Flexible Network-Based Intrusion Detection and Prevention System on Software-Defined Networks

Conference Paper · November 2015

DOI: 10.1109/ACOMP.2015.19

4 authors, including:

An Van Le
Ho Chi Minh City University of Technology (H…
4 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Cuong Tran
Ho Chi Minh City University of Technology (H…
1 PUBLICATION **2** CITATIONS
SEE PROFILE

Flexible Network-based Intrusion Detection and Prevention System on Software-defined Networks

An Le
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology
Ho Chi Minh City, Vietnam
Email: 51100010@hcmut.edu.vn

Phuong Dinh
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology
Ho Chi Minh City, Vietnam
Email: 51102644@hcmut.edu.vn

Hoa Le
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology
Ho Chi Minh City, Vietnam
Email: 51101193@hcmut.edu.vn

Ngoc Cuong Tran
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology
Ho Chi Minh City, Vietnam
Email: cuongtran@hcmut.edu.vn

***Abstract*—Software-defined networking (SDN) has recently generated considerable interest among researchers as a next generation network architecture that can overcome the limitations of the traditional network. Intrusion detection and prevention system (IDPS) can leverage the SDN approach to achieve lots of great benefits. The specialized hardware or software for network monitoring can be replaced by OpenFlow switches which might reduce the cost and decrease the latency of the whole IDPS system. SDN also enables the automatic response of the IDPS in case of intrusion detection. Besides, the administrator tasks can be much simpler and efficient thanks to the advantages of SDN. In this paper, we propose a network-based IDPS relying on SDN approach and carry out some typical experiments to evaluate its performance against Denial of Service and Probe attack**

***Keywords*-Intrusion Detection and Prevention System; Software-Defined Networks; OpenFlow.**

I. Introduction

Intrusion detection and prevention system (IDPS) is very important, especially to enterprise networks, it provides the capability of defending the inside network by monitoring network traffic, alerts the administrator when suspicious traffic is detected and filters or redirects these traffic as needed.

Fig. 1 illustrates a simple network-based IDPS system. Typical components of a network-based IDPS include IDPS sensor, IDPS management server and IDPS console [1]. IDPS sensor is responsible for monitoring and analyzing network activity. The management server is a centralized device that receives information from the sensors and manages them. A console is a program that provides an interface for the IDPS 's users and administrators. Besides, the boundary switch must copy every packet to IDPS sensor. This can be done by configuring spanning port in the normal switch. This port can see all network traffic going through the switch.

The advent of Software-Defined Networks (SDN) [2] introduces a new functionality that facilitates the deployment of Intrusion detection system. SDN has gained a lot of attention from researchers in recent years, as it addresses the lack of programmability in existing network architecture and enables easier and faster network innovation. SDN separates the data plane from the control plane and facilitates the implementation of networking applications on top. SDN architecture consists of three layers as shown in Fig. 2. The lowest layer is the infrastructure layer, also called the data plane which comprises the forwarding network elements. Its responsibilities are mainly data forwarding, monitoring local information and gathering statistics. The control layer, called the control plane, is in the middle. It is responsible for programming and managing the forwarding plane. The application layer resides atop the architecture, contains network applications. The application layer receives an abstracted and global view of the forwarding plane from the controller. It then uses that information to provide appropriate guidance to the control layer or run the network services.

In our proposed system, OpenFlow switches are used in replacement of the normal switches, the IPDS sensors and the firewall. The management server receives the information from the OpenFlow switches. In case of intrusion detection, the management server will interact with the controller to deploy new rules over the switches. There are several advantages of this approach. On one hand, total cost of the IPDS system might reduce as the sensors, firewall and the normal switch at the network boundary are no longer needed. On the other hand, the possibility of the management server being down caused by the overwhelming traffic from the sensors is eliminated. This is because when the controller deploys new rules on the OpenFlow switches, these switches

