Towards a Lightweight Detection System for Cyber Security in IoT Environments using Feature Selection

YAN NAUNG SOE[†1,2] YAOKAI FENG[†1] PAULUS INSAP SANTOSA[†2]
RUDY HARTANTO[†2] KOUICHI SAKURAI [†1]

Abstract: Deployment of IoT devices in many situations will make our daily like more comfortable and make more efficient. However, the growing of malware attacks are targeting on IoT devices. The implementation of lightweight IDS for IoT environments become a critically important and challenging task. To make the system lightweight, a correlation-based feature selection algorithm is applied to significantly reduce the number of features. We used J48 and artificial neural network to observe the detection accuracy. The performance of our system is examined in detail and the experimental result indicates that our system is lightweight and has high detection accuracy.

Keywords: IDS, Feature Selection, IoT

1. Introduction

IoT (Internet of Things) era has been coming up and our world will become more convenient and more efficient. According to the Cisco Visual Network Index, mobile data traffic will grow at a compound annual growth rate of 47 percent from 2016 to 2021, reaching 49 exabytes per month by 2021 [1]. The greater the growing of mobile and IoT infrastructure, the more challenging of cyber-security occurs. In the attempted attacks against IoT devices over the 2016, the average of IoT device was attacked once every two minutes [2]. Cybercriminals' interest in IoT devices continues to grow and many malware attacks for smart devices picked up to three times in the 2017. Kaspersky Lab have collected 121,588 malware samples in 2018 [3].

Intrusion detection system (IDS) system is an efficient technique to protect the network effectively. An IDS is a specialized tool that knows how to parse and interpret network traffic and host activity [4]. There are mainly two types of IDS, named anomaly and misused-based IDS. Anomaly detection is to distinguish the activity which is differing from the normal system activity. Misused-base IDS stores a database of known attack signatures and can compare patterns of activity, traffic, or behavior it sees in the data it's monitoring against those signatures to recognize when a close match between a signature and current behavior occurs [4]. Above of these approaches may be done by machine learning techniques to get the better detection response. The more records are trained, the more accurate sequence may be detected. Although the large number of training records is required, the large number of features becomes the challenge to implement the lightweight detection architecture on resource constraint devices especially on Raspberry Pi.

Feature selection methods can be used to select the relevant features which may be most related with desired results. The irrelevant or redundant features may not only raise the computation and storage cost and even deteriorate the detect accuracy, which has also been said by sore other researches [5]. And, they should not condense the accuracy of a predictive model [6]. There are mainly three types of feature selection methods such as filter-based, wrapper-based and embedded-based approaches [7]. Every approach has its own weaknesses. For example, finding the suitable learning is difficult in filter-based approaches. Although the filter-based approach is computationally light, the accuracy may be reduced due to the fact that the nature of the underlying algorithm is ignored. The huge number of computing will be required in the wrapper-based approaches because of the combination process of feature selection and learning process. The embedded-based approach is to incorporate the selection process with the training part.

The important challenge is that much amount of features and records is difficult to implement the lightweight detection system on resource constraint devices. Addition, the number of features, it is needed to use for detection should not be large and the classifier also must be simple. Another challenge is how to find suitable datasets for training and testing the detection system. Many researches still use KDDCUP 99 datasets [8] or its variant version KDD NSL [9]. Obviously, such datasets are too outdated. For example, they have no or not enough modern attacks and the distribution of the data in networks has also changed much.

In this paper, the main contribution is the implementation of machine learning based, lightweight IDS for IoT environment which is proposed on Raspberry Pi. In the proposed detection system, the number of features used for detection is greatly decreased using CFS algorithm. Another contribution is the examining the processing time between using feature selection and not using feature selection. The results indicate that using CFS to select feature is very effective for implementing a lightweight IDS. Specifically, it can greatly reduce the processing time almost without sacrifice of detection rates.

