Fuzzy Pattern Tree for Edge Malware Detection and Categorization in IoT

Ensieh Modiri

Azad University of Mashhad

Amin Azmoodeh & Ali Dehghantanha

Cyber Science Laboratory, University of Guelph, Ontario, Canada

David Ellis Newton

School of Computer Science, University of Salford, UK

Reza M. Parizi

Department of Software Engineering and Game Development, Kennesaw State University, GA, USA

Hadis Karimipour

School of Engineering, University of Guelph, Guelph, Canada

Abstract

The surging pace of Internet of Things (IoT) development and its applications has resulted in significantly large amounts of data (commonly known as big data) being communicated and processed across IoT networks. While cloud computing has led to several possibilities in regard to this computational challenge, there are several security risks and concerns associated with it. Edge computing is a state-of-the-art subject in IoT that attempts to decentralize, distribute and transfer computation to IoT nodes. Furthermore, IoT nodes that perform applications are the primary target vectors which allow cybercriminals to threaten an IoT network. Hence, providing applied and robust methods to detect malicious activities by nodes is a big step to protect all of the network. In this study, we transmute the programs' OpCodes into a vector space and employ fuzzy and fast fuzzy pattern tree methods for malware detection and categorization. We obtained a high degree of accuracy during reasonable runtimes especially for the fast fuzzy pattern tree. Both utilized feature extraction and fuzzy classification, which were robust, led to more powerful edge computing malware detection and categorization method.

Keywords: Malware Detection, Edge Computing, IoT, Fuzzy Pattern Tree, Machine Learning, Cyber Security.

1. Introduction

Over the last decade, the Internet of Things (IoT) has changed the face of our world and ubiquitous sensing enabled by wireless technologies is expanding into countless objects that surround us [1]. IoT, as a cutting-edge topic of Information Technology, has penetrated into all facets of everyday life ranging from health [2] and agriculture [3] to smart city [4] and energy & transport management systems [5, 6]. Numerous smart nodes are sensing, storing and communicating valuable and private information over widely distributed IoT networks without human intervention and this has caused increasing interest for cybercriminals to attack and misuse IoT[7]. Although Cloud Computing has significantly changed our computation paradigms, since a considerable volume of data is produced at the edge of IoT network, its speed of transportation is a bottleneck for cloud-based computation[8]. Furthermore, there are several security issues associated with the cloud-based computing model such as unauthorized access, lack of control or availability risks, privacy risk and etc.[9, 10, 11].

The intrinsic importance of IoT's information and applications requires that specific concerns are addressed which secure and protect IoT nodes and architecture [7, 12, 13]. Since malicious actors endeavour to attack compromised IoT nodes, malware and intrusion detection is an essential and evolving research sphere in cyber security [14, 15, 16, 17]. While there are different approaches which detect malicious activities and attack vectors[18, 19], Machine Learning has ranked amongst the top methods that robustly recognize malware and intrusions[20, 21].

Fuzzy system and inference [22] is a major category of machine learning methods that demonstrate its abilities to deal with ambiguity and unseen conditions and accurately simulates human brain decision making behavior. This potential increases the fuzzy methods' robustness against changes of malicious behavior and signature during test time. Despite proposals for methods methods using fuzzy in malware detection domains [23, 24, 25], this area is still not completely covered so and there are niche potentials to apply security and forensics issues. Fuzzy Pattern Tree (FPT) induction was recently introduced as a novel machine learning method for classification. The structure of an FPT is similar to a binary decision tree in which inner nodes are marked with generalized (fuzzy) logical and arithmetic operators, whereas the leaf nodes are associated with (unary) fuzzy predicates on a given set of input attributes[26, 27]. FPT is competitive with other machine learning rival methods in terms of prediction performance. Moreover, it tends to produce compact models that are quite appealing from an interpretation point of view due to the limited processing resource of IoT nodes. However, this method has a disadvantage which is its high computational complexity for large datasets with many examples or attributes, the runtime may become unacceptably high. Therefore, a fast Fuzzy Pattern Tree [26] method was proposed to improve the runtime learning. We demonstrate our major contribution regarding fuzzy pattern tree advantages by applying fuzzy and fast fuzzy pattern tree for malware detection, and obtained high accuracies of 100% for IoT and VxHeaven datasets, about 97% for Kaggle dataset and more than 93.13% for Ransomware dataset.

