A COMBINATION OF THE INTRUSION DETECTION SYSTEM AND THE OPEN-SOURCE FIREWALL USING PYTHON LANGUAGE

Tuan Nguyen Kim[1], Tam Nguyen Tri[2], Lam Tran Nguyen[2] and Duy Thai Truong[2]

[1]School of Computer Science, Duy Tan University, Danang, Vietnam
[2]Danang ICT Infrastructure Development Center, Danang, Vietnam

ABSTRACT

There are many security models for computer networks using a combination of Intrusion Detection System and Firewall proposed and deployed in practice. In this paper, we propose and implement a new model of the association between Intrusion Detection System and Firewall operations, which allows Intrusion Detection System to automatically update the firewall filtering rule table whenever it detects a weirdo intrusion. This helps protect the network from attacks from the Internet.

KEYWORDS

Firewall, Rule table, Intrusion detection system, Sniff, Packet capture.

1. INTRODUCTION

The enterprise network security model is responsible for monitoring every flow of traffic and every packet in/out of the network, to detect unauthorized intrusions and packets that may come from legitimate sources but are at risk of carrying malicious code into the network [1]. When an unusual activity is detected in some traffic, Intrustion Detection System (IDS) takes immediate action, such as disseminating information about irregularities or sending irregularity notifications to other systems, including the network's security administrator. It's worth noting that IDS is simply in charge of detecting and issuing warnings; other elements are in charge of dealing with problems. Firewalls commonly do this under the direction of the security administrator's law table. Obviously, this stage seems to be inactive; however, because upgrading the rule table takes time, it should be considered for improvement to increase the efficiency of network protection for organizations and companies [2].

Package Filters and Package Filtering Rules are two of the three key components of package filter firewalls, according to this definition. The cyber security management team will create a package filtering rule table for firewalls based on the network access control policy. This Rule table is used by package filters to manage network access policies. This means that the ability to identify the situation of network access to bring new laws and the speed with which the cyber security administrator updates the rule board for the firewall is important to the timely level of avoiding undesired traffic flows [3]. As a result, if we develop a firewall support system that can detect traffic flows, packets with suspicious actions, and automatically updating the package filtering rule board for the firewall, the firewall will perform better and prevent harmful traffic. This is something that the host-based IDS [8] self-made software can deal with.

Firewalls and IDS can both be hardware devices or software programs. The firewall is selected as the open-source IP Tables technology in our suggested model. IDS is a Python-based Sniffer program that we created. This is considered the article's main contribution.

As a result, IP Tables [6] conduct network access control activities while also being prepared to receive instructions from Sniffer to change the packet filtering rule table. After passing through IP Tables firewall, traffic lines and packets are delivered to Sniffer, who collects and recovers the necessary information, analyzing, statistics, and forecasting to discover the origin of the traffic, of the suspicious packet. Then, automatically, and instantly, the IP Tables rule board is updated. Under this new rule table, the IP Tables firewall will make an access control decision. The system we propose has this as its target.

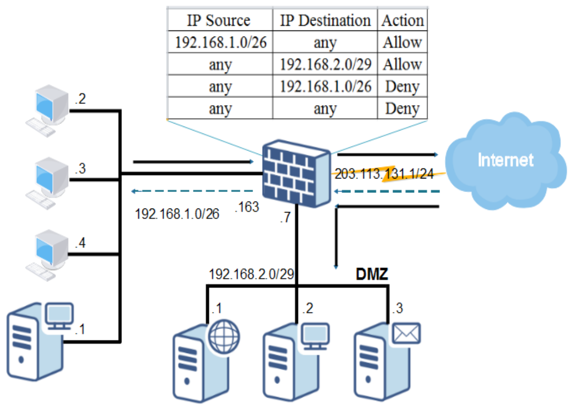
The benefit of open-source firewalls [10], such as IP Tables, is that we can easily change their rule table (packet filtering rule file, to be precise), which is made simple by the Python program. This is one of the reasons why we consider auto-updating packet filtering rules for open-source firewalls.