†1 Kyushu University
†2 Universitas Gadjah Mada

2. Related Works

Most of the public IDS are based on pattern matching and are static approaches. Attacks are becoming sophisticated and attackers can circumvent rules-based detection techniques. Thus, machine learning based approaches are thought to be promising.

Most of previous IDS researches [10]–[12] used KDDCUP 99 dataset and machine learning algorithms were used for implementing the detection system. Another popular dataset, the variant of KDDCUP 99, named KDD-NSL dataset was also used in IDS researches [11], [13], [14]. However, such datasets are too outdated and not enough the modern attack distribution records. In recent years, as more and more IoT devices are actually deployed, IDS in IoT environments has attracted attentions from many researchers and developers. The researches [15], [16] addressed specific types of threats targeting specific devices.

The researches [16]–[19] proposed IDS solutions also using Raspberry Pi. The automata based IDS [16] was implemented by constructing the previous traffic natures and matching the present traffic for detecting three types of attacks such as replay-attack, jam-attack and fake-attack. The experimental results in the works [17], [18] indicated that public IDS systems, snort and bro, can be run on Raspberry Pi device. The work [17] observed that when such traditional IDS are used on Raspberry Pi, the rules must be limited and otherwise, the Raspberry Pi system will crash. An IDS/IPS solution for RFID using Raspberry Pi is implemented in the work [19]. Although all the above researches claimed a detection system is implemented and some attacks are successfully detected, obviously how to make the detection system itself lightweight is critical and unavoidable if we want to implement an actually efficient detection system in IoT environments.

3. Proposed System

The general flow of our IDS architecture is shown in Figure 1. We used the CFS feature selection for select the most relevant features from the original dataset. There are 49 features on the original dataset. If we will use all of features from the dataset, the procedure will take too long to processes, not only training but also testing. After using CFS feature selection, we could select the most important features from the dataset. Therefore, CFS feature selection can greatly reduce the processing time. The selected features are used to process the training with the classifiers, J48. And, we also used neural network classifier because of the high classification capability of the algorithm. In the test phase, the selected features are used to decide the classification accuracy. Finally, we will compare the detection accuracy with the results of machine learning algorithm, J48 and artificial neural network also. The main objective of our system is to implement the lightweight detection system on resource constraint device, Raspberry Pi.

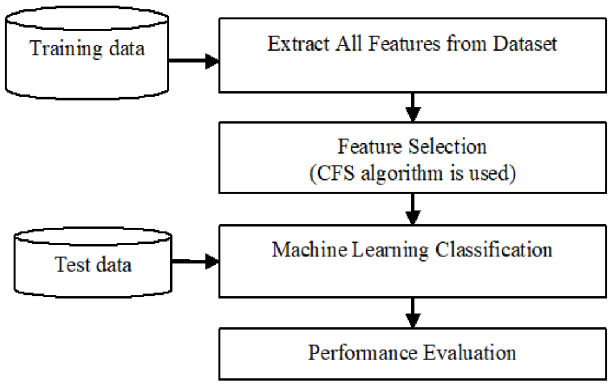


Figure 1. General flow IDS architecture

3.1 Correlation-Based Feature Selection (CFS)

To implement a lightweight detection system, a lightweight feature selection process is necessary. Correlation-based feature selection (CFS) can be used in the filter-based approach and it is a simple algorithm evaluating the corresponding relations between the outputs and correlated input features [20]. This algorithm claims that feature selection for classification in machine learning can be achieved on the basis of correlation between features. Empirical evidence shows that, along with irrelevant features, redundant information also should be eliminated [21], [22]. A feature is redundant if one or more of the other features are highly correlated with it. The CFS algorithm is based on the fact that a good feature subset is one that contains features highly correlated with the class and uncorrelated with each other. Irrelevant or redundant features should be ignored also because they may raise the computation process and even worsen the detection accuracy. Equation (1) gives the merit of a feature subset and the feature subset having the highest merit will be the result of feature selection. In Equation (1), k is the number of features in the current subset, $\overline{r_{cf}}$ is the average value of all feature-class correlations and $\overline{r_{ff}}$ is the average value of all feature-feature correlations. CFS can support most of the attribute types such as binary, date, nominal, empty (missing) values and unary attributes.

