

The Flow-Reduced Malware Detection System by Controlling Inactive/Active Timeout

Jae Kyu Lee*, Hyeon Yang*, Kyeong Ho Park**, Si Young Lee**, Seong Gon Choi*

*Broadband Network Lab., Chungbuk National University, Korea

**Xabyss Inc., Korea

princeljk@cbnu.ac.kr, adieu@cbnu.ac.kr, khpark@xabyss.com,
edward.lee70@xabyss.com, choisg@cbnu.ac.kr

Abstract— This paper shows the flow reduction effect of malware detection system through inactive/active timeout control. In the gigabyte network environment, the DB search speed of the malware detection system that processes high-speed, large-capacity data is affected by the number of stored flows. Therefore, it is necessary to reduce the number of stored flows. The malware detection system captures network data with the First-N technique and adjusts the session length by modifying the inactive/active timeout. A drill-down mechanism is used for malware inspection of network data stored in DB. Inactive/active timeout is adjusted to verify that the number of flows decreases as the session length increases.

Keywords—First-N, Inactive/Active Timeout, Drill-Down Search, Flow Count, Storage Efficiency

I. INTRODUCTION

Today, due to the spread of penetration rate of information terminals and the Internet, data volume is continuously increasing. In order to accommodate increasing production volume, the network environment also developed rapidly. A network environment of gigabytes or more is currently being offered. In such network environment, since a number of packets coming into the NIC is very large, storage management is difficult with the traditional method [1]-[3].

In a network environment where high-speed traffic arrives at high-speed, the malware security system should be processed in real time. However, with the traditional method, it is difficult to process high-speed high-capacity traffic in real-time. The signature-based malware detection system searches all the signatures of all data, so operation is performed by the amount of stored data. This means that the amount of data saved will affect the searching speed [4]-[7].

To process the data coming from the network interface card in the network environment of gigabytes or more, it is difficult to process the amount of data in the traditional processing method, so it is stored in the DB through the First-N packet processing method [8].

In security systems where packets are processed in real time in a gigabyte network environment, it is necessary for the data acquisition from the database to operate at high speed. Captured packets are stored in the session on a per-flow basis. If we reduce the number of stored flows, the flow search

speed is improved. We show that adjusting inactive/active timeout reduces "number of saved flows". This verification can improve storage efficiency and speed up searching.

For implementation, we first process and store the incoming packets through NPU with First-N. Received packets create a session on a flow-by-flow basis. At the end of the session, it exits on the basis of inactive/active timeout. According to the inactive/active timeout value, the session is terminated. Change the size of inactive/active timeout and compare the results. When the timeout values are changed, the total number of flows and the number of flows stored per second change. Setting a longer timeout increases the total number of flows and reduces the number of flows stored per second. Conversely, setting a shorter timeout reduces the total number of flows and increases the number of flows stored per second. Therefore, in this paper, we increase inactive/active timeout settings to verify that the number of stored flows per second decreases.

In Chapter 2, we introduce related works on the number of stored flows and the searching speed. Chapter 3 explains the First-N method, which is the basic theory of this paper, and inactive/active timeout. Chapter 4 explains the implementation environment, implementation method, and analysis of implementation results. Chapter 5 presents conclusions.

II. RELATED WORK

The number of packets coming into the NIC in a gigabyte class network environment is very large. When all packets are accepted, Storage is burdensome. From the viewpoint of cost, it is also required to efficiently manage storage more than increasing storage capacity. Also, for flow capture, work involving the high storage expenses involves the risk of capture loss. This phenomenon will interfere with the preservation of important packets and will also have a fatal impact on malware detection [1]-[3].

In signature-based malware systems, the number of signatures and the complexity of the signatures affect the detection speed. Storing all incoming packets from the network, the number of Storage signatures is more than the efficiently managed Storage. Obviously, the number of signatures is large and the speed of detecting malware from Storage takes time. In order to improve the retrieval speed,

techniques such as proposing a structure through signature classification [4], deleting duplicate packets, and saving only N bytes of packets are studied [5]–[7].

Generate flow based sessions for efficient storage with First-N method. However, there do not define the session inactive/active timeout. If Inactive / Active Timeout is not defined, inefficient storage in flow preservation can be performed [8].

Because of these problems, several studies related to packet capture and storage efficiency are under its way. We propose a way to reduce the number of flows saved with a fundamental solution to all these problems. In order to reduce the number of saved flow, we present the concept of Inactive / Active Timeout adjustment used in First-N method session creation and termination. In this method, if the received packet by NIC is processed with First-N, we adjust Inactive / Active Timeout which is the session termination criteria short and long. The number of flows saved through Timeout adjustment decreases and enables efficient storage. Also, reducing the number of saved flows improves the detection speed of malware systems using the Drill-Down search method.

