Evolved IoT Malware Detection using Opcode Category Sequence through Machine Learning

Sunghyun Moon
Dankook University
Yongin-si, South Korea
32181481@dankook.ac.kr

Youngho Kim
GI VITA
Seoul, South Korea
youngho.kim@gi-vita.io

Hyunjong Lee
KSign
Seoul, South Korea
hjlee@ksign.com

Donghoon Kim
Arkansas State University
Jonesboro, AR, USA
dhkim@astate.edu

Doosung Hwang
Dankook University
Yongin-si, South Korea
dshwang@dankook.ac.kr

***Abstract*—IoT devices are being exploited as entry points for cyberattacks due to security weaknesses. IoT malware variants have evolved as a result of vulnerabilities in IoT devices. This study investigates whether IoT malware can be detected with various types of malware family. An opcode sequence of malware can represent its family characteristics by utilizing opcode categories. Opcode categories are divided into 6 or 11 depending on opcode functions. Thus, a sequence of opcode categories can identify intrinsic characteristics of the family to which it belongs. By applying the entropy histogram, a 2D representation of a category sequence visually reveals innate patterns within homogeneous families. We find that benign and malware can be differentiated visually, as well as correlated and uncorrelated malware. For the designed feature representation, machine learning algorithms (5-NN, SVM, Decision Tree, and Random Forest) are used, with the best case having a mean MCC or F1-score of over 98.0%. Overall, the 11 opcode category outperforms the 6 opcode category. The experiments have shown that evolved malware can be detected with a model learned from its precedent malware.**

I. Introduction

IoT (Internet of Things) devices have become the most attractive targets for malware due to well-known weaknesses such as weak passwords, a lack of secure update procedures, and insecure network services [1], [2], [3]. As a result, cyberattacks are leveraging IoT devices as attack entry points.

`Mirai`, the notorious IoT malware, scans for vulnerable IoT devices and performs brute-force login attacks using Telnet or Secure Shell (SSH) protocols based on predefined login and password dictionaries [4]. In 2016, `Mirai` malware infected hundreds of thousands of IoT devices in an attempt to launch a distributed denial-of-service (DDoS) attack [5]. `Mirai` mutations are still being created on a daily basis, and they can spread and cause serious damage utilizing the same intrusion mechanisms as the original malware [5]. According to Kaspersky's DDoS attack report, `Simps` malware, which appeared in 2021, evolved from `Mirai` and `Gafgyt`. `Mirai` was also used by `ZHtrap` malware, which appeared in the same year [6]. There are major IoT malware families, such as `Gafgyt`, `Mirai`, and `Tsunami`. They appear to share a significant amount of code that is critical to IoT malware [7]. Many researchers are using known malware variants and their correlations to detect IoT malware [8], [9].

For IoT malware detection, machine learning (ML) explores classification rules based on feature vectors or employs similarity metrics. In general, classification prediction is accomplished through the learning process by discovering hidden pattern rules. Such detection methods can also distinguish intrinsic but hidden patterns among benign and malware. To obtain a robust learning model, it is a prerequisite to design distinguishable features of a fixed size for malware and benign. Low-dimensional features have been studied due to time or problem complexity. Operation code (opcode) and binary are used to create features, such as opcode sequences [10], [11], opcode CFG (control flow graph, [12], [13]). byte sequence sequence from ELF (executable linkable format) header [14], and 2D images from ELFs [15].

This paper deals with the IoT malware detection model in terms of category-based opcode sequence and evolutionary relationships among families. The category-based sequence is used to design features. The evolution relationship is to analyze whether a model trained with a precedent malware family is able to detect descendent IoT malware.

The contributions of this study are as follows:

- The feature of this study utilizes category-based opcode sequences. We create 11 opcode category based on 6 opcode category and the Armv6-M Architecture Reference Manual. This strategy can save training time and has the benefit of being able to deal with opcodes that differ depending on the architecture.
- The entropy histogram approach is used to visualize intrinsic patterns within IoT malware families.
- When using ascendant malware to train ML-based models, evolved or correlated IoT malware can be detected. This analyzes if a model learns from ascendant malware can detect descendant malware which is excluded in the training dataset.

This paper is organized as follows. Section II discusses related work on IoT malware detection using opcode features. Section III addresses IoT feature extraction methods. Section IV describes the experimental procedures and evaluates each feature and machine learning models for identifying IoT malware. Finally, Section V concludes this paper with future works.