Typically, the following criteria are used to evaluate the utility of machine learning aided techniques in malware detection:

- True Positive(TP): indicates that a malware is correctly identified as a malicious application.
- True Negative(TN): indicates that a benign is detected as a non-malicious application correctly.

- False Positive (FP): indicates that a benign is falsely detected as a malicious application.
- False Negative (FN): indicates that a malware is not detected and labeled as a non-malicious application.

Based on the criteria described above, the following metrics will be introduced to quantify a given system:

Accuracy indicates the number of samples that a classifier correctly detects, divided by the number of all malware and goodware:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision is another metric that indicates the ratio of predicted malware samples that are correctly predicted:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall indicates the ratio of malware samples that are correctly predicted:

$$Recall = \frac{TP + TN}{TP + FN} \tag{3}$$

F-Measure is the harmonic mean of prediction and recall, and defined as follows:

$$F - Measure = \frac{2 * TP}{2 * TP + FP + FN} \tag{4}$$

Matthews correlation coefficient is used in machine learning as a measure of the quality of classification and defined as follow:

$$F - Measure = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

Cross-validation is one of the most widely used techniques in machine learning to assess a detection method's outcomes gained from experiments that can be generalized into an independent dataset. There are many cross validation techniques such as Leave-P-Out, K-fold and Repeated Random Sub-sampling and K-fold validation which are suitable for datasets with limited size [28].

In the next section, we briefly review related work. Section 3 describes our collection, preprocessing and evaluation methodology. Section 4 presents our proposed approach, followed by its evaluation in Section 5. Section 6 concludes this paper and suggests a future research agenda.

2. Related Work

Analyzing the extracted information from graph of programs is a practical solution for malicious activity detection. Azmoodeh et al. [29] developed a deep learning based method to detect Internet Of Battlefield Things (IoBT) malware via the devices Operational Code (OpCode) sequence and they achieve 99.68% of accuracy. They extracted an adjacency matrix of OpCode's control flow and then applied deep learning on eigen-decomposition of the matrix. Yuxin et al. [30], proposed a method based on graph matching using the common behavior graph of the malware family. They used a dynamic taint analysis technique for finding the dependency relations between system calls, extracted the common behavior graphs and lastly, employed graph matching based on the maximum weight of subgraphs to detect malicious code and achieved 82% of recall. Hashemi et al. [31] extracted OpCodes of Windows benign and malware executable files and then formed a graph for each sample and turned the generated graph into a vector using Power Iteration procedure to train classifiers such as Support Vector Machine and Adaboost. They achieved an accuracy and a F-measure of 96.09% and 95.98% respectively.

The IoT as state-of-the-art ubiquitous technology in modern life is a prominent target for cybercriminals. Therefore, protecting IoT against cyber attackers is a crucial requirement in generating trust IoT. Haddadpajouh et al. [32] employed the capability of Deep Recurrent Neural Networks and proposed a method to accept OpCodes as features for three different Long Short Term Memory settings and achieved 98.18% of detection rate to detect IoT malware. In another study, Su et al.[33] converted IoT's malware binary to image and then leveraged a light-weight convolutional neural network for classifying their families and achieved 94.0% of accuracy for the classification of goodware and DDoS IoT malware. Sharmeen et al.[34] analyzed static, dynamic, and hybrid detection methods for industrial IoT malware and discovered machine learning approaches are commonly used to classify malware and benign in this domain. They indicated that the permission list, API call list, and the system call list are major features to detect mobile malware with a high accuracy to ensure Zero-day detection.