2. Related Knowledge

In this section, we'll summarize the most basic information regarding two cyber security technologies that are necessary in today's high-security enterprise networks: Firewalls and IDSs. This is required so that the meaning of the combination in their functioning in the network model that we present in this article may be simply understood (in section 3).

2.1. The packet filtering firewall's functioning principle

A firewall is a network access control device that also functions as an in/out network port (commonly referred to as a network gateway). As a result, all traffic flows in and out between external networks, most commonly the Internet, and the enterprise's internal network are subject to the Firewall's supervision and control. To determine whether a traffic/packet flow is allowed to pass through a firewall, it uses a network access policy, which is designed as a Policy Rule Table, and the appropriate information contained in the traffic/packet flow - usually IP Address, Protocols, and Port number.



IP Source	IP Destination	Action
192.168.1.0/26	any	Allow
any	192.168.2.0/29	Allow
any	192.168.1.0/26	Deny
any	any	Deny

Figure 1. The network diagram with the existence of the firewall

Firewalls are divided into two types based on their operation principles. They are the filtering firewall (Packet filtering firewall) functioning at the Network layer of the OSI or TCP/IP network model and the Application layer firewall (also known as Proxy) working at the 'Application' layer of the network model. Nowadays, enterprise networks often use a firewall that combines the operation priciples of these two types of firewalls.

The unit that manages the in/out of the news; package survey filters, also known as Packet Filters; and package filtering rules are the three primary components of the firewall filter package. The Packet Filter uses the package filtering rule table, which was put by the cyber security administrator into the firewall, to decide whether to allow the packet to get through to its destination or not.

Figure 1 represents an enterprise network with a package filter firewall acting as a network gate. Every packet from the Internet to the corporate network, as well as from the internal network to the Internet, must pass through the firewall.

The network access policy of this enterprise network can be seen in the network diagram: Only packets from the Internet are allowed to enter network zone 192.168.2.0/24, the enterprise network's DMZ (DeMilitarized Zone) zone; packets from the Internet are not allowed to enter network zone 192.168.1.0/24, the user network area inside. Any traffic flow from the user network zone and the DMZ network zone, on the other hand, can reach the Internet. The cyber security administrator standardizes this policy into packet filtering laws, which are then constructed into the Package Filtering Rule Table (shown in the diagram) and deployed on the firewall. Before the new network access control policy takes effect, the cyber security administrator must update the Package Filtering Rule Table for the firewall.

The network access policy of this enterprise network can be seen in the network diagram: Only packets from the Internet are allowed to enter network zone 192.168.2.0/24, the enterprise network's DMZ zone; packets from the Internet are not allowed to enter network zone 192.168.1.0/24, the user network area inside. Any traffic flow from the user network zone and the DMZ network zone, on the other hand, can reach the Internet. The cyber security administrator standardizes this policy into packet filtering laws, which are then created into the Package Filtering Rule Table (shown in the diagram) and installed on the firewall.

2.2. Principle of Operation of Intrusion Detection System

According to [4], IDS is a network access monitoring system that can be a hardware device or a software application. It can be a hardware device or a software program. It's responsible for monitoring traffic flows into and out of the network, or to a specific computer on the network, to discover and notify relevant departments about unusual access, unlawful access to the network system, or to specific machines on the network.

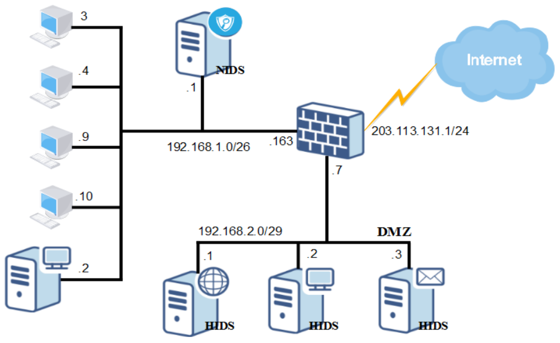


Figure 2. A network diagram with Network-based IDS and Host-based IDS in the background

In fact, IDS can detect and analyzing all user and system actions, as well as reviewing system files, configuration files, and operating systems. It can also check the integrity of system and data files, discover problems in the system's setup, and detect and issue warnings if the system is at risk. IDS, in particular, may analyze trends based on known attacks.