II. Related Work

Because signature-based approaches are difficult to detect unknown malware, most security vendors have recently

TABLE I: 6 and 11 opcode categories

Opcode	Category	Opcode
AND, EOR, SUB, RSB, ADD, ADC, SBC, RSC, TST, TEQ, CMP, CMN, ORR, MOV, BIC, MVN, CDP	**C1**	B, BL, BLX, BX
MUL, MULL, MLA, MLAL	**C2**	AND, EOR, SUB, RSB, ADD, ADC, SBC, RSC, TST, TEQ, CMP, CMN, ORR, MOV, MOVT, BIC, MVN, CDP
B, BL, BX	**C3**	ASR, LSL, LSR, ROR, RRX
LDM, STM, LDR, STR, LDC, STC, MRC, MCR	**C4**	MUL, MULL, MLA, MLAL
MRS, MSR	**C5**	SXT, UXT
DMB, DSB, ISB, NOP, SEV, SVC, WFE, WFI, SWI, SWP, ADR, FLDM, YIELD	**C6**	REV, REV16, REVSH
	C7	MRS, MSR, CPS
	C8	LDR, STR
	C9	LDM, STM, PUSH, POP
	C10	LDC, STC, MRC, MCR
	C11	DMB, DSB, ISB, NOP, SEV, SVC, WFE, WFI, UDF, SWI, SWP, ADR, FLDM, YIELD

adopted an ML-based analysis for malware detection and mitigation. ML methods can learn feature patterns from malware families found so far and use the learned model to detect new malware with high efficiency and accuracy. An opcode sequence is one of the popular features to detect IoT malware over ML approach.

Kang *et al.* [10] presented an approach based on the opcode *n*-gram function for classifying Android malware families using NB, SVM (Support Vector Machine), partial Decision Tree (DT), and Random Forests (RF). Opcode CFG (control flow graph) can configure learning features that reflect the structure indicating executing order among instructions [12]. Su *et al.* [15] designed 2D grayscale image features from malware in terms of file size and byte sequence. Shahzad *et al.* [16] extracted 383 structural features, including section headers, symbolic sections, and program headers, from ELF files. Darabian *et al.* [11] expressed malware features by applying MSP (maximal sequential pattern) from opcode sequences. To extract MSPs, maximum order subpatterns were extracted by using MapReduce-based MG-FSM. The opcodes were classified into 6 categories by function, and training features were constructed by calculating the frequency of category change in the maximum-order pattern.

Although the aforementioned approaches were able to achieve high accuracy in malware detection and classification, they did not reflect the diversity of CPU architectures and an robust feature of IoT executables [17]. Because different CPU architectures employ different instruction sets, the opcode sequence of an executable on one CPU architecture is different from that of another architecture. A methodology designed using IoT malware found in a specific architecture is unlikely to be applied in other IoT environments. Feature engineering for machine learning analysis is dependent on CPU architecture. Thus, generalization performance cannot be expected due to various CPU architectures.

Researchers investigate the relationship or correlation between IoT malware through code reuse. As a result, it is able to reconstruct family lineage, and trace evolution [7], [18], [4], [19]. This study can help to detect various malware through the genealogy of the family. Cozzi *et al.* [7] listed `Gafgyt`, `Mirai`, and `Tsunami` as top 3, which share large common functions compared with other malware families. Donno *et al.* [18] showed how the different families are related or unrelated to each other. Vignau *et al.* [19] addressed a taxonomy and the evolution of IoT malware using the evolutionary development.

