International Journal of Grid and Distributed Computing
Vol. 11, No. 3 (2018), pp.109-124
http://dx.doi.org/10.14257/ijgdc.2018.11.3.10

Network Payload and Correlation Analysis in Bigdata Environments

Jeong-Joon Kim[1], Yong-Soo Lee[2], Jin-Yong Moon[3] and Jeong-Min Park[4*]

[1]*Korea Polytechinc University, Gyeonggi-do, Korea*
[2]*Yeoju Institute of Technology, Gyeonggi-do, Korea*
[3]*Gangdong College, Gyeonggi-do, Korea*
[4]*Korea Polytechinc University, Gyeonggi-do, Korea*
[1]*jjkim@kpu.ac.kr,* [2]*diclee@yit.ac.kr,* [3]*jmoon37@gmail.com,* [4]*jmpark@kpu.ac.kr*

Abstract

Existing analyzers such as Wireshark are difficult to analyze and classify various protocols based on payload data, and it is difficult to handle large-scale data identification and extraction for event logging.

Traffic analysis methods are classified into port based analysis, payload based analysis, statistical information based analysis, and correlation based analysis based on traffic characteristics used in analysis, and classified into packet based analysis and flow based analysis based on analysis unit. However, due to the recent development of hacking technology, traffic with encrypted payload is increasing, making it difficult to utilize existing traffic analysis methods[5,6]. To solve this problem, a traffic analysis method applying various analysis techniques based on Big Data is being studied. Therefore, in this paper, we have carried out a study to extract various features from network traffic using big data base analysis technology

Keywords*: Bigdata, Networks Traffic, Feature Extraction, Analyzer, Payload*

1. Introduction

Protocol analyzers provides diagnoses of various problems on the network, implementation of specialized analysis, monitoring of network traffic, statistical data, and reporting functions. Typical protocol analyzers include pay protocol analyzers such as Sniffer, internet advisor, Domino, and free protocol analyzers such as Wireshark. Wireshark is a tool that supports decoding and analyzing network protocols of large scale. One advantage of Wireshark is that it captures packets over a period of time, such as tcpdump, and then interactively analyzes and filters the content using various protocols, ports, and other data. In addition, Wireshark supports a wide variety of protocol decoders to inspect the contents of packets and conversations in detail[1,2].

However, existing analyzers such as Wireshark are difficult to analyze and classify various protocols based on payload data, and it is difficult to handle large-scale data identification and extraction for event logging. Traffic analysis methods are classified into port based analysis, payload based analysis, statistical information based analysis, and correlation based analysis based on traffic characteristics used in analysis, and classified into packet based analysis and flow based analysis based on analysis unit. However, due to the recent development of hacking technology, traffic with encrypted payload is increasing, making it difficult to utilize existing traffic analysis methods[3,4]. To solve this problem, a traffic analysis method applying various analysis techniques based on Big

Received (December 27, 2017), Review Result (February 25, 2018), Accepted (March 5, 2018)
* Corresponding Author

ISSN: 2005-4262 IJGDC

Data is being studied. Therefore, in this paper, we have carried out a study to extract various features from network traffic using big data base analysis technology.

2. Related Work

2.1. Payload Data Classification Method

In the payload of the packet in order to solve this problem for each application having a payload-based analysis method for analyzing the presence or absence of a particular string, including traffic through the (Signature) it has been proposed. Check the contents of analytical performance due to traffic (analytical rate and accuracy) is very high, but has many limitations such as signature generation and management, encryption, traffic, high computational complexity, packet fragmentation, invasion of privacy[5,6].

Traffic encryption is based on a statistical analysis method using only the statistical characteristics such as packet window size, time interval between packets have been proposed without seeing the traffic information to solve the privacy problem. This methodology generate statistical information from header information of the packet may compensate for the limitations of the existing traffic classification methodology. However, when using an application level protocol such as the engine or based applications have hard limits this detailed application specific analysis because it has the same statistical characteristics.

Recently, to overcome the limitation of the conventional traffic analysis method, a method of changing the packet unit traffic to the flow unit and analyzing the correlation between them has been proposed. Flow refers to a set of packets of the same 5-tuple (SrcIP, SrcPort, DstIP, DstPort, Transport Layer Protocol). Traffic is analyzed using statistical information such as flow size and period, and the connection type between flows. Since various features can be used than packet based analysis methods, various analyzes are possible. However, it can’t be analyzed until the flow generation is completed, and there is a disadvantage that the overhead of calculating the statistical information of the flow occurs. Furthermore, it is difficult to distinguish between applications having similar statistical information[7-9].

The traffic analysis methodology based on analysis unit can be divided into packet unit and flow unit. Packet-based traffic analysis analyzes traffic by extracting traffic information (header information or bit streaming) that can distinguish a specific application in the packet by signature.

It is advantageous in real-time analysis because it analyzes based on packets generated in real-time. However, since the extraction target range is limited to a single packet, it is very difficult to extract a signature representing a specific application. In addition, when the traffic analysis and the overhead that must analyze the header information and payload information in every packet. Flow-based traffic analysis can use various traffic characteristics because it analyzes traffic using statistical information, connection type, and generation time of a plurality of packets constituting a flow. Therefore, the signature generation process is relatively easy, but the accuracy of traffic analysis by application is very low as the traffic of applications with similar statistical characteristics (using the same engine protocol) increases, and it is difficult to control in real time because the analysis is performed when the flow is completed. The analysis methods proposed to date have the more power, but still many limitations to overcome existing limitations.

2.2. Existing Big Data Based Traffic Analysis Study

A. Real-time network traffic analysis platform design based on Big Data

We compared the real - time traffic analysis performance of Big Data solutions(MongoDB, HBase/Hadoop, Lucene/Solr). The structure of the proposed system (RENTAP) for comparison is shown in the following figure.