III. FLOW-REDUCED MALWARE DETECTION SYSTEM

In this chapter, we will look at the session method generated based on First-N flow and show how to apply the Inactive / Active concept. Then explain the procedure of 'Drill-Down Search' and explained how it relates to the flow number.

A. Generate a First-N Flow-Based Session

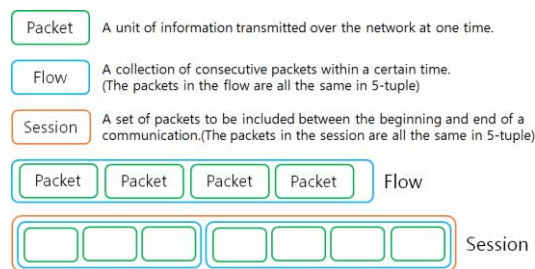


Figure 1. Session Creation Process

First-N is used to minimize/optimize the real-time detection and storage of traffic in gigabyte-class network environments. It recognizes and extracts in real time only packets corresponding to initial N packets coming into the network or initial N size data. Except for the special case like KeepAlive, it is possible to detect less than the first 10 packets of the session. So only N packets are stored.

Figure 1 helps to understand of packets, flows, and sessions. A packet is a unit of information transmitted at a time on a network, and a flow is a group of packets consecutive within a certain period of time. First-N creates a session based on the flow. When you create a session based on a packet, the unit will store more amount than the flow larger than the packet, it takes time to be proportional to the amount of storage. Also, the detection speed for generating extra index information is lowered. In order to secure these drawbacks, you can create a flow-based session and have a fast scan speed.

B. Metadata Search Method

In the case of large capacity packets in an incoming network environment, use a technology that can search at high speed. Enable high-speed search using Drill-Down Search technology. In order to extract the stored network packet information, the Drill-Down search technology is required in order of session, flow, and packet based on 5-tuple (Source IP / Port, Destination IP / Port, Protocol) information a packet is detected. It is possible to retrieve necessary packets at high speed without saving index information for high-speed retrieval of packet information by separately storing the generated flow and session information. Figure 2 shows the Drill-Down Search process. First, search for sessions according to specific search criteria. Search session information file using session ID. At this time, the file of session information has the largest unit compared with flow and packet, so the detected amount is the smallest. Find all the IDs of the flows included in a specific session and extract the information of the flow list. Find and extract all packet IDs in the next extracted specific flow file and read the data. We can see that in this process the Drill-Down Search method searches for the flow file. You can see that decreasing the number of files searched in the flow file helps to improve search speed.

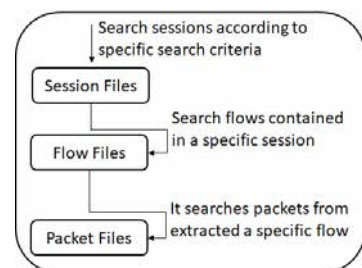


Figure 2. Drill-Down Search Process

C. Inactive/Active Timeout

Inactive / Active Timeout is a function used when you terminate a session with the First-N method. We are trying to reduce the number of flows saved in this way. Inactive / Active Timeout is used as a criterion for the end point of session creation and obtains the change in size to the current number of received flows and the number of saved flows. The Inactive time means that if packets with the same 5-tuple are not transmitted or received for a certain period of time, packets up to that time will be stored collectively in one flow. The active timeout means that when packets with the same 5-tuple are continuous for a certain period of time, multiple packets are collectively stored in one flow.

IV. IMPLEMENTATION RESULT

A. Implementation Environments

We have created an experimental environment to verify the Inactive / Active timeout efficiency. First, have to make that packets arrive in the NIC of a gigabyte network environment. For the packet generation, packets are generated on the network by executing a computer, downloading a file through

the Internet, playing back a movie, etc. Packets coming into the NIC are processed by the First-N method and stored in Storage. At this time, we use the Inactive / Active Timeout adjustment method to give two experimental values and measure the result value.

Setting Inactive / Active timeout adjustment User Interface Software used C++ for development of Support Service based on LAMP platform. We used CodeIgniter for UI Framework for web service development and used PHP which is a server-side scripting language. Finally, we used an interface with the standard SQL language to access MariaDB where the metadata is stored.

