Development of a Network Intrusion Detection System Using Apache Hadoop and Spark

Keisuke Kato* and Vitaly Klyuev†
Graduate Department of Computer and Information Systems
The University of Aizu, Japan
Email: *s1190085@gmail.com, †vkluev@u-aizu.ac.jp

***Abstract*—Cyber attacks on network communication are executed against companies, governments, and even individuals. A number of these attacks is drastically increased over the last decade. Nowadays, protecting private data, latest research data, etc. is a crucial problem. Therefore, developing an intelligent system to detect the attacks is required. In this paper, we propose an anomaly based network intrusion detection system. The system is capable to analyze huge datasets in a short period of time. We utilized 90.9 GB of a real network packet dataset provided by the Information Security Centre of Excellence at the University of New Brunswick. The system analyzes the packet capture files of this dataset in the environment by using Apache Hadoop and Spark. An approach to implement the system is based on Hive SQL and unsupervised learning algorithms. The accuracy of the proposed detection system is 86.2% with 13% of the false positive rate. These results are promising to detect attacks in real-time.**

I. Introduction

According to Symantec's report [1], IT departments at companies, governments, etc. need to be proactive in reducing the risk from persistent intrusions, malware, and security violations. Security should start digging through the data proactively during non-breach response time. Therefore, developing an intelligent intrusion detection system is highly demanded. The system itself must be capable to analyze data quickly.

There are mainly two types of intrusion detection systems (IDS): misuse (signature) based and anomaly based. A misuse based system is designed to detect the already known attacks by using signature rules defined by system administrators and/or security experts. The system needs to update the database of the signatures, if a new type of attacks is reported. Therefore, the misuse based system can detect the known attacks with high accuracy. However, the system cannot be helpful against unknown attacks such as zero-day attacks. An anomaly based system checks the network behavior and classifies it as normal or abnormal. This system can detect both known and unknown attacks by using statistical based, knowledge based, or machine learning based techniques. The main challenge in the anomaly based systems is how to get high accuracy with the low false positive rate.

Our prior work [2] studied the development of an intelligent detection system against distributed denial of service (DDoS) attacks by analyzing 44 GB network packet data. The packet capture (pcap) file is a binary file. A specific library or application is needed to read the data. Wireshark [3] was selected as a main tool to analyze the provided 14 pcap files. The applied technique extracts some features including source IP and destination IP addresses, time interval in seconds between packets, and packet size in bytes from the dataset. After that, it analyzes the extracted data. We utilized a support vector machine (SVM) for classification of packets. Experiments demonstrated that the detection accuracy is high and the results of the network packet analysis are very useful to develop detection systems against DDoS attacks. The results illustrated that analyzing the network packets is one of most important tasks. However, it takes very long time to extract feature data from pcap files applying Wireshark. It took more than 10 days to get the results. To develop a practical system, we need to analyze the pcap file quickly. We also need to find a way to process this file directly.

The main goal of this study is to develop the intrusion detection system to handle and analyze a huge size of rafw network packet data and detect the malicious attacks without long training time. Therefore, we utilized Apache Hadoop [4] and Hive [5] to store and analyze the data. Apache Hadoop allows the distributed processing of large datasets across a cluster of computers using simple programming models. It is designed to scale up from a single server to thousands of machines, each offering local computation and storage. Thus, we can scale up the proposed detection system to distributed detection systems in the future.

In this research, we focused on developing a practical intelligent anomaly based IDS. The proposed IDS can handle network packet data directly, analyze them, and detect the attacks with high accuracy and the low false positive rate. We selected Apache Spark [6] to work with the dataset directly. We applied principal component analysis (PCA) [7] for reducing feature dimension and Gaussian mixture model (GMM) [8] to cluster the network behaviors into normal and attack classes.

The remainder of this paper is organized as follows. In Section II, we review recent publications in this research area. In Section III, we discuss details of the provided dataset and the reasons why we selected it. In Section IV, we describe the details of implementation using Hadoop, Hive, and Spark. In Section V, we discuss anomaly detection algorithms and present the experimental results. The concluding remarks

978-1-5090-5569-2/17/$31.00

summarizing key results and characterizing future research are in Section VI.

II. Related Work

In this section, we review the recent important achievements in network security, development for network intrusion detection systems, and network packet analysis.

Sommer and Paxson [33] discussed the main problems of network intrusion detection using machine learning algorithms. They introduced a set of guidelines for applying machine learning to network intrusion detection. They argued that the important point in this research area is a reproducible experiments.

Buczak and Guven [9] summarized the approaches using machine learning and data mining methods for cyber analytics for intrusion detection. They mentioned that the most of publications on anomaly detection methods actually utilized anomaly and misuse detection methods. In this study, we used only anomaly detection mechanisms.

Bachupally et al. [10] presented an approach to analyze the network packet data and detect anomalous connections to the network by using Hadoop Data File System (HDFS). They extracted some features from 131 MB pcap data by using Wireshark, exported them to a CSV file, and uploaded to the HDFS environment. However, they handled small data and also used Wireshark to export data to the CSV file. It means that their approach requires long time for getting results of the analysis.