978-1-4673-8234-2/15 $31.00 © 2015 IEEE
DOI 10.1109/ACOMP.2015.19

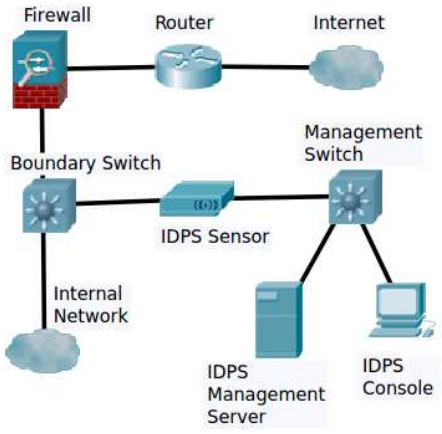

Figure 1. Example of traditional network-based IDPS system

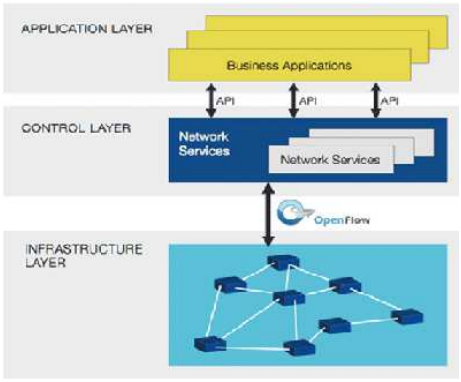

Figure 2. SDN architecture [2]

will filter the traffic themselves, they may even drop the packet according to their rules without continuing to redirect the traffic to the management server. In the traditional IDPS approach, the normal switch and the sensor will continuously send the traffic to the management server even when this traffic is classified as harmful.

Decision Tree and Support Vector Machine model are commonly used for IDPS system with similar efficiency [3, 4]. In this paper, we performed multiple tasks to evaluate the new approach. We use the C4.5 algorithm [9] and the 1999 DARPA dataset to build the decision tree for classifying the traffic in the management server. We then carry out some experiments and evaluation of the system on a small testbed in our network laboratory.

The rest of paper is organized as follows. Section II highlights some related works that we have surveyed. Section III introduces our IDPS system architecture. The implementation of the system is shown in section IV. Section V presents some experiments on the system. Finally, we draw some conclusions and outline some future works in section VI.

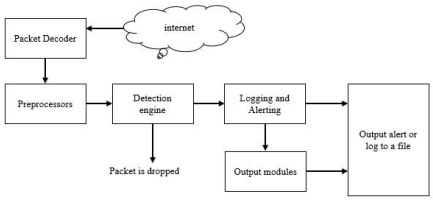

Figure 3. Architecture of Snort [5]

II. RELATED WORKS

Intrusion Detection and Prevention System has been examined and deployed by many researchers and organizations so far. Snort [5] is a lightweight intrusion detection system (IDS) that can log packets coming across the network. Fig. 3 describes how Snort as well as other IDS works:

- Packet Decoder: takes packets from different types of network interfaces (Ethernet, SLIP,PPP), prepare packets for processing.
- Preprocessor: (1) prepare data for detection engine; (2) detect anomalies in packet headers; (3) packet defragmentation; (4) decode HTTP URI; (5) reassemble TCP streams.
- Detection Engine: the most important part, applies rules to packets.
- Logging and Alerting System.
- Output Modules: process alerts and logs and generate final output.

However, there are several issues with this architecture:

- Latency: in-bound IPS requires inspection and blocking action on each network packet, which consumes cloud system resources and increases the detection latency.
- Resource Consumption: running the IDPS services usually consumes significant resources. For instance, configuring SPAN port mirroring technology will duplicate all traffic and forward to a port that an IDS sensor is connected
- Scalability: Rules database is larger and larger (e.g. snort version 2.3.2, there are 2,600 rules). As a result, too hard to update.
- Inflexible Network Reconfigurations: traditional IPS does not have network programmability feature to reconfigure.

