공학석사학위논문

SDN에서의 정보 엔트로피를 활용한 네트워크 이상 상황 탐지

Information Entropy based Network Anomaly Detection in Software-Defined Networking

2018년 2월

서울대학교 대학원
컴퓨터공학부

김 나 언

SDN에서의 정보 엔트로피를 활용한 네트워크 이상 상황 탐지

Information Entropy based Network Anomaly Detection in Software-Defined Networking

지도교수 김 종 권

이 논문을 공학석사학위논문으로 제출함

2017년 10월

서울대학교 대학원

컴퓨터공학부

김 나 언

김나언의 석사학위논문을 인준함

2017년 12월

위 원 장	전 화 숙	(인)
부위원장	김 종 권	(인)
위 원	권 태 경	(인)

ABSTRACT

Information Entropy based Network Anomaly Detection in Software-Defined Networking

Naeon Kim
Dept. of Computer Science and Engineering
The Graduate School
Seoul National University

Due to the spread of various smart devices and concomitant rise of network traffic rate, the importance of network infrastructure to accommodate them is increasing. As the network environment changes, a flexible and efficient network infrastructure that can easily deal with this has become necessary. So, Software-Defined Networking (SDN) has gained a lot of attention in recent years. SDN separates the physical forwarding function and logical control function of the network, and it centrally controls the network via API. However, the centralized control and programmable characteristics bring a lot of security issues. To mitigate security threats in SDN, many researchers have tried to monitor network traffic. Particularly to monitor SDN network, they collect flow statistics from the OpenFlow switches and detect network anomalies in the controller. But when the network scale becomes large, the flow statistics collecting process burdens the communication between the OpenFlow switches and the controller. In this thesis, we design a flow aggregation module in the OpenFlow switch based on the flow-based nature of SDN. This lightens the controller's overhead caused by the flow statistics collecting process and achieves a distributed flow statistics collection in SDN. In view of computing overhead and scalability, we apply information entropy to monitor and detect network anomalies in the controller. Information entropy is an important concept of information theory, which is a measure of the uncertainty or randomness associated with data. We prove the practicality of proposed network anomaly detection mechanism by using real legitimate and malicious traffic traces. The proposed mechanism achieves a high detection accuracy with a low false positive rate and significantly improves CPU utilization, compared with previous work.

Keywords: Software Defined Networking, SDN, Network Anomaly Detection, Network Traffic Monitoring, Information Entropy, Network Security

Student Number: 2016-21186

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter I

Introduction

Since networks are growing in size and complexity and the plurality of applications are running on them, network attackers attempt to exploit weak points of network architectures to corrupt or destroy valuable information [1], [10]. It is likely that networks would be compromised, causing individuals and companies suffering from great loss. In order to protect networks from being attacked, network anomaly detection has become an important research area [2].

And Software-Defined Networking (SDN) is an emerging networking paradigm that gives hope to change the limitations of current network infrastructures [3]. SDN exploits the benefits by decoupling the control plane from the data plane in routers and switches. The control plane can send instructions down to the data planes. And the switches in the data plane just simply forward data by checking the flow tables that are installed by the controller in the control plane. This helps to make changes globally without a device-centric configuration on each hardware and largely simplifies the switches' tasks.

SDN can easily analyze each packet in the controller for anomalous situation. However, network monitoring over the controller can significantly increase the overhead on the controller. It can result into degradation of the network performance. And due to its centralized control, it is easier to attack the controller with certain attacks. So as to overcome such problems, efficiently monitoring the network traffic for network anomalies in SDN is required.

These days, a lot of researchers have done researches on anomaly detection in SDN network. By implementing four anomaly detection algorithms in Small Office/Home Office (SOHO) environments, the authors tried to detect network security problems in home networks [4]. But this may work well when the size of network is small. They didn't consider the overhead to the controller in networks with high traffic. In [5], the authors periodically collected all flow entries in the OpenFlow switches. Nonetheless, the approach caused a considerable communication overhead between the OpenFlow switches and the controller. And some researchers took advantage of using sampling approach [6], such as sFlow [7]. However, the approach brings a tradeoff between sampling rate and anomaly detection accuracy.

In this thesis, we propose information entropy based anomaly detection mechanism in SDN. We aim to lighten the computation overhead of the controller using information entropy. And we make the OpenFlow switch collect and forward per-flow statistics counts to the controller, reducing the communication overhead between the OpenFlow switches and the controller. Our proposed mechanism also observes SDN network accurately in real time.

The main contributions of this thesis can be summarized in the following three points:

- In view of the scalability, the proposed mechanism uses information entropy which can deal with a large amount of traffic. And information entropy has a relative low calculation overload, reducing the overhead on the controller.

- Taking advantage of per-flow statistics in the OpenFlow switches, the switches only forward simple count information to the controller. So, it minimizes the communication overhead between the OpenFlow switches and the controller.
- Experiments on real legitimate and malicious traffic traces have shown that the CPU usage in the controller is lower than centralized approach and anomaly detection accuracy is also high with a low false positive rate.

The remainder of this thesis is organized as follows. Chapter 2 gives comprehensive background on SDN and Chapter 3 describes the related work of network anomaly detection in SDN. Before introducing our anomaly detection mechanism, we analyze network anomalies in the manner of three-dimension in Chapter 4. In Chapter 5 and Chapter 6, we explain the proposed mechanism in detail and show our experiment setups and performance analysis. Finally, Chapter 7 concludes this thesis.

Chapter Ⅱ

Background

Software-Defined Networking (SDN) is an emerging network architecture [3], shown in the Figure 2.1. SDN can be defined by two main characteristics: decoupling of the control plane from the data plane and programmability on the control plane. It addresses the lack of programmability in existing network architectures and enables easier and faster network innovation.

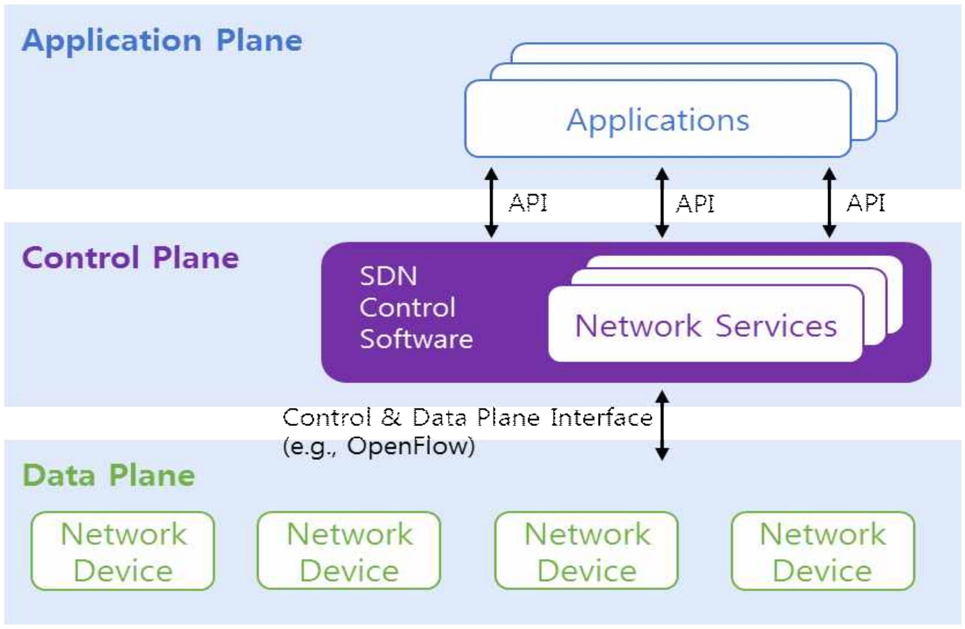


Figure 2.1: SDN Architecture

With the figure of SDN architecture, it is divided into the application plane, the control plane and the data plane. The control plane plays a role as an intermediary layer between the application plane and the data plane. A controller in the control plane communicates with the application via the northbound API and with switches in the data plane via the southbound API. And the intelligence of the network can be removed from switching devices and placed on the controller. Then, packet forwarding is one primary function of the data plane. The control plane provides a programmatic interface for developing management applications. From a management point of view, programmability opens the opportunity to reduce the complexity of distributed configuration and ease the network management tasks.