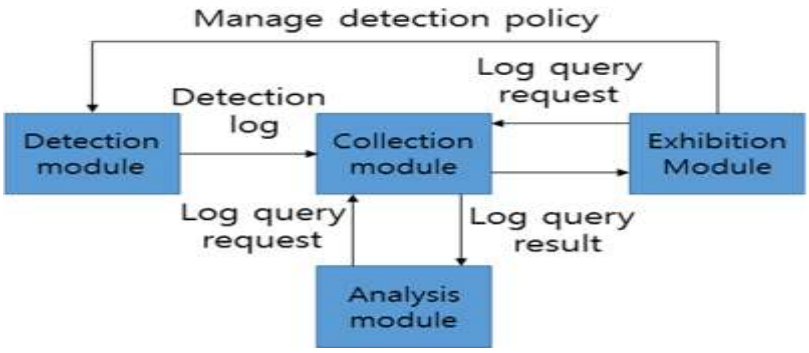

Figure 1. System Architecture

The acquisition module is responsible for storing the results collected by the detection module on each device and performs indexing using Lucene / Solr. Based on the collected logs, the analysis module performs event analysis for analyzing the occurrence of an intrusion event, *etc.*, and performs risk assessment on the internal network based on the event analysis. The exhibition module configures the console screen so that the administrator can monitor the log and risk analysis status.

B. Analysis of big data traffic for efficient campus network resources

Save for a long time to collect traffic from the enterprise network and define the classification criteria for multilateral statistical analysis to analyze the use of the campus network, purpose, frequency, generator, *etc.* made it possible to classify according to the purpose. The figure below shows the system structure for collection and storage.

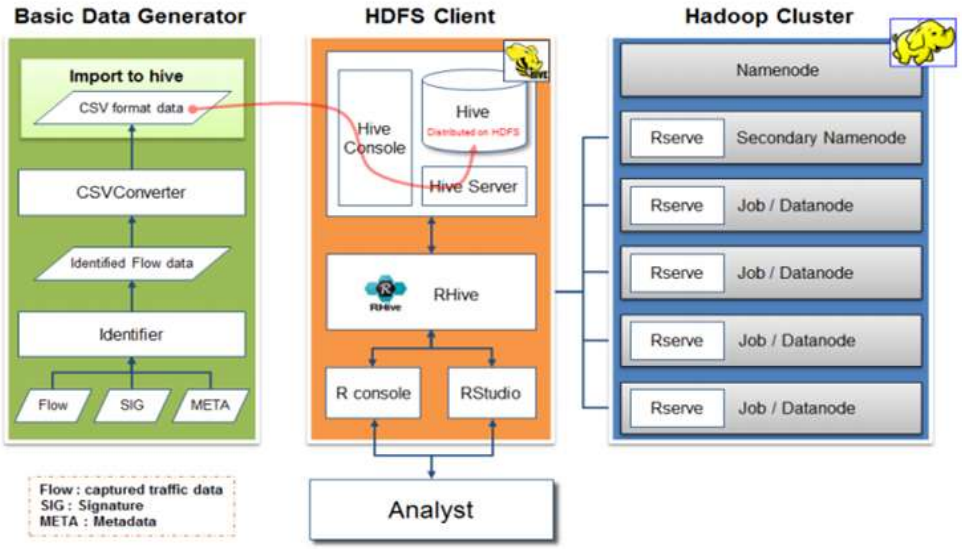

Figure 2. System Architecture

In order to collect the basic data, traffic data is organized into CSV, stored in HDFS, and analyzed using RHive. Then, we can organize data by layer, time, and user based on flow data, and analyze it by building, department, group, and purpose based on user data.

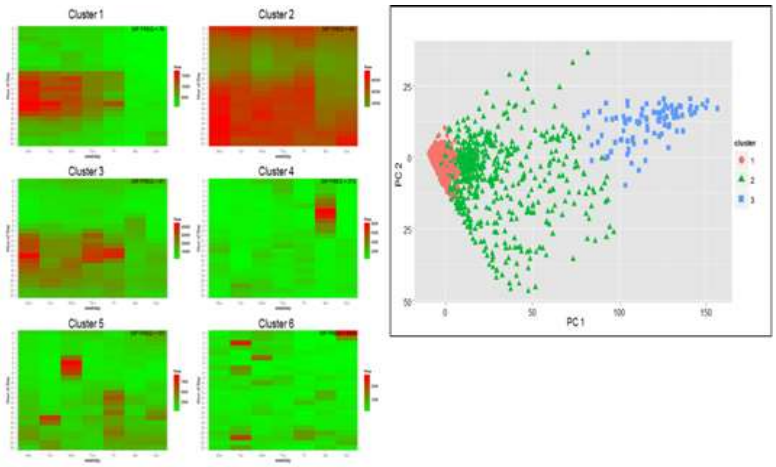

Figure 3. Analysis Result

Clustering and Heatmap creation have shown that a few heavy-torrent users are increasing their traffic and have found that they need to be acted on when the traffic is coming.

3. System Design and Implementation

Field-Control Device (Embedded Device) for Event Logging's overall flow chart for identifying and extracting accident data is shown in the following figure.

