Automatic Blocking Mechanism for Information Security with SDN

Yi-Chih Kao[1]*, Jui-Chun Liu[1], You-Hong Wang[1], Yu-Huang Chu[2], Shi-Chun Tsai[3] and Yi-Bing Lin[3]
[1]Information Technology and Service Center, National Chiao Tung University, Hsinchu, Taiwan
{ykao, g0737, youhong}@mail.nctu.edu.tw
[2]Chunghwa Telecom Laboratories
yhchu@cht.com.tw
[3]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
{sctsai, liny}@cs.nctu.edu.tw

Abstract

Information security attacks initiated within an organization are the worst nightmare to all information management personnel. Although many potential solutions had been proposed for various attack scenarios, a complete field verification for these solutions has not yet been implemented in a complex network environment. In this paper, we propose a reliable, low cost and programmable *proximal defense* architecture by orchestrating software-defined networks (SDN) controller, SDN switches, legacy switches and application level firewall. Our defensive system can instantly detect various external-to-internal and internal-to-external attacks and block them via the closest programmable device to the attack source. The greatest advantage of this scalable architecture is that we can incrementally construct our defensive system from the original network and security control. Thus, internal users will not notice the migration and all events can be fully recorded for analysis. In addition, stability tests are conducted on both the original network architecture and auto-blocking SDN architecture. The experiments showed that the average response time after 2000 tests and the average throughput of uploading a 100-MB file for both architectures are almost the same. Furthermore, we test our system in a complex campus network environment by simulating a malicious behavior to verify its functionality. All test results live up with good expectations.

Keywords: Network Security, Software Defined Networking, Auto Blocking

1 Introduction

Most organizations possess computers and network equipment of different models and years, which may experience different levels of security problems. This is a challenge for network administrators. A typical university computer center could acquire numerous valuable devices over years, of which the management host may have a system-patch program installed to maintain strong security. However, if no patch programs are installed on these devices, the system is exposed to high security risks. Campus networks are vulnerable to such security issues. Major security problems include the openness of the campus, the large scale of the network structure, the diversity of users, and a decentralized network [18]. Meanwhile, campus network can easily be infected, if users browse social media privately. This leads to great difficulties in campus network management. The related maintenance and management consumes a significant amount of manpower and budget.

In a legacy information security defensive architecture, a firewall is usually deployed at the front end of the external-to-internal network to monitor network traffic through incoming and outgoing packets.

*Corresponding author: Information Technology and Service Center, National Chiao Tung University, 1001 University Road, 30010, Hsinchu, Taiwan, Tel: +886-922-186-666

When an external attack occurs, detection and blocking function can be effectively performed. However, most network administrators and security controllers hold an incorrect belief that most network events are often caused by external attacks, and with a firewall installed, the system should be safe and free from attacks. Many network security incidents result from inappropriate network usage, or from infection or intrusion before becoming a springboard, and result in loopholes in information security management. Thus, to manage network security effectively, cutting-edge security technology is needed to detect anomalous network behaviors and attack events in real-time as well as to block devices closest to the attack sources (proximal defense).

The management of network security defense in large-scale and complex network structures is usually difficult. In 2008, McKeown et al. [12] proposed software-defined networks (SDNs), which presented advantages over legacy networks. In SDN, separation of the control and data planes enhances controller programmability and flexibility. Thus, the controller can communicate with southbound devices remotely, which reduces hardware cost. Kanazawa University Hospital [13] demonstrated an SDN use case by implementing SDN into their network environment, allowing network devices to be easily managed centrally, thereby lowering network management overheads. Bawany et al. [8] provided an in-depth survey and discussion on SDN-based distributed denial-of-service (DDoS) attack detection and mitigation mechanisms, and sort them by detection techniques. They then proposed an SDN-based proactive DDoS defense framework called ProDefense, which can be utilized for secure applications such as smart cities. In this study, we propose an SDN-based automatic blocking mechanism to avoid malicious attacks initiated from internal users.

The remainder of this paper is organized as follows. Section 2 discusses relevant literature on automatic blocking mechanisms; Section 3 proposes a feasible solution to improve existing problems; Section 4 verifies the feasibility of the system architecture and evaluates its effectiveness and stability; and finally, Section 5 summarizes the research findings and future applications.

2 Background and Related Works

This section surveys the applications of Auto Block and SDN technology in the field of information security for internal-to-external security issues.

2.1 Automatic Blocking Mechanisms in Information Security Protection

Today, most enterprises purchase application-layer firewalls as traffic in-and-out gateways of their external networks for detecting and blocking various information security attacks. However, application-layer firewalls usually have limitation on the number of processable sessions per second (i.e., max sessions and new sessions per second). Moreover, they are required to perform intrusion detection system (IDS) and intrusion prevention system (IPS) functions. When a server encounters various types of DDoS attacks, numerous sessions are generated instantly, which may overload the firewall central processing unit (CPU). When the loading of the firewall CPU reaches 85% or higher, it may not be able to maintain normal operation or even cause network service interruption [5]. Therefore, it is essential to reduce firewall CPU load under a DDoS attack through a mechanism that can block attack events automatically.