$$M_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}} \quad (1)$$

3.2 Artificial Neural Network

Machine learning is a subset of Artificial Intelligence (AI) in the field of computer science that often uses statistical techniques to give computers the ability to learn with data. It could understand how to program them to learn, to improve automatically with experience and its impact would be dramatic [23]. Artificial Neural Networks (ANN) provides a general, practical method for learning real-valued, discrete-valued, and vector-valued functions [23]. It is important machine learning tool used for classification and clustering. It is an attempt to build machine that will mimic brain activities and be able to learn. ANN is able to perform classification and even discover new trends or patterns in data. Basic ANN is composed of three

layers such as input, output and hidden layer. Each layer can have number of nodes and nodes from input layer are connected to the nodes from hidden layer. Nodes from hidden layer are connected to the nodes from output layer. Those connections represent weights between nodes. The typical neural network process flows are shown in Figure 2. There are mainly two phases such as forward and backward process. The backward process is to adjust the weight vectors between each connection and can generate the optimal weight vectors for testing/detection purpose.

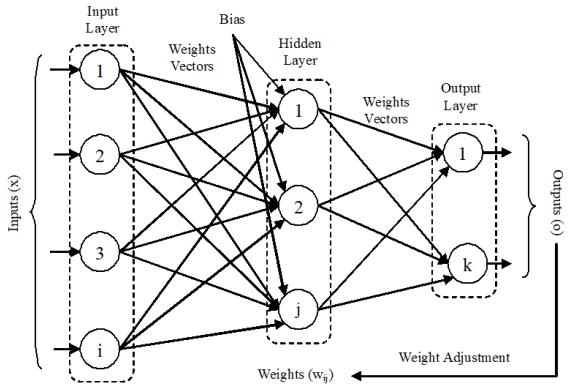


Figure 2. Typical neural network process

3.3 J48

J48 (C4.5) is one of decision tree generation algorithms developed by Ross Quinlan [24]. It is the descendant of the ID3 algorithm and can be used as a statistical classifier. It is a tree-like structure which consist root node and leaf nodes are derived from it. The leaf nodes may represent classes or class attributes. It is constructed by using information gain and entropy criteria. And then, it generates the rules for the target outcomes. It can handle both continuous and discrete features. By using the pruning techniques, the overfitting problem can be solved. It can also be used on training data having incomplete data and different weighted features. J48 is used in many researches and actual systems.

4. Experiments

Our experiments are done by using Python programming language. We used the supporting python libraries, specially on scikit-learn [25]. We used this library for implementing not only decision tree but also neural network. And also, keras [26]and tensorflow [27] libraries are also used for implementing neural network program. Raspberry Pi 3 Model B is the experiment platform to implement our lightweight IDS. The dataset UNSW-NB 15 [28], [29] is used as training data and test data. J48 is used for classification and is to mention the possible features subsets, could be handled. We also implemented the neural network program. We will compare the detection accuracy between J48 and neural network. The effectiveness of CFS algorithm is examined, including its influence on processing time and detection rates.

4.1 Dataset

UNSW-NB15 was created in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [28], [29]. Totally, 175,341 instances are randomly selected from this dataset including nine different kinds of attacks as shown in Table 1.