The OpenFlow protocol has been accepted as the *de facto* interface between the controller and the OpenFlow switches [8]. The controller communicates with the OpenFlow switches through a secure channel. The controller manages the flow tables of the OpenFlow switch by adding, modifying and deleting flow entries and rules. Each flow entry also maintains counters to collect flow statistics for particular types of flow. Due to the flow statistics provided by the OpenFlow switches, the controller has a global view of the network. Therefore, sophisticated and effective monitoring solutions can be developed using these capabilities of the OpenFlow.

When the OpenFlow handles a new flow coming into the OpenFlow switch, the controller handles the new flow reactively or proactively according to the controller's implementation, as shown in the Figure 2.2.

Under a reactive control mode, the controller responds to packet_in events by setting up policy rules for only that particular flow. A good example of reactive approach is Ethane [9]. In this case, there will be a small performance delay as the first packet of each new flow is forwarded to the controller for establishing the policy rule. The delay caused by processing the first packet may be negligible in many case.

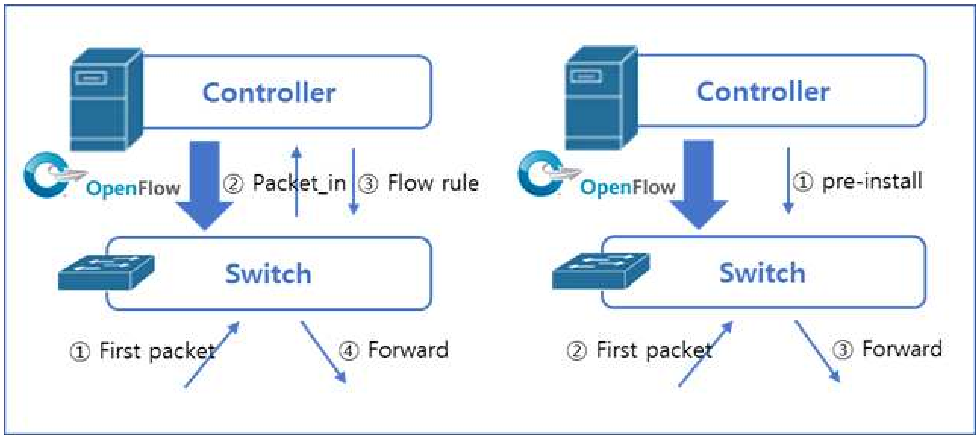


Figure 2.2: Reactive and Proactive mode of SDN

Alternatively, the controller using proactive control mode sets up flows in advance so that the controller should not get any packet_in events. In this case, the policy rule related to the first packet is already in the flow table, so no time is required to install the policy rule. Therefore, proactive approach avoids the extra latency on the first packet of the flow.

Chapter Ⅲ

Related Work

Efficiently monitoring and analyzing all the traffic in conventional network have been steadily studying for a long time [10]. In conventional network, it was difficult and complicated to observe all the traffic on the network. However, SDN network manages the whole network centrally on the controller, making the first packet of each flow pass through the controller. Monitoring and analyzing SDN network are more efficient and less overhead than conventional network. So, monitoring and detecting network anomalies in SDN are considered as one of the important issues [11]. And there are also researches that observe various networks such as WLAN by utilizing advantages of SDN network [12]. When monitoring SDN network, the controller may have the overhead of computation required for network analysis, which may result in degrading the overall performance of the network. Therefore, researchers should choose the appropriate method that does not burden the controller.

In [4], the authors implemented four anomaly detection algorithms which used in Small Office/Home Office (SOHO) environments. However, the authors only considered networks with low traffic as they focused on home environments and they did not take into account the overhead on the controller in high traffic network.

In [5], to detect network anomalies, especially a DDoS attack, in the controller, the authors implemented a flow collector module. The flow collector module periodically gathered requesting flow entries of all flow tables from the OpenFlow switch. Nonetheless, this module also didn't consider scalability and the overhead of the controller when the network is fairly large.

So, some researchers leveraged the sFlow protocol [7] which is the sampling technology embedded within switches and routers. In [6], the authors reduced the overhead of flow statistics collection using sampling method and the communication overhead between the OpenFlow switches and the controller. It mitigated the controller's overload in the large network. However, sampling method may affect the accuracy of network monitoring and anomaly detection.

And nowadays, many researchers try to add some intelligence to OpenFlow switches. In [13], the authors leveraged the local CPU to run simple measurement modules to collect and analyze more data at the OpenFlow switches.

Many researchers have also used information entropy based approaches, which have proven to be effective particularly in determining the randomness of a data set. Information entropy based approaches are capable of real-time detection and can handle large amounts of traffic data [6]. One of the most important advantages using information entropy is a low computation overhead. We also take advantage of these properties to monitor and detect network anomalies in SDN network.

Chapter Ⅳ

Analysis of Network Anomalies

4.1 Network Anomaly Detection

Network anomaly detection refers to the problem of finding exceptional patterns in network traffic that do not follow the expected normal behavior [2]. These unusual patterns are commonly referred to as anomalies and outliers in the context of network anomaly detection. For example, an anomalous traffic pattern in a network may mean that a hacked computer is sending out sensitive data to a host. Otherwise, it may mean a flash crowd that is a legitimate flooding traffic as the number of users who access the server increases significantly during any specific event. Hence, network anomaly detection needs to distinguish the legitimate flash crowd from illegal network anomalies. In this thesis, we concentrate on a DDoS attack, a hostscan attack and a portscan attack among various network anomalies.

4.2 DDoS

A Denial-of-Service (DoS) is the attack that prevents the victim host from providing normal services or receiving services on the Internet. When the traffic of the DoS attack comes from various sources, it is called a Distributed Denial-of-Service (DDoS) attack [14]. The DDoS attack uses numerous compromised hosts to increase the transmission of meaningless packets and may make the victim host overloaded and crashed.

When the DDoS attack occurs, the packets transmitted from distributed malicious hosts have the following three-dimensional characteristics. In case of source IP addresses, the distribution varies widely. However, a large amount of packets are destined for a particular victim host. So, the distribution of destination IP addresses is concentrated. These characteristics can be expressed in three-dimension as the Figure 4.1(a). In short, the number of source IP addresses increases infinitely because source IP addresses are usually manipulated at random. On the other hand, destination IP addresses have convergence characteristics because of focusing on the victim host.