Anwar et al. [11] classified the network attacks based on the security policy and proposed optimum response option based on statistical analysis of attacks during the response selection process. They mentioned that there are many studies about IDS and the researchers paid less attention to intrusion response systems (IRS). According to them, it is essential to have an appropriate IDS and IRS to detect and respond to the potential intrusion and attacks. They introduced several optimum response solutions. Disconnecting from the network and shutting down the host or network system are among them. They stated that the first one is more appropriate than the second one.

Zanero and Savaresi [12] presented the unsupervised learning techniques for development of an intrusion system. They introduced a two-tier architecture. The first tier is implemented by using an unsupervised learning algorithm to reduce the network packets payload. The second tier is an anomaly detection algorithm. They chose the K-Means algorithm, the principal direction algorithm, and Kohonen's Self Organization Maps (S.O.M) algorithm to classify the packets. They found that the S.O.M algorithm is the best one among the chosen algorithms.

Aghdam and Kabiri [13] studied the important features to develop an intelligent intrusion detection system. They applied Ant Colony Optimization (ACO) to reduce the dimension of the feature space and improve the performance of IDS. Their experiments with the KDD Cup 99 dataset [14] showed that the proposed method can reduce the number of features by approximately 88% and the error rate reduced by around 24%.

Alseiari and Aung [15] proposed the real-time distributed intrusion detection system (DIDS) based on unsupervised stream data mining. They utilized the sliding window approach that monitors the data traffic flow. They showed good performance, high detection accuracy and low false positive rates of the mini-batch K-Means algorithm.

Kyong et at. [34] discussed the scalability of Apache Spark by comparing different results (WordCount, Naive Bayes, Grep, and K-Means). They demonstrated that Spark has enough scalability up to 60 cores and some difficulties from 61 to 120 cores because of a garbage collection (GC) problem.

From these studies, we found that one of the most difficult problem is to handle a large packet data and analyze it. We selected Hadoop, Hive and Spark for scalability and flexibility of the proposed system. Spark is a fast general-purpose cluster computing system. Its execution speed is faster than Hadoop MapReduce in main memory. Many libraries are available for it. Thus, in this study, we designed the analysis environment to handle the large packet data by using Hadoop, Hive, and Spark.

III. Datasets for IDS

A. Comparison of IDS Datasets

There are many studies using different datasets in this area. Datasets are classified into two types: datasets provided by institutions, and datasets created by individuals. Dataset owners may put them into the public domain. However, creating a dataset for an IDS is very difficult for any researcher and there are a few works utilizing datasets of the second category. The publicly available datasets DARPA-99 [16] and KDD-99 [14] are the most popular. We should note that these datasets were created more than 15 years ago, and according to Folino et al. [18], they cannot provide real scenarios. As we mentioned before, the goal of this study is to develop a practical intrusion detection system. Therefore, the aforementioned datasets may not be suitable for this purpose.

Shiravi, Tavallaee et al. [19] created UNB ISCX 2012 intrusion detection evaluation dataset [20] for developing an IDS. They compared their dataset with several widely used ones. The main features of the UNB ISCX 2012 dataset are as follows: It consists of labeled network traces including full packet payloads in pcap format. The total size of the dataset is 90.9 GB. It covers seven days of network activity. The capturing packets were executed from 00:01:06 on Jun 11th, 2010 to 00:01:06 on Jun 18th, 2010. This dataset has four types of attack scenes: infiltrating the network from the inside, HTTP denial of service (DoS), distributed denial of service (DDoS) using an IRC Botnet, and brute force SSH. Therefore, the UNB ISCX 2012 dataset comply with the requirements for evaluating an IDS. Table I shows its brief summary. This dataset is publicly available for researchers.

Table I
DETAILS ON THE UNB ISCX 2012 DATASET

Date	Description	File size	Number of Normals	Number of Attacks	Percentage of Attacks
2010/6/11 Friday	Normal Activity No malicious activity	17.31GB	378667	0	0.0
2010/6/12 Saturday	Normal Activity Non-classified attacks	4.53GB	131111	2082	1.56
2010/6/13 Sunday	Infiltrating the network from inside Normal Activity	4.25GB	255170	20358	7.4
2010/6/14 Monday	HTTP Denial of Service Normal Activity	7.37GB	167609	3771	2.2
2010/6/15 Tuesday	Distributed Denial of Service using an IRC Botnet	25.15GB	534320	37378	6.5
2010/6/16 Wednesday	Normal Activity No malicious activity	18.99GB	522263	0	0.0
2010/6/17 Thursday	Brute Force SSH Normal Activity	13.3GB	392392	5203	1.31
Total		90.9GB	2381532	68792	2.8

```
<TestbedTueJun15-1Flows>
<appName>BitTorrent</appName>
<totalSourceBytes>391223</totalSourceBytes>
<totalDestinationBytes>4114484</totalDestinationBytes>
<totalDestinationPackets>7322</totalDestinationPackets>
<sensorInterfaceId>1</sensorInterfaceId>
<totalSourcePackets>5650</totalSourcePackets>
<sourcePayloadAsBase64></sourcePayloadAsBase64>
<sourcePayloadAsUTF></sourcePayloadAsUTF>
<destinationPayloadAsBase64></destinationPayloadAsBase64>
<destinationPayloadAsUTF></destinationPayloadAsUTF>
<direction>L2R</direction>
<sourceTCPFlagsDescription>F,P,A</sourceTCPFlagsDescription>
<destinationTCPFlagsDescription>F,P,A</destinationTCPFlagsDescription>
<source>192.168.2.107</source>
<protocolName>tcp_ip</protocolName>
<sourcePort>1989</sourcePort>
<destination>80.246.149.72</destination>
<destinationPort>6880</destinationPort>
<startDateTime>2010-06-14T21:54:57</startDateTime>
<stopDateTime>2010-06-15T01:43:38</stopDateTime>
<Tag>Normal</Tag>
</TestbedTueJun15-1Flows>
```