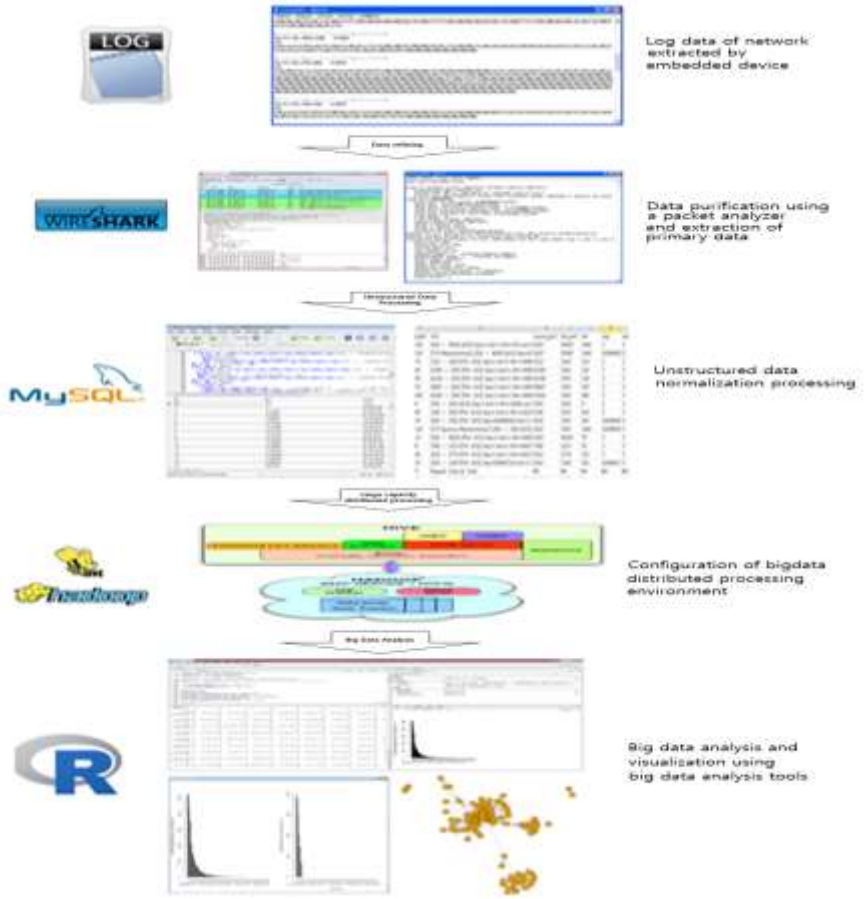

Figure 4. Flow Chart

The network log data extracted from the embedded device was refined by Wireshark packet analyzer and extracted as CSV file format and the unstructured data was processed by using MySQL. After that, secondary processing and refining were performed using Hadoop dispersion tools (Hadoop, Mapreduce, Pig, Hive, *etc.*) and data analysis and visualization were performed using Big Data Analysis Tool R.

In this study, we tried to identify and extract incident control data for field control device (embedded device) for event logging using thread data. Based on the protocol, the number of packets (average, minimum, maximum) and the amount of packets (average, minimum, maximum) are analyzed to identify and extract abnormal packet frequencies or packet amounts. In this experiment, if the network traffic log data of about 15 gigabytes is used, the overall process is as follows.

① The network log data extracted from the embedded device is refined by Wireshark packet analyzer and converted into CSV file format
② Formulate unstructured data using MySQL
③ Perform secondary processing and purification using Hadoop dispersion tools (Hadoop, Mapreduce, Pig, Hive, *etc.*)
④ Analyze and visualize data using big data analysis tool R

A. Convert log data (PCAP) extracted from embedded device to CSV file format
1) PCAP file loading using Wireshark

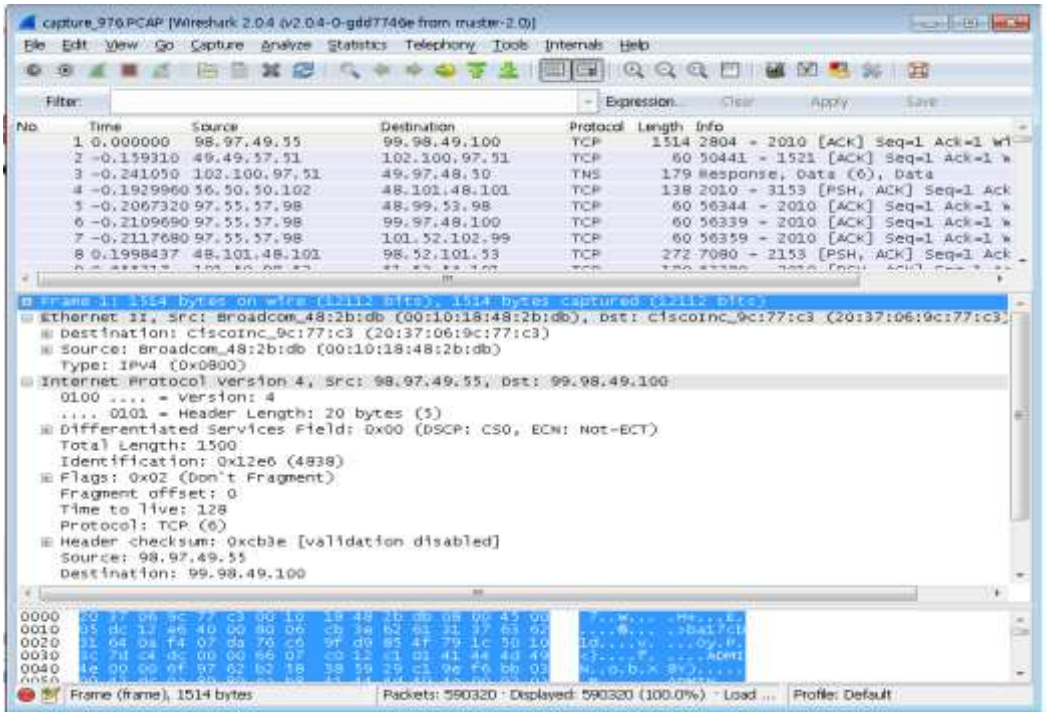

Figure 5. PCAP File Loading

- Wireshark loads important data by layer of log data.