According to our survey, IDPS that exploits SDN advantages is still a new topic. Yogita Hande et al. proposed a general architecture of IDS with SDN without deploying or evaluating it [6]. Suresh Kumar et al. introduced the OpenFlow switch with IDS inside of it [7]. Two new actions are added to the OpenFlow switch: IP verification and Packet verification. The switch contains an IDS database and a new table. The IDS database contains the definition of various attacks. New table stores IDS IP information, which contains

the list of suspicious IP. When a new packet arrives at the switch, it will check the source IP address in the IDS IP information table. If the source IP is found in the table, which means that this packet is sent from a suspicious IP address, the switch will take drop packet action. If the source IP is not found in the IDS IP information Table, then it starts intrusion detection by checking the packet data with database. If intrusion is detected, then make an entry in IDS IP information table and drop the packet. However, this is against the SDN philosophy of separating the data plane and the control plane in which the data plane just responsible for forwarding the packet according to the rules deployed from the control plane. Moreover, this may cause latency at the switch by the new extra task which may influence the network performance.

III. SYSTEM ARCHITECTURE

With the ability of centralization in SDN environment, the problem of IDS will be easily overcome. The general architecture is given in Fig 4.

- SDN controller: An OpenFlow controller is an application that manages flow control in a software-defined networking (SDN) environment. SDN controllers functions according to OpenFlow protocol. The SDN controller acts as a sort of operating system (OS) for the network. All communications between applications and devices have to go through the controller. The OpenFlow protocol connects controller software to network devices so that server software can tell switches where to send packets. The controller uses the OpenFlow protocol to configure network devices and choose the best path for application traffic.
- OpenFlow switch: OpenFlow provides an open protocol to program the flow table in different switches and routers. An OpenFlow Switch consists of 3 parts: A Flow Table, A Secure Channel, The OpenFlow Protocol. One of advantages of Openflow switch is the ability of duplication packet, served for inspection process. Moreover, Openflow switch also acts as firewall to prevent many attacks.
- Feature Extractor: Choosing a good set of features is not easy due to the diversity of protocols existing at the network layers and the amount of features extracted from each particular protocol. Current network intrusion detection research community does not provide such a complete feature set that can cover all the network-level intrusions [8], and our paper tends to focus on particular set of intrusions when it comes to detection scope, DOS and Probe. So, based on a TCP connection, some features are extracted from header. Moreover, depending on which mode is running, if training mode is selected, all extracted features are passed to training module, otherwise features are used to check to detect normal or abnormal connection.

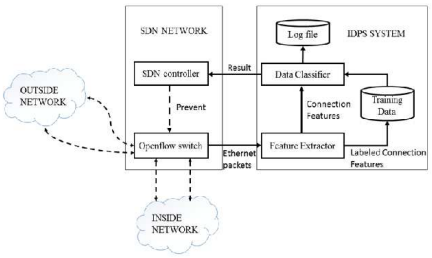

Figure 4. IDS based on SDN architecture

- Data Classifier: A classifier detects whether this is an attack or not using machine learning. There are many models can be implement such as decision tree, support vector machine. The classifier is trained by TCP, UDP flooding, and tested via a real DDoS flooding attack. If some malicious flows are detected by the classifier, the IDS immediately warns the SDN controller about the attack.
- Log Module: This module is able to save information of the attacks and creates data for training process.

IV. IMPLEMENTATION

A. Packet capture

As mentioned above, IDS system in traditional network has unsolved issues. Besides, it depends on hardware technology, in case of too high bandwidth, IDS is prone to suspension. Another problem is that span-ports are used for multi-purposes [1]. For SDN, the issue will be covered since each port in the OpenFlow switch can act as a spanning port. In this paper, we deploy a computer with many network cards as an Openflow vSwitch. Many flexible configurations can be adapted for mirror ports [9][10]. These mirror ports will send the Ethernet packet to the Feature Extractor module.