Table 1. Nine kinds of attacks in UNSW-NB15 [29]

Attack Type	Description
Fuzzers	Attempting to cause a program or network suspended by feeding it the randomly generated data
Analysis	It contains different attacks of port scan, spam and html files penetrations
Backdoors	A technique in which a system security mechanism is bypassed stealthily to access a computer or its data
DoS	A malicious attempt to make a server or a network resource unavailable to users, usually by temporarily interrupting or suspending the services of a host connected to the Internet.
Exploits	The attacker knows of a security problem within an operating system or a piece of software and leverages that knowledge by exploiting the vulnerability.
Generic	A technique works against all block-ciphers (with a given block and key size), without consideration about the structure of the block-cipher.
Reconnaissance	Contains all Strikes that can simulate attacks that gather information.
Shellcode	A small piece of code used as the payload in the exploitation of software vulnerability.
Worms	Attacker replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it.

The 49 features are recommended in the original datasets [28], [29]. In our experiment, we also used the selected input features such as service, sbytes, sttl, dttl, slode, ct_dst_sport_ltm, ct_srv_dst and, attack label.

4.2 Supporting Python Libraries

We used the supporting libraries to implement the python program, especially on Scikit-lean, Keras and Tensorflow.

4.2.1 Scikit-learn

Scikit-learn is the supporting tool to implement many machine learning algorithms efficiently [25]. It also provides the function to split datasets into multiple subsets including for splitting train and test set. We used this function to split the dataset by one-third of dataset as test set and two-third of dataset as train test, also.

4.2.2 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of the dependencies including tensorflow. It was developed with a focus on enabling fast experimentation [26]. It is running on the top of the Tensorflow [30]. It can also support the multilayer perception.

4.2.3 Tensorflow

Tensorflow is more complex library for distributed numerical computation using data flow graphs. It makes it possible to train and run very large neural networks efficiently [25]. Tensorflow was originally created at Google and it is an open source software library for high performance numerical computation [27]. It can strongly support for machine learning and deep learning for many other scientific domains.

4.3 Performance Evaluation

The performance (processing time and detection accuracy) of our system for detecting each of the different nine kinds of attacks is examined. The investigated detection accuracy based on these criteria, TP (true positive), FP (false positive), TN (true negative) and FN (false negative). The detection accuracy is compared by TPR (true positive rate) and FPR (false positive rate).

In our experiment, we found that, in the case of all the original 49 features are used, our Raspberry Pi could only handle about 80% at the most of the dataset. That is, the system crashed when more instances were used. Thus, in our experiment, 140,273 instances are used, which almost reached the up-limitation of our system when all the 49 features are used. Of the 140,273 instances, 92,580 instances (about two thirds of the total) are used for training and 47,693 instances (about one third of the total) for test. Anyway, after the CFS algorithm is used to decrease the number of input features to seven, our Raspberry Pi can handle the whole dataset.

4.3.1 Processing Time

According to our experiment, the processing time for each kind of attacks was decreased greatly with the help of the CFS algorithm. Because of the space limitation, only the total training times for all the nine kinds of attacks are shown in Table. 2. Again, J48 algorithm is used as the classifier in our system. The total training time is 80.98 seconds for processing 92,580 instances in the case of using CFS selected features. On the other way, the training time is 210.29 seconds for the same instances in the case of using all features, without using CFS feature selection. These result shows that the usage of CFS feature selection can assist the processing time to be faster than the original one that is around three times of the processes.

Table 2. Processing time (with seconds) comparison between using CFS and not using CFS in the case of J48

	Using CFS	**Not using CFS**
Training	80.98	210.29
Testing	8.75	17.87

The total testing time comparison also shows in the Table 3. We used 47,693 records for testing purpose for detection accuracy. We compare the processing time between using CFS and not using CFS in the case of J48 classifier in the table. This table also shows that the CFS feature selection can assist the processing time to be faster, and it can decrease to the half of normal processing time. The usage of CFS selected features will need around 8.75 seconds for the 47,693 instances. Therefore, only 0.19 milliseconds are needed for single processing unit. The processing time will increase up to 0.37 milliseconds if we will use all features of the dataset. These results shows CFS can support the detection system to be lighter.

