Anomaly Detection on Intrusion Detection System Using CLIQUE Partitioning

N. Nastaiinullah
Telkom University
Bandung 40257, Indonesia
n.nastain.92@gmail.com

Adiwijaya
Telkom University
Bandung 40257, Indonesia
adiwijaya@telkomuniversity.ac.id

A. P. Kurniati
Telkom University
Bandung 40257, Indonesia
angelina@telkomuniversity.ac.id

***Abstract*—The development of information and network technology makes network security become important. Intrusion is one of the issues in network security. To prevent intrusion happens, intrusion detection system (IDS) is built. One of IDS category is anomaly detection. This category detects intrusion event based on data profile. Clustering is one way to observe data profile. There are a lot of clustering algorithms proposed for anomaly detection on IDS, but most of them find clusters in the highest dimension of data. CLIQUE Partitioning (CP) is one of the clustering algorithm that can find clusters from the subspace of data. Testing is done to analyze system's performance based on computational time, completeness, and false alarm rate. CP algorithm shows good performance from completeness point of view (94.59%) and false alarm rate (2.54%). From computational time, CP shows good performance based on the amount of tuple, but the performance is not too good from the quantity of feature side.**

*Keywords—**anomaly detection; IDS; CLIQUE Partitioning; subspace; cluster***

I. INTRODUCTION

Intrusion is all the actions that threaten the availability, integrity and secrecy of network resource such as user account, file system, kernel system etc. Intrusion detection system (IDS) is a system that is used to observe and analyze an event that occurs on a computer whether that event is intrusion or not[4,7,15].

IDS can be divided into two categories : first misuse detection, second anomaly detection. Misuse detection detects intrusion event based on existing pattern. This pattern consists of event collections including intrusion. Anomaly detection detects intrusion event based on profile. Profile consists of events that usually user does (normal activity). If there is an event that is different from profile, this event will constitute intrusion candidate. Whether determination of event is intrusion or not, it depends on how big the deviation is. If the deviation is small, the event needs adding to profile as a new event. If its deviation is big, that event is intrusion [7,15].

Many researches had been conducted in anomaly detection on IDS, such as [2] by automated feature weighting using fuzzy subspace, [9] by using Y-means and N-CP clustering, [15] by using Support Vector Machine, and [13] by using Bayesian network. Most of them use clustering for detecting anomaly, but almost all the clusters found in the highest dimension of data. In this research, the method used for anomaly detection at IDS is CLIQUE (Clustering in Quest) partitioning (CP). The reason of CP use is its ability to find out cluster in all attribute combinations. It is related to basic idea from CP that there is possibility not all attributes are needed to find out cluster from dataset. CP constitutes combination of grid-based clustering technique and density-based clustering. Beside anomaly detection, CP has been used for another purpose like web usage [17].

II. DATA PREPROCESSING AND CLIQUE PARTITIONING

In this section, data preprocessing method and the clustering algorithm (CP) that is used in this anomaly detection system are explained. There are z-score normalization and Principal Component Analysis (PCA) for data preprocessing and CLIQUE Partitioning for clustering algorithm. Z-score is one of the ways of data normalization that is generally used. The advantage of z-score is its use when maximum value or minimum data is no known and it can reduce the domination from a value at the data. Given A as an attribute from dataset, the new value of v (v is an old value from a value in attribute A) can be determined using

$$v' = \frac{v - \Delta}{\sigma_A} \quad (1)$$

where Δ is the average from attribute value A, v' is a new value from v after z-score has been done, and σ_A is a deviation standard from attribute A.

PCA is one of the ways to do the reduction of attribute. The advantage of PCA is able to reduce attribute from the data by defending a lot of variations in the original data. PCA reduces attribute by counting eigenvalues from each attribute. There are four steps to find out eigenvalues from a particular attribute. First, count the average of value in that attribute, example: $[x_1, x_2, \ldots, x_n]$ is the value in attribute A and $\bar{A}$ is the average from that value, so $\bar{A}$ can be calculated by $\bar{A} = [(\sum_{1 \le i \le n} X_i)/n]$. Second, reduce every datum with the average value. For example, A is matrix n x m where n is the data amount and m is the attribute amount that consists of new value from A1 until Am after passing the second step, so the third step is to find out matrix covarian and A' with a pattern $C = (A'. A'^{T}) / n$. The last step is to find out eigenvalues from each attribute. After eigenvalues from each attribute got, each attribute is arranged based on the biggest eigenvalues.

