ICIC Express Letters
Part B: Applications
Volume **9**, Number **3**, March **2018**

ICIC International ©2018 ISSN 2185-2766
pp. **257–264**

DETECTING MALICIOUS ACTIVITIES IN A COMPUTER CLUSTER FOR DEVELOPING DYNAMIC HONEYPOT

Tohari Ahmad, Thiar Hasbiya, Royyana Muslim Ijtihadie and Waskitho Wibisono

Department of Informatics
Institut Teknologi Sepuluh Nopember
Jalan Raya ITS, Sukolilo, Surabaya 60111, Indonesia
tohari@if.its.ac.id

Received October 2017; accepted December 2017

Abstract. *The need of securing computer networks in this digital era has remarkably raised. This is because advance in the information technology affects the risk of the networks. Moreover, highly confidential data are likely stored in a database server within the network. In order to mitigate possible security problems, we propose a mechanism which is able to detect a possible network attack. Next, honeypot is deployed according to this detecting result. It is by considering both internal and external factors, including the number of receiving packets, source machine being used for sending those packets, and dead processes. The experimental results which are taken by using a real environment show that the proposed method has been able to achieve better performance than existing algorithms.*
Keywords: Computer attack, Data security, Honeypot, Information security, Network security

1. **Introduction.** Information technology, especially computer networks, has been developing for some decades, from small to large scales. This means that the number of available devices, such as smart phone and laptop, significantly increases which leads to also raising the number of possible applications. In 2020, it is predicted that the number of smart phone users itself is about 2.87 billion [1]; this number excludes that of PC or laptop users. It is very likely that a user has more than one device, which is connected to the network. Consequently, there are many applications that run on the network which actually is not realized by users.

All those devices potentially access web applications through the Internet. Due to its popularity, this number of accesses may reach thousands at the same time. Therefore, a mechanism which can be used to manage this heavy and spike traffic is needed [2]. To overcome this issue, the use of computer clusters can be considered. Furthermore, a computer cluster is suitable to use for fulfilling availability and reliability requirements because in a cluster, there are some machines which support each other. In a cluster, moreover, the traffic is distributed among the computers in it to meet the fairness factor. In many cases, this needs load balancer to manage the traffic between machines [3, 4].

The use of computer clusters, however, requires the user to provide many machines, which may drive malicious activities resulting in computer attacks. In this case, the attackers possibly do not care about which machine they are attacking since it is done randomly by targeting any computer in the respective cluster. In addition, this attack may be launched from a botnet, a machine which has been comprised [5, 6]. In the worst case, this attack is distributed among botnets whose purpose is to shut down either the service or the machine of the target itself [7].

In order to mitigate this problem, we may use honeypot which can be implemented for trapping an attack. Moreover, honeypot can also be used to detect malicious packets or counteract an attack [8]. In consequence of its characteristics, honeypot can camouflage its machine with that of real systems but be isolated from them. This environment distracts the attackers from the real target. In a computer cluster, however, we cannot develop honeypot to all machines; also, it is not easy to determine a machine to be honeypot.

In this paper, we anticipate an attack to a machine within a computer cluster, so that the system is able to recognize whether the receiving packets are an attack or not. Next, some responses can be given to counteract the attacks, such as converting the attacked machine to honeypot, once it is detected harmful. Furthermore, in this proposed method we utilize both internal and external factors of computer and network environments to make such decision. It is worth noting that this algorithm may be combined with other detecting systems according to the respective requirements.

Even though in a specific evaluation it has a lower sensitivity level, in general the proposed method delivers better performances than some existing algorithms. This is applied for accuracy, specificity and precision levels.

This proposed research is structured as follows. Section 2 describes some research which corresponds to this proposed method. Section 3 explains the method being proposed. Section 4 shows the analysis of the experimental results and finally, the conclusion is drawn in Section 5.