Naveed et al. [15] proposed storing warning messages generated by Snort in a database as a reference to the intrusion behavior and then execute Cisco access control list (ACL) rules in the router according to the warning messages. Haq et al. [11] proposed an auto mode, when the attack possibility of a device exceeds a (configurable) threshold, it would automatically block IP or specific malicious traffic of the attacking device. These blocking policies can be set by the administrator in advance, including blocking granularity, blocking duration, and setting a threshold for generating notifications and the device to be blocked. Although the automatic mode reduces management costs, it still has significant limitations, such as false alarms potentially resulting in the blocking of legitimate traffic. Fu et al. [9] proposed an automatic blocking mechanism that uses syslog generated by the firewall as a reference and executes an ACL command on the external-to-internal gateway router to block the IP of external-to-internal malicious attacks. Campus network verification proved that it could effectively defend external-to-internal DDoS attacks. However, the attack defense of internal networks was not thoroughly studied. Applying the same defensive mode for internal-to-external attacks will inevitably lead to an excessive CPU load as well as on the external-to-internal gateway router.

2.2 The White List and Filtering Mechanism in an SDN Controller

Bakker et al. [7] used SDN/OpenFlow to establish a virtual ACL firewall in a switch within a network to block the malicious behavior of internal hosts, which cannot be detected by a network boundary firewall. ACL rules are divided into different flow tables in the OpenFlow pipeline. The packets flowing into the switch first reach Table 0, which processes blacklist-related traffic. Any packet that meets the flow-table rule is discarded, whereas any packet that meets the table-miss rule is sent to Table 1 through the goto command. Table 1 processes white list-related traffic. If any packet meets the flow-table rule, it is sent to Table 2, whereas packets conforming to the table-miss rule are discarded. Finally, Table 2 transmits packets that conform to the flow-table rule to the switch port, and packets that conform to the table-miss rule are passed to the controller for processing. The study indicated that using a blacklist and a whitelist simultaneously engenders greater advantages than using only one. The permissive policy of a blacklist is suitable for situations where specific traffic must be blocked. When the policy is changed to allow only a small amount traffic to pass the firewall and excessive ACL rules are required, the restrictive policy of a whitelist exerts its effect. Giotis et al. [10] pointed out that although their algorithm can identify the characteristics of attack behavior, some benign network errors may be regarded as malicious.

2.3 Application of SDN Technology in Information Security

Ammar et al. [6] proposed a security-enhanced architecture based on SDN that integrates safety measures, such as a firewall and IDS/IPS, transmitting syslog records collected to the syslog server. These log data are then further analyzed and filtered by a security agent. The attack behavior is detected according to the feature comparison rules set by the administrator. After the system detects an attack, the security agent notifies the controller to block the source interface. However, the authors of this paper only used the Mininet software package [2] to establish the experimental test environment, and did not conduct a field test for their ideas. Nam et al. [14] asserted that few studies have used a combination of open-source software and SDN for enhancing security. Therefore, they proposed using an open-source Suricata IDS/IPS system [4]. The study uses a mirroring method to transmit the packet traffic of a network device to a server executing intrusion detection, which then analyzes the contents of the mirror packets to identify if it conforms to attack behavior. If an attack occurs, the Suricata system issues a command to an SDN switch through the controller to realize the SDN firewall function. Although the author proposed an innovative perspective, such a solution presents concerns regarding effectiveness when it is applied to a complex network environment. After all, the performance of software firewall cannot compete with physical firewall devices. Shin et al. [17] claimed that the characteristics of SDN dynamic flow control could help a network to achieve security. In their system, when the switch receives a packet, the packet is transmitted to the application plane through the controller, and the firewall application module analyzes the contents of the packet for violations of security policies. After the packet is determined to be malicious, the firewall asks the switch to block the packet through the controller. Their solution could only provide simple ACL functions and was unable to update identification patterns of the firewall module. Yoon et al. [20] analyzed the advantages and disadvantages of a firewall application module under an SDN architecture. They believed that although the SDN firewall does not need to install additional network security devices, the firewall application module may consume considerable resources and eventually lead to degradation of network performance. Furthermore, numerous flow entries may completely occupy the flow table. This paper showed that many studies have mentioned SDN security applications, but only provided conceptual prototypes, and the effectiveness of which has not been verified in a real network environment. Sonchack et al. [19] proposed the OpenFlow extension framework (OFX) architecture, which uses the computing capability of the switch to process the logic and mechanism of security application programs that were originally processed by the controller, and allows such programs to define a customized data layer on the existing SDN software/hardware infrastructure. This approach makes many security application programs more practical on existing hardware because it reduces the computing cost of processing complex packets and decreases the amount of information between the data and the control planes. Even though this method overcomes performance problem, many shortcomings still exist and remain unsolved. For example, it cannot be deployed on OpenFlow switch hardware that uses ASICs, and the expansion modules only provide specific application functions.