Robustness and an ability to deal with unseen conditions is an important motivation to apply fuzzy learning on the cybersecurity domain. Afifi et al.[23] proposed a multi agent system for monitoring and data preparation. Then, they introduced a hybrid method to combine an adaptive neuro fuzzy inference system (ANFIS) with a particle swarm optimization (PSO) for mobile malware detection. After extensive studying of fuzzy hashing methods, Li et al.[24] proposed a clustering approach to cluster malware based on their novel fuzzy hashing algorithm and obtained acceptable clustering outcome for different malware families. Bernardi et al.[25] presented a dynamic mobile malware detection method using process mining and fuzzy logic that characterized the behaviour of malware and utilized fuzzy logic for malware classification and obtained high precision for different mobile malware families.

3. Fuzzy Pattern Tree

3.1. Fuzzy Pattern Tree

A fuzzy pattern tree [27] contains a tree-like structure in which the inner nodes are fuzzy logic arithmetic operators and the leaf nodes are associated with fuzzy predicates on input attribute, although a pattern tree has a bottom to top induction which means that a node somehow takes the values of its descendants as input and combines them using fuzzy operators. Finally, submit the output to its predecessor (Figure 1). During an iterative approach, several pattern trees are generated for each class and at the end of each iteration, the best pattern

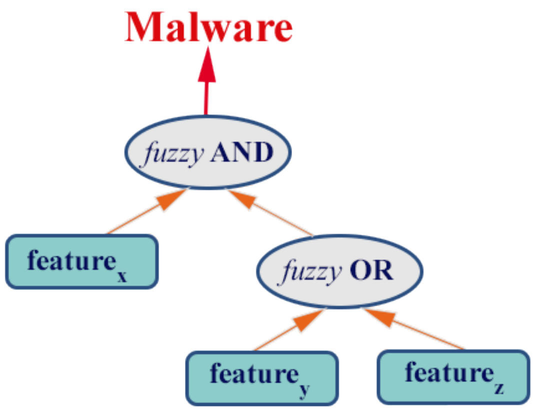

Figure 1: Fuzzy Pattern Tree

tree that has least prediction error on the class is selected to expand leaf nodes at the next stage. In order to use fuzzy pattern tree for malware detection, we applied it on processed datasets of OpCode sequences (described in Section 4) in which an instance is a vector $X = (x_1, x_2, ..., x_m) \in \mathrm{X}_1 \times \mathrm{X}_2 \times .... \times \mathrm{X}_\mathrm{m}$. X_i is the domain of i_{th} attribute and for each domain, X_i is discretized by means of a fuzzy partition $F_i = \{F_{(i,1)}, ..., F_{(i,d_i)}\}$ into a collection of fuzzy subsets that $F_{i,j} : X_i \longrightarrow [0, 1] \quad (j = 1, ..., d_i)$ and $\sum_{j=1}^{d_i} = F_{i,j}(x_i) > 1$ for all $x \in X_i$. The $F_{i,j}$ defines a unary fuzzy predicator on individual attribute values that are often associated with linguistic labels such as "small" or "large", for example, $F_{i,2} \simeq X_i$ *is medium*. On the other hand, each instance is associated with a class label of malware or benign in the output space $\mathbb{Y} \in \{malware, benign\}$ or generally, $\mathbb{Y} \in \{Class_1, ..., Class_k\}$. Leaf nodes are associated with a fuzzy partition $F_{i,j}$ and each inner node is a fuzzy operator, which belongs to:

- t-norms and t-conorms [35]
- weighted and ordered-weighted average (OWA) [36, 37]