2. **Computer Cluster Attack.**

Classifying packets being received by a computer in a cluster can be performed in many ways. This classification system decides whether the incoming packets are a sequence of attacks or just normal data. Some popular methods include machine learning [9, 10] such as artificial neural network (ANN), support vector machine (SVM) and decision tree. ANN processes an input by imitating human neural network whose potential limitations have been overcome by SVM, while decision tree develops rules according to the available data for taking a decision.

In a cluster, a machine works along with others to do computation. This has made it more powerful to perform heavy jobs. Nevertheless, this may also deliver more vulnerabilities to the system if the network administrator cannot maintain their performance and security. Maintaining clusters itself, however, may take much time and high cost. On the other hand, a lack of security in a certain machine causes the attacker easily exploits it by performing a horizontal scanning attack or subnet scanning, as illustrated in Figure 1.

As shown in Figure 1, horizontal scan is done by scanning certain port numbers in some hosts [11]. In this scanning, an attacker does not need to know the exact address

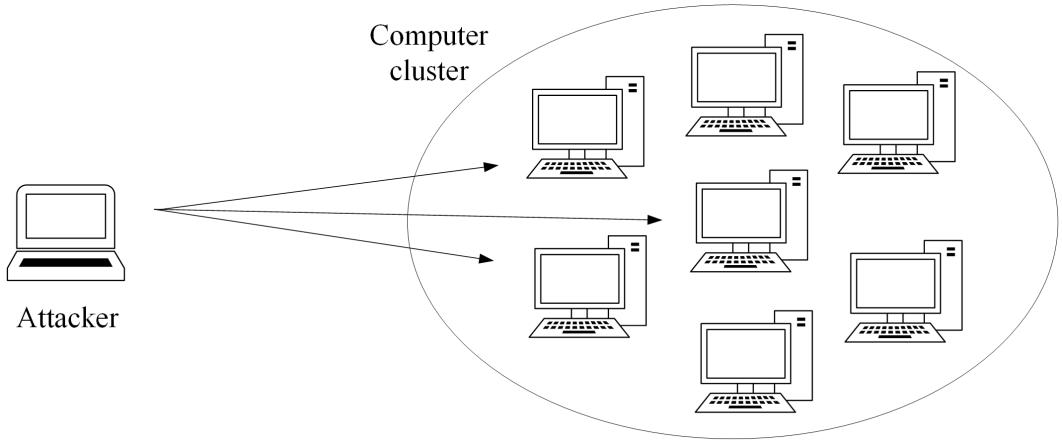

FIGURE 1. Horizontal scanning attack

of the machines in the cluster. He/she just does it randomly to machines in the public subnets. Furthermore, more computers can be compromised leading to higher possibility to successfully attack the real target. Next, the result of this activity is used for finding vulnerabilities and exploiting the target. For example, the compromised machines can be employed as botnets to launch a distributed denial of service (DDOS) attack [12, 13].

In the DDOS implementation, many botnets simultaneously send a huge packet to the target. This results in a high traffic and load to the respective machine, which lead to its inability to process the jobs. Consequently, either the service, the machine itself or both is shut down. In this case, the attacker has successfully taken over both the intermediate and final machines.

Recently, low level DDOS attacks, such as SYN flooding and malicious packets, may not work well because advanced tools are able to detect it. Instead of sending malicious packets in the transport layer, an attacker may employ real packets in the application layer. For example, the attacker uses malicious URL request in HTTP to exploit the vulnerabilities of a web server, and carries out a brute force attack to FTP application protocol in other networks.

3. **Detecting Malicious Activities.** In this paper, we propose a mechanism to detect an attack and deploy honeypot according to the previous detection result. That is, if the incoming packets cause a machine in the cluster harmful, then the system converts this attacked machine to honeypot, and the respective services are moved to another safer machine, as depicted in Figure 2. Practically, this proposed method can be simultaneously used with existing intrusion detection systems (IDS), such as [14].