We compare previous solutions and ours in Table 1. As shown in this table, a firewall is either implemented through a firewall application module (app module) or a physical firewall where the stability and performance problems have been discussed previously. The same reason could explain the advantage of physical IPS/IDS functions over IDS/IPS application module (app module) and open source software. Although Nam et al. mentioned automatic blocking system using Openflow command, the detailed architecture was not delivered. To resolve the above issues, we propose proximal defense which highlights the difference from the other researches. Lastly, we verify our work in campus network through its functionality, reliability and stability.

Table 1: Comparison of related works

Author	Firewall	IPS/IDS	Auto Block	Proximal Defense	Concept Proof
Fu et al. [9]	physical	physical	script server	N/A	physical
Ammar et al. [6]	application module	physical	N/A	N/A	simulate
Nam et al. [14]	application module	open source	openflow command	N/A	N/A
Shin et al. [17]	application module	application module	N/A	N/A	N/A
Yoon et al. [20]	application module	application module	N/A	N/A	physical
Sonchack et al. [19]	hybrid	N/A	N/A	N/A	physical
This research	physical	physical	script server	V	physical

3 Proposed Framework

Although the mechanism proposed by Fu et al. [9] achieved automatic blocking in an external-to-internal defensive architecture, no practical solution for internal-to-external defense exists. Moreover, even a firewall can filter Internet packets, the internal users' attack cannot be detected by the firewall. To resolve these issues, we integrate the SDN technology with an automatic defensive mechanism to propose a new programmable automatic defensive architecture. Our method makes firewalls detect abnormal network activity closer to the client to achieve the internal-to-external automatic defensive mechanism.

In the aforementioned studies, all functions are connected to the northbound bridge through the SDN controller. This cannot withstand a major security attack. Thus, considering the constraints of the existing network environments and resources, we propose a new architecture that can maintain basic operational capabilities when a major attack occurs.

3.1 Programmable Automatic Defensive Architecture

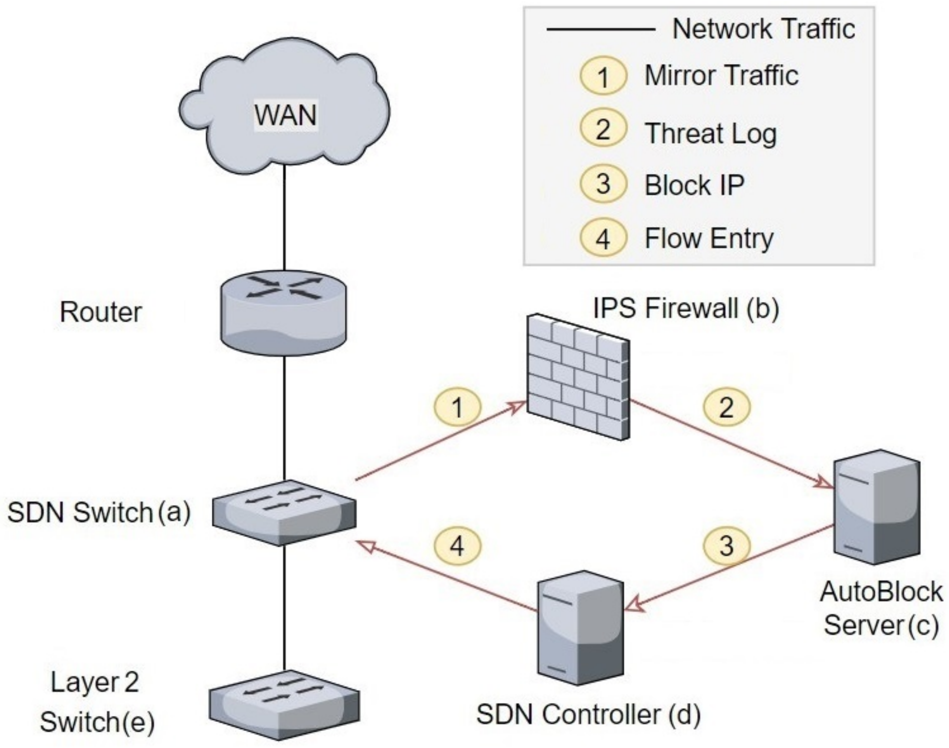

Figure 1: Diagram of our automatic defensive architecture

As shown in Fig. 1, the core components in this architecture includes an SDN switch, an IPS firewall, an Auto Block server, an SDN controller, and a layer 2 switch. Our approach proceeds as in the following steps. Step 1, the SDN switch (Fig. 1(a)) mirrors all network traffic to the IPS firewall and manages flow control to enable intelligent networking. Step 2, an IPS firewall (Fig. 1(b)) is used to receive network traffic, analyze the content information such as malicious behavior source, destination IP, and event type (e.g., malicious attack and a computer infected by a virus) and generate an event record (syslog). Step 3, the Auto Block server (Fig. 1(c)) is a server combining multiple scripts with different functional modules which will be further explained in the following section. When a malicious behavior is identified as an attack event, the Auto Block Server instructs the SDN controller to block the source IP. Step 4, the SDN controller (Fig. 1(d)) executes management control by assigning flow entry to the SDN switch and provides the relevant application programming interfaces (APIs) for external program to access. The layer 2 switch (Fig. 1(e)) provides the functions of address learning, forward/filter decisions, and loop avoidance.