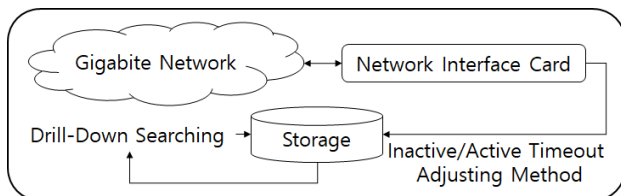


Figure 3. Architecture of Implementation Environment

An implementation method for the verification is to experiment with the Inactive / Active Timeout set value short, long and divided into two cases. Compare the values of the results that occurred in the two different cases and check the change in the number of saved flows.

The setting values for the verification are as follows.

- Short-Timeout Length (Short-Time Timeout) – Set Inactive/Active Timeout to 10 seconds / 50 seconds.
- Long-Timeout Length (Long-Time Timeout) – Set Inactive/Active Timeout to 40 seconds / 160 seconds.

Experiment Measurement time will proceed for more than 5 minutes to collect enough data so that results can be accurately derived from the following graph. The short-time Timeout is used as a comparison and analysis value for only 5 minutes from 19: 47 to 19: 52, and the long-time timeout is used as a comparison and analysis value of only 5 minutes from 19: 54 to 19: 59.

B. Implementation Results

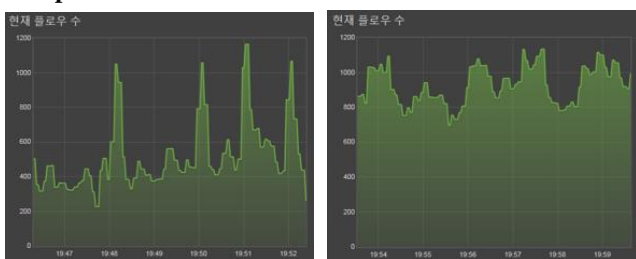


Figure 4. Current flow number measurement graph (left timeout for short-time timeout, right for long-time timeout)

Figure 4 is a graph obtained by measuring the current flow number of the short-time Timeout and the long-time Timeout. Analyze the resulting value of the other Timeout setting. In the case of the short-time Timeout, the current flow number is measured on the average of about 400. In the case of long-

time Timeout, the current number of flows was measured on average about 900.

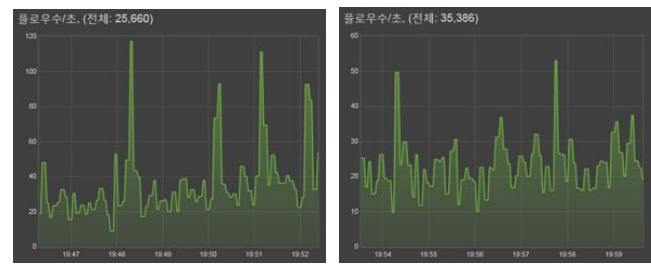


Figure 5. Number of flow saved per second measurement graph (left timeout for short-time timeout, right for long-time timeout)

Figure 5 is a graph that measures the number of saved flows per second for short-time Timeout and long-time Timeout. Analyze the result value of the Timeout setting value in two different cases. In the case of short-time Timeout, the number of saved flows was measured on average about 30. In the case of a long-time Timeout, the number of saved flows was measured on average about 20. For maximum saved flow values, the short time Timeout was measured to be close to 120 and the long-time Timeout was measured for the number of saved flows less than 55.

When it is set to the length of Timeout for a short time, the end of the session is finished faster than when the session is set to Timeout for a long time. Therefore, the current number of flows displayed since the session was created is measured relatively small. When sessions are finished, they are saved, so when you measure in the same time zone for 5 minutes, the number of flows saved every second is increased with a short Timeout setting where the criteria of session termination are short and termination occurs frequently. In the case of a long timeout, the end time of the session is 110 seconds longer than that of the short timeout, and the active timeout is 30 seconds longer. The end of the session ends late by this difference. For sessions that are maintained for a relatively long time, of course, the number of flows displayed on the network and displayed is numerous. However, as the session is held longer, the number of session terminations decreases. A decrease in the number of termination is equivalent to a decrease in the number of data storage times. Therefore, looking at the number of flow saved every second, it is 20 on average. Shows the number of stored flow numbers that is less than 30 which is the average value of short Timeout. Even when comparing the flow sizes saved every second, there are times when the short timeout exceeds the flow size of 60. The long timeout is measured with a flow size of 60 or more.

V. CONCLUSIONS

High-speed, and large-capacity traffic requires that the amount of packets be stored very efficiently, and the searching speed of stored packets should be increased. We implemented with the method of adjusting inactive/active timeout in First-N mode. Inactive/active timeout were experimented by modifying two different setting values. As a result, the results of setting timeout to a longer value saved the number of flows

per second while maintaining more sessions. Because the number of stored flows is reduced, the search speed is increased by reducing the amount of search that is performed by the Drill-down search, which performs the process of searching the flow file.