A fuzzy pattern classifier is a collection of fuzzy pattern trees $PT = \{PT_i | i = 1, ..., k\}$ and each PT_i is the pattern tree associated with class $y_i \in \mathbb{Y}$. For classifying a new instance x, a prediction is made in favor of the class whose tree produces the highest score:

$$\hat{y} = argmax(PT_i(\mathbf{x}))\ for\ y_i \in \mathbb{Y} \tag{6}$$

3.1.1. Top-Down Induction

During the learning phase and iteratively, by expansion of the leaf nodes, several pattern trees for each class are generated and at the end of each iteration the best fuzzy pattern tree for each class is selected with respect to the *root mean squared error(Equation 7)*, which serves as a continuous and differentiable surrogate of 0/1 loss.

$$error(PT_z) = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y^j - PT_z(x^j))} \tag{7}$$

In the first step, fuzzy partitions are constructed for each input attribute and the candidate set C^0 is initialized using all primitive pattern trees P in which each primitive tree is a tree that constructed with only one node that is labeled by a fuzzy term $F_{i,j}$. Thus, the optimal PT is initialized by the model that has the lowest empirical error i C^0. During each iteration, a new set of candidate models C^t that include all expansions of the current best model $M*$ is reproduced. An expansion of a primitive tree is a model that is constructed by replacing a leaf node L by a sub tree of two leaf nodes and a parent fuzzy operator node. All existing candidates in C^t are evaluated based on their error score and M^* replaced by the new best models.

3.2. Fast Fuzzy Pattern Tree

Although the fuzzy pattern tree has demonstrated its capability to compete with state-of-the-art classification methods, it suffers from computational complexity especially for large datasets or those have curse of dimensionality. To overcome this disadvantage, the fast fuzzy pattern tree as a modified version was presented[26]. The first modification endeavors to speed up the detection of the best model among a set of candidate models, and with the second one restricted the total number of candidates.

3.2.1. Racing Algorithm

The Racing Algorithm intends to accelerate the selection of the best model from a set of candidate models. By calculating the performance of all models, those who have lower performance and are unlikely to exhibit the best productivity at the end are eliminated in early steps. The race iterates over a random number of datasets. For each model, two variable are stored; the number of examples n_j and the mean estimate of its error err_j. Each two models are compared pairwise and the model with the higher error is omitted. It is an heuristic estimation of the models' error and to obtain more accurate knowledge about the model a confidence interval is defined such as $[l_j, u_j]$ for the error of M_j and $[l_k, u_k]$ for M_k. Then a decision about the better model can be made if the intervals do not overlap, e.g. $l_j > u_j$. Using the Hoeffding inequality [26] that upper-bounds the deviation of the mean from the true expectation of a random variable with high probability gives:

$$[l_j^n, u_j^n] = err_j \pm \sqrt{\frac{B^2 log(\frac{2}{\delta})}{2n}} \tag{8}$$

where δ is an external parameter that controls the reliability of a confidence interval and the constant B is an upper bound on the value of the random variable and n is the number of observations.

3.2.2. Potential Heuristic

The idea behind Potential Heuristic is to narrow the search in each iteration with the algorithm creating new candidate models C^t by expanding every leaf node. This heuristic restricts expansion to a constant number $p < |C^t|$ of leaf nodes with highest potential. In fact, this method calculates the potential of each node to expand and selects the ones with highest potential. The potential of each node is defined as follows: Let L be a leaf of a candidate tree $M_L \in C^t$. A model $M_(L_{Opt})$ by replacing L in M_L by L_{Opt}, where L_{Opt} is a fictitious oracle leaf that provides the ideal output y_i for every training instance (x_i, y_i) in the dataset. Then, we define the potential of L in terms of the possible

Dataset	#instances	#classes
Vx-Heaven	22000	2
Ransomware	555	6
IoT	1207	2
Kaggle	9601	9

Table 1: Datsets Information

reduction of empirical risk:

$$P_M(L) = err(M_L) - err(M_{L_{Opt}}) \tag{9}$$

4. Dataset

4.1. Dataset Description

In this study, we applied fuzzy and fast fuzzy pattern tree on four different data sets namely IoT, Vx-Heaven, Kaggle and Ransomware. IoT dataset [29] includes malware and benign samples from ARM-Based architecture. Ransomeware dataset [38] contains OpCodes' of six different ransomware families. Vx-Heaven dataset includes 22000 Microsoft Windows samples belonging to malware and benign categories. Kaggle[1] dataset includes Microsoft Windows malware from nine different families. Table 1 shows the dataset information.