3.2 Auto Block Server Modules

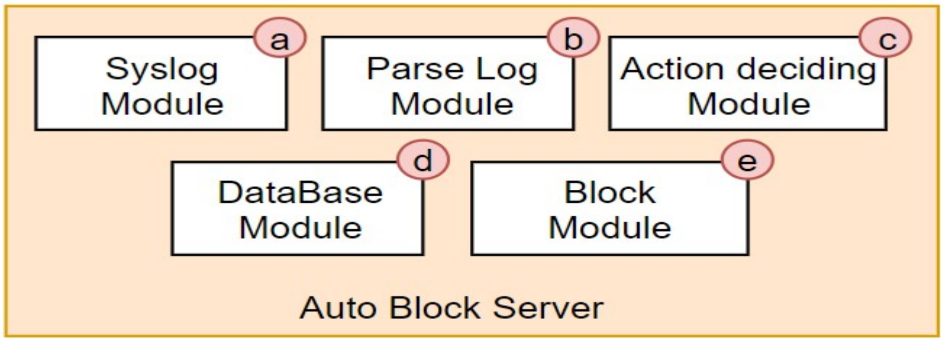

Figure 2: Diagram of the auto block server module

Fig. 2 shows the details of the Auto Block server (Fig. 1(c)), which consists of several modules including syslog module, parse log module, action deciding module, database module, and block module. The syslog module (Fig. 2(a)) is responsible for receiving event records from the IPS firewall; the parse log module (Fig. 2(b)) sorts out syslog by certain patterns; the action deciding module (Fig. 2(c)) is diagnosing whether it is necessary to block the event host according to the severity of the event, a whitelist is also established in this module to prevent benign network traffic from being mistaken for a threat and blocked; the database module (Fig. 2(d)) is where the relevant records are saved; the block module (Fig. 2(e)) calls APIs of the SDN controller to initiate the blocking.

3.3 Auto Blocking Defensive Mechanism

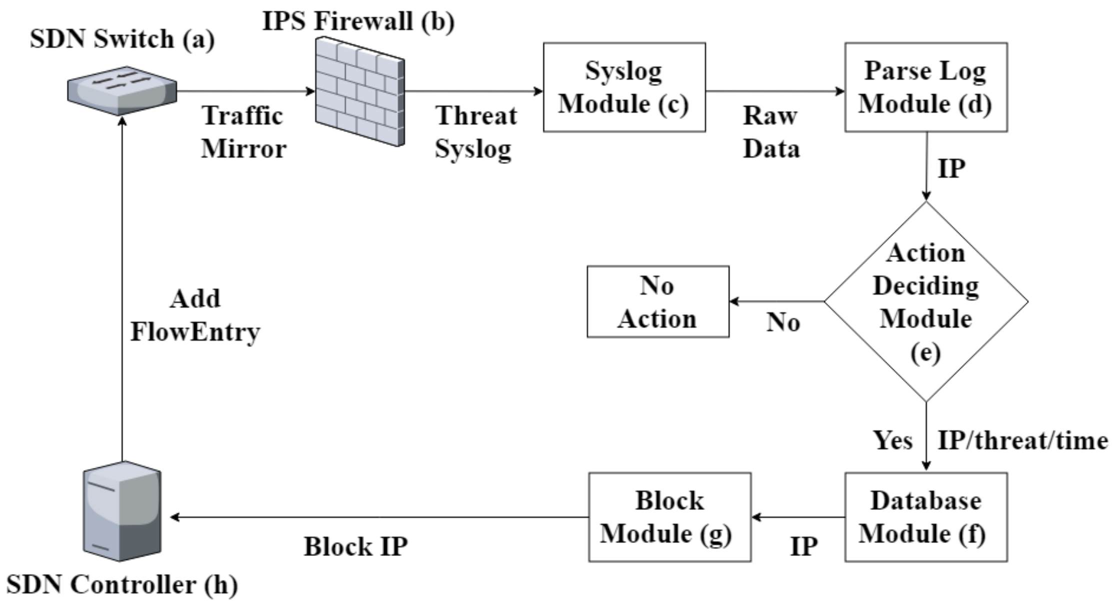

Figure 3: Auto blocking defensive mechanism flow chart

To realize our automatic defensive mechanism and overcome the limitations of legacy network architecture, such as unable to monitor underlying network traffic, we connect end user devices onto the layer

2 network switch. In Fig. 3, through the SDN switch (Fig. 3(a))we mirror all network traffic to the IPS firewall (Fig. 3(b)) for analyzing mirrored network traffic. The syslog module (Fig. 3(c)) receives the threat log generated by the IPS firewall and then diagnoses whether blocking the event host is necessary according to the severity of the event. Then the parse log module (Fig. 3(d)) checks the blocking patterns by parsing raw data from syslog module. When the attack event reaches the blocking criteria (clipping level), the action deciding module (Fig. 3(e)) takes action along with whitelist. If the blocking threshold is not met, no action will take place; otherwise the IP address, threat variety and the time when the attack occurred will be sent to the database module (Fig. 3(f)). The block module sends parameters such as IP and blocking duration to the SDN controller (Fig. 3(h)) through API to block violating IP stored in the database module. The corresponding flow entry is further issued by the SDN controller (Fig. 3(g)) to the SDN switch. Thus, IP blocking can be performed on the network device closer to devices which exhibit abnormal network behavior, thereby reducing the damage caused by attacks.