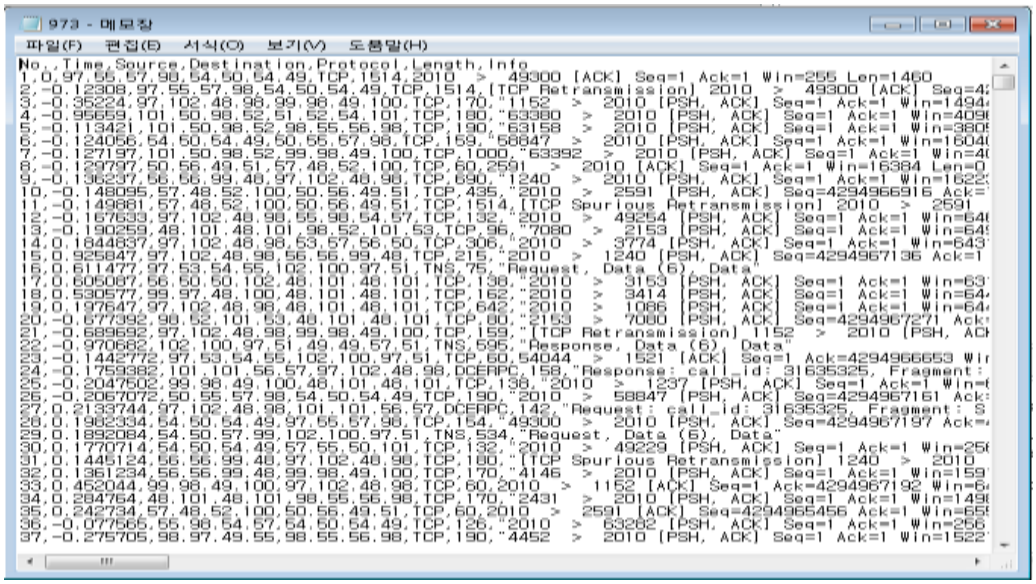

Figure 6. CSV File Convert

- Converts log data to CSV file format via Wireshark
- The extracted CSV file structure contains unnecessary information and unstructured data, so we import it into Mysql for processing and refine it.

B. Formulation processing using Mysql

1) Create Schema

Figure 7. Mysql Table Schema

- Defines the table schema needed to import log data in CSV file format into Mysql

2) Import CSV to MySQL

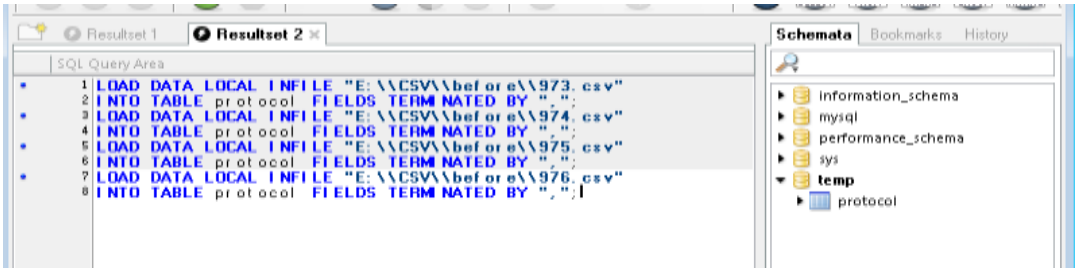

Figure 8. Mysql Import (CSV File)

- Performs SQL script to bulk load multiple CSV files converted via Wireshark
- Removes unformatted unnecessary information when importing CSV file