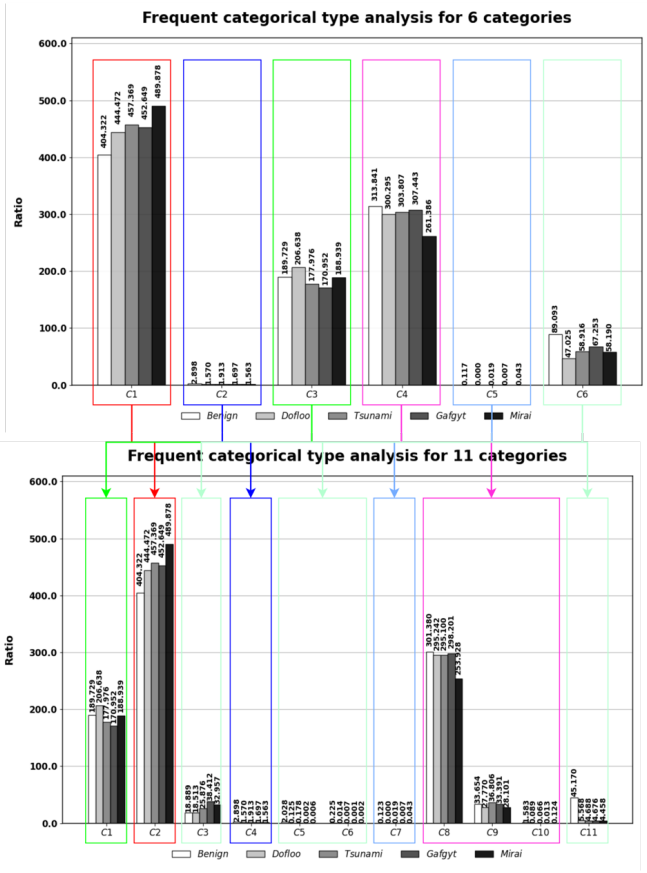


Fig. 1: opcode distribution from 6 category to 11 category

III. Research Approach

A. Opcode categorization

The dataset is denoted by $D = \{P_1, P_2, \ldots, P_N\}$ where P_i is a benign or malware executable file and N is the total of files. The `objdump`[1] is a tool for extracting opcodes from P_i. The extracted opcode sequence is defined as $P_i =< o_1, o_2, \ldots, o_{n_i} >$ where n_i is the number of opcodes in P_i. The approach proposed by Darabian *et al.* [11] is adopted to encode P_i with opcode categories. According to Table I, every opcode o_j of P_i is mapped to c_k in which o_j is included. The opcode category sequence of P_i becomes $C_i =< c_1, c_2, \ldots, c_{n_i} >$.

Table I shows 6 and 11 opcode categories, and each category is divided according to the functional type of the opcode. Table II illustrates how 6 opcode category [11] is expanded to 11 opcode category by reflecting the most frequent functional types. Based on Armv6-M Architecture Reference Manual[2], the 11 opcode category is created from the 6 opcode category that Darabian *et al.* [11] proposed.

C1 is branch instructions. According to their functions in the data-processing instructions, C2, C3, C4, C5, and C6 are divided into five categories. C7 is status register access instructions. According to the load and store instructions, C8, C9, and C10 are divided into three categories. C11 has miscellaneous instructions, including exception-generating instructions.

Figure 1 presents the frequency ratio of 6 and 11 opcode categories in the IoT malware dataset. As 6 opcode category is changed to the 11 opcode category, other instruction is 1/2 times less for benign files and 1/10 times less for other malware. Other instructions are distributed through C3, C5, C6, C7, and C9 in 11 opcode category; most of instructions are moved to C3 and push and pop of C9. The ratio of benign and malware files are similar in C9, while C3 is found 1.5 to 2 times more frequently in malware files than in benign files.

The code sequences of malware variants evolved from the same family tend to be very similar. Opcode based IoT malware detection is performed by comparing the inserted or modified parts of two opcode sequence. Since the number of opcode categories is less than the number of opcodes, it is easier to compare opcode category sequences than opcode sequences and allows for quick comparison.

B. Feature extraction

The entropy histogram transition [20] is introduced to design feature vectors from opcode category sequences. The entropy histogram E_i is computed from C_i within a sliding window of size 256 and stride size 128. If C_i has K windows, then $E_i =< w_1, w_2, \ldots, w_K >$ where $w_j =< e_1, e_2, \ldots, e_\tau >$ and e_k is the entropy value of the k^{th} opcode category frequency within the j^{th} sliding window. Here, τ is 6 or 11, indicating the number of opcode categories (Table I). So, the proposed feature $F_i = [f_{\ell j}]_{L\times\tau}$ of P_i becomes a 2D matrix. The column of F_i represents opcode category indexes and the row of F_i indicates L discretized values. For a small constant δ, F_i is calculated from all windows of E_j and $\ell = 1, 2, \ldots, L$.

$$f_{\ell k} = f_{\ell k} + e_k \text{ if } e_k \in (\delta(\ell-1), \delta\ell] \text{ and } k = 1, 2, \ldots, \tau$$

Each entropy value is mapped to one of L sorted ranges. The value $f_{\ell k}$ accumulates the e_k of all sliding windows if e_k is between $\delta(\ell-1)$ and $\delta\ell$.

The change of the kth opcode changed within the specific sliding window area is caught by $f_{\ell k}$.