The focus of this work is to implement the mechanism of automatic detection and blocking internal-to-external attack events. In a legacy network architecture, firewalls and IPS devices are usually installed on the external network backbone and are unable to control lower-layer users' network behavior. The main advantage of our architecture is that it mirrors SDN switch traffic to the IPS firewall to detect abnormal behavior. In addition to monitoring underlying network traffic, a physical IPS firewall device is far superior to a software firewall module in terms of effectiveness, stability, and abnormal behavior diagnosis. Furthermore, our Auto Block module remits the process of manual modification of the switch after an event log is manually determined, and realizes automated assertion, event log parsing, and API calling. By adopting programmable SDN network environment, faster and more accurate information security protection can be achieved.

4 Implementation and Performance Evaluation

For related research projects, we have used the Network Adapter with Programmability and Automation (NAPA) SDN controller [16] developed by Chunghwa Telecom Laboratories because of its availability, stability and reliability. Therefore, we selected the NAPA controller for our SDN environment. The NAPA controller is a web-based visualization SDN network management system developed with Ryu [3]. It provides the functions of flow entry management, device management, network topology management, and traffic monitoring terminal identification, which are necessary for an SDN controller. The management system of NAPA controller provides useful maintenance and service tools, which make it easy to manage multiple brands of SDN-based network devices.

To verify the stability and feasibility of our system, we integrated SDN into our network environment and tested its response time performance with the tool *My traceroute* (MTR) by continuously sending ICMP PING packets to the border gateway router. We also used iperf3 [1] to measure the throughput by uploading files of different sizes. In addition, we simulated an attack by implanting malicious programs on tested end user device to observe how the proposed mechanism reacts.

4.1 Campus Network Architecture

The campus network backbone architecture is presented in Fig. 4, where the border gateway core router (Fig. 4(a)) is connected to the internal gateway router (Fig. 4(c)) through an IPS firewall (Fig. 4(b)). A layer 2 switch (Fig. 4(d)) is connected to the internal gateway router directly. An SDN switch (Fig. 4(e)) is connected to the internal gateway router and then connected to the layer 2 switch (Fig. 4(f)) to provide network service for end users. This SDN switch is managed and controlled by the SDN controller (Fig. 4(g), CHT NAPA) and mirrors all network packets to the IPS firewall. An Auto Block server (Fig. 4(h)) is used to receive threat log generated by the IPS firewall. End user device 1 (Fig. 4(i)) and end user device 2 (Fig. 4(j)) represent testing devices within the original network architecture and the architecture with auto-blocking SDN respectively.