4.2. Feature Extraction

Binary executable files contain a long sequence of OpCodes which are instructions to be performed on each device's processing unit. Although leveraging OpCodes as features for learning algorithm is a common approach[39], the curse of dimensionality turns it to a challenging topic to utilize. In this study, we leverage Azmoodeh et al.[29] approach to extract features for fuzzy pattern tree method. At the first stage, we extracted the OpCode sequence of each sample and generated a dictionary of unique OpCodes. Then, we assessed the utility

[1]`https://www.kaggle.com/c/malware-classification`

of each OpCode by a class-wise information gain(CIG) approach[40]. CIG was proposed to overcome global feature selection imperfection and endeavors to recognize more useful features based on each dataset's classes. CIG calculate information gain for each unique OpCode-Class and this is our main criteria for selecting OpCodes as features. Equation 10 shows the CIG calculation for a two-class dataset where $P(v_f = 1, C_i)$ denotes the probability of feature f appearing in C_i, and $P(v_f = 0, C_j)$ is the probability of feature f being absent from C_j. C_B and C_M denote benign program and malicious applications, respectively.

$$\begin{aligned} CIG(f, C_B) &= P(v_f = 1, C_B) * log\frac{P(v_f = 1, C_B)}{P(v_f = 1)P(C_B)} \\ &+ P(v_f = 0, C_M) * log\frac{P(v_f = 0, C_M)}{P(v_f = 0)P(C_M)} \\ CIG(f, C_M) &= P(v_f = 1, C_M) * log\frac{P(v_f = 1, C_M)}{P(v_f = 1)P(C_M)} \\ &+ P(v_f = 0, C_B) * log\frac{P(v_f = 0, C_B)}{P(v_f = 0)P(C_B)} \end{aligned} \tag{10}$$

Afterwards, the most beneficial features were selected to generate control flow graph of each sample. A Control Flow Graph (CFG) is a data structure that represents the order of OpCodes in an executable file. A graph, $G = V, E_i$, has two sets: V and E. V Where denotes the graphs vertices and $E_{i,j}$ shows the relation between V_i and V_j[29]. In order to generate OpCodes' CGF, we employed Equation 11 to calculate CFG's edges E_{v_i,v_j} .

$$\begin{aligned} E_{v_i,v_j} &= \sum_{s \epsilon S} \frac{2}{1 + \alpha * e^{min(|s-t-1|)}} \\ S = \{index\ of\ all\ appearance\ of\ &OpCode_{V_i}\ in\ sample's\ OpCode\ sequence\} \\ t\epsilon\{index\ of\ all\ appearance\ of\ &OpCode_{V_j}\ in\ sample's\ OpCode\ sequence\} \\ \alpha\ is\ tuning\ paramter&(here\ \alpha = 1) \end{aligned} \tag{11}$$

At the final stage, Azmoodeh et al.[29] demonstrated the usefulness of CFG's eigenspace instead a complete graph. Therefore, we extracted the first k eigenvectors and eigenvalues of graph as final features where k is the number of classes.

5. Experiments

5.1. Settings

The experiments to demonstrate the ability of a fuzzy pattern tree for malware detection have been implemented on a Microsoft Windows 10 station with 8GB of memory and a Core i7 CPU. Preprocessing for generating datasets was implemented by Python 3.7 and classification phase was developed by MATLAB 2016.