4.3.2 Detection Accuracy: J48 Classifier

The investigation result on detection rates is shown in Figure 3 and 4.

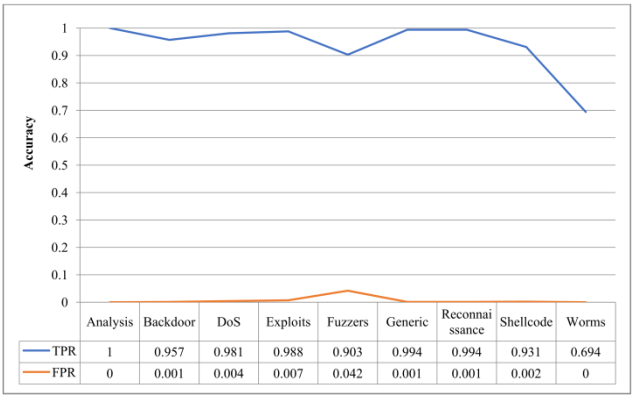


Figure 3. The TPR (True Positive Rate) and FPR (False Positive Rate) not using CFS feature selection in the case of J48

The true positive rate (TPR), not using CFS is shown in Figure 3. Almost of the attack types except “worms”, the attack detection (true positive) rates are more than 90 percent. However, the processing time will be needed and only 80 percent of dataset can be used to detection system. Another way, CFS feature selection, we used to select the effective features for dataset. After using CFS feature selection, this detection system can be handled all instances of dataset. Figure 4 shows the detection accuracy of J48 algorithm using CFS feature selection. We will compare the detection accuracy between using CFS features and not using CFS features in Figure 4 and 5. In this comparison, we only compare the detection result by using 80 percent of dataset because Raspberry Pi can only handle to process 80 percent of records if we will use all features. According to the results, the detection rate for worms attack is slightly increased after applying the CFS selected features although other attacks’ detection rates are slightly decreased. However, the processing time will be significant

decreased and the large number of training and testing records can be used. The more number of training records may assist the detection results to be stable.

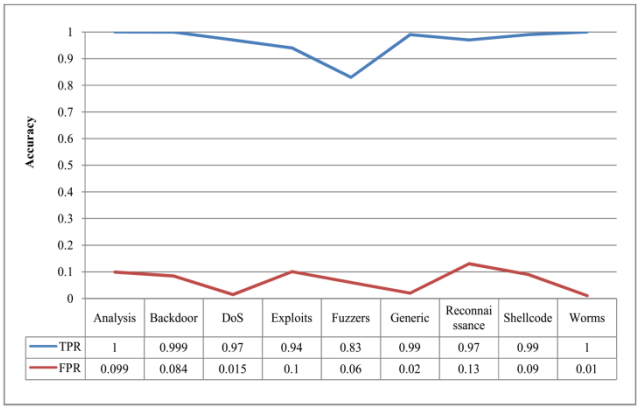
	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms
TPR	0.839	0.943	0.965	0.973	0.838	0.995	0.99	0.81	0.769
FPR	0.003	0.001	0.006	0.018	0.085	0.002	0.002	0.003	0

Figure 4. The TPR (True Positive Rate) and FPR (False Positive Rate) using CFS feature selection in the case of J48

From Figure 3 and 4, we can observe that, if the CFS algorithm is used for feature selection, the processing time has been reduced greatly for both training and testing, without clear sacrifice of detection rates.

4.3.3 Detection Accuracy: Artificial Neural Network (ANN)

In the case of implementing the artificial neural network, we also used the selected features by CFS because this feature selection is effective, as mentioned in section 4.3.1 and 4.3.2. The detection results are shown in Figure 5. Almost of TPR of ANN based classification are better than using J48 algorithm and these are around 100 percent detection rate. On the other side, FPR of J48 results are better than ANN. However, these results are very depending on our implemented neural network program which is based on python libraries. Our mission, to implement lightweight system, we constructed the network structure with single hidden layer with the limited number of epochs. And also, we also used sigmoid activation function to adjust the weight vectors.