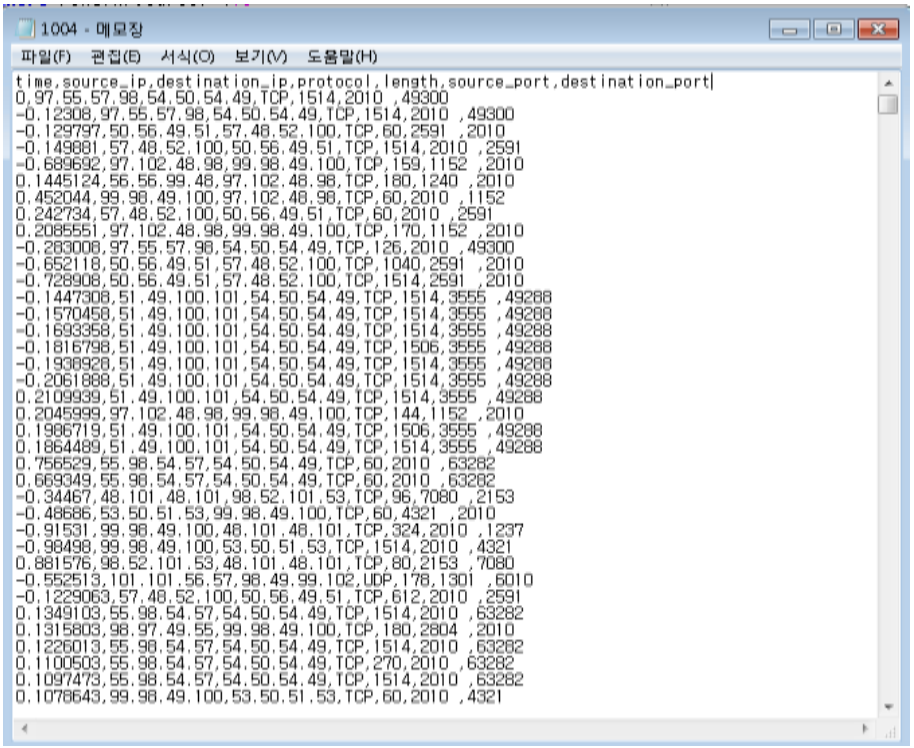

Figure 9. CSV File

C. Purification using Hadoop

1) Import Mysql data into Hadoop

- Import tables in Mysql to Hadoop using Hadoop's sqoop tool

```
sqoop import
--connect jdbc:mysql://localhost/test
--table log

hadoop fs -ls log
Found 4 items
-rw-r--r-- 1 root hdfs 272 log/part-m-00000
-rw-r--r-- 1 root hdfs 241 log/part-m-00001
-rw-r--r-- 1 root hdfs 238 log/part-m-00002
-rw-r--r-- 1 root hdfs 272 log/part-m-00003
```

2) Loading and extracting data using pig

```
grunt> log = LOAD 'log.txt' AS (time, source_ip, destination_ip, source_port,
destination_port, protocol) USING TextLoader();

grunt> tcp_only = FILTER log BY (protocol=='TCP');
```

3) Purification and export of data using pig

```
grunt> log_group = GROUP log BY protocol;

grunt> STORE log_group INTO 'log_group.csv';
```

4) Loading and extracting data using Hive

```
hive> create table log (time int, source_ip string, source_ip string,
destination_ip string, source_port string, destination_port string, protocol
string)
> ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

hive> load data inpath '/user/root/log.txt' into table log;

hive> select * from log;

hive> select * from log where state = 'TCP';
```

5) Data cleansing using Hive

```
hive> SELECT count(protocol)
FROM log GROUP BY protocol;

hive> FROM log o
INSERT OVERWRITE DIRECTORY 'TCP' SELECT o.* WHERE protocol = 'TCP'
INSERT OVERWRITE DIRECTORY 'UDP' SELECT o.* WHERE protocol = 'UDP'
```

D. Feature extraction and visualization using big data analysis tool R

1) Import CSV file

```
    log<-read.csv("E://CSV//after//ch10006_15G.csv", header = T, stringsAsFactors
= F)
```

- Save converted csv file to object variable log of R

2) Separate the columns of the object variable log by detailed variables

```
log$time <-as.factor(log$time)
log$source_ip <- as.factor(log$source_ip)
log$destination_ip <- as.factor(log$destination_ip)
log$protocol <-as.factor(log$protocol)
log$length <-as.factor(log$length)
log$source_port <- as.factor(log$source_port)
log$destination_port <- as.factor(log$destination_port)
```

- Save the time, source_ip, destination_ip, protocol, length, source_port, and destination_port as variables in the object log.

3) Source_ip criteria analysis

a. Maximum frequency, minimum frequency, average frequency analysis

```
frequence_showPivot <- function(x) # Functions that return the highest
frequency, the lowest frequency, and the average frequency
{
print(max(summary(x[,2]))) # The most frequent number
print(min(summary(x[,2]))) # The least frequent number
print(mean(summary(x[,2]))) # Average frequency
}
frequence_showPivot(log)
```


- Summary (x [, 2]) means source_ip, calculates the maximum, minimum and average frequency using the frequence_showPivot function and outputs it

```
> frequence_showPivot <- function(x) # Functions that return the highest frequency,
+ {                                   # the lowest frequency, and the average frequency
+   print(max(summary(x[,2]))) # The most frequent number
+   print(min(summary(x[,2]))) # The least frequent number
+   print(mean(summary(x[,2]))) # Average frequency
+ }
>
> frequence_showPivot(log)
[1] 1238398
[1] 33
[1] 69758.58
```

Figure 10. Source_ip based Max, Min, Avg

- The maximum frequency is 1238398, the minimum frequency is 33, and the average frequency is 69758.58.

b. Top 10 frequency tables by source IP

```
head(sort(table(log$source_ip), decreasing = T),10) # Top N frequency tables
```

```
97.102.48.98 99.98.49.100 102.100.97.51 48.101.48.101 97.55.57.98 56.56.99.48
1238398 836239 693485 577512 409585 383961
55.98.54.57 54.50.54.49 53.50.51.53 57.48.52.100
345325 312713 269824 227447
```

c. Top 10 Frequency Graphs by Source IP

```
plot(head(sort(table(freqDestination), decreasing = T),10)) # Top 10
```