IDS provides a variety of advantages, including ease of deployment in an existing company network without disrupting the current network, and the ability to provide rapid and varied warnings regarding problems. The potential of an IDS to detect unexpected traffic flows and suspicious actions can aid systems and administrators in detecting and preventing multiple network attacks. System administrators can simply modify IDS's "*what needs monitoring*" by modifying its "*signature set*" (with signature-based IDS type). IDS can also log, identify, and report on changes to important files on computers connected to the inland network. However, IDS has several limitations, such as the ability to deliver false positive or false negative warnings to the system regarding irregularities and suspicions, which can negatively affect the system's normal operation. The ability of an IDS to examine encrypted packets is likewise restricted, and therefore does not assist in discovering the source of a network attack. Like other cyber security systems/network services, IDS can be hacked via DoS (Deny of Service) and "misleading" attacks.

The differences between IDS and IPS (Intrusion Prevention System) cybersecurity systems should be clear. They read and analyze network packets before comparing them to known attacks known as "*signature set*" (also known as signature episodes, or threads). While IDS merely detects unusual packets and issues warnings, IPS is an access control system that can accept or reject a packet depending on a set of rules. IDS requires people and/or another system to view the results and take action, whereas IPS just requires regular updates to known threats and the addition of new threats.

When IDS is needed to monitor traffic both going in and out across the network, it can be placed between the firewall and the router (Internet connection); it can be placed in the DMZ zone: when only ids monitor traffic entering the DMZ zone; or it can be placed behind the firewall (right in front of the inner network): when IDS is needed to monitor traffic between the internal and external networks. HIDS is often installed on each server in the DMZ zone to monitor and warn of unusual access or suspicious activity.

Host-based IDS (HIDS), Network-based IDS (NIDS), Protocol-based IDS (PIDS), Application Protocol-based IDS (APIDS), and Hyber IDS are the five most popular types of IDS (HyIDS). In terms of where IDS is located in the network or scope, whether it's on the entire network or just one computer, and how it monitors traffic flow, there are two main types: HIDS and NIDS.

- NIDS: This form of IDS is usually installed in a network position where it may monitor all traffic flow in and out of all network devices. NIDS monitors and analyzes all traffic flows over its entire network region in real time, then compares them to known threats. When a "match" or irregularity is found, NIDS notifies the systems involved, including the network's security administrator, of the unusual actions.
- HIDS: This form of IDS analyzes packets as they open individual hosts in the network. HIDS is typically found near computers and other devices where it receives tracking tasks. When HIDS detects suspicious or harmful behavior from its "user," it will send alerts to the systems involved or the system administrator, letting them know what to do next. HIDS must also track key files on the host it is monitoring, and if it detects that these files have been altered or destroyed, it must transmit alerts to the appropriate systems.

If you rely on IDS intrusion detection, it includes signature-based IDS and Anomaly-based IDS.

- Signature-based IDS: This form of IDS uses a signature set - a database of known threats and attacks - to determine whether traffic flows are unusual or have suspicious behaviors by "matching" the data analyzed from traffic flows to the signature set's data. This sort of IDS serves the same functions as antivirus software and is widely used and effective. However, the efficiency of IDS is dependent on the focus, so it is entirely adaptable to "new abnormalities" and "new forms of attacks" and a large signature set will reduce network bandwidth.
- Anomaly-based IDS: This form of IDS is used to detect traffic flows including "new abnormalities," or "new types of attacks." It monitors traffic flows to determine "normalcy," which is one of the foundations for IDS to detection systems. In truth, this strategy required a high level of "intelligence" on the part of the IDS to successfully complete the task of detecting traffic, particularly unusual packets that the system had not previously detected. Currently, an IDS that has a high level of "intelligence" must be developed using a machine learning approach.