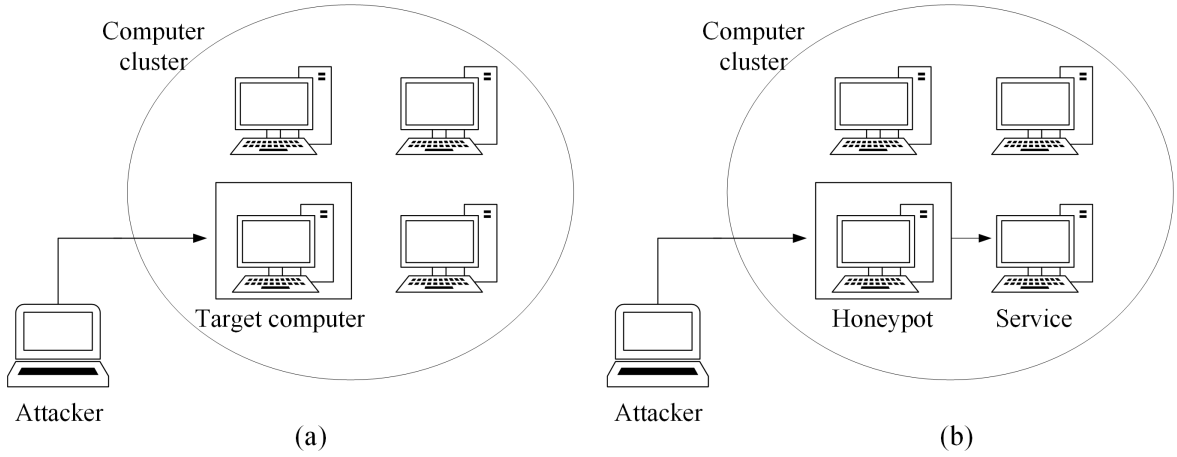

FIGURE 2. An example of attack models: (a) attack to a computer in the cluster; (b) the attacked computer changes to honeypot

It is shown that an attacker which had carried out a horizontal scanning attack (see Figure 1) found a machine to be a target. Next, he/she sends malicious packets to it. Once receiving these packets, the respective computer should decide whether the machine is being attacked or not by calculating a certain security value. In case it exceeds the specified threshold, it can be inferred that it is harmful, honeypot should be deployed in this computer, and the service is moved to another computer.

3.1. **Security factors.** In deciding whether a machine is harmful, we consider some factors.

- Amount of packets which has been received for a period of time. It is likely that an attacker is sending a huge number of packets as in a DDOS attack.

- Level of CPU usage while a series of packets are received. Because of many requests, CPU is busy to handle them, which results in overload.
- Number of dead processes, which represents the amount of uninterrupted processes while running services. This dead process is likely to happen because of significant CPU usage.
- Number of sources where the incoming packets are initiated. We consider that more sources increase the possibility of attacks. This is because, even though not always, a DDOS attack is possibly running at that time.
- Total packets which have been sent by a user and may not be a significant factor of attacks, unless they are sent concurrently.
- Number of CPU core on the targeted machine. It is assumed that higher number of cores may increase the ability of the machine to defend against an attack, which means less harmful.

It is depicted that different from common classification attack methods, we include an internal factor of the target machine: CPU usage and number of CPU cores. This is because the capability of CPU determines whether high computation can be handled or not.

3.2. **Detection process.** Overall, the steps to detect this malicious activity are provided in Figure 3, and an example of parameters used for detection is presented in Table 1. At the beginning, real data are collected by running the scenario. That is, packet data are sent for at least 1 minute as the minimum time of an attacking tool to perform a sequent attack. In the receiving side, the computer collects some data: CPU usage, number of dead processes, attacks and attackers. These data have various ranges whose gap is relatively high. Therefore, these data are normalized such that their value is between 0 and 1. In this case, normalization is carried out by dividing the collected data by its respective maximum value, which has been specified in the beginning (this maximum value is provided in Table 2). From these parameters, we determine values for the next process. These values are as follows.

