50% when the data transmission rate is higher than 20Mbps. Compared with the existing IDS deployed in serial mode, its throughput increases 2-4 times the transmission rate is between 40 Mbps and 100 Mbps. The same trend continues in Fig.11a and Fig.11b. The RTT of our proposed IPS is very close to the RTT of without using IPS when the system workload continuously increases from 20 to 100 Mbps. During the same period, the RTT of the existing IPS is much higher than the other two cases. The experimental results suggest the proposed IPS incurs reasonable overhead, which is much lower than the existing solution.

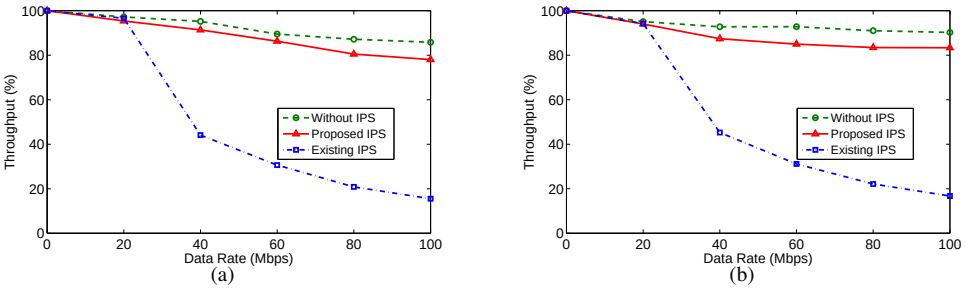


**Fig. 10.** Throughput of Three IPS Schemes: (a) From Internet Host to DC Server; (b) From ON Host to DC Server

## 7. Discussion

According to the evaluation results in the literature about attack traceback (details in Section 2.2) and the experimental evaluation results of our proposed traceback method, we compared and analyzed the various methods (see Table 5) in the aspects, such as traceback accuracy, overhead and compatibility, etc.

Compare with other methods, our traceback method based on the inverse HSA performs higher accuracy of locating attack source and lower overhead. Our proposed method only needs to locate the attack proxy host for internal attacks or the border router for external attacks, and block the malicious traffic injecting from their uplink switch port. And it can find the attack path and the attack source when only one packet is detected as an intrusion attempt by its detection engine component, so our proposed IPS can avoid malicious

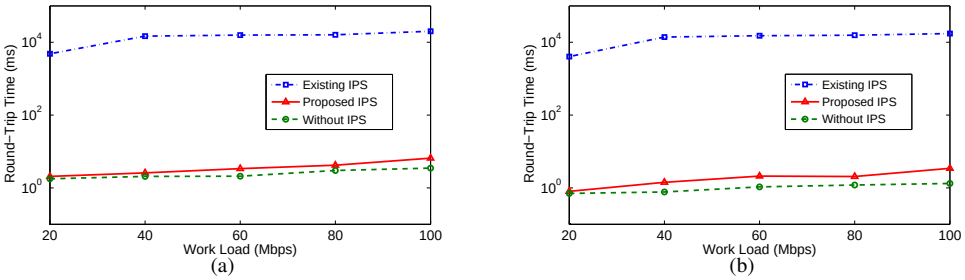


**Fig. 11.** Round-Trip Time of Three IPS Schemes: (a) From Internet Host to DC Server; (b) From ON Host to DC Server

traffic from soaring and prevent intrusion attacks more effectively. Also, the traceback method and the IPS architecture can be compatible with the existing network infrastructure (such as routers, switches, etc), don't need special devices or modify their protocols, and can support the hybrid SDN networks because the OF-disenabled Middle Boxes can be modeled as the inverse HSA as same as the SDN switches. However, our method has some shortcomings, such as needing the prior knowledge of network topologies and no supporting about the post attack analysis, this will be the next step for us to study in the future.

## 8. Conclusion

As traditional networks, the software-defined campus network also suffers from intrusion attacks. Current solutions for intrusion prevention cannot meet the requirements of the campus network. Existing methods of attack traceback are either limited to specific protocols or incur high overhead. To protect the data center of the campus network from internal and external attacks, we propose an Intrusion Prevention System (IPS) based on the coordinated control between the detection engine, the attack traceback agent, and the software-defined control plane. The proposed IPS has the following advantages:

First, it can accurately and timely find the best switch port for defense and prevent the malicious traffic from injecting the network at the first time due to our traceback algorithm based on the inverse HSA. We expand the Header Space Analysis (HSA) framework to construct the inverse forwarding functions and design a novel protocol-independent algorithm to trace attack packets back to their origins and infer the best switch port for defending different attacks using the inverse forwarding functions. It can locate the attack host for internal attacks or the border router for external attacks, and block the malicious traffic injecting from their uplink switch port. In this manner, the malicious traffic would not travel through additional switches and increase the transmission burden of the network. Compare with other traceback methods, our method based on the inverse HSA performs higher accuracy of locating attack source and lower overhead. We replicate the Stanford backbone network to verify the effectiveness of our algorithm, it can pinpoint the attack source for three kinds of attacks and infer the best switch port for defense, and its average execution time is about 2.27 ms, which can work in real-time.