CLIQUE Partitioning (CP) is one of clustering techniques from description type task on data mining. Generally, CP is

978-1-4799-3580-2/14/$31.00 ©2014 IEEE

projecting *dataspace* into some subspaces, and divide each subspace into *e* unit (*e* is an input parameter). Then, CP is looking for and finding the subspace with the dense unit based on *t* (*t* is the threshold which decide certain unit that could be dense or parse, *t* is an input parameter). Thus, the dense and connected units are collected to be cluster.

Subspace is part of *dataspace*. For example, certain dataset has n attribute, subspace can be formed from the combination of the n attribute in each dimension level. In 1 dimension subspace, the total of its subspace is C(n,1). In 2 dimension subspace, the total of its subspace is C(n,2) and so on and so forth until the highest dimension subspace which is included into the *dataspace*, its total of subspace is C(n,n).

Every attribute in subspace will be divided into several grids. Unit is the little part among all attributes in subspace. Subspace in n dimension which was divided into e unit has e^n unit.

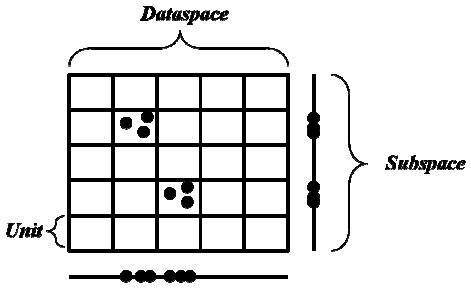

Figure 1: *Dataspace, Subspace,* and*Unit*

CP consist of 3 steps, those are; identification of subspace that contain dense units, identification of cluster from the dense units, and generating minimal description for the cluster.

a. Identification of Subspace that Contain Dense Units

The first step is dividing each 1 dimension subspace into certain amount of units based on input parameter, so its range and status from each unit are defined. The range of unit is defined as [x,y), x is the minimum limit of unit, while y is maximum limit of unit. [x,y) is right-open interval ($x \leq data < y$). The status of each unit is dense or parse based on the amount of data in the unit. If the data fulfill the threshold, then its status is dense, otherwise parse.

Next, a subspace, which 1 dimension higher from dense units in prior subspace, is built. The status of higher subspace is also checked, because it is not guaranteed that certain unit in certain subspace will n-dimension will be in dense status in subspace (n+1)-dimension. The step is done repeatedly until no one new generated subspace can be found (all units in new subspace are not having dense status).

The subspaces will always be generated until there is no new subspace can be generated anymore. If the data consist of many attributes, for example 20, so there will be many subspaces that need to be checked in clustering labeling in order the pruning in subspace is conducted. One type of pruning is *Minimal Description Length/ MDL-based Pruning* (MDL).

There are 6 processes in MDL, those are:

1. Counting the coverage of formed subspace. The coverage is the total of dots in dense units in subspace. The coverage of j subspace is symbolized *XSj.*
2. Arranging subspace based on its coverage from the highest up to the lowest (descending).
3. The collected subspace will be divided into 2; those are pointed subspace (I) and reduced subspace (P). The problem is finding the most optimum *cutpoint*. To find it, count the code length (CL) from all possible *cutpoint*.
4. To find the CL, first, count the coverage average from I and P for all possible *cutpoint*. For example, the amount of subspace is *n,* the pointed *cutpoint* is *i,* the coverage average from I is $\mu_I(i)$, the coverage average of from P is $\mu_P(i)$, so the value of $\mu_I(i)$ is $\mu I(i) = \left[\left(\sum_{1\leq j\leq i} Xs_j\right)/i\right]$ and the value of $\mu_P(i)$ is $\mu P(i) = \left[\left(\sum_{i+1\leq j\leq n} Xs_j\right)/(n-i)\right]$.
5. After getting the average of each I and P, so CL is: $CL(i) = \log_2(\mu I(i)) + \sum_{1\leq j\leq i} \log_2(|Xs_j - \mu I(i)|) + \log_2(\mu P(i)) + \sum_{i+1\leq j\leq n} \log_2(|Xs_j - \mu P(i)|)$
6. After getting the CL from each *cutpoint*, choose the *cutpoint* with the lowest CL as the optimum *cutpoint*.