We have verified that increasing the length of inactive / active timeout reduces the number of stored flows. Applying this result to the drill-down search mechanism will reduce the number of stored flows and improve the search speed.

Research is needed to optimize the search speed by appropriately adjusting the inactive / active timeout according to the specification of the network equipment using the drill-down search mechanism.

ACKNOWLEDGMENT

This work was supported by “Human Resources Program in Energy Technology” of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea. (No. 20164030201330)

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(M SIP) (No. NRF-2015R1A2A2A03004152).

*corresponding author is S.G. Choi (choisg@cbnu.ac.kr)

REFERENCES

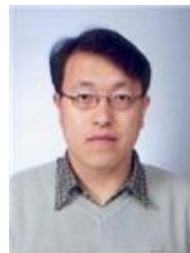
- [1] Jihyung Lee, Sungryoul Lee, Junghee Lee, Yung Yi, KyoungSoo Park, “FloSIS: A Highly Scalable Network Flow Capture System for Fast Retrieval and Storage Efficiency,” 2015 USENIX Annual Technical Conference(USENIX ATC’15), July 8-10, 2015
- [2] Ying-Dar Lin, Po-Ching Lin, Tsung-Huan Cheng, I-Wei Chen, and Yuan-Cheng Lai, “Low-Storage Capture and Loss Recovery Selective Replay of Real Flows,” IEEE Communications Magazine, Vol. 50, No. 4, April 2012.
- [3] Seon-Ho Shin, Jootoung Lee, Ji-Man Jeong, HyunBong Kim, Jong-Hyun Kim, Ikkyun Kim, and Myung Keun You, “PrIDE: A Protocol—Independent De-Duplication Engine for Packet Recording,” IEEE Network, Vol. 30, No. 6, November / December 2016.
- [4] Jun-Sang Park, Sung-Ho Yoon, and Myung-Sup Kim, “Performance Improvement of Signature-based Traffic Classification System by Optimizing the Search Space,” Korea Society for Internet Information, 12 (2011), No. 3, 89-99.
- [5] Chia-Hao Hsu, and Sheng-De Wang “ Embedded Network Intrusion Detection System with a Multi-Core Aware Packet Capture Module”
- [6] Woo-Suk Jung, Jun-Sang Park, and Myung-Sup Kim, “Performance Improvement of Traffic Identification by Categorizing Signature Matching Type,” The Journal of Korean Institute of Communication and Information Sciences, 15-07, Vol. 40, No. 07, 2015.
- [7] Yan Zhang, and Nirwan Ansari, “On Protocol-Independent Data Redundancy Elimination,” IEEE Communications Surveys & Tutorials, Vol. 16, No. 1, First Quarter 2014.
- [8] Seung Min Ryu, Si Young Lee, Chang Woo Ryu, and Seong Gon Choi, “A Method of the First-N Packets Storage by Using the Advanced-PCA,” Contemporary Engineering Sciences, Vol. 9, 2016, no. 14, 647-653



Jae Kyu Lee received B.S degree in the College of Electrical & Computer Engineering, Chungbuk National University, Korea in 2017. He is currently an M.S. candidate in School of Electrical & Computer Engineering, Chungbuk National University. His research interest is network security.



Hyeon Yang received B.S and M.S. degree in the College of Electrical and Computer Engineering, Chungbuk National University, Korea in 2011 and 2012 respectively. He is currently a Ph D. candidate in School of Electrical & Computer Engineering, Chungbuk National University. His research interests include smart grid and network security.



Kyeong Ho Park received B.S from Kyeongbuk National University in 2000 and M.S degree from Korea Polytechnic University in 2013. He had been researching and development in Samsung electronic's research center. Since 2015, He has been developing the network security device in Xabyss Inc.



Si Young Lee received B.S, M.S., and a Ph.D. degree from Kyeongbuk National University in 1995, 1997, and 2000 respectively. He has been researching director and CTO for a few venture company since 2000, and CEO in Xabyss Inc. since 2014. His research interest is a software-defined network device (SDND).



Seong-Gon Choi received B.S. degree in Electronics Engineering from Kyeongbuk National University in 1990, and M.S. and a Ph.D. degree from KAIST, Korea in 1999 and 2004, respectively. He is currently an assistant professor in the College of Electrical & Computer Engineering, Chungbuk National University. His research interests include smart grid, IoT, mobile communication, high-speed network architecture, and protocol.