- The number of concurrent sending packets (C), which can be presented in (1).

$$C = \frac{a * b}{\max(a) * \max(b)} \tag{1}$$

- Normalized number of dead processes (D), as specified in (2).

$$D = \frac{e}{\max(e)} \tag{2}$$

- Normalized number of sending packets (P), as defined in (3).

$$P = \frac{c}{\max(c)} \tag{3}$$

- Normalized number of CPU usages (U) as in (4).

$$U = \frac{f}{\max(f)} \tag{4}$$

By assuming that overloaded packets which are received by a machine increase the CPU usage and number of dead processes, we define 2 categories of running computers in a cluster: harmless and harmful which are denoted by 0 and 1, respectively. An example of these normalized data is depicted in Table 3.

In order to determine whether the computer in a cluster is harmless or harmful, we propose a security level θ whose value is specified in (5), where α, β, γ, δ are the weights

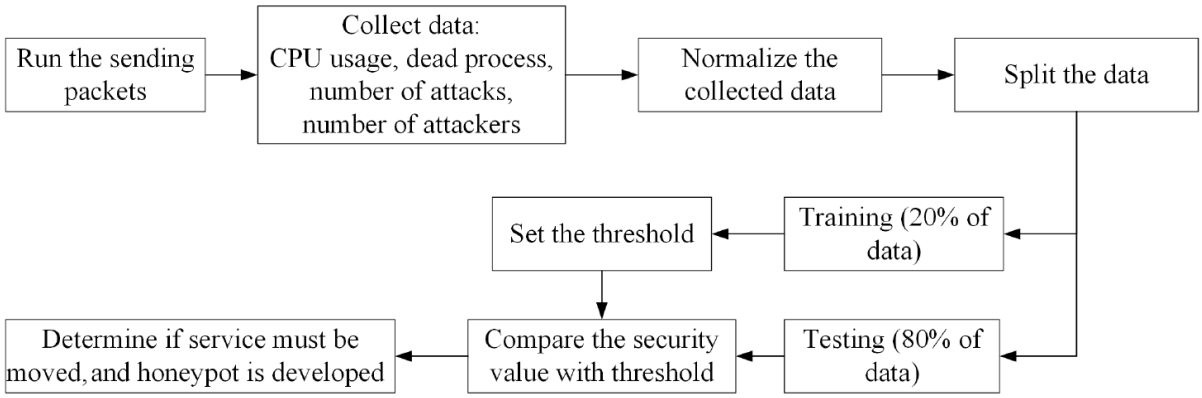

FIGURE 3. Detecting method

TABLE 1. Examples of collected data

Number of attackers	Concurrent attacks per attacker	Total number of sent packets	CPU core	Number of dead processes	CPU usage
1	16	200	1	0	4.35
2	28	400	1	1	11.27
3	22	1000	1	13	9.86

TABLE 2. Maximum value of parameters

Notation	Factor	Maximum value
a	Number of attackers	10
b	Concurrent attacks per attacker	64
c	Total number of sent packets	1000
d	CPU core	4
e	Number of dead processes	250
f	CPU usage	100

TABLE 3. Examples of normalized data

Concurrent attacks (C)	Dead process (D)	Packets (P)	CPU usage (U)	Label
0.025	0.064	0.2	0.0435	0 (harmless)
0.0875	0.112	0.4	0.1127	1 (harmful)
0.103125	0.088	1	0.0986	0 (harmless)

of C, D, P, U respectively. The status of the corresponding machine, whether it is harmless or harmful, depends on the specified threshold τ (see (6)).

$$\theta = (\alpha * C) + (\beta * D) + (\gamma * P) + (\delta * U) \tag{5}$$

$$\text{status} = \begin{cases} \text{harmless}, & \text{if } \theta \leq \tau \\ \text{harmful}, & \text{if } \theta > \tau \end{cases} \tag{6}$$

3.3. **Security threshold.** As what has been described before, the collected data are split into two groups which are for training and testing whose proportion is 20% and 80% of data, respectively. Each of this group has similar proportion to the whole data, in terms of the value of each factor. Next, the records in the training data are further classified to either harmless or harmful according to those values. Furthermore, the average values