b. Identification of Clusters

The input of this step is the dense units of each subspace. After getting the dense unit, then the cluster can be done through the units. Cluster is the combination of dense units that close to each other in one subspace.

c. Generating Cluster's Minimal Description

The minimum description of each cluster can be identified after clustering its identification. It can be done through identifying the size of rectangle as maximum as possible and the amount of rectangle as minimum as possible in cluster. Rectangle is the shape (whether it is square or rectangle) that can be defined from its cluster form.

In Figure 2, the grey colored area is called cluster. There are many rectangles that can be defined, for example the rectangle of ($300 \leq A < 400$) ^ ($5 \leq B < 10$). However, the optimum form of rectangle to define the cluster is having the maximum size and minimum amount. There are two rectangles with maximum size, those are ($200 \leq A < 400$) ^ ($5 \leq B < 15$) and rectangle ($100 \leq A < 300$) ^ ($10 \leq B < 20$).

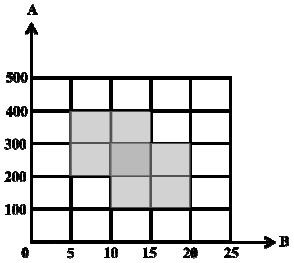

Figure 2: Rectangle in Cluster

That is why, the description of minimum cluster based on the picture is ((200 ≤ A < 400) ^ (5 ≤ B < 15)) V ((100 ≤ A < 300) ^ (10 V B < 20)).

III. System Design

In this part, the performance of system that consists of data processing, clustering process, cluster labeling and performance measuring will be explained. The data KDD Cup 1999 that is the subset of DARPA Intrusion Detection Evaluation Dataset launched in 1998 by MIT Lincoln Library is used [2].

The dataset is retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. It has 4.898.431 tuples and 41 attributes (without class). From the 41 attributes, 7 attributes are with symbolic type, and the rest is numeric. There are 23 classes in dataset, that is one normal class and 22 intrusion classes which can be grouped into 4 intrusion types.

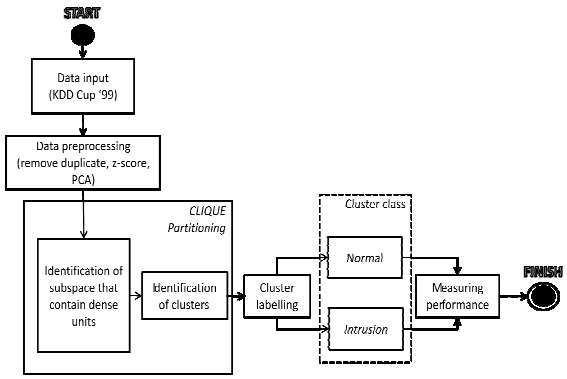

Figure 3: Flow System Diagram

1) Data preprocessing

Preprocessing was conducted on the data cleaning, the data transformation, and data reduction. The result of the data cleaning is a unique dataset (different from each other). From this process, the number of datasets was successfully reduced as much as 3.902.068 tuple, from 4.894.431 tuple into 996.363 tuple by removing redundant data.

Transformation data includes attributes conversion of symbolic into numeric and data normalization. Attributes conversion of symbolic into numeric is conducted in a simple way. In a symbolic attributes which only has 2 categories, the conversion is conducted by boolean conversion (0 or 1). In other symbolic attributes, the conversion is conducted by looking for many categories, then each number from 1 to the number of categories replaces exactly one symbolic category, for instance A to 1, B to 2, and so on. In normalization, the algorithm used is the z-score normalization. Data reduction is a reduction in the number of attributes using PCA algorithm.