	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms
TPR	1	0.999	0.97	0.94	0.83	0.99	0.97	0.99	1
FPR	0.099	0.084	0.015	0.1	0.06	0.02	0.13	0.09	0.01

Figure 5. TPR (True Positive Rate) and FPR (False Positive Rate) using the CFS feature selection in the case of ANN

4.3.4 Observations

Using the CFS algorithm greatly made our system to be lightweight. In particular, after the number of features was significantly decreased, the processing time for training and that for testing are reduced greatly.

In the case of J48 algorithm, our detection system had similar detection accuracy (see Figure 3 and 4). That is, our system successfully accomplished lightweight with almost no sacrifice of detection accuracy.

In the case of ANN algorithm, our detection system had more accurate result in almost of the attack types. Our lightweight detection system even has better detection accuracy in almost of the cases, except “Fuzzers” attack which is slightly decreased around 0.008. The experiment results show that the irrelevant or redundant features may only raise the computation and storage cost.

5. Conclusion and Future Work

Due to the limited computing resources of IoT devices and targeting of cyber-attacks, how to design and implement lightweight intrusion detection systems for IoT environments has been an urgent issue. In this study, based on machine learning technology, we implemented a lightweight system for detecting cyber-attacks in IoT environments. In our system, we use a lightweight and efficient feature selection algorithm, CFS, to reduce the number of irrelevant features. Our system was implemented on a Raspberry Pi system and its performance was examined using UNSW-NB15 dataset. According to our experiment, our system can handle all the instances in this dataset and has a much higher speed for training and having almost no sacrifice of detection accuracy. In the future work, also we will continue the implementation of the lightweight detection architecture on Raspberry Pi to get the high detection rate with low false alarm.

Reference

[1] Cisco, “Cisco Visual Networking Index (VNI),” *Globle Forecast Update*, pp. 1–35, 2017.

[2] Symantec, “Internet Security Threat Report,” April, 2017.

[3] V. K. Mikhail Kuzin, Yaroslav Shmelev, “New Trends in the World of IoT Threats - Securelist,” *Kaspersky Lab*. 2018.

[4] A. R. Baker and J. Esler, "Snort IDS, IPS Toolkit". Syngress Publishing, Inc. Elsevier, Inc. 30 Corporate Dr. Burlington, MA 01803, 2007.

[5] Y. Feng, H. Akiyama, L. Lu, and K. Sakurai, “Feature Selection for Machine Learning-Based Early Detection of Distributed Cyber Attacks,” *IEEE Cyber Science and Technology Congress (CyberSciTech), CyberSciTech2018*, pp. 173–180, 2018.

[6] M. Kuhn and K. Johnson, “An Introduction to Feature Selection,” *Applied Predictive Modeling*. pp. 487–519, 2013.

[7] Zena M. Hira and D. F. Gillies, “A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data,” *Advances in Bioinformatics*, vol. 2015, no. 1, 2015.

[8] S. D. Bay, D. Kibler, M. J. Pazzani, P. Smyth, “The UCI KDD Archive of Large Data Sets for Data Mining Research and Experimentation,” *SIGKDD Explore*, vol. 2, p. 81, 2000.

[9] L. Dhanabal and D. S. P. Shantharajah, “A Study On NSL-KDD Dataset For Intrusion Detection System Based On Classification

Algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[10] P. Aggarwal and S. K. Sharma, "An Empirical Comparison of Classifiers to Analyze Intrusion Detection," *International Conference on Advances Computing and Communication Technologies, ACCT*, vol. 2015–April, pp. 446–450, 2015.

[11] S. Samdani and S. Shukla, "A Novel Technique for Converting Nominal Attributes to Numeric Attributes for Intrusion Detection," *8th International Conference on Computing, Communicaion and Networking Technologies ICCCNT 2017*, no. 1, pp. 1–5, 2017.