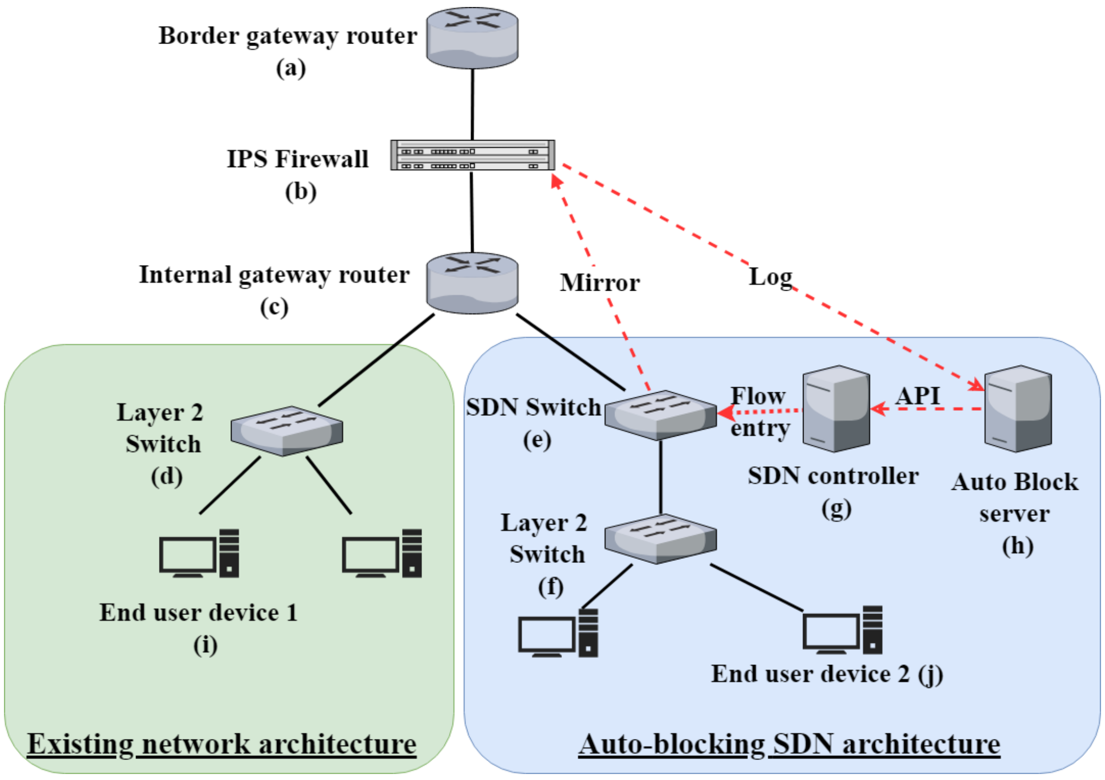

Figure 4: Diagram of the campus network architecture

4.2 Response Time Test

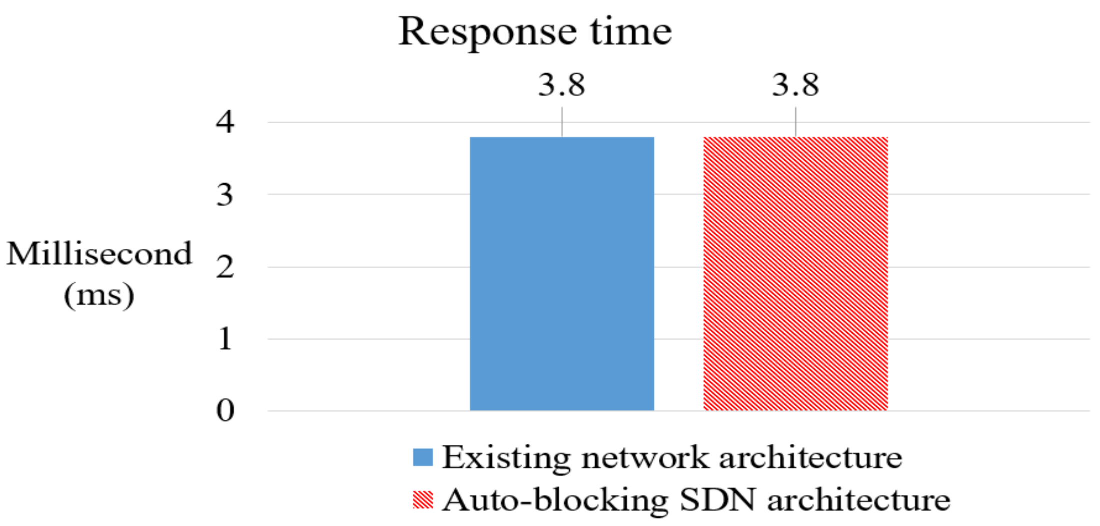

Figure 5: Response time comparison of existing and auto-blocking SDN architecture

To compare the service stability of auto-blocking SDN architecture with existing network architecture, we used the tool *My traceroute* (MTR) on the test computers by repeatedly sending ICMP PING packets to the border gateway router (Fig. 4(a)) for 2,000 times. Fig. 5 illustrates the results of the average response time of the original network architecture and auto-blocking SDN network architecture. The average response time for End user device 1 (Fig. 4(i)) in the existing architecture was 3.8 milliseconds without packet loss after the test, while the average response time for End user device 2 (Fig. 4(j)) in the auto-blocking SDN architecture was also 3.8 milliseconds without packet loss. Thus, we conclude that the auto-blocking SDN architecture did not affect the network performance.