The main difference between these two types of IDS can be noticed in the fact that signature-based IDS uses a known signature set to detect irregularities. Anomaly-based IDS, on the other hand, detects irregularities based on "normal" traffic.

Figure 2. shows a business network diagram that combines both NIDS and HIDS. HIDS is installed on servers in the DMZ zone (Web, File, Mail) in this network to detect any illegal or unexpected access to these servers. NIDS is installed behind the firewall to monitor all traffic entering and exiting the network.

When a network traffic flow passes through it, the signature-based IDS type operates as follows: it copies the traffic and then separates the necessary information in headers and/or payloads from it. The IDS then proceeded to "match" this information against a database of known threats. If a "match" is identified, IDS notifies the appropriate systems. When the IDS detects a new irregularity or a new threat from traffic flowing through it, the database is updated to prepare for future "matching”. The Anomaly-based IDS type observes all traffic flows passing through first, acquiring important data to determine "normalcy." Then rely on "this normality" to find the "irregularities" hiding inside it. This sort of IDS is extremely good at detecting new threats, "new attacks," but it need "intelligence" to do so [7].

3. The Proposed Model

3.1. Network model

Figure 3 depicts our proposed model, which is based on an operational combination of firewall and Host-based IDS:

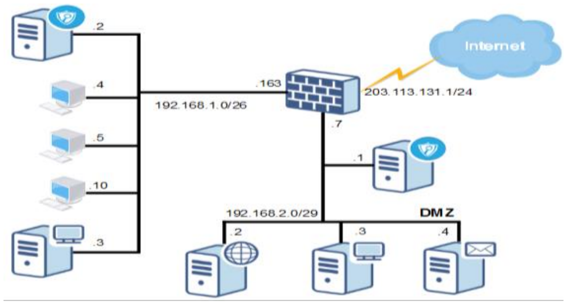


Figure 3. A network diagram with a firewall and Host-based IDS in use

In this model:

- The firewall in responsible for controlling access to the entire network in this model. In effect, the firewall only enables connections from the Internet to the DMZ zone, while preventing connections from the Internet to the inside user network. Any connection from within the network, however, is permitted. This firewall is built with the IPTables open-source firewall software. IPTables permission commands were used to create the package filtering rule table at first.
- The IDS is responsible for observing the flow of traffic entering the DMZ. If the IDS detects an anomaly or suspicious activity from a certain traffic stream or packet, it will report this to the relevant components/objects. In particular, Sniffer can automatically update the packet filtering rule table on IPTables. We use the Scapy library of the Python language to build a Sniffer program that performs this function and task of the IDS.

The proposed model has a new feature in that we have established a connection in the operation and completed the task assigned between the Sniffer program and the open-source firewall IPTables: Once the origin of the traffic or packet line is determined to be abnormal or suspected of acting suspiciously, Sniffer immediately sends a command to IPTables to update the rule table filters its packets. This new package filtering rule table will use firewall IPTables to prevent unwanted traffic flows and packets from accessing the network.

The benefits of the proposed model are obvious: in addition to the usual functions of a specialized IDS, Sniffer is totally proactive in automatically modifying the package filtering rule table on the IPTables firewall. This not only helps to prevent the organization's plan to sabotage the network from the Internet from happening, but it also helps the cybersecurity administrator save time [9].

3.2. Related work

First, we do the necessary work to put the proposed model's network into operation, as designed. We then proceed to establish the IPTables Firewall Package Filtering Rule Table in accordance with the network access policy, in which we direct all traffic lines through IDS, the sniffer program runner, after passing through the IPTables firewall.