4.3 Hostscan and portscan

A hostscan attack and a portscan attack are used as a preparation for several network attacks. Before attacking, an attacker needs to have information about the vulnerable target host and port number [15].

The hostscan attack is the process of finding active hosts in the network. Using the hostscan attack, the attacker determines which hosts in the network are vulnerable to network attacks. After the victim hosts are determined, the attacker leads with the porstscan attack.

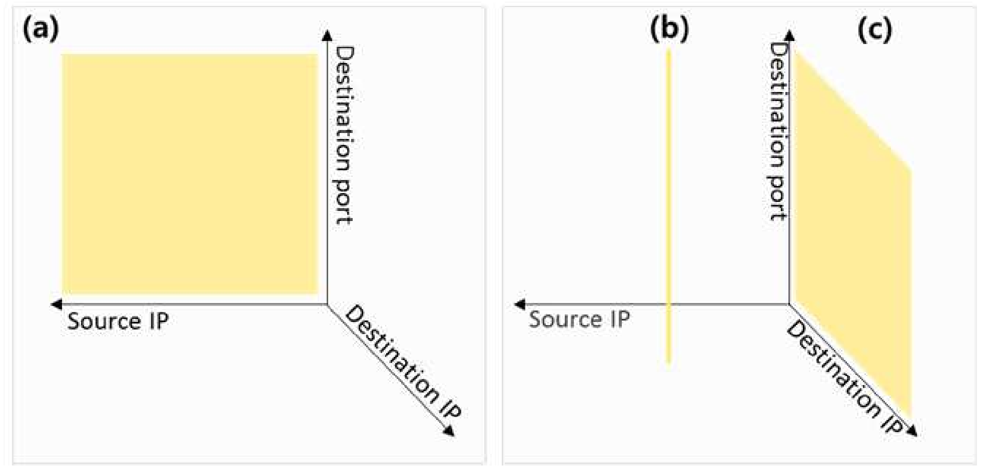


Figure 4.1: Three-dimensional analysis of network anomalies

The portscan attack is the method of determining whether a port is open on the victim host by observing the responses to connection attempts. The porstscan attack consists of sending a message to a port on the victim host and listening to a response which can indicate the port status. Thus, the attack can be helpful in launching other attacks on the victim.

The characteristics of the above attacks are as follows. The hostscan attack sends packets from a specific source IP address to various destination IP addresses, as shown in the Figure 4.1(c). In the case of the portscan attack, packets are sent from a particular source IP address to various ports at a specific destination IP address. It is illustrated as the Figure 4.1(b).

To wrap it up in a nutshell, we analyze three different network attacks in the manner of three-dimension and we find out that these attacks have prominent three-dimensional features.

Chapter V

Proposed Mechanism

In this chapter, we first briefly introduce adversary model and proposed mechanism compared with centralized approach. Then, we sequentially describe flow aggregation module, anomaly detection module using information entropy and anomaly mitigation module.

5.1 Adversary Model

An adversary can exploit the reactive mode of SDN network to launch an attack. Figure 5.1 depicts our adversary model. To make a DDoS attack and a hostscan attack, the adversary sends massive table-miss packets to a victim host. They are randomly forged, including source and destination IP address and port number. In this case, the total number of packets sent to each address is not 1 time but more than 2 times. And the time interval is less than 0.01 seconds, which is shorter than legitimate traffic. To process table-miss packets, the victim switch directly connected with the victim host buffers them and sends packet_in messages to the controller. According to OpenFlow 1.4 version, a packet_in message should contain the whole packet when the memory of an OpenFlow switch is full.

The adversary generates massive table-miss packets to jam the bandwidth between the OpenFlow switch and the controller. And the controller exhausts computational resources by installing flow rules for table-miss packets. The adversary exploits these features to paralyze SDN network.

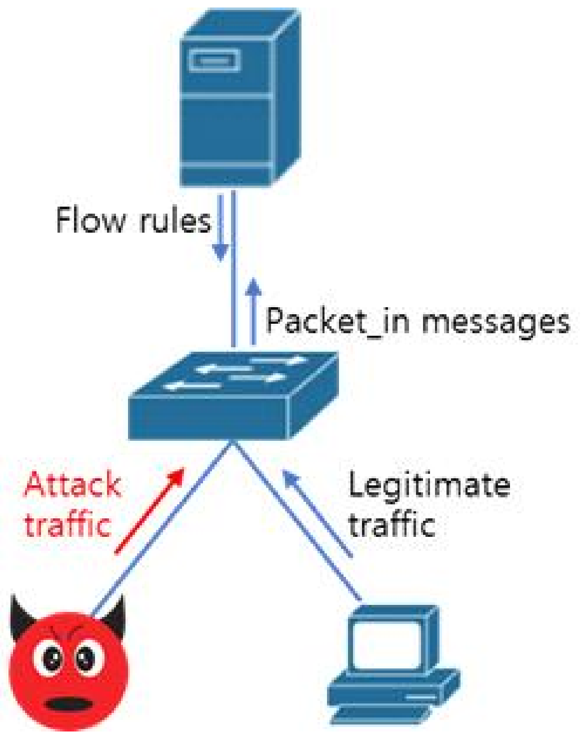


Figure 5.1: Adversary Model

5.2 Information Entropy based Network Anomaly Detection in SDN

To exploit the characteristics of network anomalies expressed in three-dimension, we use information entropy to detect abnormal situation in SDN.

The information entropy proposed by Shannon is an important concept of information theory, which is a measure of the uncertainty or randomness associated with data coming over the network. If it is more random, it contains more information entropy. The information entropy shows its minimum value 0 when all information is same and its maximum value $\log n$ when all information is different. Hence, comparing the information entropy value of some packets provides a mechanism for detecting changes in the randomness.

To acquire the information required for calculating information entropy value, we put our flow aggregation module at the OpenFlow switch which is close to the victim host, as shown in the Figure 5.2(b). The OpenFlow switch forwards the result of flow aggregation process to the controller. This achieves a distributed flow statistics collection in SDN and reduces the heavy communication overhead without collecting all of flow entries from the OpenFlow switch. The controller uses the count received by the OpenFlow switch to calculate the information entropy value, which detects abnormal situation in the network. The information entropy based anomaly detection mechanism can handle large amounts of traffic data and detect network anomalies in real time. And the mechanism has also a relative low calculation overload.

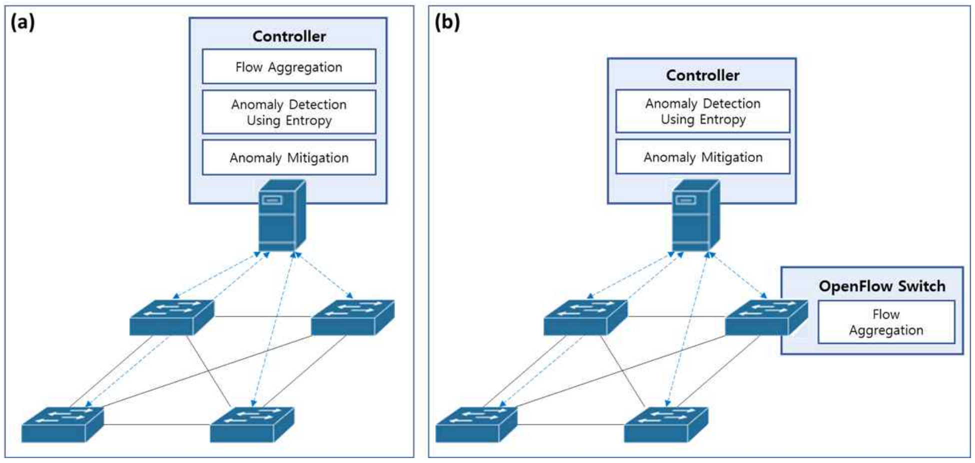


Figure 5.2: Anomaly detection and mitigation mechanism for (a) centralized approach (b) proposed approach

However, in the centralized approach, all of modules required for detecting network anomalies are located in the controller, as shown in the Figure 5.2(a). The controller periodically receives flow tables from all the OpenFlow switches and then aggregates per-flow statistics. Using per-flow statistics, it calculates information entropy value of each attribute and observes anomalous situation in SDN network. The anomaly detection module in the controller triggers anomaly mitigation module if information entropy value differs from the normal state.