4.3 Throughput Test

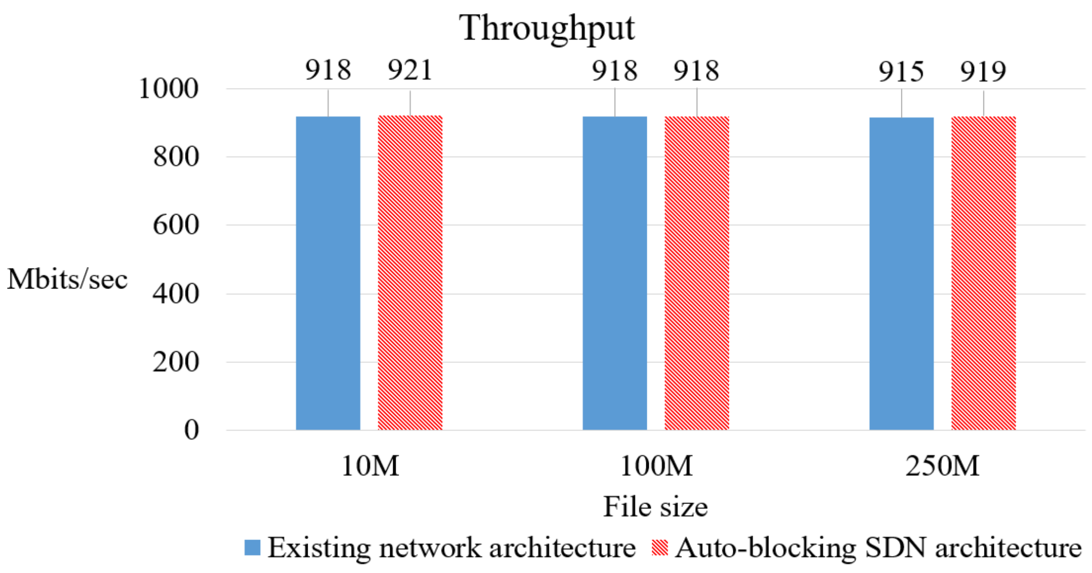

Figure 6: Throughput comparison of the original and auto-blocking SDN architectures

Iperf3 is a network performance measurement tool which provides various network information analysis that helps us better understand network status. To compare the throughput of both network architectures, we used Iperf3 to measure the throughput by uploading files of different sizes (10, 100, and 250 MB) to border gateway router (Fig. 4(a)) on End user device 1 (Fig. 4(i)) and End user device 2 (Fig. 4(j)) respectively. Each file size was experimented for 10 minutes and the results were shown every 60 seconds, and the average throughput was summarized in Fig. 6. The average throughput of 10-MB file in the existing architecture is 918 MB/s, while auto-blocking SDN architecture is 921 MB/s; the average throughput of 100-MB file in the existing architecture is 918 MB/s, while auto-blocking SDN architecture is also 918 MB/s; the average throughput of 250-MB file in the existing architecture is 915 MB/s, while auto-blocking SDN architecture is 919 MB/s. That is, there is only negligible difference between the two architectures as the testing result revealed. It indicates that our implementation neither causes performance degradation, nor affects user experience.

4.4 Malicious Behavior Simulation

To verify the function of our architecture, we simulate an infected computer querying the DNS server for the malicious domain. It can be done with the *nslookup* command to query a malicious domain.

For both architectures, the malicious DNS packets were identified as spyware by the IPS Firewall which triggered packet drop. The IPS Firewall generated and sent an event record of network traffic analysis at a rate of approximately once per minute. Table 2 lists each threat analysis record, which includes the time when the abnormal behavior detected (Time), type of abnormal event (Type), abnormal behavior ID (ID), name of the abnormal behavior (Name), source IP/attacker (Attacker), destination IP/victim (Victim), packet application-layer protocol (Application), and severity (Severity).

Table 2: Test results of the attack simulation

Time	Type	ID	Name	Attacker	Victim	Application	Severity
10:31:38 AM	spyware	4058685	Suspicious DNS Query	x.x.x.x	y.y.y.y	dns	medium
10:31:40 AM	spyware	4058685	Suspicious DNS Query	x.x.x.x	y.y.y.y	dns	medium
10:31:42 AM	spyware	4058685	Suspicious DNS Query	x.x.x.x	y.y.y.y	dns	medium
10:31:44 AM	spyware	4058685	Suspicious DNS Query	x.x.x.x	y.y.y.y	dns	medium

Moreover, the Auto Block server (Fig. 4(h)) provided SDN controller (Fig. 4(g)) parameters such as IP and blocking duration (Hard Timeout), then the SDN controller diagnosed event records from the IPS Firewall and added a flow entry (as in Table 3) to the SDN switch (Fig. 4(e)) to successfully block the malicious IP.

Table 3 shows the corresponding flow entry generated by the SDN controller. It includes the following information: FLOW ID, Timestamp, ofp_version, ControllerGroup, ControllerId, Priority, Idle_timeout, Hard_timeout, Packet_count, Byte_count, Cookie, Send_flow_rem, Actions, and MATCHFIELDS. FLOW ID is automatically produced by the SDN switch; Timestamp is the time when the flow entry is added; ofp_version is the version of openflow protocol in use, which means Openflow version 1.3 is used in the SDN switch; ControllerGroup is the group of the controller which generates this flow entry and is manually defined as x.x (IP); ControllerId is the controller which generates this flow entry in x.x ControllerGroup; Priority defines the priority of flow entry to be executed; Idle_timeout is the timeout during which if no packet matches the rule, the flow entry will be removed; Hard timeout is the absolute time that regardless of packet matching the rule or not, the flow entry will be removed when the Hard_timeout is reached. In this example, Idle_timeout is 0 and Hard_timeout is 60, which means that the flow entry will expire after 60 seconds; Packet_count is the number of processed packets matching the rule; Byte_count is the total received packets byte size matching the rule; Cookie is the information stored by the controller to transmit message between controller and switch, which is not used in this case; Send_flow_rem sends a message to controller when the flow entry is removed, which is enabled; Actions is the action after the condition is met; MATCHFIELDS is the rule matching condition. The None action in the flow entry indicates that the SDN switch will not forward any corresponding packets of the source IP described in MATCHFIELDS.