B. Features Extraction

The three-way handshaking process of a TCP protocol leads to a variety of connection features that can be used for better detection of the network attacks. Moreover, recent TCP attacks show a certain anomaly regarding the number of SYN, RST, and FIN flags of the TCP connection. As a result, connection features have been proposed in recent research [8] and can be divided into two main categories: basic features and derived features [11].

Basic Features encapsulates all the attributes extracted from a TCP/IP connection are shown in Table I. Derived features, also called traffic features, include features that are computed during a period of a time window interval, which measure the similarities and abnormalities between different TCP connections. There are two main types of window intervals, namely time based and connection based intervals.

Table I
BASIC FEATURES OF INDIVIDUAL TCP CONNECTIONS. [11]

Feature name	Description	Type
numSrc	Number of same IP-source connection	continuous
numSrcSamePort	Number of same IP-source and Port connection	continuous
numSrcDiffPort	Number of same IP-source and different port connection	continuous
numSYNSameSrc	Number of same IP-source and same SYN flag connection	continuous
numRSTSameSrc	Number of sameIP-source and same RST flag connection	continuous
numDst	Number of same IP-destination connection	continuous
numDstSamePort	Number of same IP- destination and Port connection	continuous
numDstDiffPort	Number of same IP- destination and different port connection	continuous
numSYNSameDst	Number of same IP- destination and same SYN flag connection	continuous

Features over a time based window interval can be used to detect fast attacks that happen within a short time period (e.g. Worm and DDoS attacks), while the features over a connection based interval (e.g. the latest k connections) have the potential to detect slow scanning attacks (e.g., one scan per minute; one scan per hour). In this paper, we only use features extracting from last 100 connections, detail is show in Table II.

C. C4.5 algorithm[12]

The C4.5 algorithm is introduced by Ross Quinlan. C4.5 is an improvement to his earlier ID3 algorithm. It is one of the best algorithms, both from a theoretical point of view and in practice. C4.5 is used to produce decision tree which is used for classification purpose. C4.5 algorithm have some remuneration over ID3 like, C4.5 uses the concept of information gain ratio as a splitting criteria, can also deal with continuous value attributes along with discrete value attributes , can make use of various techniques of pruning to stay away from over-fitting of decision tree. Normalized information gain ratio can be used as the splitting criteria in C4.5. The algorithm picks the best attributes which splits the dataset into its subsets which contain either one class or the other. The attributes having highest information gain can be used to split the data. The algorithm is repetitively applied to its sub-tree. The leaf node in the decision tree created by C4.5 represents any one class. The splitting process brings to a halt when the number of instances to be split is below a certain threshold.

C4.5 address some advantages:

- Avoiding over-fitting the data.
- Determining how deeply to grow a decision tree.
- Reduced error pruning.
- Rule post-pruning.
- Handling continuous attributes. e.g., temperature.
- Choosing an appropriate attribute selection measure.
- Handling training data with missing attribute values.
- Handling attributes with differing costs.

D. Processing part

Based on the system architecture mentioned in section III, we divide the system into two main phases: preprocessing phase and classification phase.

Table II
DERIVED FEATURES [8]

Feature name	Description	Type
duration	Length of the connection (in seconds)	continuous
numPacket	The number of packets	continuous
numTCPFin	The number of FIN flags	continuous
numTCPSyn	The number of SYN flags	continuous
numTCPReset	The number of RST flags	continuous
numTCPPush	The number of PUSH flags	continuous
numTCPAck	The number of ACK.	continuous
numTCPUrg	The number of URG flags	continuous
numPktSrc	The number of packet from source to destination	continuous
numPktDst	The number of packet from destination to source	continuous
src-bytes	Number of data bytes from source to destination	continuous
dst-bytes	Number of data bytes from destination to source	continuous
Protocol	Specify protocol	continuous
Service	Specify service	continuous