2) Clustering process

Clustering process is conducted by using CP algorithm. Data input is data that has been passed through preprocessing. First, this data mapped into 1-dimensional *subspaces* and partitioned into units. From those units that qualified as dense, the 2-dimensional *subspaces* are generated, and so on. The output is all contained clusters on all subspaces.

3) Cluster labelling

To determine whether the label of a cluster is intrusion or not is by calculating the percentage of data types which has larger clusters based on the number of data of each type as a whole.

4) Performance measuring

System performances is determined based on three aspects, they are computational time, completeness (CR), and false alarm rate (FAR). Computational time can be seen from the comparison of a running time towards the number of data input. Completeness can be calculated from the number of intrusion which is detected by the system than the number of actual intrusion. False alarm rate can be calculated from the number of normal data which detected incorrectly by the system as a data intrusion than the actual amount of normal data.

IV. Performance Evaluation

The tests carried out to analyze the data and parameters contained in the system to produce an anomaly detection system using algorithm CP with good performance. Determining system performance includes computational time, completeness, and false alarm rate.

1. The influence of the amount of tuple and attribute to computational time

In this experiment, the influence of the amount of tuple and attribute to computational time was examined. The constant parameter value is t = (0.05*(amount of tuple)), e = 100, and MDLuse = 0. For the amount of tuple analysis, there were 9 experiments with different amount of tuples, but having the similar amount of attribute which was 7. The tuples in this analysis were selected randomly. For the amount of attribute analysis, there were 8 experiments with different amount of attributes, but having the similar tuple which was 993.636.

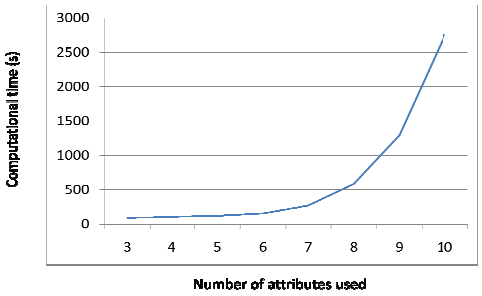

Figure 4: The influence of the number of attributes used to computational time

In Figure 4, there was an increase in computational time to the amount of attribute used which was exponential. This occurred because there were more subspace and dense unit generated as the effect of the amount of subspace combination

that could be formed. For example, *dataspace* was divided into 10 units and all units had dense status. If the amount of attribute used is 2, then there will probably be three subspaces. In 1 dimension subspace, there are 2 subspaces, and each of which consists of 10 units. In 2 dimension subspace, there is 1 subspace, which has 10*10 = 100 units. It means that there were 120 units examined. If the amount of attribute used is 1, then there will be 1 subspace. In 1 dimension subspace, there will be 1 subspace and each of which has 10 units. It means that there were 10 units examined. It was only 1 dimension difference, but the amount of units examined was bigger.

In Figure 5, the influence of the amount of tuple to computational time was inconsistent, sometimes it went up and sometimes it went down. The inconsistency shown in the graphic was influenced by random tuple selection in dataset.

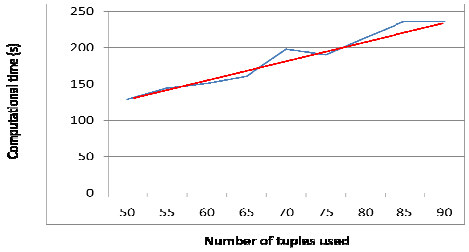

Figure 5: The influence of the amount of Tuple to Computational Time

However, the change of computational increase tended to be linier (it can be seen from red line [*trendline*]). This occurred because the amount of tuple did not affect the amount of subspace combination that could be formed. Larger amount of tuple could indeed affect the amount of dense unit in a subspace because dataset is more varied, which means more dense units to be examined. However, the increase of the amount of the examined dense unit is much smaller than the amount of dense unit that must be examined if dataset attribute is added.