Table 3: A sample flow entry installed on the SDN switch

[FLOW_ID20]	
Timestamp	= Mon Dec 24 14:00:52 2018
ofp_version	= 4
ControllerGroup	= x.x
ControllerId	= 1
Priority	= 40032
Idle_timeout	= 0
Hard_timeout	= 60
Packet_count	= 262
Byte_count	= 56266
Cookie	= 0
Send_flow_rem	= true
Actions	= None
MATCHFIELDS	OFPXMT_OFB_IPV4_SRC = x.x.x.x

5 Conclusion and Future Work

Because firewalls are used to handle both external-to-internal and internal-to-external attack events, this often leads to high CPU load and affects network service effectiveness. In this work we effectively integrate and orchestrate the functionality of SDN and firewall to automatically detect and block internal-to-external attacks and thus reduce the CPU load of the firewall.

When the hardware firewall detects internal-to-external attack events, the SDN controller can issue responding flow rules to the SDN switch to automatically block the event source IP. We have installed the proposed architecture in a complex campus network environment and the experimental result shows that our system maintains its service level, and it does not pose any observable influence for the users. Our defense system is scalable which can be constructed in stages and seamlessly integrated with already existing defensive mechanism. The physical application level firewall can be shared and save construction cost, and thus maintain the consistency of firewall control policies.

With the powerful artificial intelligence technology, we may analyze the log of firewall more effectively and establish a whitelist mechanism to automatically block security incidents and reduce misjudgment rate. We leave it as a future work.

6 Acknowledgements

This research was supported in part by the Ministry of Economic Affairs of Taiwan under contract MOEA 107-EC-17-A-I1-0010, and was financially supported by the Center for Open Intelligent Connectivity from the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) of Taiwan.

References

[1] Iperf3, https://iperf.fr/iperf-download.php.

[2] Mininet, http://mininet.org/.

[3] Ryu sdn framework community, https://osrg.github.io/ryu/.

[4] Suricata open source ips/ids, https://suricata-ids.org/.

[5] Troubleshooting high cpu on a firewall device, https://kb.juniper.net/InfoCenter/index?page=content&id=KB9453.

[6] M. Ammar, M. Rizk, A. Abdel-Hamid, and A. K. Aboul-Seoud. A framework for security enhancement in sdn-based datacenters. In *Proc. of the 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Larnaca, Cyprus*, pages 1–4. IEEE, November 2016.

[7] J. N. Bakker, I. Welch, and W. K. Seah. Network-wide virtual firewall using sdn/openflow. In *Proc. of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA*, pages 62–68. IEEE, November 2016.

[8] N. Z. Bawany, J. A. Shamsi, and K. Salah. Ddos attack detection and mitigation using sdn: methods, practices, and solutions. *Arabian Journal for Science and Engineering*, 42(2):425–441, February 2017.

[9] S.-J. Fu, H.-W. Hsu, Y.-C. Kao, S.-C. Tsai, and C.-C. Tseng. An autoblocking mechanism for firewall service. In *Proc. of the 2017 IEEE Conference on Dependable and Secure Computing, Taipei, Taiwan*, pages 531–532. IEEE, August 2017.

[10] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. *Computer Networks*, 62:122–136, April 2014.

[11] O. Haq, Z. Abaid, N. Bhatti, Z. Ahmed, and A. Syed. Sdn-inspired, real-time botnet detection and flow-blocking at isp and enterprise-level. In *Proc. of the 2015 IEEE International Conference on Communications (ICC), London, UK*, pages 5278–5283. IEEE, June 2015.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

[13] K. Nagase. University hospital case study: OpenFlow deployment to campus lan, https://jpn.nec.com/univerge/pflow/pdf/ONS2013_kanazawa.pdf, 2013.

[14] K. Nam and K. Kim. A study on sdn security enhancement using open source ids/ips suricata. In *Proc. of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, South Korea*, pages 1124–1126. IEEE, October 2018.

[15] M. Naveed, S. un Nihar, and M. I. Babar. Network intrusion prevention by configuring acls on the routers, based on snort ids alerts. In *Proc. of the IEEE 6th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan*, pages 234–239. IEEE, October 2010.

[16] NTT Corporation. International service provider collaboration in sdn pushes forward ip packet transport to employ commodity products, http://www.ntt.co.jp/news2017/1712e/171212a.html, December 2017.

[17] S. Shin, L. Xu, S. Hong, and G. Gu. Enhancing network security through software defined networking (sdn). In *Proc. of the 2016 25th International Conference on Computer Communication and Networks (ICCCN), Waikoloa, HI, USA*, pages 1–9. IEEE, August 2016.

[18] U. K. Singh, C. Joshi, and N. Gaud. Measurement of security dangers in university network. *Measurement*, 155(1):6–10, December 2016.

[19] J. Sonchack, J. M. Smith, A. J. Aviv, and E. Keller. Enabling practical software-defined networking security applications with ofx. In *Proc. of the 2016 Network and Distributed System Security (NDSS), San Diego, CA, USA*, pages 1–15, February 2016.

[20] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang. Enabling security functions with sdn: A feasibility study. *Computer Networks*, 85:19–35, July 2015.