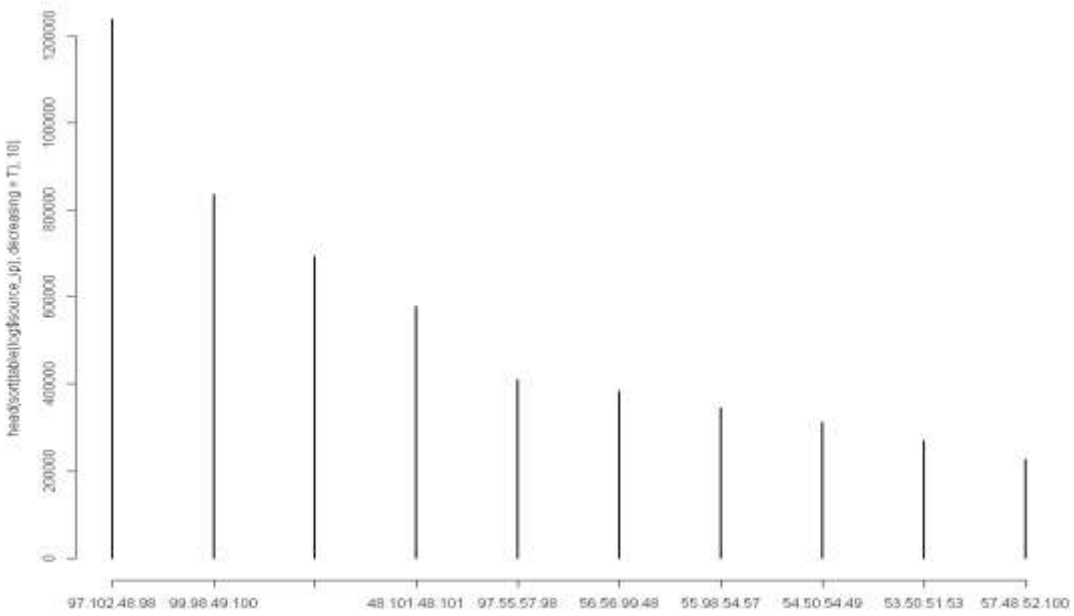

Figure 11. Destination_ip based Top 10(from source_ip)

d. Most frequent Destination_ip sent from Source_ip

```
a <- head(sort(table(log$source_ip), decreasing = T),1)
> freqDestination <-log$destination_ip[log$source_ip == names(a)]
> head(sort(table(freqDestination), decreasing = T),10) # frequency table
```

- Store the maximum frequency source_ip in the variable a and output the top 10 destination_ip from the maximum frequency source_ip.

```
freqDestination
  56.56.99.48 101.101.56.57  99.98.49.100 48.101.48.101   97.55.57.98 56.102.102.57   55.98.54.57   52.53.53.57
       262173        195874        185631        109867         86579         86068         82985         33396
  54.50.54.49   55.98.54.53
        32594         28584
```

Figure 12. Most Frequent source_ip(to destination_ip)

e. Top 10 Destination_ip graphs sent from Source_ip

```
plot(head(sort(table(freqDestination), decreasing = T),10)) # Top 10
```

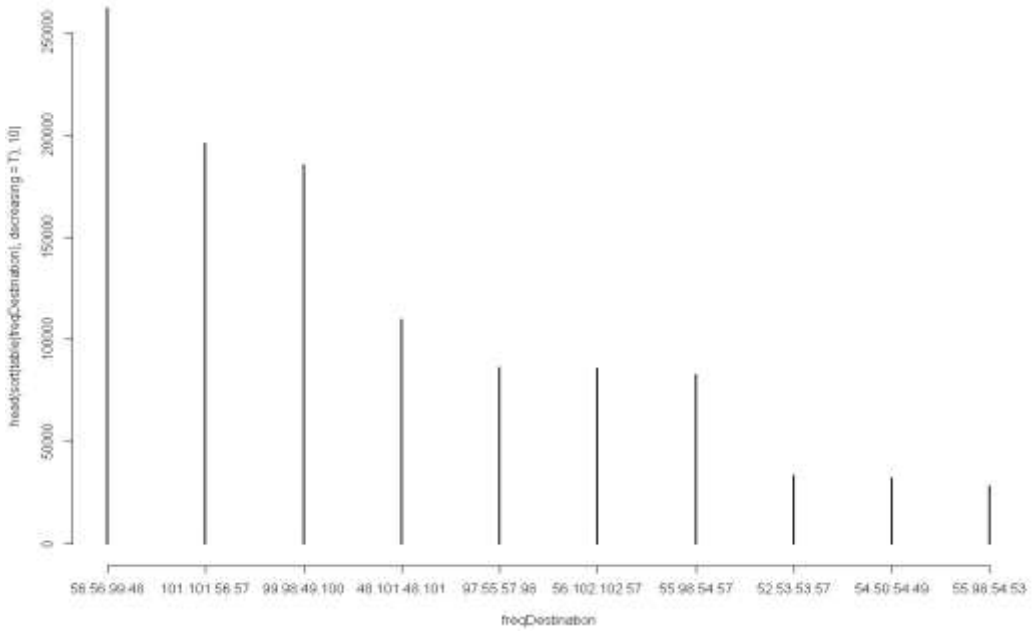

Figure 13. Source_ip based Top 10(to destination_ip)

4) Protocol Analysis

a. Protocol Maximum, Minimum, Average Frequency

```
frequence_showPivot <- function(x) # Functions that return the highest frequency, the lowest
frequency, and the average frequency
{
print(max(summary(x[,4]))) # The most frequency
print(min(summary(x[,4]))) # The least frequency
print(mean(summary(x[,4]))) # Average frequency
}
frequence_showPivot(log)
```



```
frequence_showPivot(log)
[1] 6937073
[1] 1
[1] 634168.9
```

b. Frequency by protocol

```
head(sort(table(log$protocol), decreasing = T, na.rm = TRUE))
```

```
TCP UDP ANSIC12.2 DISTCC H1
6937073 38231 526 17 4 2
```

c. Frequency by protocol graph

```
a <- head(sort(table(log$protocol), decreasing = T, na.rm = TRUE))
pct <- round(a/sum(a)*100,1)
lab <- paste(pct,"%")
pie(a, col = rainbow(length(a)),label=lab)
legend(0.7,1.05,c(names(a)),cex=0.8, fill=rainbow(length(a)))
```