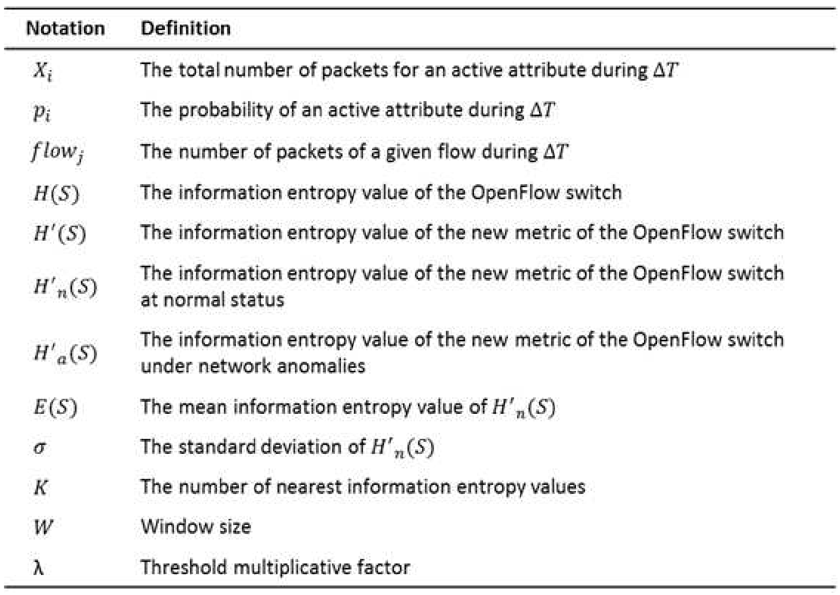
5.3 Flow Aggregation

The OpenFlow switch has at least one flow table, which has many flow entries. We specify the definition of flow used in this thesis.

- Flow refers to a set of sequential packets passing through the same network during time period. The sequential packets have same common properties such as source IP address, destination IP address, source port number, destination port number and network protocol.

And notations used in this thesis are defined in the Table 5.1.

Table 5.1: Notations used in the proposed mechanism



Notation	Definition
X_i	The total number of packets for an active attribute during ΔT
p_i	The probability of an active attribute during ΔT
$flow_j$	The number of packets of a given flow during ΔT
$H(S)$	The information entropy value of the OpenFlow switch
$H'(S)$	The information entropy value of the new metric of the OpenFlow switch
$H'_n(S)$	The information entropy value of the new metric of the OpenFlow switch at normal status
$H'_a(S)$	The information entropy value of the new metric of the OpenFlow switch under network anomalies
$E(S)$	The mean information entropy value of $H'_n(S)$
σ	The standard deviation of $H'_n(S)$
K	The number of nearest information entropy values
W	Window size
λ	Threshold multiplicative factor

Firstly, we do the flow aggregation process. In this thesis, we set source and destination IP address and destination port as an attribute to aggregate flow statistics. The OpenFlow switch calculates the number of packets for an active attribute during $\triangle T$. For example, the count number for specific source IP address during $\triangle T$ can be calculated as follows.

$$X_i = \sum_{j=1}^{k} flow_j [IP_{src} = IP_i] \ \dots (1)$$

Then, the OpenFlow switch can get $X = \{X_1, X_2, \dots, X_N\}$ as the count number of the packets collected by the specific attribute during $\triangle T$. Then, the OpenFlow switch forwards the count number to the controller every $\triangle T$.

Since the OpenFlow switch naturally records the flow statistics such as per-port counters, per-flow counters and per-table counters, we leverage the per-flow counters to obtain the number of packets related to specific flow during $\triangle T$.

5.4 Anomaly Detection

The controller calculates the frequency of each attribute to approximately estimate its probability, using the count number forwarded by the OpenFlow switch.

$$p_i = \frac{X_i}{\sum_{i=1}^{N} X_i} \ \dots (2)$$

Then, the controller can get $P = \{p_1, p_2, \dots, p_N\}$ as the probability distribution of each attribute and calculates its information entropy value as follows.

$$H(S) = -\sum_{i=1}^{N} p_i \log p_i \ \ldots (3)$$

Although the existing approaches consider the attributes independently, we expect that it is more efficient to compare all the attributes to detect the abnormal situation in the network. Thus, we identify the abnormal traffic through the information entropy difference value of attributes. In case of a DDoS attack and a hostscan attack, we define the new metric $H'(S)$ as formula (4). And in case of a portscan attack, we define $H'(S)$ as formula (5).

$$H'(S) = \left| H_{src}(S) - H_{dst}(S) \right| \ \ldots (4)$$

$$H'(S) = \left| H_{port}(S) - H_{src}(S) - H_{dst}(S) \right| \ \ldots (5)$$

The information entropy value can be divided into two parts: before anomalous situation and under anomalous situation like a DDoS attack and a hostscan attack. The information entropy value is correspondingly denoted by $H_n'(S)$ and $H_a'(S)$. Thus, $H_n'(S)$ remains stable because the network traffic is relative stable without anomalous situation. Let $E(S)$ and σ be the mean information entropy value and the standard deviation at the stable phase. When the network traffic is out of the normal pattern, the following formula will be met.

$$H_a'(S) - E(S) > \delta \ \dots (6)$$

In order to distinguish between legitimate flash crowd and illegal network anomalies, we consider illegal network anomalies if the formula (6) is satisfied at least K times during the window W. Then, the controller triggers the anomaly mitigation module. And we set the threshold δ using multiplicative factor λ as follows.

$$\delta = \lambda\sigma \ \dots (7)$$

5.5 Anomaly Mitigation

After the network anomalies has been detected, another crucial thing to do is the anomaly mitigation. The anomaly detection module triggers the anomaly mitigation module and passes the related information. The simplest way to response anomalous situation is installing a drop action to the OpenFlow switch. Our anomaly mitigation module also follows the drop action. After mitigation process, the controller recalculates $E(S)$ and σ. We have left the efficient anomaly mitigation module as future work.

Chapter Ⅵ

Evaluation

6.1 Experiment Setup

To evaluate the proposed mechanism, we used Mininet 2.2.1 [16] as the network emulator. Mininet is the standard network emulation tool for SDN, creating a realistic virtual network of hosts, switches, controllers and links. For our flow aggregation module, we implemented the algorithm in Open vSwitch [17], a famous SDN software switch. And we implemented the anomaly detection module and the anomaly mitigation module as the application of Pox 2.0 [18] written in Python language. All experiments were done at Ubuntu 14.04.3 LTS, Intel Core i5 CPU and 8GB RAM.