TABLE 4. Difference value between harmless and harmful of training data

Criterion	Harmless $\overline{X}_N(L)$	Harmful $\overline{X}_N(F)$	Difference Δ_N
C	0.1918	0.5015333	0.3097333
D	0.07904	0.1779	0.09886
P	0.5711	0.6609	0.0898
U	0.06056	0.2627	0.20214

of those groups are provided in Table 4. Here, Δ_N represents the difference between the average values of criterion N belonging to harmless (L) and harmful (F) groups whose value is specified in (7).

$$\Delta_N = \left|\overline{X}_N(L) - \overline{X}_N(F)\right| \tag{7}$$

By using the values in Table 4, we calculate the constants $\alpha, \beta, \gamma, \delta$, as specified in (8). It can be found that the value of those constants depends on its proportion to the whole difference value. It is shown that α has the highest value, and has the most influence to the value of θ. From this, we have a formula to calculate the security level of a machine which is used for detecting its harmfulness, as presented in (9).

$$\left.\begin{aligned} \alpha &= \tfrac{\Delta_C}{\Sigma\Delta_N} \\ \beta &= \tfrac{\Delta_D}{\Sigma\Delta_N} \\ \gamma &= \tfrac{\Delta_P}{\Sigma\Delta_N} \\ \delta &= \tfrac{\Delta_U}{\Sigma\Delta_N} \end{aligned}\right\} \tag{8}$$

$$\theta = (0.44 * C) + (0.14 * D) + (0.12 * P) + (0.3 * U) \tag{9}$$

4. **Experimental Results.** As in the previous description, the experiment is carried out by using 80% of the collected data. The Hydra tools are used for simulating a brute force attack on FTP protocol, as in [15]. The respective data set is generated by constructing a system comprising 10 computers to attack a targeting machine by using vsftpd FTP server. The number of attackers, concurrent attacks per attacker, total number of attacks and CPU core are adjusted during the experiment whose proportion is similar to that in the previous research [16]. Additionally, the experiment is carried out in Python 2.7 with SciPy and sklearn package in Ubuntu 16.04 running on Intel Core i5-2410M 2.3GHz and 2 GB of RAM.

Equation (9) is applied to all harmless and harmful training data whose result is then plotted in Figure 4. It is depicted that θ of harmless and harmful groups is mostly closer to 0 and 1, respectively. Nevertheless, there is an overlap area whose value can be either harmless or harmful. The threshold τ is determined by heuristically calculating possible values, from 0.25 to 0.37. It is found that $\theta = 0.31$ delivers the best result. Therefore, in the experiment we use this value for deciding the status of the machine.

This status is determined by measuring those in (5) and comparing its value with (6) whose results are presented in Table 5. For the evaluation purpose, in this research we also implement some algorithms: artificial neural networks (ANN), support vector machine (SVM) and decision tree. For further analysis, we vary the threshold values as depicted in Figure 5.

As in Table 5 and Figure 5, we analyze the sensitivity, specificity, accuracy and precision. Here, sensitivity means the number of attacks which can be correctly detected among the whole attacks; specificity means the number of non-attacks which can be correctly recognized among the whole non-attacks; accuracy means the number of attacks or non-attacks which can be correctly identified among all evaluations; precision means the