5.2. Results

In order to investigate and assess the competency of fuzzy and fast fuzzy pattern tree methods for malware detection, we have conducted three different approaches to obtain comparable results. The first experiment in Section 5.2.1 compares fuzzy and fast fuzzy with other state of the art classification algorithms namely Support Vector Machine, Decision Tree, k-Nearest Neighbor and Random Forest. Section 5.2.2 includes outcomes for fuzzy and fast fuzzy pattern tree with more emphasis on the racing algorithm and potential heuristic.

5.2.1. State-of-the-art Classifiers Comparison

Dataset	PTTD	PTTD-FAST	SVM	KNN	Random Forest	Decision Tree
VX-Heaven	100	100	100	99.9955	100	100
IOT	99.8347	99.8347	98.177	99.9174	99.9174	99.9167
Kaggle	97.0427	95.136	93.7086	99.4144	99.6505	99.4051
Ransomware	88.7641	83.0649	93.9416	94.5303	95.9805	93.0909

Table 2: Accuracy of different classifiers

5.2.2. Fuzzy and Fast Fuzzy Pattern Tree Comparison

Dataset	Accuracy	Precision	Recall	F-measure	MCC	Time(Second)
Heaven	100	100	100	1	1	301.106437
IoT	99.8347	100	98.7302	0.99344	0.9922	29985.142449
Kaggle	97.0427	86.0988	87.2818	0.85879	0.84769	6507.781891
Ransomware	88.7641	91.2389	72.8486	0.71799	0.70787	2438.27705

Table 3: Performance of Fuzzy Pattern Tree Algorithm

Dataset	Accuracy	Precision	Recall	F-measure	MCC	Time(second)
Vx-Heaven	100	100	100	1	1	17.1706407
IoT	99.174	99.55	95.455	0.97393	0.94916	51.010565
Kaggle	90.093	77.81	77.506	0.72821	0.7079	66.853548
Ransomware	86.061	81.448	85.392	0.81741	0.74146	9.123403

Table 4: Performance of Fast Fuzzy Pattern Tree with Racing Algorithm

Dataset	Accuracy	Precision	Recall	F-measure	MCC	Time(second)
Vx-Heaven	100	100	100	1	1	61.68227
IoT	100	100	100	1	1	32.612093
Kaggle	89.861	84.907	87.1047	0.69092	0.71603	6287.274425
Ransomware	93.132	82.786	81.318	0.7925	0.77486	80.85638

Table 5: Performance of Fast Fuzzy Pattern Tree with Potential Heuristics

6. Concluding Remarks

Edge Computing as a state-of-the-art trend related to the Internet of Things has naturally drawn considerable attention. There are several risks associated with cloud-based computing such as unauthorized access, lack of control or availability risks & privacy risk and therefore, edge computing as an alternative solution aids cyber scientists in providing an alternative practical security mechanism. Hence, edge malware detection research is attracting more consideration and investigation into its potentiality for for applied solutions.

In this study, we employed fuzzy and fast fuzzy pattern or malware detection and demonstrated that these methods are capable of accurately detecting malicious codes and can compete with rival classification methods such as SVM, KNN, Random Forests and Decision Trees. What makes fuzzy approaches more remarkable for their pattern recognition is their potential to deal with ambiguities and human brain simulation.

During our research, we implemented fuzzy pattern and fast fuzzy pattern trees and their variations. Based on experimental results, the fuzzy pattern tree could accurately classify malware and benign for IoT and Vx-Heaven dataset with an accuracy of 99.834% and 100% respectively. Fast fuzzy pattern tree using Potential Heuristics performed 100% accuracy for both dataset with much more lower run-time. As for malware categorization, we applied both methods on Kaggle and Vx-Heaven datasets and achieved 97.0427% and 88.76% of accuracy by fuzzy pattern tree and 90.093% and 93.132% of accuracy by fast fuzzy pattern tree. In the future, we plan to boost fuzzy pattern tree accuracy and present a distributed variation of a fuzzy pattern tree that will be more efficient regarding edge computing over an IoT network.