Based on the result of experiment and analysis discussed previously, it can be concluded that the amount of computational time is linier to the amount of tuple, but is exponential to the amount of attribute. This is due to the increase of even 1 dimension means more units to be examined so that computational time could get larger significantly.

2. The influence of MDL use to system performance

In this experiment, the goal to be achieved is examining the influence of MDL use to system performance. During the experiment, the constant parameter value is t = (0.05 * (amount of tuple)) and e = 100. The experiment was conducted 5 times with different amount of attributes, but having the similar amount of tuple which is 993.636 tuples.

Table 1. Performance Comparison based on MDL Use

Number of attributes used	With MDL			Without MDL		
	Computational time (s)	C (%)	FAR (%)	Computational time *(s)*	C (%)	FAR (%)
6	178.7	100	99.9	154.6	98.6	91.2
7	259.8	100	100	270.9	94.6	2.5
8	586.8	99.9	97.5	586	94.5	1.7
9	1270.7	99.9	98.9	1294.3	94.5	1.6
10	2743.7	99.8	98.9	2752.1	94.5	1.5

Based on Table 1, it can be seen that completeness (C) with MDL will have higher value than that without it, but false alarm rate (*FAR*) is much better without MDL. In the clustering result with MDL, almost all data was considered intrusion. This is due to the normal cluster data lost because they were cut during MDL. The characteristic of MDL is cutting subspace that has low coverage, but that subspace could be important because there is cluster that can determine accurate data class in it. In terms of computational time / running time, the use of MDL does not necessarily reduce computational time because more time is needed to execute MDL. If there are a lot of subspaces cut, then computational time could probably be faster, but if there are few subspaces cut, then the time spent could be more than that without MDL since the amount of the cut execution time is smaller than the execution time of MDL.

Based on the result of experiment and analysis discussed, it can be concluded that the use of MDL is not effective since it cuts many important subspaces which have decisive cluster to clustering result. In terms of computational time, the use of MDL does not always result in faster computational time because the subspaces cut might be few or none at all.

3. The influence of t parameter to system performance

In this experiment, the influence of t parameter to system performance was observed. Dataset used is that with 993.636 tuple and 7 attributes. The constant parameters are e = 100 and *MDLuse* = 0.

Table 2. The influence of t parameter to performance

t (% data)	C (%)	FAR (%)	Computational time (s)	Number of dense unit	Number of cluster
0.05	94.58	2.54	273.6	319	255
0.075	94.59	2.73	197.6	191	191
0.1	94.59	2.73	197.9	191	191
0.125	94.59	2.73	175.8	127	127
0.15	94.59	2.73	177.8	127	127
0.175	94.59	2.73	173.9	127	127
0.2	94.59	2.73	178.7	127	127
0.225	94.59	2.73	175.7	127	127
0.25	94.59	2.73	172.5	127	127
0.275	94.59	2.73	168.6	127	127

Based on Table 2, it can be seen that *C* and *FAR* increased. This is due to the higher t is, the less the dense unit is. The less the dense unit is means the fewer the clusters are formed. The fewer the formed clusters, the more data are considered outlier. Outlier data are considered as intrusion so that there are a lot of data assumed as intrusion while it can

happen where the data assumed as intrusion are actually normal data. This causes *C* and *FAR* to get larger value.

Figure 6 shows the decrease of dense unit is linear to the decrease of computational-time.

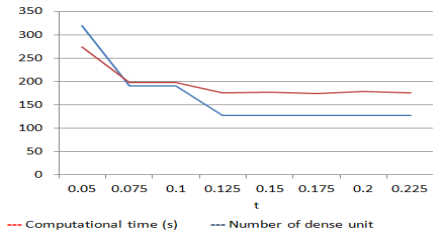

Figure 6: Comparison between the number of Dense Unit with Computational Time according to *t*

This is caused by the less amount of dense unit that must be examined by system so that less time is needed to examine dense unit. Based on the result of experiment and analysis, it can be concluded that the bigger the value of *t* is, the bigger the value of *C* and *FAR* will be. The bigger the value of *t* is, the less the computational time is required.