Figure 1. Sample Code of the XML file

B. Features of the UNB ISCX 2012 dataset

In the UNB ISCX 2012 dataset, there are two types of data: pcap files on daily basis and label files in XML. Figure 1 shows a sample code of the XML file. The Tag attribute shows the label of the network behavior as Normal or Attack. We executed Java programs to extract some attributes from the label files to CSV files. We selected the following attributes: *totalSourceBytes*, *totalDestinationBytes*, *totalDestinationPackets*, *totalSourcePackets*, *source*, *protocolName*, *sourcePort*, *destination*, *destinationPort*, *startDateTime*, *stopDateTime*, *Tag*. As result, a 119.9 MB file was created. We only used this file to check whether the classification by the detection system is correct or not. Selection of these attributes was done according to the technique presented in [2]. We should note that accurate selection of the right attributes is an open problem.

According to report [1], DDoS attacks are growing in number and intensity. They last for 30 minutes or less. While the DDoS attack executes, the connection time is not far from the normal one. Figure 2 shows the mean of the packet bytes per day. The peak was on June 15th when DDoS attacks were executed. Figure 3 shows the mean of the connection time per day. There is no anomaly on June 15th. The peak value was reached on the last day when the brute force ssh attack was executed. This analysis allows us to conclude that this dataset with its real data is useful for this study.

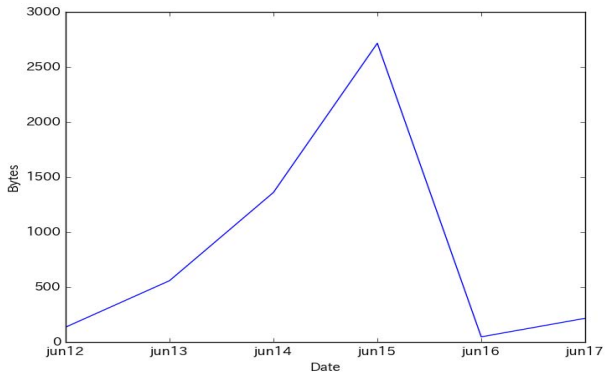

Figure 2. Mean of the Packet Bytes for each day

IV. IMPLEMENTATION DETAILS

In this section, we discuss the details of the system implementation. We used several different tools to develop the system. As we mentioned before, one of the biggest problem is how to store the pcap data to the database and read the data from it. It is also very important for scalability and flexibility. Therefore, to meet these requirements, Apache Hadoop and

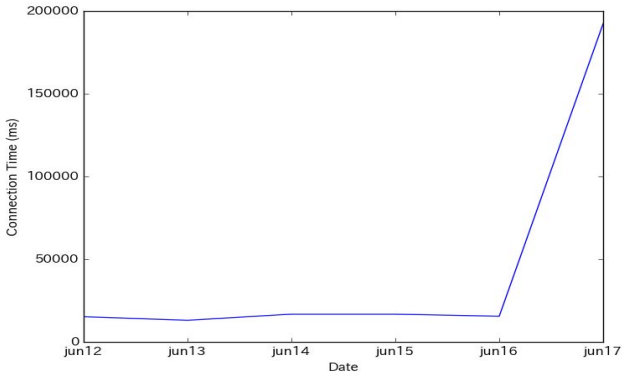

Figure 3. Mean of the Connection Time for each day

Hive were selected. However, there is no default option or library to handle the pcap file directly on Hadoop and Hive. We utilized the Hadoop PCAP library developed by Reséaux IP Europeeńs Network Coordination Centre (RIPE-NCC) [21]. The library allows us to store the pcap data to Hive tables.

Apache Spark [6] is a cluster computing engine to analyze the large-scale data easily. As we mentioned before, Spark can be significantly faster than Hadoop for large-scale data processing by exploiting memory during computing and other optimizations. Spark is also fast when data is stored on disk. We can utilize the libraries such as Spark SQL for structured data processing, MLlib for machine learning algorithms, GraphX for graph processing, and Spark Streaming. Spark Streaming is one of main tools to build scalable fault-tolerant near real-time applications. We can write streaming jobs in the same way as batch jobs.

To communicate Spark with Hive and Hadoop, we import HiveContext to execute HiveSQL on Spark. We also used resilient distributed dataset (RDD) which is one of the most important features in Spark. RDD is a collection of objects partitioned across the machines of the cluster. We can analyze the data and create a fault-tolerant application by using RDD because RDD automatically recovers from failures. After reading the data from the Hive table in Spark, the proposed approach can build the basic function, implement and execute the detection algorithm, and evaluate the results.