References

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future generation computer systems 29 (7) (2013) 1645–1660.

[2] I. You, K.-K. R. Choo, C.-L. Ho, et al., A smartphone-based wearable sensors for monitoring real-time physiological data, Computers & Electrical Engineering 65 (2018) 376–392.

[3] M. Roopaei, P. Rad, K.-K. R. Choo, Cloud of things in smart agriculture: Intelligent irrigation monitoring by thermal imaging, IEEE Cloud Computing 4 (1) (2017) 10–15.

[4] X. Li, J. Niu, S. Kumari, F. Wu, K.-K. R. Choo, A robust biometrics based

three-factor authentication scheme for global mobility networks in smart city, Future Generation Computer Systems 83 (2018) 607–618.

[5] D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges, Ad hoc networks 10 (7) (2012) 1497–1516.

[6] Z. M. Fadlullah, A.-S. K. Pathan, K. Singh, Smart grid internet of things, Mobile Networks and Applications 23 (4) (2018) 879–880.

[7] M. Conti, A. Dehghantanha, K. Franke, S. Watson, Internet of things security and forensics: Challenges and opportunities (2018).

[8] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet of Things Journal 3 (5) (2016) 637–646. `doi:10.1109/JIOT.2016.2579198`.

[9] D. Svantesson, R. Clarke, Privacy and consumer risks in cloud computing, Computer Law & Security Review 26 (4) (2010) 391 – 397.

[10] D. Catteddu, Cloud computing: Benefits, risks and recommendations for information security, in: C. Serrão, V. Aguilera Díaz, F. Cerullo (Eds.), Web Application Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 17–17.

[11] J. Heiser, M. Nicolett, Assessing the security risks of cloud computing, Gartner Report 27 (2008) 29–52.

[12] S. Watson, A. Dehghantanha, Digital forensics: the missing piece of the internet of things promise, Computer Fraud & Security 2016 (6) (2016) 5–8.

[13] C. J. DOrazio, K.-K. R. Choo, L. T. Yang, Data exfiltration from internet of things devices: ios devices as case studies, IEEE Internet of Things Journal 4 (2) (2017) 524–535.

[14] J. Gardiner, S. Nagaraja, On the security of machine learning in malware c&c detection: A survey, ACM Computing Surveys (CSUR) 49 (3) (2016) 59.

[15] J. Peng, K.-K. R. Choo, H. Ashman, User profiling in intrusion detection: A review, Journal of Network and Computer Applications 72 (2016) 14–27.

[16] E. M. Rudd, A. Rozsa, M. Gnther, T. E. Boult, A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions, IEEE Communications Surveys Tutorials 19 (2) (2017) 1145–1172.

[17] H. HaddadPajouh, A. Dehghantanha, R. Khayami, R. Choo, et al., Intelligent os x malware threat detection, Journal of Computer Virology and Hacking Techniques.

[18] F. N. Dezfouli, A. Dehghantanha, R. Mahmod, N. F. B. M. Sani, S. B. Shamsuddin, F. Daryabar, A survey on malware analysis and detection techniques, International Journal of Advancements in Computing Technology 5 (14) (2013) 42.

[19] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, A. Hamzeh, A survey on heuristic malware detection techniques, in: Information and Knowledge Technology (IKT), 2013 5th Conference on, IEEE, 2013, pp. 113–120.

[20] N. Milosevic, A. Dehghantanha, K.-K. R. Choo, Machine learning aided android malware classification, Computers & Electrical Engineering 61 (2017) 266–274.

[21] A. Shalaginov, S. Banin, A. Dehghantanha, K. Franke, Machine learning aided static malware analysis: A survey and tutorial, Cyber Threat Intelligence (2018) 7–45.