The y-axis of the 2D entropy map captures changes in opcode function type (x-axis) within a sliding window that changes over time. It reflects the characteristics of malware with a fixed length. In addition, fixed-size features are required for machine learning analysis.

IV. Experiment

A. Experimental setting

Malwares[3] offered 19,890 IoT benign/malware files executing on ARM CPU structure as a dataset for malware detection. according to Kaspersky's categorization criteria, the gathered malware consists of four families, such as `Dofloo`, `Gafgyt`, `Mirai`, and `Tsunami` as in Table III.

TABLE III: EX1 dataset

Type	Family	No. of Data
Benign	-	2,593
Malware	Dofloo	844
	Gafgyt	8,582
	Mirai	7,404
	Tsunami	467
Total		19,890

TABLE IV: EX2 dataset

Dataset	Type	Family	No. of Data
Training Set	Benign	-	2,126
	Malware	Tsunami	467
Test Set	Benign	-	467
	Malware	Gafgyt	467

TABLE V: EX3 dataset

Dataset	Type	Family	No. of Data
Training Set	Benign	-	2,093
	Malware	Tsunami	467
		Gafgyt	8,582
Test Set	Benign	-	500
	Malware	Mirai	500

[1] https://sourceware.org/binutils/docs/binutils/index.html
[2] https://developer.arm.com/documentation/ddi0419/latest
[3] https://www.malwares.com/

TABLE II: category transition

High level	Low level	6 category	11 category
Branch instruction	-	C3	C1
Data-processing instructions	Standard data-processing instructions	C1	C2
	Shift instructions	C6	C3
	Multiplication instructions	C2	C4
	Packing and unpacking instructions	C6	C5
	Miscellaneous data-processing instructions	C6	C6
Status register access instructions	-	C5	C7
Load and store instruction	Load and store (single) instructions	C4	C8
	Load multiple and store multiple instructions		C9
	Load and store coprocessor instructions		C10
Other instructions	-	C6	C11

TABLE VI: EX4 dataset

Dataset	Type	Family	No of Data
Training Set	Benign	-	2,093
	Malware	Dofloo	844
Test Set	Benign	-	500
	Malware	Tsunami	467
		Gafgyt	8,582
		Mirai	7,404

Depending on the evolutionary relationship among four families, we define the four classification problems for malware detection. Each experiment's dataset is divided into four parts. EX1 dataset is composed of the full dataset to check the robustness of an ML-based malware detection (Table III). EX2 dataset is composed of benign and malware including `Tsunami` and `Gafgyt` (Table IV). Ex2 evaluates whether a model trained on the ancestor `Tsunami` can detect `Gafgyt` as `Tsunami` descent [19], [18]. EX3 dataset is composed of benign and malware including `Tsunami`, `Gafgyt`, and `Mirai` (Table V) in order to test whether an ML model can detect `Mirai` samples when it is trained on benign and malware from `Tsunami` and `Gafgyt`. `Mirai` is known to have evolved from `Tsunami` and `Gafgyt`. EX4 dataset is composed of benign files and maleare including `Dofloo`, `Tsunami`, `Gafgyt`, and `Mirai` (Table VI). In EX4 experiment, we analyze how data learned with unrelated malware (`Dofloo`) affects other malware (`Tsunami`, `Gafgyt`, and `Mirai`).

IoT malware has been updated over time as a result of code reuse and improvement. We analyze whether a model trained with only the ancestor IoT malware family can detect descendant malware. Four classification experiments are conducted based on the dataset configuration. Each experiment uses five features as follows.

- MSP: MSP (maximal sequential pattern) feature was proposed by Darabian et al. [11]. This feature is used to compare to the ones we created (listed above).
- $\mathrm{OCS}_{n=2,c=6}$: OCS (Opcode Category Sequence) is 2-gram feature generated from 6 opcode category.
- $\mathrm{OCS}_{n=2,c=11}$: OCS is 2-gram feature generated from 11 opcode category.
- $\mathrm{EHOC}_{l=6,c=6}$: EHOC (Entropy Histogram for Opcode Category) is a feature generated from 6 opcode category.
- $\mathrm{EHOC}_{l=6,c=11}$: EHOC is a feature generated from 11 opcode category.