Hadoop, Hive, and Spark are hadoop-related projects at Apache and each developer and contributor carefully implements libraries and APIs to be able to call their functions/procedures from different projects. We can analyze a large-scale dataset in a short period by using Hadoop and Spark. As we mentioned before, the prior work [2] took more than 10 days to get the results. In this study, the calculation process to produce results lasted for 40 minutes. In addition, we can develop a cost efficient intelligent detection system which can scale up from one cluster to thousand of clusters. There are a few studies that use these tools because Spark was released in 2009 and version 1.0.2 was released in August 2014. Therefore, there are a small number of publications on how to use Spark. However, recently, many research teams including the world famous companies started to use Spark in their research and production. For example, Baidu teams are using Spark for data processing tasks [22].

To the best of our knowledge, this study is the first one to use Spark for processing network packet data and developing an intrusion detection system.

In this study, the experiments were executed on MacBookPro with Intel Core i7 2.0 GHz having 16 GB of main memory. We used Hadoop 2.7.1, Hive 1.2.1, and Spark 1.5.2. When we start Spark's server on localhost, we can see the process as shown in Figure 4. We set up four workers on Hadoop and assigned 3 GB memory for each worker. Figure 5 shows the job tracking on the Spark's application. We executed the application by using 8 cores. The Event Timeline shows the work process of the application. The bars show the status of the tasks. Figure 6 is a fragment of the screenshot. It illustrates how jobs are assigned and executed on each cluster on Apache Spark. On Figure 7, we can see the growth of the input data on each cluster during the execution, processing time of the data, etc. Illustrations shown in Figures 4-7 give the notion of the complexity of the operations to process data and the power of visualization tools to control these operations.

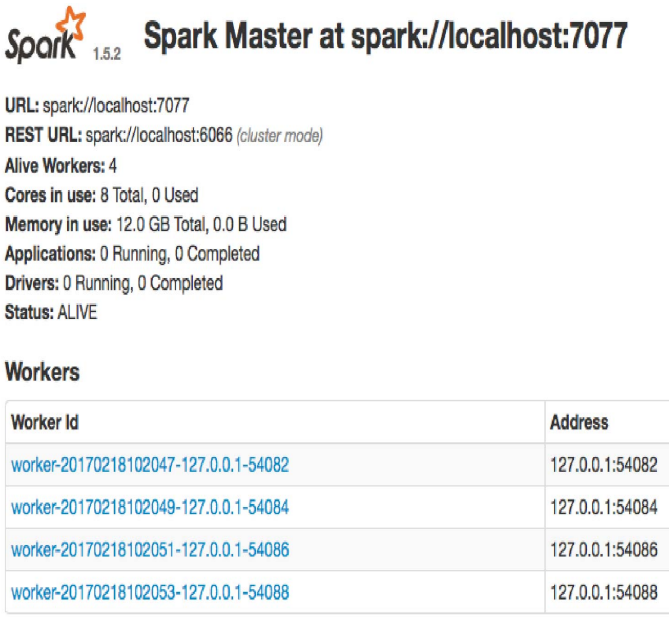

Figure 4. Apache Spark's Master Web UI

V. Proposed System

A. Anomaly Detection Algorithm

Figure 8 shows the workflow of the proposed system. The 90.9 GB of pcap data are stored to the Hive table. The system reads source/destination IP addresses, the ports, protocol type, time stamp, the day label of the packets, bytes of the packet from this table. It uses the key-value data type for analyzing the packet data. As a type of the key, it selects source/destination IP addresses, ports, protocol type, and the