Our main contribution is not only to propose a novel model combining IDS and Firewall, but also to provide a Python-based Sniffer software. The major functions of the Sniffer program are as follows:

- Function *process_packet*(): Sniffer's process packet() function is responsible for copying all packets going through the specified ports (specified in the files: *Listports* and *Blacklist*). Then, from the header of this packet, obtain the necessary details, such as the packet type (TCP or UDP), source IP address, source port, and so on. The function test for active will take this data as an argument.
- Function *test_for_active()*: This function compares the contents of packets received from the proceeded packet with the contents of Listports and Blacklist. If a match is identified from a specific traffic or packet, Sniffer counts how many times it appears; if the number of times exceeds the defined threshold, test for active updates the IPTables rules to prevent the traffic from going online. This function is also in charge of detecting and notifying suspicious traffic flows, which are often traffic flows provided in large from a single IP address or many IP addresses to Sniffer.
- Function *setup_logfile():* Set the logfile parameters that Sniffer uses to save information about the traffic, packets that it is specified to pay attention to during observation with the setup

logfile() function.

- Class *limitedPool*: This class contains the functions *init_* and *_setup queues*. In multi-threaded processing cases, the function init_ performs the initial construction of queues for ThreadPools. We've set a maximum queue size of 10,000 people (containing 10000 threads). The setup queues function is used to configure queue parameters.
- Sniffer was able to handle all the packets supplied to it thanks to Python's support [5], even when up to 10,000 packets were sent at the same time.
- *update rule iptables.py* program: This firewall, which is based on the SSH protocol, which is used by Sniffer to deliver commands to IPTables, must update the rule table to block packets containing strings that Sniffer considers are a threat to the internal network system.

Sniffer uses these two functions to transmit commands to change IPTables, ensuring that traffic flows and undesirable packets are prevented from accessing the network.

- And report2email.py program, which is based on the SMTP protocol and is used by Sniffer to send emails to cybersecurity administrators with the content of unusual notifications.

3.3. Testing and evaluation of the proposed model

We set the proposed model to the test in three different scenarios: Scenario 1: String filtering test This scenario is designed to see how well Sniffer and IPTables work together in alerting and stopping packet lines that contain a string that has been flagged for attention and is on the blacklist.

- ***String filtering test (scenario 1):*** This script is intended to test the cooperation between Sniffer and IPTables in a warning and blocking packet streams that contain a specified string that should be noted and indicated. out on the blacklist.

The result is as expected: if many packets come into the Sniffer, and their payload contains the same string recorded in the blacklist, these packets will be blocked by IPTables immediately for a set length of time.

- ***Source IP filtering test (scenario 2):*** This scenario is designed to see how Sniffer reacts when a specific IP address generates many connections. IPTables must monitor traffic coming from this IP address, and Sniffer must identify that this is a suspicious connection.

As a result, if there is a large amount of traffic, go into a sniffer, which has the same source IP address, within a certain period, IPTables will quickly stop this traffic stream.

- ***SYN Flood attack prevention test (scenario 3):*** In this scenario, Sniffer must react when TCP SYN sends it too many packets (possible DoS attack in SYN Flood). In this case, Sniffer will have to issue a request to IPTables, which will block traffic streams containing TCP SYN packets, leaving iptables to do the rest of the work.

Not surprisingly, if a significant number of packets TCP SYN, go into sniffer, within a given length of time, the traffic line with this initial initialing packet is immediately blocked by IPTables.

➢ We also performed an operational coordination performance assessment between Sniffer and IPTables during the testing procedure. To check the filtering and packet blocking results of the proposed model, we use the Tcpdump and Wireshark tools:

- ***For string filtering***, we attack iptables by producing and sending about 1000 packets to IPTables within the set time period, including a large number that exceeds the allowable threshold, and the packets containing strings have been requested sniffer tracking (here is the command "ls –ls"). We dump packets on the firewall and the Sniffer server at the same time. Then, after analyzing two dump findings, it was discovered that: There are over 1000 packets to firewall, but only over 800 packets make it to the server.