The Figure 6.1 is our experimental network topology which was created by mininet. The network attackers were launched from outside of SDN network. All victim hosts were connected directly to the OpenFlow switch 4 (S4). And SDN network and attackers' network were connected through a gateway with 1Gbps bandwidth. The bandwidth of all other links was 800Mbps.

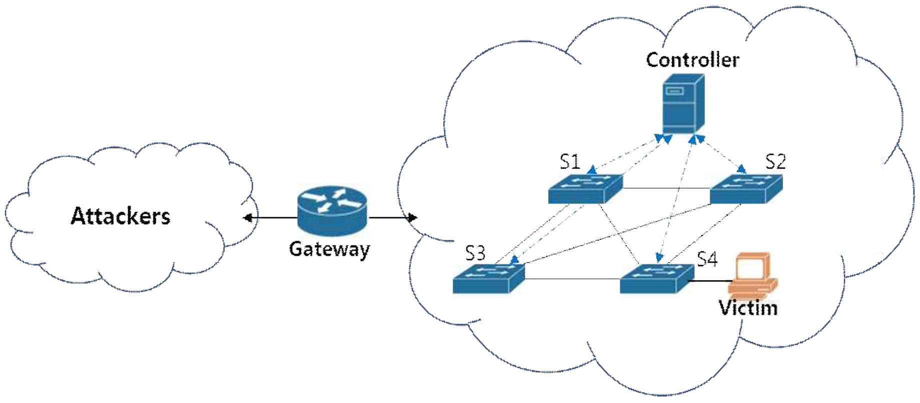


Figure 6.1: Experimental network topology

6.2 Network Traffic Dataset

In order to evaluate the proposed mechanism with real network environment, we used real legitimate and malicious network traffic traces. For legitimate traffic, we leveraged CAIDA Anonymized Internet traces 2013 Dataset [19]. Since the data size is huge, we extracted two data which make up approximately 100Mpbs and 500Mbps for low and high rates of network traffic. We used Tcpreplay tool [20] to replay the legitimate traffic traces in mininet.

The malicious network traffic trace has been mixed with the legitimate network traffic trace after 250 seconds from the start of the experiment. We leveraged CAIDA DDoS Attack 2007 Dataset [21] for the DDoS attack experiment and DARPA 2000 Dataset [22] for the hostscan attack experiment. The CAIDA attack dataset includes the rate of the attack started at a mere 200Kbps and reached

80Mbps. So, we can evaluate the proposed mechanism under both low and high rates of the DDoS attack traffic. And we use the python tool Scapy [23] to generate a portscan attack from attackers to the victim.

6.3 Performance Analysis

We first experimented a DDoS attack on one victim host in the network of the switch S4. The total simulation time was 550 seconds and we injected the DDoS flooding attack from 250 seconds to 300 seconds. And we set $W=3, K=2, \lambda=3$ and $\triangle T=5$. The information entropy value tendency on the source and destination IP address of the switch S4 are shown in the Figure 6.2 and the Figure 6.3, respectively. The information entropy value tendency using the new

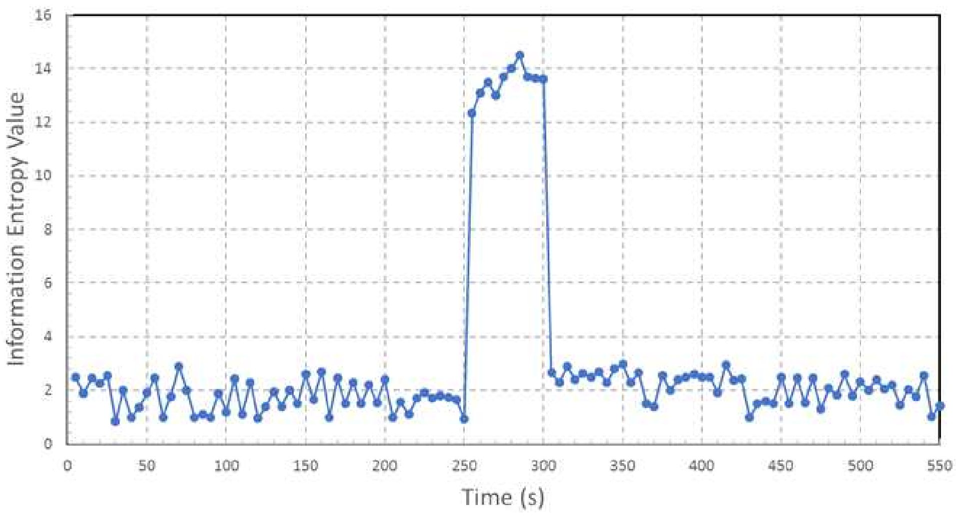


Figure 6.2: The information entropy value of source IP address under the DDoS attack

metric is shown in the Figure 6.4. It sharply increased at 255 seconds. As shown in the result graphs, it was efficient to detect the DDoS attack by monitoring all attributes together using the new

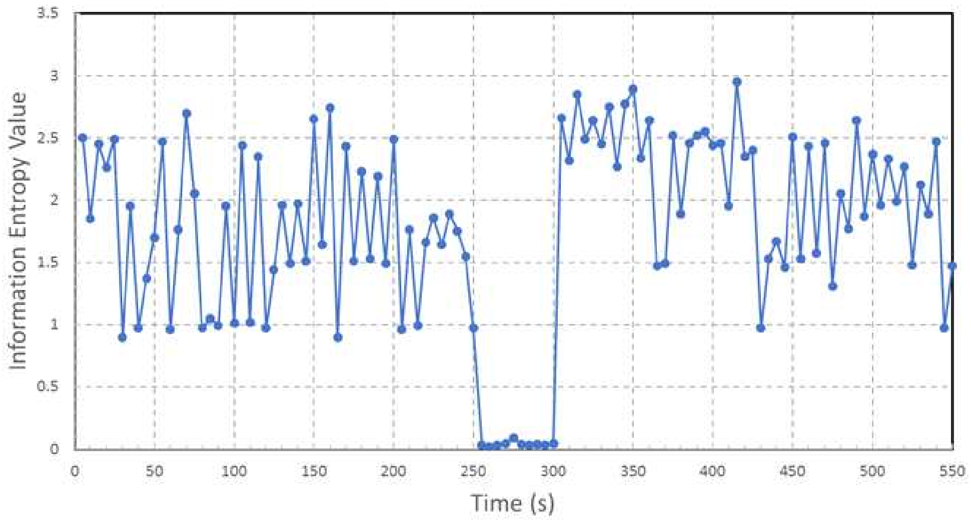


Figure 6.3: The information entropy value of destination IP address under the DDoS attack