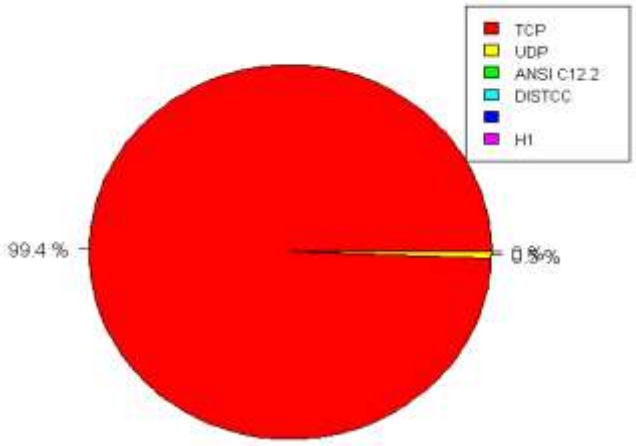

Figure 14. Protocol Frequency

5) Correlation analysis

a. IP correlation analysis

```
library(igraph)
par(mfrow=c(1,1))
par(mar=c(0,0,0,0)) # Set the top and bottom margins of the graph to 0
gh<-graph.data.frame(unique(log[,2:3]))
gh
plot(gh, layout=layout.fruchterman.reingold, vertex.size=8, edge.arrow.size=0.5, vertex.label=NA)
```

```
[1] 97.55.57.98 ->54.50.54.49 99.98.49.100 ->97.102.48.98
[3] 54.50.54.49 ->53.57.56.50 55.56.53.48 ->102.100.97.51
[5] 97.102.48.98 ->56.102.102.57 97.102.48.98 ->99.98.49.100
[7] 54.50.54.49 ->56.56.99.48 99.98.49.100 ->101.50.98.52
[9] 51.52.54.101 ->101.50.98.52 98.52.101.53 ->48.101.48.101
[11] 98.97.49.55 ->99.98.49.100 97.102.48.98 ->48.101.48.101
[13] 99.98.49.100 ->98.97.49.55 57.48.52.100 ->53.50.51.53
[15] 56.56.99.48 ->97.102.48.98 50.56.49.51 ->57.48.52.100
+ ... omitted several edges
```

Figure 15. IP Correlation Analysis

b. Port correlation analysis

```
library(igraph)
par(mfrow=c(1,1))
par(mar=c(0,0,0,0)) # Set the top and bottom margins of the graph to 0
gh<-graph.data.frame(unique(log[,6:7]))
gh
plot(gh, layout=layout.fruchterman.reingold, vertex.size=8, edge.arrow.size=1.5, vertex.label=NA)
```

```
IGRAPH DN-- 14259 27118 --
+ attr: name (v/c)
+ edges (vertex names):
[1] 2010 ->49300 2010 ->1152 2010 ->4040 2306 ->1521 2010 ->63833 1152 ->2010
[7] 2010 ->3335 2010 ->63392 2010 ->63380 2153 ->7080 2804 ->2010 2010 ->1086
[13] 2010 ->2804 2010 ->1348 1240 ->2010 2591 ->2010 1086 ->2010 2010 ->1237
[19] 7080 ->2153 1237 ->2010 2010 ->2591 1521 ->54045 1441 ->2010 1738 ->2010
[25] 2010 ->3414 2010 ->63282 2010 ->1738 2010 ->1441 2505 ->1521 2776 ->1521
[31] 2010 ->49254 2010 ->1240 2010 ->58847 2010 ->49259 3414 ->2010 4321 ->2010
[37] 1957 ->14000 2010 ->4321 2855 ->14000 2010 ->3153 2010 ->56337 2010 ->58790
[43] 3153 ->2010 2010 ->4452 4146 ->2010 1521 ->54044 3755 ->1521 3555 ->49288
+ ... omitted several edges
```


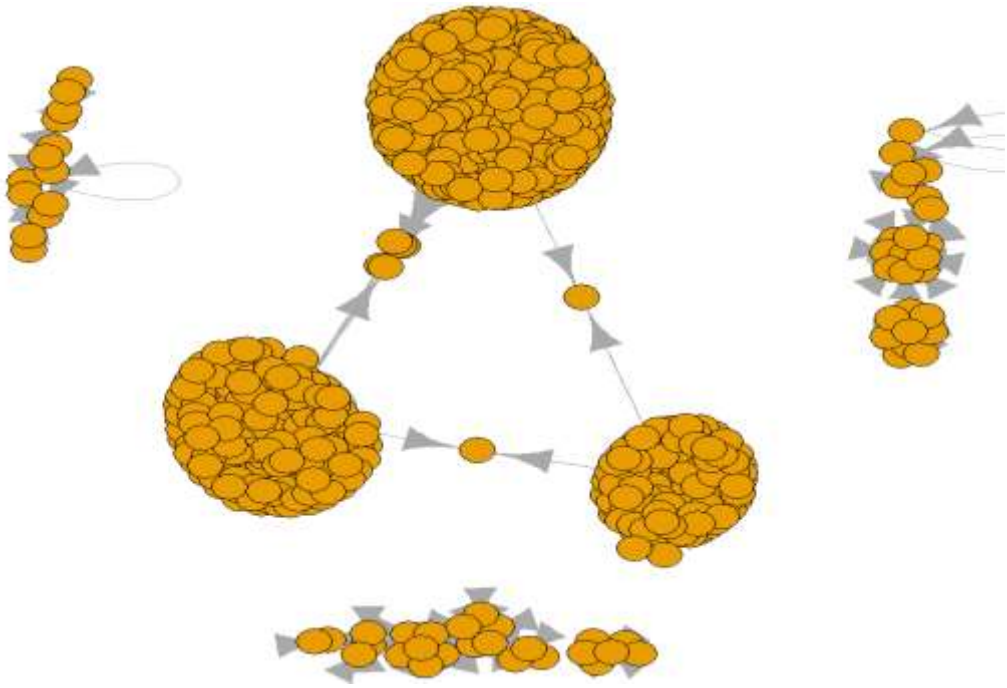

Figure 16. Port Correlation Analysis

4. Conclusion

PLCs and other field control devices do not have the ability to log important event information, so accident analysis was difficult. Therefore, it is necessary to log important event information of field control devices such as PLC and IED, and to secure information that can be analyzed when a cyber accident occurs. Field controller for embedded event logging (embedded device) After analyzing industrial protocol analysis and trends, we propose a payload data classification method and conducted a study on identification and extraction of incident data using real data.