Table 1. The packets received on the IPTables firewall

1021	35.844333	10.7.3.100	192.168.10.50	TCP	60 934 $\longrightarrow$ 2222 [SYN]
1022	35.984048	10.7.3.100	192.168.10.50	TCP	60 16453 $\longrightarrow$ 2222 [SYN]
1023	35.127766	10.7.3.100	192.168.10.50	TCP	60 47486 $\longrightarrow$ 2222 [SYN]
1024	35.267755	10.7.3.100	192.168.10.50	TCP	60 19205 $\longrightarrow$ 2222 [SYN]
1025	36.415242	10.7.3.100	192.168.10.50	TCP	60 32231 $\longrightarrow$ 2222 [SYN]
1026	36.570646	10.7.3.100	192.168.10.50	TCP	60 50113 $\longrightarrow$ 2222 [SYN]
1027	36.713131	10.7.3.100	192.168.10.50	TCP	60 57756 $\longrightarrow$ 2222 [SYN]
1028	36.863405	10.7.3.100	192.168.10.50	TCP	60 25465 $\longrightarrow$ 2222 [SYN]

Table 2. The packets received on the Sniffer server

683	26.090433	10.7.3.100	192.168.10.50	TCP	60 47532 $\longrightarrow$ 2222 [SYN]
684	26.235342	10.7.3.100	192.168.10.50	TCP	60 21959 $\longrightarrow$ 2222 [SYN]
685	26.375629	10.7.3.100	192.168.10.50	TCP	60 31919 $\longrightarrow$ 2222 [SYN]
686	26.515015	10.7.3.100	192.168.10.50	TCP	60 59450 $\longrightarrow$ 2222 [SYN]
709	26.660699	10.7.3.100	192.168.10.50	TCP	60 56269 $\longrightarrow$ 2222 [SYN]
769	26.805431	10.7.3.100	192.168.10.50	TCP	60 29910 $\longrightarrow$ 2222 [SYN]
808	26.942159	10.7.3.100	192.168.10.50	TCP	60 55237 $\longrightarrow$ 2222 [SYN]

As a result of Tcpdump, packets containing the strings "ls – ls" were blocked in IPTable during sniffer and IPTables.

- ***To prevent SYN Flood attacks***, we create and send a large number of TCP SYN packets from the outside, via IPTables, to the Sniffer server. The packets were then sent to and from Sniffer. The following are the Tcpdump dump results:

Table 3. Packets TCP_SYN and RST, ACK

617	25.255051	192.168.10.50	147.205.111.88	TCP	54 2222 $\longrightarrow$ 12617 [RST, ACK]
624	25.538365	242.216.67.55	192.168.10.50	TCP	60 15384 $\longrightarrow$ 2222 [SYN]
631	25.670945	76.238.179.36	192.168.10.50	TCP	60 47907 $\longrightarrow$ 2222 [SYN]
635	25.804151	158.83.198.86	192.168.10.50	TCP	60 52472 $\longrightarrow$ 2222 [SYN]
639	25.950671	147.122.241.47	192.168.10.50	TCP	60 12835 $\longrightarrow$ 2222 [SYN]
646	26.082268	73.24.49.61	192.168.10.50	TCP	60 26716 $\longrightarrow$ 2222 [SYN]
650	26.219748	180.213.100.213	192.168.10.50	TCP	60 55000 $\longrightarrow$ 2222 [SYN]
651	26.220044	192.168.10.50	180.213.100.213	TCP	54 2222 $\longrightarrow$ 55000 [RST, ACK]
655	26.360506	82.253.163.252	192.168.10.50	TCP	60 56391 $\longrightarrow$ 2222 [SYN]
659	26.511481	220.167.184.100	192.168.10.50	TCP	60 16225 $\longrightarrow$ 2222 [SYN]
663	26.638535	184.161.231.8	192.168.10.50	TCP	60 29016 $\longrightarrow$ 2222 [SYN]
670	26.776625	254.187.37.69	192.168.10.50	TCP	60 18580 $\longrightarrow$ 2222 [SYN]
679	26.935711	247.192.20.31	192.168.10.50	TCP	60 4524 $\longrightarrow$ 2222 [SYN]
684	27.088005	72.62.78.17	192.168.10.50	TCP	60 3017 $\longrightarrow$ 2222 [SYN]
688	27.226594	145.177.52.153	192.168.10.50	TCP	60 785 $\longrightarrow$ 2222 [SYN]
689	27.226694	192.168.10.50	145.177.52.153	TCP	54 2222 $\longrightarrow$ 785 [RST, ACK]
693	27.372435	211.171.132.100	192.168.10.50	TCP	60 26950 $\longrightarrow$ 2222 [SYN]