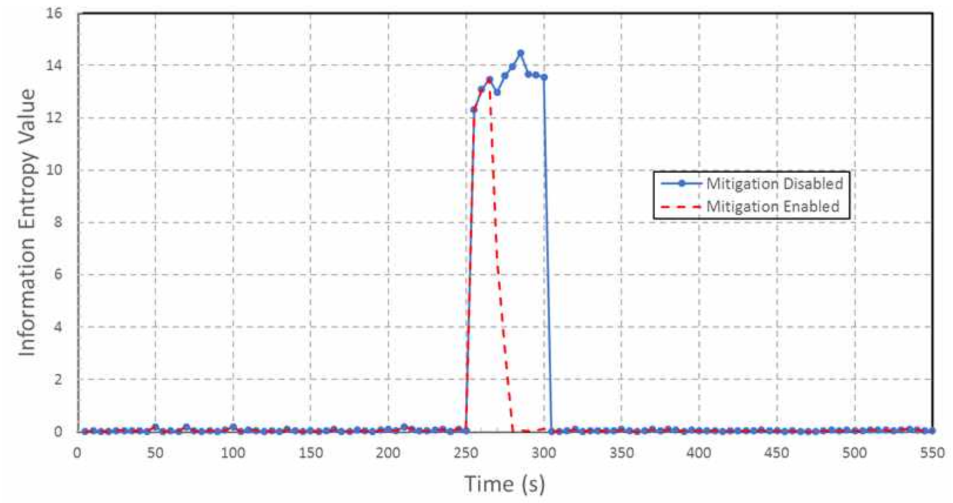


Figure 6.4: The information entropy value of new metric under DDoS attack

metric than by independently monitoring each attribute. The Figure 6.4 also shows the information entropy tendency with or without the anomaly mitigation module. The mitigation module was triggered at 265 seconds. It is obvious that once the network anomaly has been detected and mitigated, the information entropy values returned to the expected range.

The experiment of a hostscan attack was conducted in the same environment as the DDoS attack. The information entropy value tendency on the source and destination IP address of the switch S4 are shown in the Figure 6.5 and the Figure 6.6, respectively. Compared to the DDoS experiment, it was more difficult to detect the hostscan attack by independently observing the information entropy value of each attribute. The information entropy value tendency using the new metric is shown in the Figure 6.7, drastically increasing at 255 seconds. It was more efficient to compare attributes together in the hostscan attack as well as the DDoS attack. The information entropy value also returned to the expected range, once the network anomaly has been detected and mitigated. In this case, the anomaly detection module triggered the anomaly mitigation module at 270 seconds.

And the experiment of a portscan attack was also conducted in the same environment as the hostscan attack. In case of the portscan attack, we use different metric from the DDoS attack and the hostscan attack. The information entropy value tendency using the new metric includes the information entropy value of destination's port, destination IP address and source IP address. It is shown in the Figure 6.8, remarkably increasing at 255 seconds. The anomaly mitigation module was activated at 270 seconds and then the information entropy values returned to the normal pattern.

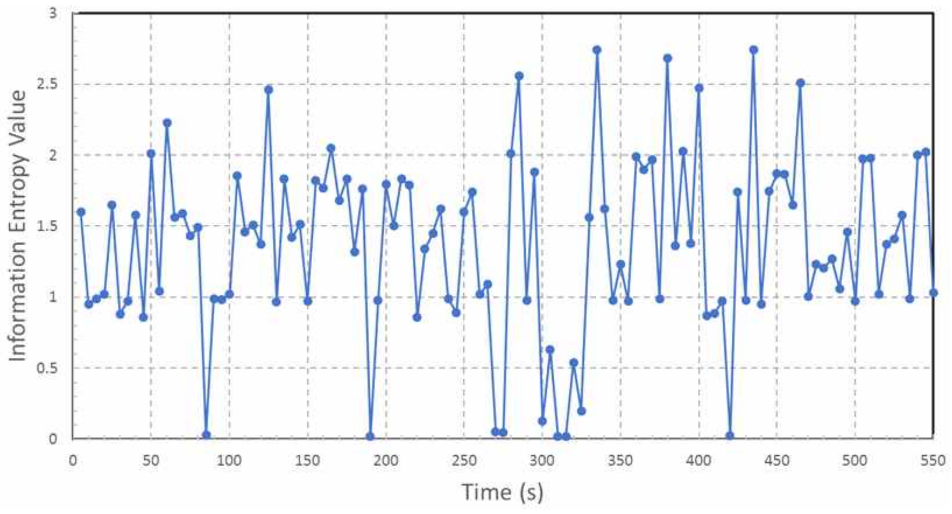


Figure 6.5: The information entropy value of source IP address under the hostscan attack

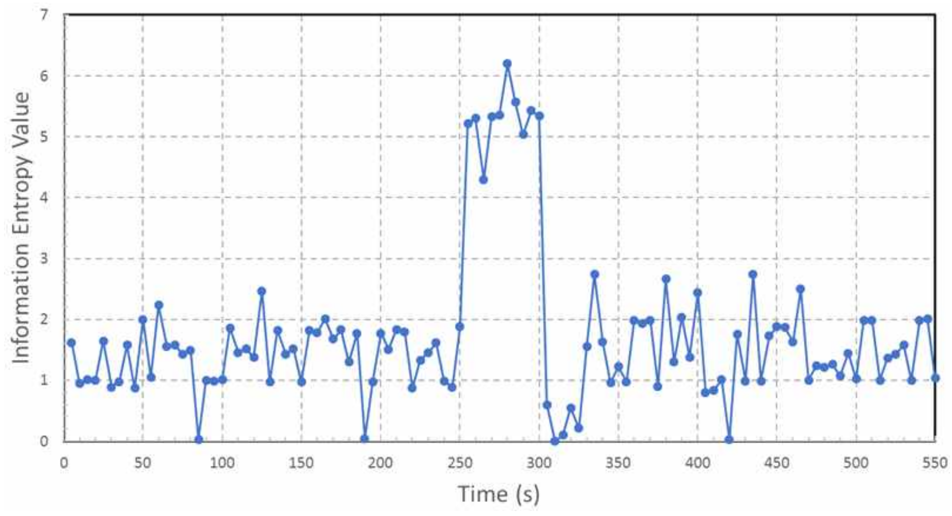


Figure 6.6: The information entropy value of destination IP address under the hostscan attack

To evaluate the effect of different numbers of victim hosts to our anomaly detection module, we changed the rate of victim hosts, such as one victim host and 20% victim hosts in the network of the switch S4. We used same experiment environment as the DDoS

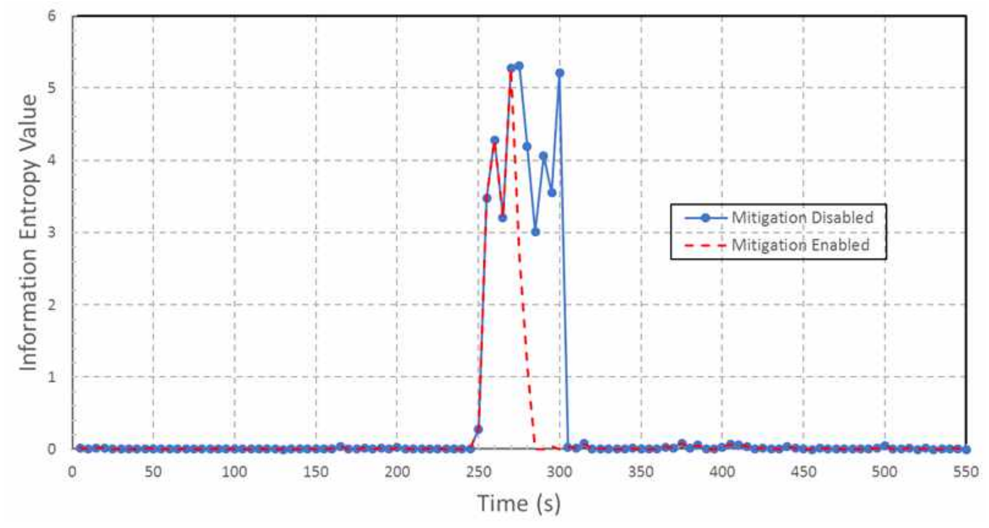


Figure 6.7: The information entropy value of new metric under the hostscan attack