**Table 5.** Comparison of Various Traceback Methods

| | **Link Test** | **ICMP Trace** | **Logging** | **Overlay Network** | **PPM** | **DPM** | **Our Method** |
|---|---|---|---|---|---|---|---|
| Traceback Accuracy | Medium | Good for less packets | Medium | Good | Medium | Good | Good |
| Device Overhead | High | High | High | Low | Medium | Medium | Low |
| Bandwidth Overhead | High | High | High | Nil | Low | Low | Nil |
| Traceback with number of packets | Huge | Huge | One | Small | Large | Small | One |
| Device Compatibility | Good | Good | Need special routers | Add special routers | Modify protocols | Modify protocols | Good |
| Prior Knowledge of Topologies | Need | Not Need | Not Need | Need | Not Need | Not Need | Need |
| Post Attack Analysis | Not Possible | Possible | Possible | Not Possible | Possible | Possible | Not Possible |

Second, it incurs reasonable overhead due to coordinated control design and collecting network traffic with port mirroring. It leverages the coordinated control between the detection engine, the attack traceback agent, and the software-defined control plane. The three components work together to detect intrusion attacks, as well as to plan and enforce the corresponding defense mechanisms swiftly. Our proposed IPS is deployed to bypass the data center switch and collect network traffic with port mirroring. Compared with the existing IDS deployed in serial mode, this design can avoid a single point of failure, reduce the probability of network congestion, and defend the data center's internal attacks. The experimental results show that its throughput increases 2-4 times than the existing IDS deployed in serial mode when the transmission rate is between 40 Mbps and 100 Mbps.

Finally, it can meet the real-time requirements for defense internal attacks and external attacks in different scales, and avoid malicious traffic from soaring and prevent intrusion attacks more effectively. We have implemented a prototype of the proposed IPS and conducted several experiments to evaluate its performance. The experimental results show that the overhead of our IPS is very low, which enables it to meet the real-time requirements. The average defense time is between 10 and 14 ms for the data center internal attacks of different scales. For external attacks, the maximum defense time is about 76 ms for a large-scale network with 100 switches.

The algorithm for finding the best defense switch port is based on stateless forwarding devices and known initial network topologies. In the future, we will improve the algorithm to make it support more stateful forwarding devices regardless of the initial network topology.

**Acknowledgments.** This work was partially supported by the National Natural Science Foundation of China (Grant N0. 61261019 and 61762071), the Inner Mongolia Science & Technology Plan (Grant No. 201802027), the Inner Mongolia Science and Technology Innovation Guidance and Incentive Fund (Grant No. 111-0406041701), and the Inner Mongolia Autonomous Region Natural Science Foundation (Grant No. 2016MS0614 and 2018MS06023).

## References

1. Floodlight OpenFlow Controler (2019), `https://github.com/floodlight/floodlight`
2. Belenky, A., Ansari, N.: Ip traceback with deterministic packet marking. IEEE communications letters 7(4), 162–164 (2003)
3. Bellovin, S.M., Leech, M., Taylor, T.: Icmp traceback messages (2003)
4. Bhavani, Y., Janaki, V., Sridevi, R.: Ip traceback through modified probabilistic packet marking algorithm using record route. In: Proceedings of the Third International Conference on Computational Intelligence and Informatics. pp. 481–489. Springer (2020)
5. Bitner, J.R., Reingold, E.M.: Backtrack programming techniques. Communications of the Acm 18(11), 651–656 (1975)
6. Bridge, L.: Linux Bridge (2020), `https://wiki.linuxfoundation.org/networking/bridge`
7. Chao, G., Sarac, K.: Toward a practical packet marking approach for ip traceback. International Journal of Network Security 8(3), 271–281 (2009)
8. Chen, P.J., Chen, Y.W.: Implementation of sdn based network intrusion detection and prevention system. In: International Carnahan Conference on Security Technology. pp. 141–146 (2016)