During the TCP "3-step shake-up", usually, every time the destination party receives a packet [TCP_SYN] it must immediately send a response packet [TCP_RST, ACK]. But here, due to suspicion of a SYN_Flood DoS attack, Sniffer asked IPTables to restrict response to connection initiation packets [TCP_SYN]. This has been proven by Tcpdump.

- ***For source IP filtering***: This test is conducted similarly to "string filtering", but here requires Sniffer to detect and prevent abnormalities coming from traffic flows coming from the same source IP address. Dump results and analysis of packets received at IPTables and Sniffer are shown in the following two shapes:

Table 4. IPTables firewall packets received

1932	271.587888	10.7.3.100	192.168.8.164	TCP	60 10541 ⟶ 2222 [SYN]
1935	272.634401	10.7.3.100	192.168.8.164	TCP	60 1379 ⟶ 2222 [SYN]
1938	273.681978	10.7.3.100	192.168.8.164	TCP	60 29845 ⟶ 2222 [SYN]
1943	274.769926	10.7.3.100	192.168.8.164	TCP	60 54837 ⟶ 2222 [SYN]
1946	275.809951	10.7.3.100	192.168.8.164	TCP	60 41488 ⟶ 2222 [SYN]
1949	276.854445	10.7.3.100	192.168.8.164	TCP	60 23832 ⟶ 2222 [SYN]
1953	277.906213	10.7.3.100	192.168.8.164	TCP	60 27645 ⟶ 2222 [SYN]
1959	278.954523	10.7.3.100	192.168.8.164	TCP	60 41319 ⟶ 2222 [SYN]

Table 5. The packets received on the Sniffer server

666	177.06191	10.7.3.100	192.168.10.50	TCP	60 56740 ⟶ 2222 [SYN]
673	178.11423	10.7.3.100	192.168.10.50	TCP	60 2805 ⟶ 2222 [SYN]
727	179.20589	10.7.3.100	192.168.10.50	TCP	60 28246 ⟶ 2222 [SYN]
776	180.24032	10.7.3.100	192.168.10.50	TCP	60 2851 ⟶ 2222 [SYN]
826	181.27678	10.7.3.100	192.168.10.50	TCP	60 4149 ⟶ 2222 [SYN]
876	182.32243	10.7.3.100	192.168.10.50	TCP	60 35744 ⟶ 2222 [SYN]
878	183.39045	10.7.3.100	192.168.10.50	TCP	60 45856 ⟶ 2222 [SYN]
880	184.42113	10.7.3.100	192.168.10.50	TCP	60 32774 ⟶ 2222 [SYN]
882	185.47298	10.7.3.100	192.168.10.50	TCP	60 37169 ⟶ 2222 [SYN]

As a result of Sniffer and IPTables' views and actions, packets from IP: 10.7.7.100 (in large quantities) were blocked in IPTables, according to Tcpdump.

Because most open-source firewall systems allow command-line interaction, our usage of IPTables in our test and evaluation scenarios does not compromise the general validity of the proposed model.

From the results of the assessment, we are confident that the proposed system can completely play a role in detecting and illegally entering the enterprise in-house network such as IDS systems and other individual firewalls in the information security product market.