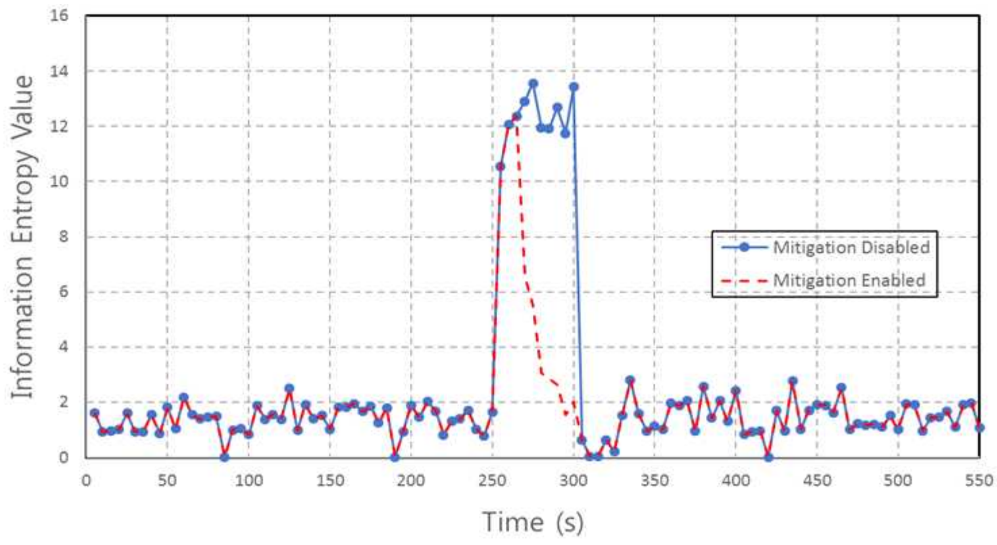


Figure 6.8: The information entropy value of new metric under the portscan attack

attack scenario. As shown in the Figure 6.9, the change of information entropy value was less when network anomalies occurred. It is difficult to distinguish the network anomalies from the normal

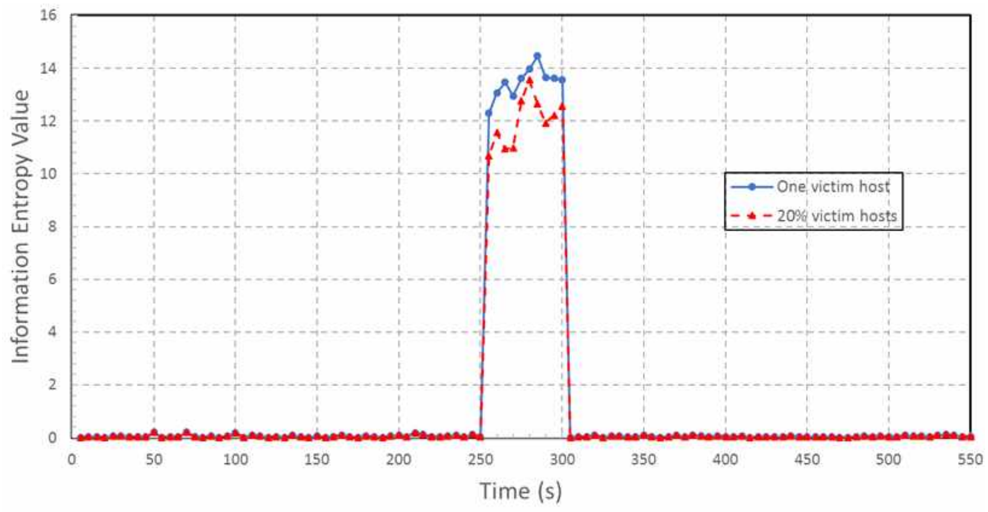


Figure 6.9: The information entropy value for different rates of victim hosts

pattern if the rate of victim hosts increases further.

We present the ROC (Receiver Operating Characteristic) curves to show the trade-off between the detection accuracy and the false positive rate. We measured these metrics under the DDoS attacks to one victim host. The parameters for experiment were same as others. The Figure 6.10 shows that the anomaly detection module can achieve 100% detection accuracy while has approximately 24~28% false positive rate for both 100Mbps and 500Mbps legitimate traffic. Since a high traffic network is beneficial to randomize the traffic, the module detects better in 500Mbps legitimate traffic.

Lastly, we evaluated CPU usage for the OpenFlow switch and the controller in order to assess the capability of our anomaly detection mechanism. The CPU usage is a key performance indicator for the operation of the controller. In the Table 6.1, we compared the average CPU usage of the OpenFlow switches for our distributed approach and the centralized approach. As we can observe, the CPU utilization

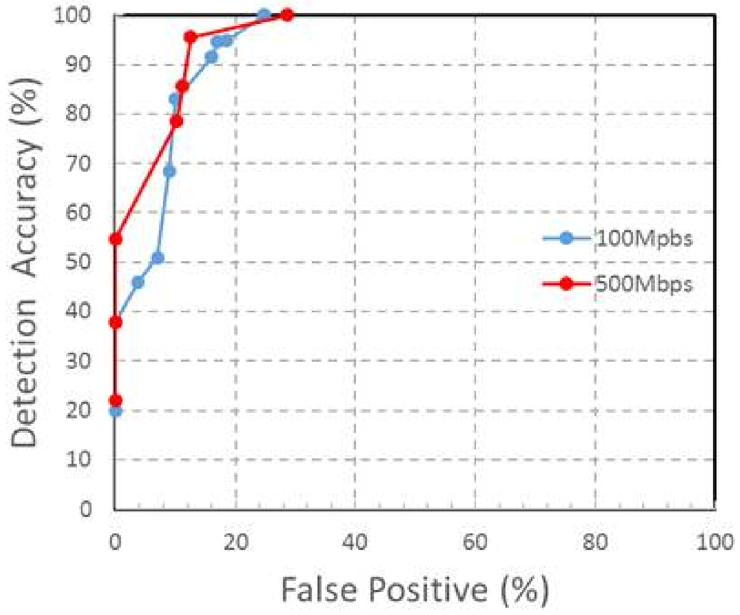


Figure 6.10: ROC curves for both 100 and 500Mbps legitimate traffic

of our distributed approach slightly increased from 61% to 62% when compared with the centralized approach. This is because our distributed approach aggregated per-flow statistics in the OpenFlow switch but it was negligible. As shown in the Table 6.2, we depicted the number of our distributed approach and the centralized approach, in terms of CPU usage for the controller. The required CPU usage of our distributed approach was reduced significantly from 63% to 41%, when compared with the centralized approach. Since the information entropy was used to detect network anomalies and the OpenFlow switch forwarded the necessary information to calculate the information entropy, the overhead of the controller can be notably reduced.