9. Chi, Y., Jiang, T., Li, X., Gao, C.: Design and implementation of cloud platform intrusion prevention system based on sdn. In: Big Data Analysis (ICBDA), 2017 IEEE 2nd International Conference on. pp. 847–852. IEEE (2017)
10. Cisco: Snort (2018), https://www.snort.org
11. Community, R.S.F.: Ryu SDN Framework (2018), https://osrg.github.io/ryu/
12. Erickson, D.: Beacon (2013), https://openflow.stanford.edu/display/Beacon.html
13. Foundation, T.L.: Open vSwitch (2016), http://www.openvswitch.org/
14. hping3: hping3 (2005), http://www.hping.org/hping3.html
15. Inc., M.: GandCrab ransomware (2020), https://www.malwarebytes.com/gandcrab/
16. Inc., N.: DDoS Threat Report 2020 Q1 (2020), https://blog.nexusguard.com/threat-report/ddos-threat-report-2020-q1
17. Izaddoost, A., Othman, M., Rasid, M.F.A.: Accurate icmp traceback model under dos/ddos attack. In: 15th International Conference on Advanced Computing and Communications (ADCOM 2007). pp. 441–446. IEEE (2007)
18. James Hongyi Zeng, P.K.: Automatic Test Packet Generation(ATPG) (2015), https://github.com/eastzone/atpg
19. Kazemian, P.: Header Space Library (Hassel) (2014), https://bitbucket.org/peymank/hassel-public/
20. Kazemian, P., Varghese, G., McKeown, N.: Header space analysis: Static checking for networks. Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12) pp. 113–126 (2012)
21. Lin, C.W.: Pigrelay (2015), https://github.com/John-Lin/pigrelay
22. LXC: LXC (2018), https://linuxcontainers.org/lxc/introduction/
23. Lyon, G.: Nmap: the Network Mapper (2018), https://nmap.org/
24. Ma, M.: Tabu marking scheme to speedup ip traceback. Computer Networks 50(18), 3536–3549 (2006)
25. McKee, N.: sflowtool (2018), https://github.com/sflow/sflowtool
26. McKeown N.: Software-defined networking (2009), http://infocom2009.ieee-infocom.org/technicalProgram.htm
27. Mininet: Mininet (2018), http://mininet.org/
28. ONF: OpenFlow Spec v1.3.5 [online] Technical report ONF TS-023. Tech. rep., Open Networking Foundation (2015), https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.3.5.pdf
29. Park, K., Lee, H.: On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack. In: Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213). vol. 1, pp. 338–347. IEEE (2001)
30. Satheesh, N., Sudha, D., Suganthi, D., Sudhakar, S., Dhanaraj, S., Sriram, V., Priya, V.: Certain improvements to location aided packet marking and ddos attacks in internet. Journal of Engineering Science and Technology 15(1), 94–107 (2020)
31. Saurabh, S., Sairam, A.: Icmp based ip traceback with negligible overhead for highly distributed reflector attack using bloom filters. Computer Communications (01 2014)
32. Snoeren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Kent, S.T., Strayer, W.T.: Hash-based ip traceback. Acm Sigcomm Computer Communication Review 31(4), 3–14 (2001)
33. Stone, R.: Centertrack: an ip overlay network for tracking dos floods. In: Usenixsecurity Symposium, August (2000)
34. Xing, T., Huang, D., Xu, L., Chung, C.J., Khatkar, P.: Snortflow: A openflow-based intrusion prevention system in cloud environment. In: Second Geni Research and Educational Experiment Workshop. pp. 89–92 (2013)

35. Yoon, C., Park, T., Lee, S., Kang, H., Shin, S., Zhang, Z.: Enabling security functions with sdn: A feasibility study. Computer Networks 85(C), 19–35 (2015)
36. Zeng, H., Kazemian, P., Varghese, G., McKeown, N.: Automatic test packet generation. IEEE/ACM Transactions on Networking 22(2), 554–566 (2014)

**Guangfeng Guo** received two B.S. degrees in Computer Science & Technology and Educational Technology from Inner Mongolia Normal University in 2003 and received an M.S. degree in Computer Application Technology from Tianjin University in 2009. He is a Ph.D. student in Computer Application Technology at Inner Mongolia University. He is also an associate professor in Baotou Teachers' College at Inner Mongolia University of Science & Technology. His research activity is in network security and mobile computing.

**Junxing Zhang** is a Professor in the College of Computer Science at the Inner Mongolia University. He is also the Director of the Inner Mongolia Key Laboratory of Wireless Networking and Mobile Computing. He received a B.S. degree in Computer Engineering from the Beijing University of Posts and Telecommunications, an M.S. degree in Computer Science from the Colorado State University, and a Ph.D. degree from the University of Utah. His research interests include network measurement and modeling, mobile and wireless networking, network security and verification, etc. Prof. Zhang was awarded the title of Grassland Talent by the government of the Inner Mongolia Autonomous Region in 2010. He has published over 40 papers in various internationally recognized journals and conferences, and led several national and provincial research projects. He also served as a peer reviewer for several international journals and conferences, such as IEEE Transactions on Mobile Computing, Wireless Networks, and ICNP.

**Zhanfei Ma** is a Professor in Baotou Teachers' College at Inner Mongolia University of Science & Technology. He received a B.S. degree in Computer Science and Education from Inner Mongolia Normal University in 1997, received an M.S. degree in Computer Software and Theory in 2002, and received a Ph.D. degree in Computer Application Technology from University of Science & Technology Beijing in 2008. His research interests include network information security and artificial intelligence.

*Received: February 6, 2020; Accepted: November 30, 2020.*