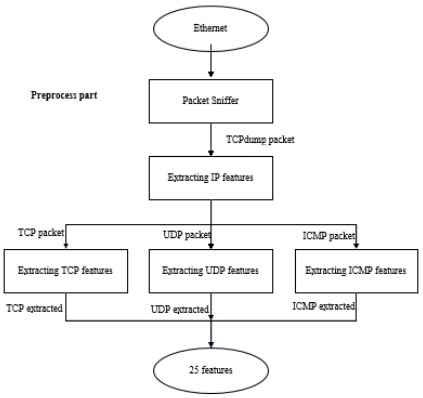

Figure 5. Preprocessing phase

1) Preprocessing phase: In the preprocessing phase, we use the packet sniffer, which is built with J-pcap library, to store network packet information including IP header, TCP header, UDP header, and ICMP header from each promiscuous (crude) packet. After that, the packet information is divided by considering connections between any two IP addresses (source IP and destination IP). Each record consists of 25 data features and an answer class as shown in Table I and Table II. Examples of the data records obtained from the preprocess part are shown below:

- Record1: 43188,TCP,SF,10,http,2,2,0,2,9,0,5,5, 597,2945,17,8,9,8,0,8,8,0,8,0,normal.
- Record2: 0,UDP,SF,1,domain,0,0,0,0,0,0,1,0,85, 0,19,14,5,0,0,38,34,4,0,0,normal.
- Record3: 0,UDP,SF,1,udp,0,0,0,0,0,0,1,0,85,0, 11,0,11,0,0,18,0,18,0,0,normal.
- Record4: 0,ICMP,SF,1,icmp,0,0,0,0,0,0,1,0,113 ,0,20,3,17,0,0,38,4,34,0,0,normal.

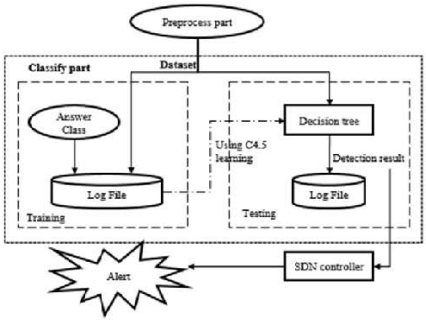

Figure 6. Classification phase

2) Classification phase: The classification phase presented in Fig. 6 consists of 2 main processes: training and testing using java library of WEKA tool [13].

We train the C4.5 Decision tree model with known answer class of each record from the preprocess part. As soon as the detection result is given, SDN controller will produce appropriate actions.

V. EXPERIMENT

A. Testing with Darpa 99 dataset

In our paper, the 1999 Darpa dataset [14] is used to calculate the superiority of the proposed system. The Darpa 99 dataset contains normal data and four types of attack: probing, denial of service (DoS), user to root (U2R), and remote to local (R2L). However, we only focus on probing and Dos. Some of detail attacks are listed in table.

After extracting all labeled packets dump in Darpa 99 dataset into 25 features, 10% of the extracted set are treated as training data, and the others are used as testing data. Experiments were performed using CPU E8500, 2GB of RAM. The following measurements which are often used to evaluate the efficiency of the classifier:

- True positive (TP): The number of sample that is correctly classified into the ith class.
- False positive (FP): The number of samples being wrongly classified into the ith class.
- True negative (TN): The number of outer samples that is correctly classified.
- False negative (FN): The number of ith class samples which is wrongly classified into the other classes.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

As shown in Table IV, our system can detect Dos and Probe attacks at high precision. Although, there are 1031 instances of attack that are misclassified as normal instances, the result illustrates the proposed system can efficiently detect attacks.