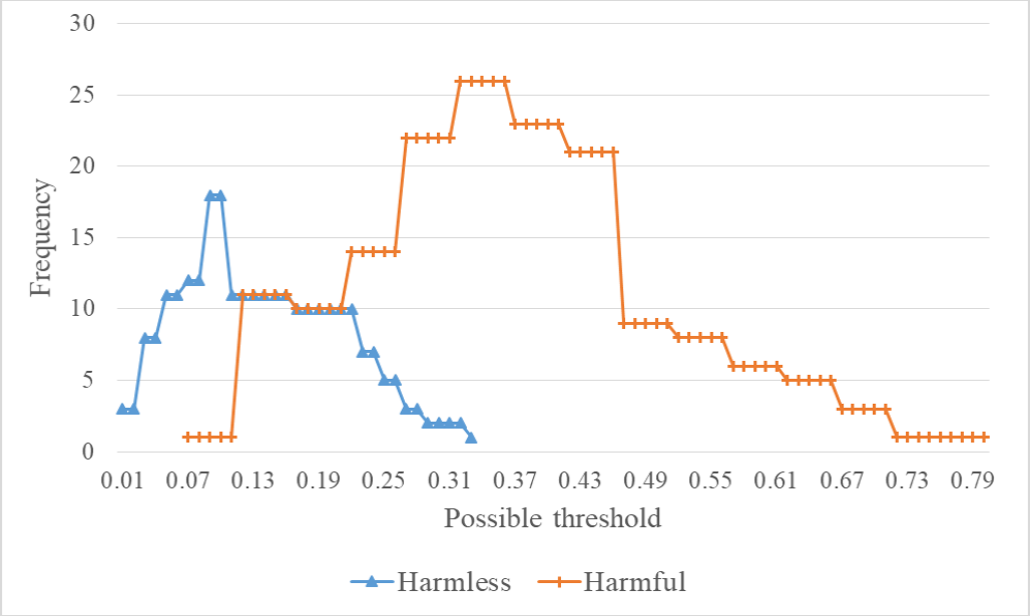

FIGURE 4. Composition of training data

TABLE 5. Experimental results

Algorithm	Accuracy	Sensitivity	Specificity	Precision
ANN	0.89	0.87	0.90	0.88
SVM	0.90	0.94	0.88	0.84
Decision Tree	0.87	0.83	0.91	0.89
Proposed Method	0.90	0.80	0.97	0.97

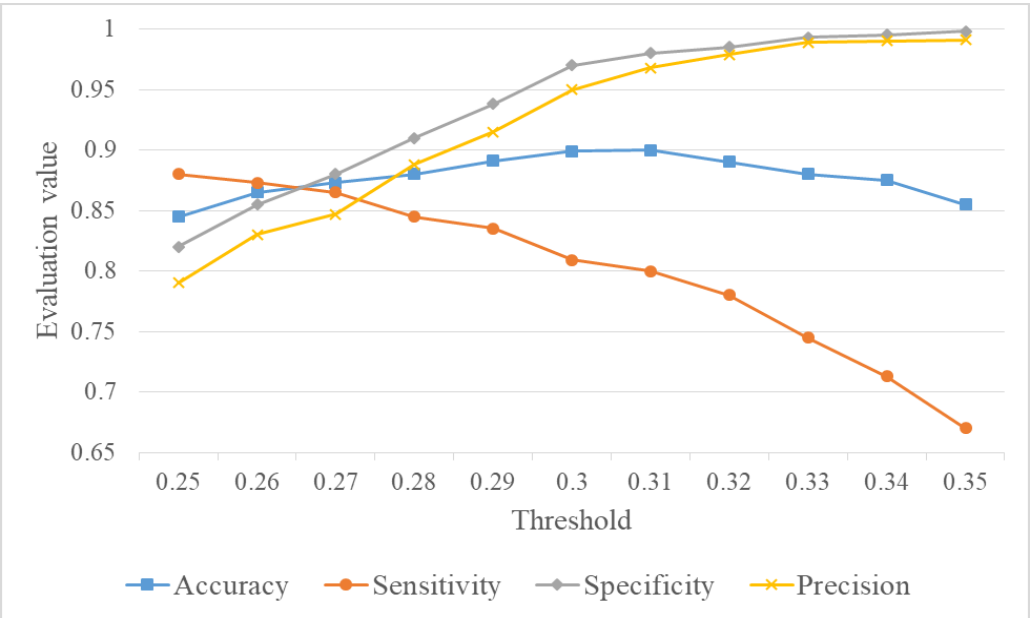

FIGURE 5. Evaluation results with various threshold values

number of attacks which can be correctly identified among correct or incorrect identified attacks.