4. The influence of *e* parameter to system performance

In this experiment, the influence of *t* parameter to system performance was observed. Dataset used was that of 993.636 tupleand 7 attribute. The constant parameters are *t* = (0.05 * (the amount of tuple)) and *MDLuse* = 0.

Table 3.The influence of *e* parameter to performance

e	C	FAR	Computational time (s)	Number of dense unit	Number of cluster
10	98.43	78.55	201	255	191
20	98.34	61.90	219.5	255	191
30	98.34	60.88	236.3	319	191
40	94.55	2.36	216.3	255	255
50	94.57	2.4	211.8	255	255
60	94.57	2.4	241.2	319	255
70	94.57	2.4	215.6	255	255
80	94.57	2.4	254.7	319	255
90	94.57	2.4	253.1	319	255
100	94.58	2.54	258.8	319	255
110	94.58	2.54	265	319	255
120	94.58	2.54	258.9	319	255
130	94.58	2.54	259.3	319	319
140	94.58	2.54	277.8	319	255
150	94.58	2.54	274.4	319	255

Based on Table 3, it can be seen that if the value of *e* is getting bigger, then the value of *C* and *FAR* will decrease, but the decrease of *FAR* is far more significant. This is caused by the bigger the value of *e* is, the higher is the accuracy of cluster formation due to the smaller unit size that results in better value of C and FAR. However, when *e* = 100 and so forth, the value of *C* and *FAR* starts to increase. This shows that if *e* is too high, then the accuracy level will be too high (unit size will bee too small) so that the number of points in the units is too small. This resulted in the decreasing amount of clusters and the more points that are not included as clustering and considered as outlier that causes increase in the value of C and FAR.

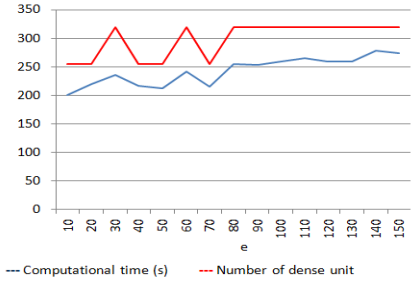

Figure 7: Comparison between the amount of Dense Unit with Computational Time according to *e* Parameter

In Figure 7, it can be seen that the change of dense unit amount is almost linear to the change of computational time. This is caused by the less dense unit that must be examined by system so that less time is required to examine dense unit.

Based on the result of experiment and analysis discussed, it can be concluded that the bigger the value of *e* is, the less is the value of *C* and *FAR*. However, when *e* is in certain value, then the value of *C* and *FAR* will increase. The value of *e* affects computational time depending on data distribution pattern because the dense units generated are different.

6. Performance Comparison with Previous Research

The comparison between completeness and FAR value from CP algorithm of this paper with the result of previous research is illustrated as follows:

Table 4.Anomaly Detection Performance in IDS

No	Method	CR (%)	FAR (%)
1.	*z-normalization + N-CP clustering* [9]	86,63	2,72
2.	*Fuzzy c-means + k-means vector clustering* [2]	96,5	2
3.	SSV + SCM [8]	91,92	0,061
4.	*z-normalization + unsupervised clustering* [5]	88	8,14
5.	*PCA + CLIQUE partitioning* (Proposed scheme)	94,59	2,54

Based on Table 4, it can be seen that anomaly detection scheme used in this journal is able to produce better completeness and false alarm rate value than that of previous research.

V. CONCLUSION

Based on the analysis result of objective and subjective measurement of anomaly detection system in IDS using CP, the system constructed can be implemented by better performance, namely CR 94.59% and FAR 2.54%. CLIQUE Partitioning Algorithm can result in good computational time in terms of the amount of tuple (the increase of tuple amount is linear with the increase of computational time), but it is not good in terms of the amount of attribute (the increase of tuple

amount is exponential with the increase of computational time).