[12] P. Kushwaha, H. Buckchash, and R. Balasubramanin, "Anomaly based Intrusion Detection using Filter Based Feature Selection on KDD-CUP 99," *Proceeding 2017 IEEE Region 10 Conference (TENCON), Malaysia*, pp. 839–844, 2017.

[13] H. Haddad Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, "A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 6750, no. c, pp. 1–1, 2016.

[14] B. Subba, S. Biswas, and S. Karmakar, "A Neural Network based system for Intrusion Detection and Attack Classification," *2016 Twenty Second National Conference Communication*, pp. 1–6, 2016.

[15] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of Sinkhole Attacks for Supporting Secure Routing on 6LoWPAN for Internet of Things," *Proceeding 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, pp. 606–611, 2015.

[16] Z. Guo, I. G. Harris, Y. Jiang, and L. F. Tsaur, "An efficient Approach to Prevent Battery Exhaustion Attack on BLE-based Mesh Networks," *International Conference on Computing, Networking and Communication ICNC 2017*, pp. 1–5, 2017.

[17] A. K. Kyaw, Y. Chen, and J. Joseph, "Pi-IDS: Evaluation of Open-source Intrusion Detection Systems on Raspberry Pi 2," *2015 2nd International Conference on Information Security and Cyber Forensics, InfoSec 2015*, pp. 165–170, 2016.

[18] J. Pacheco, X. Zhu, Y. Badr, and S. Hariri, "Enabling Risk Management for Smart Infrastructures with an Anomaly Behavior Analysis Intrusion Detection System," *Proc. - 2017 IEEE 2nd Int. Work. Found. Appl. Self* Syst. FAS*W 2017*, pp. 324–328, 2017.

[19] T. Zitta, M. Neruda, and L. Vojtech, "The Security of RFID Readers with IDS/IPS Solution using Raspberry Pi," *2017 18th Int. Carpathian Control Conf. ICCC 2017*, pp. 316–320, 2017.

[20] M. Hall, "Correlation-based Feature Selection for Machine Learning," *Methodology*, vol. 21i195-i20, no. April, pp. 1–5, 1999.

[21] R. Kohavi and G. H. John, "AIJ Special Issue on Relevance Wrappers for Feature Subset Selection," pp. 1–43, 1997.

[22] M. M. Korobov, "Feature Subset Selection using the Wrapper Method: Overfitting and Dynamic Search Space Topology," *The First International Conference on Knowledge Discovery and Data Mining*, 1995.

[23] T. M. Mitchell, "Machine Learning," 2009.

[24] A. Ashari, I. Paryudi, and A. Min, "Performance Comparison between Naïve Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool," *International Journal of Advance Computing Science and Application*, vol. 4, no. 11, pp. 33–39, 2013.

[25] A. Géron, "Hands-on machine learning with scikit-learn & tensorflow." 2017.

[26] "Keras: The Python Deep Learning library." [Online]. Available: https://keras.io/#support [Accessed: 01-Feb-2019] .

[27] "An Open Source Machine Learning Framework for Everyone." [Online]. Available: https://www.tensorflow.org. [Accessed: 01-Feb-2019].

[28] N. Moustafa and J. Slay, "The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems," *Proceeding - 2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2015*, March 2016, pp. 25–31, 2017.

[29] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 network data set)," *2015 Military Communiccations and Information System Conference, MilCIS 2015 - Proceeding*, December, 2015.

[30] A. Gulli and S. Pal, "Deep Learning with Keras," Packt Publishing Ltd., 2017.

Acknowledgments The authors are grateful for the financial support provided by AUN/SEED-Net Project (JICA). This research is also partially supported by Strategic International Research Cooperative Program, Japan Science and Technology Agency (JST), JSPS KAKENHI Grant Numbers JP17K00187 and JP16K00132.