4. CONCLUSION

In this paper, we propose a new model for merging the operations of IDS and open-source firewall IPTables, which we have successfully implemented. We've also created a Sniffer software that acts as the network system's IDS. This rule table change is completely automated; therefore, it happens extremely quickly.

The accuracy of making a statement about the suspicious actions of an IDS depends on the collection and analysis of information, from the flow of traffic into/out of the network and depends on its statistical and forecasting capabilities. It can be said that the "intelligence" of this department determines so greatly the accuracy in determining which traffic flows are suspicious actions among the many traffic flows passing through IDS. Also, with the increasing ability to "fake" and "deceive" hackers in cyberspace, without high "intelligence", IDS is difficult to determine exactly whether a packet is carrying malicious code or not. Our Sniffer's limitation is that it has a low "intellect." Sniffer's "intelligence" will be improved in the future by moving it toward a Machine Learning approach.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] Kanika & Urmila, (2013) "Security of Network Using IDS and Firewall", *International Journal of Scientific and Research Publications*, Vol. 3, Iss. 6, pp.1-4.

[2] Waleed Bul'ajoul, Anne James & Mandeep Pannu, (2015) "Improving network intrusion detection system performance through quality of service configuration and parallel technology", *Journal of Computer and System Sciences*, Vol. 81, Iss. 6, pp. 981-999.

[3] Sulaman & Sadar Muhammad, (2012) "An Analysis and Comparison of The Security Features of Firewalls and IDSs", *Linköpings universitet, Institutionen för systemteknik*, pp. 87-2011.

[4] D. Ashok Kumar & S.R Venugopala, (2017) "Intrusion Detection Systems: A review", *International Journal of Advanced Research in Computer Science*, Vol. 8, No. 8, pp.356-370.

[5] Mrinal Wahal, Tanupriya Choudhury & Manik Arora, (2018) "Intrusion Detection System in Python", *8th International Conference on Cloud Computing, Data Science & Engineering*, pp. 348-353.

[6] Michael & Rash, (2007) "Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort", No *Starch Press*, pp. 81-173.

[7] SH Kok, Azween Abdullah, NZ Jhanjhi, Mahadevan Supramaniam, (2019) "A Review of Intrusion Detection System using Machine Learning Approach", *International Journal of Engineering Research and Technology*, Vol. 12, No. 1, pp. 8-15.

[8] Thomas M. Chen, Patrick J. Walsh, (2014) "Guarding Against Network Intrusions", *In: Network and System Security (Second Edition)*, Chapter. 3, pp. 57-82.

[9] Takeda K., Takefuji Y., (2001) "Pakemon – A Rule Based Network Intrusion Detection System", *International Journal of Knowledge Based Intelligent Engineering Systems*, Vol. 5, No. 4, pp. 240-246.

[10] Dmitrij Melkov & Šarūnas Paulikas, (2021) "Analysis of Linux Os Security Tools For Packet Filtering and Processing", *Vilnius Gediminas Technical University*, pp. 1-5

AUTHORS

Tuan Nguyen Kim was born in 1969, received B.E, and M.E from Hue University of Sciences in 1994, and from Hanoi University of Technology in 1998. He has been a lecturer at Hue University since 1996. From 2011 to present (2021) he is a lecturer at School of Computer Science, Duy Tan University, Da Nang, Vietnam. His main research interests include Computer Network Technology and Information Security.

Duy Thai Truong was born in 1997, He graduated from Duy Tan University in 2019. He is a Security Engineer at Danang Department of Information and Communications, Vietnam. He is working at here from November 2019 to present. His main is research on information security and penetration testing.

Tam Nguyen Tri was born in 1998, He graduated from Duy Tan University in 2020. From November 2020 to present (2021), he is a Security Researcher at Danang ICT Infrastructure Development Center.

Lam Tran Nguyen was born in 1998, He graduated from Duy Tan University in 2020. From November 2020 to present (2021), he is a Security Researcher at Danang ICT Infrastructure Development Center.