Based on threshold of the amount of data in dense unit and subspace distribution scheme, CLIQUE Partitioning Algorithm could result in good computational time because the amount of computational time is linear with the amount of dense unit formed. Based on the use of MDL, the amount of computational time depends on the amount of reduction by MDL. If there are a lot of reductions, then computational time value will apparently get smaller. However, if there are only few reductions or none at all, then computational time value will be bigger.

CLIQUE performance also depends on its initial state, hence it has high probability that the solution generated is a local optimum solution. Learning ability, such as Genetic Algorithm, Fuzzy Logic, and Particle Swarm Optimization, can be added to CLIQUE to ensure the solution generated is a global optimum solution.

REFERENCES

[1] Agrawal, Rakesh et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. San Jose, IBM Almaden Research Center, 1998

[2] Dat Tran Wanli Maa Sharma, D. *Automated Network Feature Weighting-based Intrusion Detection Systems*. IEEE, ISBN 978-1-4244-2172-5, INSPEC Accession Number 10425608, DOI: 10.1109/SYSOSE.2008.4724144, 2008

[3] Bethi et al. *CLIQUE Clustering Approach to Detect Denial-of-Service Attacks*. IEEE, ISBN 0-7803-8572-1, INSPEC Accession Number 8435285, DOI: 10.1109/IAW.2004.1437856, 2005

[4] Debar, Hervé. An Introduction to Intrusion-Detection Systems. Jurnal CiteSeerX, 2002.

[5] Eskin, Eleazar et al. *Intrusion Detection with Unlabeled Data Using Clustering*. New York, Department of Computer Science Columbia University, 2001

[6] Finley, Thomas dan Thorsten Joachims. *Supervised Clustering with Support Vector Machines*. CiteSeerX, 2005.

[7] Han, Jiawei and Michelin Kamber. *Data Mining : Concepts and Techniques Second Edition*. Morgan Kauffmann Publishers, 2006

[8] Hernández- Pereira, E. et al. *Conversion Methods for Symbolic Features: A Comparison Applied to an Intrusion Detection Problem*. A Coruña, Laboratory for Research and Development in Artificial Intelligence (LIDIA) Computer Science Department University of A Coruña, 2009

[9] Kannan, Sivanadiyan Sabari. Y-Means Clustering Vs N-CP Clustering With Canopies for Intrusion Detection. Chennai, Madras University, 2005

[10] Kuchimanchi, Gopi K. *Dimension Reduction Using Feature Extraction Methods for Real-time Misuse Detection Systems*. New York, Workshop on Information Assurance and Security United States Military Academy, 2004

[11] Lakhina, Shilpa et al. Feature Reduction using Principal Component Analysis for Effective Anomaly-Based Intrusion Detection on NSL-KDD. Bhopal, Technocrats Institute of Technology, 2010

[12] Leung, Kingsly and Christopher Leckie. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. CiteSeerX, 2005.

[13] O.A. Marlita, Adiwijaya, A.P. Kurniati. *Anomaly Detection* pada *Intrusion Detection System* (IDS) menggunakan Metode *Bayesian Network*. Jurnal Penelitian dan Pengembangan Telekomunikasi 17 : 1 pp. 53-61, 2012.

[14] Tatsumi, Keiji et al. *Multiobjective Multiclass Soft-Margin Support Vector Machine Maximizing Pair-Wise Interclass Margins*. SpringerLink, Volume 5506/2009, 970-977,DOI:10.1007/978-3-642-02490-0_118, 2009.

[15] Yao, JingTao et al. *An Enhanced Support Vector Machine Model for Intrusion Detection*. SpringerLink, Volume 4062/2006, 538-543, DOI:10.1007/11795131_78, 2006

[16] Zhang, Yong et al. *A Novel Fuzzy Compensation Multi-class Support Vector Machine*. SpringerLink, Volume 27, Number 1, 21-28, DOI:10.1007/s10489-006-0027-x, 2007

[17] Santhisree, K and Damodaram, A. *CLIQUE : Clustering based on density on web usage data : Experiment and test results.* IEEE, ISBN 978-1-4244-8679-3, INSPEC Accession Number 12096647, DOI: 10.1109/ICECTECH.2011.5941893, 2011