Table III
EXPERIMENT ATTACKS

Attack type	Attack name
Probe	Ipsweep, NTinforscan, Nmap
DOS	Mailbomb, SYN flood, Ping of Death, Smurf, Teardrop, Crashiis, Back

Table IV
EXPERIMENT ATTACKS RESULT

	Normal	DOS	Probe	TP	FP	Precision	Recall
Normal	1303651	40	29	1	0.051	1	1
DOS	136	3567	0	0.964	0	0.989	0.964
Probe	157	1	1841	0.921	0	0.984	0.921

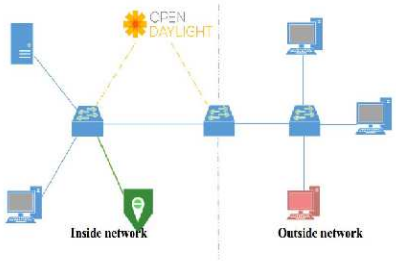

Figure 7. Testing with real system

B. Testing with real system

We also generated attacks from a computer as shown in Fig.7 consisting of 3 types of DoS and 8 types of Probe attack to a server located inside network. Many Internet services in Ethernet were used in order to generate normal network activities.

In Fig. 8, we evaluate the IDPS-SDN system under different network throughput, which also states how well the IDPS can handle the attacks. This evaluation mainly indicates that

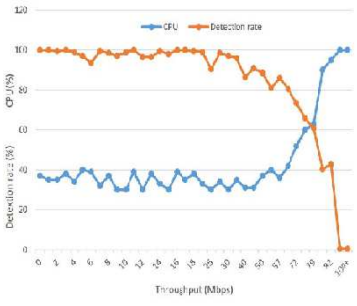

Figure 8. IDPS-SDN performance under different network throughput and CPU consumption

our system can generate alert under high workload of DoS attack. When the network throughput reaches over 80 Mbps, IPS can only generate 60% alert.

VI. Conclusion

We have proposed a network-based Intrusion detection and prevention system relying on SDN. Our approach might reduce the cost of specialized hardware and software required in traditional IDPS while maintaining the same performance. We also exploited some advantages of the SDN architecture to minimize the overload harmful traffic and make the task of the network administrator simpler via the centralized controller. In practice, it is possible to only deploy the OpenFlow switches at the boundary of the internal network without any firewall but still ensure the IDPS operation.

Our experiments are carried out in the scope of a small laboratory with the fixed dataset. Therefore, the proposed IDPS may face some issues in practical run. Besides, the tests are only on DOS and probe attack. We will expand our testbed and do further experiments on real traffic and other kinds of attack to make the system more comprehensive.

References

[1] Skarfone, K., and Peter Mell, "Guide to intrusion detection and prevention systems."

[2] Open Networking Foundation, "Software-defined networking: The new norm for networks," ONF White Paper, 2012.

[3] Peddabachigari, Sandhya, Ajith Abraham, and Johnson Thomas, "Intrusion detection systems using decision trees and support vector machines," International Journal of Applied Science and Computations, USA, 2004, pp. 118-134.

[4] Peddabachigari, Sandhya, "Modeling intrusion detection system using hybrid intelligent systems," Journal of network and computer applications, 2007, pp. 114-132.

[5] Rehman, Rafeeq, Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID, Prentice Hall Professional, 2003.

[6] S. Kumar, T. Kumar, G. Singh, Open Flow switch with intrusion detection system, IJSRET, vol. 1, 2012, pp. 1-4.

[7] Kumar, Suresh, "Open flow switch with intrusion detection system", International J.Schientific Research Engineering & Techonology (IJSRET) 1, 2012, pp. 1-4.

[8] Ghorbani, Ali A., Wei Lu, Mahbod Tavallaee, network intrusion detection and prevention, Springer US, 2010.

[9] git.openvswitch.org:Gitopenvswitch/summary, http://git.openvswitch.org/cgibin/gitweb.cgi?p=openvswitch.

[10] OpenVswitch:Port-mirroring, n40lab, http://n40lab.wordpress.com/2013/02/23/openvswitch-port-mirroring/.

[11] KDD-CUP-99-Task-Description, https://kdd.ics.uci.edu/databases/kddcup99/task.html

[12] Quinlan, J. Ross, C4. 5: programs for machine learning, Elsevier, 2014.

[13] Weka 3 - Data Mining with Open Source Machine Learning Software in Java, http://www.cs.waikato.ac.nz/ml/weka/

[14] MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation, https://www.ll.mit.edu/ideval/data/1999data.html