It is shown that the proposed method has the highest accuracy, along with SVM. In terms of sensitivity and specificity, it is the lowest and the highest, respectively. It can be

inferred that the proposed method is able to minimize possible false alarms. This factor is important, especially for a security administrator to avoid a large number of incorrect signals. Similarly, the proposed method also outperforms other algorithms, in terms of precision. It is able to achieve as high as 0.97. This depicts its capability to recognize a possible attack which may exist in normal packets.

5. **Conclusions.** In this paper, we present a method which determines whether a computer in a cluster is harmful or not. This status depends on both external and internal factors of the machine itself. In case its status is harmful, the machine is to be honeypot, and all services are moved to another machine.

The experimental result shows that this method, overall, is better than other classifier algorithms, although its performance is lower in a specific evaluation. In the future, some factors may be included in the consideration, for example, the memory usage.

Acknowledgment. This research is supported by ITS and the Ministry of Research, Technology and Higher Education of Indonesia.

REFERENCES

[1] Statista, *Number of Smartphone Users Worldwide from 2014 to 2020 (in Millions)*, https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, 2016.

[2] D. Valocchi, D. Tuncer, M. Charalambides, M. Femminella, G. Reali and G. Pavlou, SigMA: Signaling framework for decentralized network management applications, *IEEE Trans. Network and Service Management*, vol.14, no.3, pp.616-630, 2017.

[3] E. Jafarnejad Ghomi, A. Masoud Rahmani and N. Nasih Qader, Load-balancing algorithms in cloud computing: A survey, *Journal of Network and Computer Applications*, vol.88, pp.50-71, 2017.

[4] E. Gravelle and S. Martinez, An anytime distributed load-balancing algorithm satisfying capacity and quantization constraints, *IEEE Trans. Control of Network Systems*, vol.4, no.2, pp.279-287, 2017.

[5] I. Indre and C. Lemnaru, Detection and prevention system against cyber attacks and botnet malware for information systems and Internet of things, *IEEE the 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp.175-182, 2016.

[6] M. Feily, A. Shahrestani and S. Ramadass, A survey of botnet and botnet detection, *The 3rd International Conference on Emerging Security Information, Systems and Technologies*, pp.268-273, 2009.

[7] R. R. R. Robinson and C. Thomas, Evaluation of mitigation methods for distributed denial of service attacks, *The 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp.713-718, 2012.

[8] I. Sembiring, Implementation of honeypot to detect and prevent distributed denial of service attack, *The 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pp.345-350, 2016.

[9] B. Yadav and V. S. Devi, Novelty detection applied to the classification problem using probabilistic neural network, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp.265-272, 2014.

[10] C.-Y. Lai and P.-C. Wang, Fast and complete conflict detection for packet classifiers, *IEEE Systems Journal*, vol.11, no.2, pp.1137-1148, 2017.

[11] J. Faircloth, Chapter 3 – Scanning and enumeration, in *Penetration Tester's Open Source Toolkit*, 4th Edition, 2017.

[12] G. Somani, M. S. Gaur, D. Sanghi, M. Conti and R. Buyya, DDoS attacks in cloud computing: Issues, taxonomy, and future directions, *Computer Communications*, vol.107, pp.30-48, 2017.

[13] H. Rahmani, N. Sahli and F. Kamoun, DDoS flooding attack detection scheme based on F-divergence, *Computer Communications*, vol.35, no.11, pp.1380-1391, 2012.

[14] I. Zainul Muttaqien and T. Ahmad, Increasing performance of IDS by selecting and transforming features, *IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, Surabaya, Indonesia, 2016.

[15] C. T. Phong, *A Study of Penetration Testing Tools and Approaches*, Master Thesis, Auckland University of Technology, 2014.

[16] T. Ahmad and K. Muchammad, L-SCANN: Logarithmic subcentroid and nearest neighbor, *Journal of Telecommunications and Information Technology*, vol.2016, no.4, 2016.