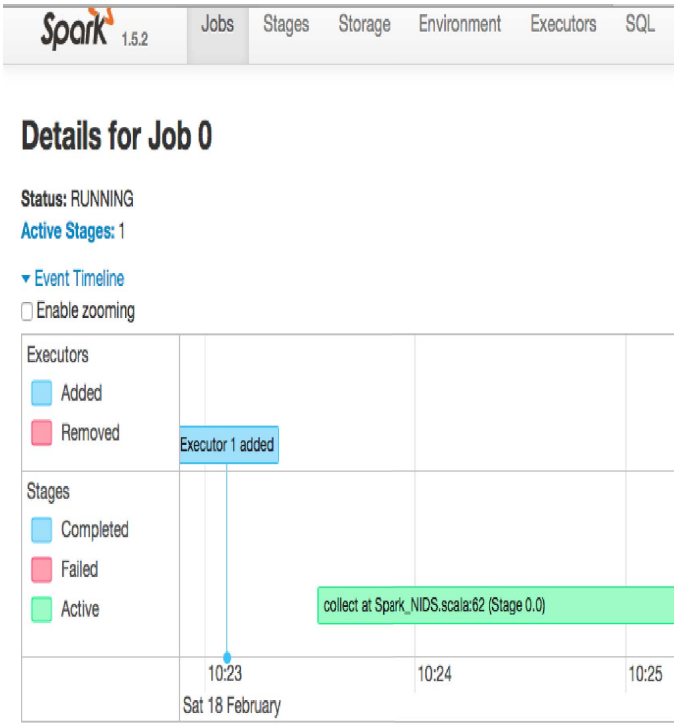

Figure 5. Job Tracking on Apache Spark

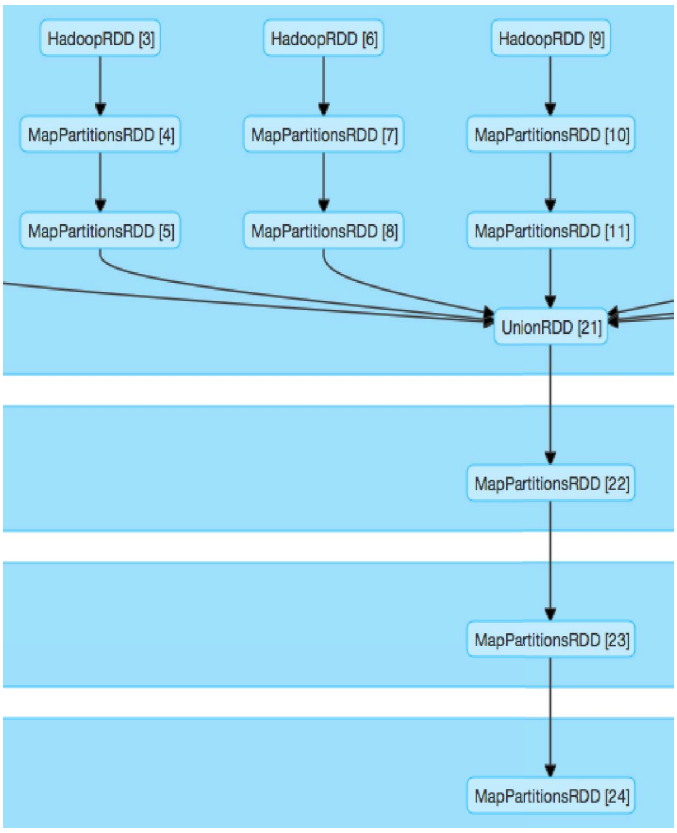

Figure 6. Jobs on Apache Spark

Tasks

Index ▲	ID	Duration	GC Time	Input Size / Records
0	0	7.9 min	6 s	2.3 GB (hadoop) / 3144062
1	1	7.9 min	6 s	2.3 GB (hadoop) / 3281838
2	2	7.9 min	6 s	2.5 GB (hadoop) / 3617313
3	3	7.9 min	6 s	1189.5 MB (hadoop) / 1731826
4	4	6.6 min	5 s	1408.0 MB (hadoop) / 2121943
5	5	7.9 min	6 s	2.4 GB (hadoop) / 3817196
6	6	7.9 min	6 s	2.3 GB (hadoop) / 3050344

Figure 7. Details for the task on Apache Spark

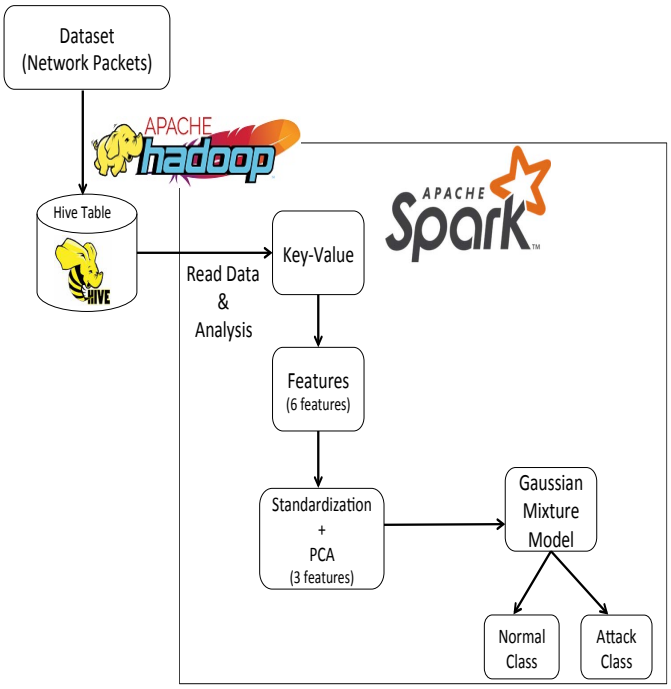

Figure 8. Workflow of the IDS System

day label of the packets. As a type of the value, we selected 6 following features: the number of the packets, a size in bytes of the packet, mean value bytes in the packet, bytes in the packet value per second, time interval between packets, mean of the time interval. Our prior study [2] showed that these values are useful as the features for detecting anomaly behaviors.

In machine learning, normalization and/or standardization is a key technique to get reliable results. Values for some features may vary from small to very big numbers. In such situations, analyzed processes may go off the scale by using unnormalized data. To avoid such effects, we normalized the features by using unit variance and zero mean.

Many researchers utilized supervised learning algorithms to detect anomaly behaviors. However, applying unsupervised learning algorithms for an IDS is important in practice because

labeled data may not be available in a real situation.