[22] E. Hüllermeier, Fuzzy methods in machine learning and data mining: Status and prospects, Fuzzy sets and Systems 156 (3) (2005) 387–406.

[23] F. Afifi, N. B. Anuar, S. Shamshirband, K.-K. R. Choo, Dyhap: dynamic hybrid anfis-pso approach for predicting mobile malware, PloS one 11 (9) (2016) e0162627.

[24] Y. Li, S. C. Sundaramurthy, A. G. Bardas, X. Ou, D. Caragea, X. Hu, J. Jang, Experimental study of fuzzy hashing in malware clustering analysis, in: 8th Workshop on Cyber Security Experimentation and Test (CSET 15), USENIX Association, Washington, D.C., 2015. URL `https://www.usenix.org/conference/cset15/workshop-program/presentation/li`

[25] M. L. Bernardi, M. Cimitile, F. Martinelli, F. Mercaldo, A fuzzy-based process mining approach for dynamic malware detection, in: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2017, pp. 1–8.

[26] R. Senge, E. Huellermeier, Fast fuzzy pattern tree learning for classification, IEEE Transactions on Fuzzy Systems 23 (6) (2015) 2024–2033.

[27] R. Senge, E. Hüllermeier, Top-down induction of fuzzy pattern trees, IEEE Transactions on Fuzzy Systems 19 (2) (2011) 241–252.

[28] Y. Bengio, Y. Grandvalet, No unbiased estimator of the variance of k-fold cross-validation, Journal of machine learning research 5 (Sep) (2004) 1089–1105.

[29] A. Azmoodeh, A. Dehghantanha, K.-K. R. Choo, Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning, IEEE Transactions on Sustainable Computing.

[30] Y. Ding, X. Xia, S. Chen, Y. Li, A malware detection method based on family behavior graph, Computers & Security 73 (2018) 73–86.

[31] H. Hashemi, A. Azmoodeh, A. Hamzeh, S. Hashemi, Graph embedding as a new approach for unknown malware detection, Journal of Computer Virology and Hacking Techniques 13 (3) (2017) 153–166.

[32] H. HaddadPajouh, A. Dehghantanha, R. Khayami, K.-K. R. Choo, A deep recurrent neural network based approach for internet of things malware threat hunting, Future Generation Computer Systems 85 (2018) 88–96.

[33] J. Su, V. D. Vasconcellos, S. Prasad, S. Daniele, Y. Feng, K. Sakurai, Lightweight classification of iot malware based on image recognition, in: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Vol. 02, 2018, pp. 664–669.

[34] S. Sharmeen, S. Huda, J. H. Abawajy, W. N. Ismail, M. M. Hassan, Malware threats and detection for industrial mobile-iot networks, IEEE Access 6 (2018) 15941–15957.

[35] E. P. Klement, R. Mesiar, E. Pap, On the order of triangular norms: Comments on "a triangular norm hierarchy" by e. cretu, Fuzzy Sets Syst. 131 (3) (2002) 409–413. `doi:10.1016/S0165-0114(02)00114-8`.
URL `http://dx.doi.org/10.1016/S0165-0114(02)00114-8`

[36] B. Schweizer, A. Sklar, Probabilistic metric spaces, Courier Corporation, 2011.

[37] R. R. Yager, On ordered weighted averaging aggregation operators in multicriteria decisionmaking, IEEE Transactions on Systems, Man, and Cybernetics 18 (1) (1988) 183–190. `doi:10.1109/21.87068`.

[38] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence, IEEE Transactions on Emerging Topics in Computing (2018) 1–1.

[39] I. Santos, F. Brezo, J. Nieves, Y. K. Penya, B. Sanz, C. Laorden, P. G. Bringas, Idea: Opcode-sequence-based malware detection, in: International Symposium on Engineering Secure Software and Systems, Springer, 2010, pp. 35–43.

[40] Y. Tan, Artificial immune system: applications in computer security, John Wiley & Sons, 2016.