Table 6.1: CPU usage of the OpenFlow switch

OpenFlow Switch	100Mpbs legitimate traffic	Attack
	CPU usage (%)	
Centralized approach	47	61
Proposed approach	48.5	62

Table 6.2: CPU usage of the controller

Controller	100Mpbs legitimate traffic	Attack
	CPU usage (%)	
Centralized approach	42	63
Proposed approach	35	41

Chapter Ⅶ

Conclusion

Network anomalies can paralyze SDN network by exhausting the bandwidth and computational resources. We propose information entropy based network anomaly detection based on three modules: the flow aggregation module in the OpenFlow switch and the anomaly detection module and the anomaly mitigation module in the controller. The proposed mechanism can reduce the communication overhead between the OpenFlow switches and the controller by efficiently aggregating per-flow statistics. And it also decreases the burden of the controller by using information entropy to detect network anomalies. We implemented a prototype to evaluate the performance of the proposed mechanism using POX controller and Open vSwitch. The experimental results show that our mechanism efficiently detects network anomalies using new metrics and achieves a high detection accuracy with a low false positive rate. And compared with previous work, the proposed mechanism significantly improves CPU utilization.

BIBLIOGRAPHY

[1] Ahmed, Mohiuddin, Abdun Naser Mahmood, and Jiankun Hu. "A survey of network anomaly detection techniques." *Journal of Network and Computer Applications* 60 (2016): 19-31.

[2] Bhuyan, Monowar H., Dhruba Kumar Bhattacharyya, and Jugal K. Kalita. "Network anomaly detection: methods, systems and tools." *Ieee communications surveys & tutorials* 16.1 (2014): 303-336.

[3] Agarwal, Sugam, Murali Kodialam, and T. V. Lakshman. "Traffic engineering in software defined networks." *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013.

[4] Mehdi, Syed Akbar, Junaid Khalid, and Syed Ali Khayam. "Revisiting traffic anomaly detection using software defined networking." *International Workshop on Recent Advances in Intrusion Detection*. Springer, Berlin, Heidelberg, 2011.

[5] Braga, Rodrigo, Edjard Mota, and Alexandre Passito. "Lightweight DDoS flooding attack detection using NOX/OpenFlow." *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. IEEE, 2010.

[6] Giotis, Kostas, et al. "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments." *Computer Networks* 62 (2014): 122-136.

[7] Phaal, P. "sFlow Specification Version 5." (2004): 1-48.

[8] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." *ACM SIGCOMM Computer Communication Review* 38.2 (2008): 69-74.

[9] Casado, Martin, et al. "Ethane: Taking control of the enterprise." *ACM SIGCOMM Computer Communication Review*. Vol. 37. No. 4. ACM, 2007.

[10] Peng, Tao, Christopher Leckie, and Kotagiri Ramamohanarao. "Survey of network-based defense mechanisms countering the DoS and DDoS problems." *ACM Computing Surveys (CSUR)*39.1 (2007): 3.

[11] Kim, Hyojoon, and Nick Feamster. "Improving network management with software defined networking." *IEEE Communications Magazine* 51.2 (2013): 114-119.

[12] Jang, RhongHo, et al. "RFlow+: An SDN-based WLAN monitoring and management framework." *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017.

[13] Moshref, Masoud, Minlan Yu, and Ramesh Govindan. "Resource/accuracy tradeoffs in software-defined measurement." *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013.

[14] Mirkovic, Jelena, and Peter Reiher. "A taxonomy of DDoS attack and DDoS defense mechanisms." *ACM SIGCOMM Computer Communication Review* 34.2 (2004): 39-53.

[15] Ashfaq, Ayesha Binte, et al. "A comparative evaluation of anomaly detectors under portscan attacks." *RAID*. Vol. 8. 2008.

[16] Mininet. http://mininet.org/

[17] Open vSwitch. http://openvswitch.org/

[18] POX. http://www.noxrepo.org/

[19] “The CAIDA UCSD Anonymized Internet traces 2013.”, ttp://www.caida.org/data/passive/passive_2013_dataset.xml

[20] Turner, Aaron, and M. Bing. "Tcpreplay: Pcap editing and replay tools for* nix.", *http://tcpreplay. sourceforge. net* (2005).

[21] “The CAIDA UCSD DDoS attack 2007 dataset.”, http://www.caida.org/data/passive/ddos-20070804_dataset.xml

[22] “2000 DARPA Intrusion Detection Evaluation Dataset.”, https://www.ll.mit.edu/ideval/data/2000data.html

[23] Scapy. http://www.secdev.org/projects/scapy/

요 약

SDN에서의 정보 엔트로피를 활용한 네트워크 이상 상황 탐지

김나언
컴퓨터공학부
서울대학교 대학원

스마트 기기의 보급 확대 등으로 네트워크 트래픽이 폭증함에 따라, 이를 수용하기 위한 네트워크 인프라의 중요성도 함께 커지고 있다. 이러한 환경 변화에 따라 쉽게 대처할 수 있는 유연하고 효율적인 네트워크 인프라가 필요하게 되었으며, 최근 SDN 기술이 많은 관심을 받고 있다. SDN은 네트워크의 물리적 전달 기능과 논리적 제어 기능을 분리하였으며, 개방형 API를 통해 중앙 집중적으로 네트워크를 제어할 수 있다. 하지만 중앙 집중적인 제어와 프로그래밍이 가능한 특징들은 여러 보안 문제를 일으킬 수 있다. SDN 네트워크에서의 보안 위협을 완화하기 위해 많은 연구자들이 네트워크 트래픽을 관찰하는 연구를 하고 있다. 특히, SDN 네트워크를 관찰할 때에는 OpenFlow 기반의 스위치에서 네트워크 플로우 정보를 수집하며, 이 정보를 이용하여 컨트롤러에서 이상 상황을 감지하는 방법을 사용한다. 하지만 네트워크의 규모가 커짐에 따라 네트워크 플로우 정보를 수집하는 과정은 스위치와 컨트롤러 모두에게 오버헤드를 야기할 수 있다. 본 논문에서는 플로우 기반의 SDN 특징을 이용하여 OpenFlow 스위치에 네트워크 플로우를 수집하는 모듈을 설계하였다. 이는 컨트롤러에서 모든 네트워크 플로우 정보를 모으는데 필요한 오버헤드를 감소시켜주며, 분산적으로 네트워크 플로우 정보를 모을 수

있도록 한다. 또한, 컨트롤러에서 네트워크 이상 상황을 관찰하고, 탐지할 때에 계산 오버헤드와 확장성을 고려하여 정보 엔트로피를 활용한다. 정보 엔트로피는 정보 이론의 중요한 개념 중 하나로 데이터의 무질서 정도를 측정하는 지표이다. 본 논문에서는 실제 트래픽 데이터 셋을 이용하여 제안하는 네트워크 이상 상황 탐지 기법을 검증하였으며, 실험을 통해서 이상 상황 탐지에 있어 낮은 오탐지율을 보여주었다. 그리고 기존의 방법에 비해 CPU 사용량이 확연하게 줄었음을 입증하였다.

주요어: 소프트웨어 정의 네트워킹, SDN, 네트워크 이상 상황 탐지, 네트워크 트래픽 모니터링, 정보 엔트로피, 네트워크 보안

학번: 2016-21186