This paper is a preliminary study for extracting payload data of field control device communication protocol for event logging. Based on the results of this paper, it can be used to collect event logging information and develop a system (industrial black box) for detecting abnormal behavior The network log data extracted from the embedded device was refined by Wireshark packet analyzer and extracted as CSV file format and the unstructured data was processed by using MySQL. After that, secondary processing and refining were performed using Hadoop dispersion tools (Hadoop, Mapreduce, Pig, Hive, *etc.*) and data analysis and visualization were performed using Big Data Analysis Tool R.

In addition, it is expected that various domestic and overseas researches and related technologies will be continuously developed for analysis of field control device (embedded device) communication protocol in order to secure cyber accident prevention and countermeasures for the national infrastructure control system.

Acknowledgement

This research was supported by the KIAT(Korea Institute for Advancement of Technology) grant funded by the Korea Government(MOTIE : Ministry of Trade Industry and Energy). (No. N0002429)

This paper is a revised and expanded version of a paper entitled [Bigdata based Network Traffic Feature Extraction] presented at [2017 the 14th international workshop series, daejeon university, korea, 2017.12.21~2017.12.23].

References

[1] H. M. An, S. K. Lee, K. S. Sim, I. H. Kim, S.H. Jin and M. S. Kim, "Big-Data Traffic Analysis for the Campus Network Resource Efficiency", Journal of Koran Institute of Communications and Information Sciences, vol.40, no.3, **(2015)**, pp.541-550.
[2] J. S. Park, S. H. Yoon and M. S. Kim, "Software architecture for a lightweight payload signature-based traffic classification system", Proc. of 3rd Int.Conf. Traffic Monitoring and Analysis, vol. 11, **(2011)**, pp.136-149.
[3] S. J. Oh, "Design of a Smart Application using Big Data", Journal of IIBC, vol.15. no.6, **(2015)**, pp.79-84.
[4] D. H. Lee, J. C. Park, C. G. Yu and H. S. Yun, "On the Design of a Big Data based Real-Time Network Traffic Analysis Platform", Journal of The Korea Institute of Information Security & Cryptology, vol.23, no.4, pp.721-728, **(2013)**.
[5] S. H. Yoon and M. S. Kim, "Behavior Based Signature Extraction Method for Internet Application Traffic Identification, "Journal of Koran Institute of Communications and Information Sciences, vol.38, no.5, **(2013)**, pp.368-376.
[6] J. H. Kim, S. H. An, Y. J. Won, J. M. Lee and E. Y. Lee, "Detection of Traffic Anomalities using Mining: An Empirical Approach", Journal of KISE, vol.33, no.3, **(2006)**, pp.201-217.
[7] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks", Proc. of Internet Measurement Conf, **(2002)**, pp.137-150.
[8] F. Risso, M. Baldi, O. Morandi, A. Baldini and P. Monclus, "Lightweight, payload-based traffic classification: an experimental evaluation", Proc. of IEEE Int. Conf. Communication, vol. 08, **(2008)**, pp.5869-5875.
[9] J. H. Jun, M. J. Kim, J. H. Cho, C. W. Ahn and S. H. Kim, "Detection Method of Distributed Denial-of-Service Flooding Attacks Using Analysis of Flow Information", Journal of IIBC, vol.14. no.1, **(2014)**, pp.203-209.

Authors

Jeong Joon Kim, he received his BS and MS in Computer Science at Konkuk University in 2003 and 2005, respectively. In 2010, he received his PhD in at Konkuk University. He is currently a professor at the department of Computer Science at Korea Polytechnic University. His research interests include Database Systems, BigData, Semantic Web, Geographic Information Systems (GIS) and Ubiquitous Sensor Network (USN), *etc.*
e-mail: jjkim@kpu.ac.kr

Yong-soo Lee, he received his MS in Computer Science at Konkuk University in 1989. In 2015, he received his PhD in Information & Control Engineering at Kwangwoon University. He is currently a professor at the Department of Computer Information at Yeoju Institute of Technology. He is the Member of the Korea Institute of Internet, Broadcasting & Communication (IIBC). His research interests include Database Systems, Data Mining, BigData, Wireless Sensor Networks and Ubiquitous Sensor Network (USN), *etc.* e-mail: diclee@yit.ac.kr

Jin-Yong Moon, he received his BS in Computer Science at Suwon University in 1996. He received his MS in Computer Science at Konkuk University in 1998. Then he received Ph.D. degree from Suwon University in 2001. He is currently a professor in the department of Visual Broadcasting Media at Gangdong College. His research interests include Database Systems, Mobile Systems, Geographic Information Systems (GIS) and Multimedia Systems, *etc.*
e-mail: jmoon37@gmail.com

Jeong-Min Park(Corresponding Author), he received his Ph.D. and M.S. degrees in Department of Computer Engineering from Sungkyunkwan University, Korea, in 2009 and 2005, respectively, and his B.S. degree in Computer Engineering from Korea Polytechnic University, in 2003. Currently, He is currently an assistance professor of the Department of Computer Engineering at Korea Polytechnic University, Korea. From 2012 to 2014, he was a senior member of engineering staff, in ETRI, Korea. His research interests include Cyber-Physical System (CPS), Autonomic Computing and Software Engineering.

e-mail: jmpark@kpu.ac.kr