We selected Gaussian mixture model (GMM) algorithm [8] to detect anomaly behaviors in the data. The GMM algorithm is from an unsupervised learning category. It is widely used for clustering problems. The GMM algorithm estimates the probability of the data belonging to a class by modeling the probability density function of the data. We used the expectation-maximization (EM) algorithm to estimate the maximum likelihood given a set of data. In EM algorithm, there are E step and M step. In E step, it computes the logarithmic likelihood of the entire dataset with K samples. In M step, it finds the parameters by maximizing the logarithmic likelihood function. This procedure is iterated until the convergence.

To execute the GMM algorithm, we should specify a number of components for feature data. However, it is difficult for any researcher to know the appropriate number of components. In this study, we used PCA to reduce the dimension of feature data to three dimensions.

B. Analysis of the Experimental Results

We executed the GMM algorithm to cluster the network packet data into normal and attack classes: K is set to 2. Table II shows the confusion matrix of the result. To evaluate the clustering results, we used aforementioned label data (119.9MB CSV file) to check whether the clustering is correct or not. Evaluation of the results is not an easy task because the UNB ISCX 2012 dataset includes 97.2% of normal data and 2.8% of attack data. So, if the system classifies all data as normal, the accuracy would be 97.2%. To avoid this situation, the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR) are often used for the evaluation. When FPR is high, we misunderstand the network behavior as attack. Then, we have to take some actions for the normal network behavior. Interpretation of the aforementioned rates are given below.

- TPR: the ratio between packets with the *attack* label and packets with the same label detected by the system;
- TNR: the ratio between packets with the *normal* label and packets with the same label detected by the system;
- FPR: the ratio between packets recognized as *normal*, but detected by the system as the representatives of the *attack* class;
- FNR: the ratio between packets recognized as *attack*, but detected by the system as members of the *normal* class.

The calculations showed that the accuracy is 86.2%, TNR is 87%, and FPR is 13%. The anomaly based system can detect about 47% of the attacks without pre-training. In addition, we also calculated recall and area under the receiver operating characteristic curve (ROC), called AUC [23], [24]. Recall is the ratio between the size of correctly predicted attack class and the actual attack class. The AUC shows the goodness of the classifier. Definitions of the parameters used in these formulas are given below. If the packet belongs to the the normal class and it is classified as normal, it is counted as a TruePositive. If it is classified as a member of the attack class, it is counted as a FalseNegative. If the packet is from the attack class and it is classified as positive, it is counted as a FalsePositive. The corresponding values are 0.52 and 0.67.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$AUC = \int_0^1 \frac{TruePositive}{P} d(\frac{FalsePositive}{N})$$

where P is the sum of True Positive and False Negative, and N is the sum of True Negative and False Positive.

Table II
CONFUSION MATRIX OF THE RESULTS

Classes of the packets and their categories		Predicted Class	
		Attack	**Normal**
True Class	**Attack**	0.516	0.483
	Normal	0.168	0.831

Listing 1 shows the sample code to run Standardization, PCA, GMM, and K-Means. To standardize the feature data, we need to specify two parameters *withMean* to center the data with mean before scaling and *withStd* to scale the data to the unit standard deviation. When we run PCA to reduce the feature dimension, we call *PCA* and need to specify one parameter. It is a number of reduced dimension with input data. To run GMM, we call *GMM()* and specify each parameter. We set k to 2 and another parameters to default values: The convergence tolerance is 0.01, the max iteration is 100, the seed is selected randomly.

```
//Standardization with unit variance and zero
    mean
val scaler2 = new StandardScaler(
withMean=true, withStd=true
).fit(features)
val scaler = scaler2.transform(features)

//Principal Component Analysis
val pca = new PCA(3).fit(scaler)
val pca_model = pca.transform(scaler)

//Gaussian Mixture Model Algorithm
val gmm = new GaussianMixture().setK(2)
.run(pca_model)
val gmm_prediction = gmm.predict(pca_model)

//K-Means Algorithm
val kmeans = new KMeans().setK(2)
.setInitializationMode("k-means||")
.setMaxIterations(100)
val clusters = kmeans.run(pca_model)
val kmeans_prediction = clusters
.predict(pca_model)
```

Listing 1. Code to run Standadization, PCA, GMM, and K-Means

To run K-Means, we call *KMeans()* and specify the following parameters: k is 2, the max iteration is 100, and another parameters have default values: The number of computing to run the K-Means is 1, the initializationMode is *k-means||* [32],

the number of steps in the K-Means is 5, the epsilon is 0.0001, the seed is selected randomly.

We used scikit-learn [17] in Python to run OCSVM. OCSVM classifies new data into similar or different. The training data should be given from one class. We used the data from June 11th and 16th as training data because it includes only normal data. After that, we used data from another day as test data. To run OCSVM, we call *OneClassSVM* and set the *nu* to 0.1, *gamma* to 0.5, another parameters to the default values. The *nu* is a parameter for an upper bound on the ratio of training errors and *gamma* is a kernel coefficient for the selected kernel function. We executed OCSVM with different parameters. The result with the *nu* equal to 0.1 and *gamma* equal to 0.5 was the best one.

Folino et al. [18] proposed an ensemble based algorithm by using a distributed genetic programming framework. They executed the drift detection strategy and several classifier algorithms based on the WEKA implementation by using the UNB ISCX 2012 dataset. However, they did not mention how to handle the dataset.

They chose Statistical Test of Equal Proportions (STEPD) as the drift detection strategy. They used CAGE-MetaCombiner (CMC) which uses many leaners as base classifiers, then it makes a selection using the old strategy which replaces 1/3 of classifiers considering the insertion time in the ensemble model. They partitioned the dataset into 50% for training and 50% for testing. The results of the CMC with the old strategy showed that the recall is 0.63 and the AUC is 0.53. Comparing this result with our result, we may conclude that the recall values are similar. In addition, the higher AUC value was obtained in our experiments. It means that our classifier may give better detection performance.

We also executed K-Means algorithm [25] and one class support vector machine (OCSVM) algorithm [26]. We selected GMM, K-Means algorithm, and OCSVM because these algorithms were used in studies [12], [27], [28]. The authors showed the promising results. In addition, we can apply the algorithms for real-time processing in the future. Table III shows the comparison of the obtained results. In the case of the K- means algorithm, very high accuracy was achieved. However, the outcome was very poor for detecting the attacks. This result demonstrates that K as a number of clusters must have an appropriate value: It should not be equal to 2 (attack and normal classes). In addition, selection of the initial cluster centers in the K-Means algorithm was very sensitive for the result. In the OCSVM, we could also detect the normal behavior with very high accuracy. However, TPR was low and FNR was very high. It was very difficult to determine the boundary between normal and attack classes. Therefore, in this study, we could get the best accuracy with low FPR by using PCA and GMM algorithms.

Gupta and Kulariya [29] presented the performance of classification algorithms for network intrusion detection systems by using Apache Spark. They utilized NSL-KDD dataset [30] that is CSV format. It is a dataset suggested to solve some of the inherent problems of the KDD'99 dataset which are mentioned in [31]. They implemented supervised leaning algorithms (Logistic regression, SVM, Random forest, Gradient Boosted Decision (GB) tree, and Naive Bayes) by using MLlib on Apache Spark. And also they implemented two feature selection algorithms and compared the performance results. Using the Random forest with correlation based feature selection resulted in the best accuracy, 82.35%. Table IV and V compare their results with our results. The sensitivity is the same as TPR and specificity is the same as TNR. The accuracy of the proposed system is better. The specificity parameter is also higher. The main differences between this work and study [29] are as follows:

- We analyzed the pcap file directly. Authors of that did not analyze the raw packet data and utilized analyzed dataset.
- We implemented unsupervised leaning algorithms. Study [29] focused on supervised learning algorithms.
- We utilized the 90.9 GB packet dataset created in 2012. Our opponents utilized an improvement version of KDD'99 dataset which is approximately 4 GB network traffic data created in 1999.

Table III
K-MEANS, OCSVM, AND GMM: EXPERIMENTAL RESULTS

Algorithms	Accuracy (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
GMM	86.1	47.0	87.0	13.0	53.0
K-Means	94.3	0.004	99.9	0.03	99.9
OCSVM	89.3	5.0	94.0	5.9	95.0

Table IV
PROPOSED SYSTEM VS. CORRELATION BASED FEATURE SELECTION METHOD

Evaluation parameters		Accuracy (%)	TPR (%)	TNR (%)
Our Results (PCA and GMM)		86.2	47.0	87.0
[29]	Logistic Regression	73.43	58.67	92.93
	SVM	37.89	65.01	2.05
	Naive Bayes	23.83	0.8	54.27
	Random Forest	82.35	72.72	95.07
	GB Tree	79.91	66.99	96.98

Table V
PROPOSED SYSTEM VS. HYPOTHESIS TESTING BASED FEATURE SELECTION METHOD

Evaluation parameters		Accuracy (%)	TPR (%)	TNR (%)
Our Results (PCA and GMM)		86.2	47.0	87.0
[29]	Logistic Regression	74.37	60.01	93.34
	SVM	40.51	58.45	16.87
	Naive Bayes	35.4	0.65	81.32
	Random Forest	80.96	69.28	96.4
	GB Tree	76.57	60.85	97.33

VI. Conclusion

In this paper, we proposed the anomaly based intrusion detection system. The system is able to manage the large-scale network packet analysis in a short period of time using Apache Hadoop and Spark. To test the system, we used the real IDS dataset provided by the Information Security Centre of Excellence at the University of New Brunswick. We handled the packet capture files (pcap) directly and analyzed 90.9 GB of pcap data on Hadoop clusters. We applied PCA for reducing feature dimension and utilized GMM to detect anomaly behaviors in the network packets.

We focused on designing the practical intelligent intrusion detection system with high accuracy and low false positive rate. The results showed that the approach is promising to develop the intelligent IDS. The detection accuracy is 86.2% and the false positive rate is 13%.

References

[1] The 2016 Internet Security Threat Report, Volume 21, APRIL 2016, https://www.symantec.com/security-center/threat-report.

[2] K. Kato and V. Klyuev, *An intelligent ddos attack detection system using packet analysis and support vector machine*, International Journal of Intelligent Computing Research (IJICR), Volume 5, Issue 3, pp. 464-471, September 2014.

[3] Wireshark, https://www.wireshark.org/.

[4] Apache™; Hadoop ®, http://hadoop.apache.org.

[5] Apache Hive™, http://hive.apache.org.

[6] Apache Spark™; http://spark.apache.org.

[7] W. Wang, R. Battiti, *Identifying intrusions in computer networks with principal Component analysis*, In: Proceedings of the First International Conference on Availability Reliability and Security, pp. 270-279, 2006.

[8] D. Reynolds, *Gaussian mixture models*, Encyclopedia of Biometric Recognition, Springer, 2008.

[9] A. L. Buczak and E. Guven, *A survey of data mining and machine learning methods for cyber security intrusion detection*, IEEE Communications Surveys and Tutorials, vol. 18, no. 2, pp. 1153-1176, 2016.

[10] Y. R. Bachupally, X. Yuan, and K. Roy, *Network security analysis using big data technology*, Proceeding of IEEE SoutheastCon 2016, pp. 1-4, 2016.

[11] S. Anwar, J. M. Zain, M. F. Zolkipli, Z. Inayat, A. N. Jabir, and J. B. Odili, *Response option for attacks detected by intrusion detection system*, 2015 4th International Conference on Software Engineering and Computer Systems (ICSECS), 2015.

[12] S. Zanero and S. M. Savaresi, *Unsupervised learning techniques for an intrusion detection system*, Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 412-419, 2004.

[13] M. H. Aghdam and P. Kabiri, *Feature selection for intrusion detection system using ant colony optimization*, International Journal of Network Security, 18(3): 420-432, 2016.

[14] KDD Cup 1999 Data: The third international knowledge discovery and data mining tools competition dataset, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[15] F. A. A. Alseiari and Z. Aung, *Real-time anomaly-based distributed intrusion detection systems for advanced metering infrastructure utilizing stream data mining*, 2015 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE), pp. 148-153, 2015.

[16] 2000 DARPA Intrusion Detection Evaluation Data Set; Massachusetts Institute of technology Lincoln Laboratory, http://www.ll.mit.edu/ideval/data/2000data.html.

[17] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux, *API design for machine learning software: experiences from the scikit-learn project*, ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108-122, 2013.

[18] G. Folino, F. S. Pisani, and P. Sabatino, *An incremental ensemble evolved by using genetic programming to efficiently detect drifts in cyber security datasets*, Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, pp. 1103-1110, 2016.

[19] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, *Toward developing a systematic approach to generate benchmark datasets for intrusion detection*, Journal Computers and Security, vol. 31, no. 3, pp. 357-374, 2012.

[20] UNB ISCX Intrusion Detection Evaluation Dataset, http://www.unb.ca/research/iscx/dataset/iscx-IDS-dataset.html.

[21] Large-scale PCAP Data Analysis Using Apache Hadoop, https://labs.ripe.net/Members/wnagele/large-scale-pcap-data-analysis-using-apache-hadoop.

[22] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, *Apache spark: a unified engine for big data processing*, Communications of the ACM, Volume 59, Issue 11, pp. 56-65, 2016.

[23] T. Fawcett, *An introduction to ROC analysis*, Pattern Recognition Letters, vol. 27, no. 8, pp. 861-874, 2006.

[24] Apache Aprk MLlib - Evaluation Metrics, https://spark.apache.org/docs/1.5.2/mllib-evaluation-metrics.html.

[25] MacQueen, *Some methods for classification and analysis of multivariate observations*, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, 1967, pp. 281-297.

[26] B.Schölkopf, J.C.Platt, J.C.Shawe-Taylor, A.J. Smola, R.C.Williamson, *Estimating the support of a high-dimensional distribution*, Neural computation, pp. 1443-1471, 2001.

[27] K.Yamanishi, J.Takeuchi, G.Williams, P.Milne, *On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms*, Data Mining and Knowledge Discovery, v. 8, n. 3, pp. 275-300, May 2004.

[28] M. Zhang, B. Xu, and J. Gong, *An anomaly detection model based on one-class svm to detect network intrusions*, 2015 11th International Conference on Mobile Ad-hoc and Sensor Networks, pp. 102-107, 2015.

[29] G.P. Gupta and M. Kulariya, *A Framework for Fast and Efficient Cyber Security Network Intrusion Detection using Apache Spark*, Proceedings of the 6th International Conference on Advances in Computing Communication, Procedia Computer Science, Volume 93, pp. 824-831, 2016.

[30] UNB ISCX NSL-KDD DataSet, http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html.

[31] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, *A Detailed Analysis of the KDD CUP 99 Data Set*, Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[32] B.Bahmani, B.Moseley, A.Vattani, R.Kumar, S.Vassilvitskii, *Scalable K-Means++*, Proceedings of the VLDB Endowment, v. 5, n. 7, p. 622-633, March 2012.

[33] R. Sommer, V. Paxson, *Outside the Closed World: On Using Machine Learning for Network Intrusion Detection*, Proceedings of the 2010 IEEE Symposium on Security and Privacy, pp. 305-316, 2010.

[34] J. Kyong, J. Jeon, and S. Lim, *Improving scalability of apache spark-based scale-up server through docker container-based partitioning*, Proceedings of the 6th International Conference on Software and Computer Applications, pp.176-180, 2017.