ML algorithms are 5-NN, SVM, DT, and RF. As performance indicators, accuracy (ACC), true positive rate (TPR), false positive rate (FPR), AUC-ROC (ROC), and F1-score (F1), and Matthews Correlation Coefficient (MCC, [21]) are utilized [22]. MCC is chosen as a balance measure for unbalanced two classes. MCC values range from -1 to +1, where +1 indicates a perfect prediction, -1 indicates a wrong prediction, and 0 indicates a random prediction. Accuracy, recall, precision, and F1-score can be misleading when evaluating imbalanced classification problems because classifiers tend to predict majority classes.

B. Evaluation and Discussion

Table VII shows the size of a feature vector. The experimental results for EX1 are shown in Table VIII. Dim. (dimension) indicates that low-dimension features with fixed size are used to reduce training time. MSP, $\mathrm{OCS}_{n=2,c=6}$, and $\mathrm{EHOC}_{l=6,c=6}$ achieved greater than 98% ACC across all models. $\mathrm{OCS}_{n=2,c=11}$, and $\mathrm{EHOC}_{l=6,c=11}$, the features generated in the 11 opcode category, had an ACC of over 99%, and FPR was reduced by more than 50% when compared to the 6 opcode category. RF performed best, with a detection rate of 99.8% and FPR of 0.6% in $\mathrm{OCS}_{n=2,c=11}$ while an ACC was 99.7% and FPR of 1.0% for $\mathrm{EHOC}_{l=6,c=11}$.

Table IX shows the experimental results of EX2. The 11 opcode category outperformed the 6 opcode category. In RF, $\mathrm{EHOC}_{l=6,c=6}$ has an ACC of 96.5% while $\mathrm{EHOC}_{l=6,c=11}$ has an ACC of 98.2%. $\mathrm{OCS}_{n=2,c=6}$ has an MCC of 86.7% while $\mathrm{OCS}_{n=2,c=11}$ reaches at an ACC of 99.4%. `Gafgyt` can be detected using features created only from `Tsunami`. This result is consistent with the study that `Tsunami` and `Gafgyt` had the most shared functions [7].

TABLE VII: Dimension for feature vectors

Feature	Dimension
MSP	36
$\mathrm{OCS}_{n=2,c=6}$	36
$\mathrm{OCS}_{n=2,c=11}$	121
$\mathrm{EHOC}_{l=6,c=6}$	36
$\mathrm{EHOC}_{l=6,c=11}$	121

TABLE VIII: Experimental results for EX1

Feature	Model	ACC	TPR	FPR	ROC	F1	MCC
MSP	5-NN	0.988	0.997	0.077	0.985	0.972	0.945
	SVM	0.986	0.991	0.051	0.993	0.969	0.938
	DT	0.987	0.997	0.082	0.953	0.969	0.939
	RF	0.995	0.997	0.018	0.999	0.990	0.980
$\mathrm{OCS}_{n=2,c=6}$	5-NN	0.989	0.999	0.075	0.987	0.975	0.951
	SVM	0.989	0.995	0.052	0.989	0.975	0.950
	DT	0.991	0.998	0.054	0.969	0.980	0.961
	RF	0.997	0.999	0.014	1.000	0.993	0.987
$\mathrm{OCS}_{n=2,c=11}$	5-NN	0.992	0.999	0.053	0.991	0.982	0.964
	SVM	0.992	0.999	0.053	0.991	0.982	0.964
	DT	0.995	0.998	0.024	0.980	0.990	0.979
	RF	0.998	0.999	0.006	1.000	0.996	0.992
$\mathrm{EHOC}_{l=6,c=6}$	5-NN	0.985	0.998	0.104	0.980	0.965	0.931
	SVM	0.986	0.994	0.063	0.990	0.970	0.939
	DT	0.987	0.996	0.073	0.961	0.971	0.942
	RF	0.996	0.998	0.020	1.000	0.990	0.981
$\mathrm{EHOC}_{l=6,c=11}$	5-NN	0.991	0.998	0.058	0.991	0.980	0.960
	SVM	0.991	0.994	0.029	0.992	0.980	0.960
	DT	0.994	0.998	0.034	0.972	0.986	0.972
	RF	0.997	0.998	0.010	1.000	0.993	0.987

TABLE IX: Experimental results for EX2

Feature	Model	ACC	TPR	FPR	ROC	F1	MCC
MSP	5-NN	0.933	0.884	0.019	0.970	0.932	0.869
	SVM	0.930	0.876	0.015	0.987	0.930	0.866
	DT	0.901	0.818	0.015	0.901	0.901	0.814
	RF	0.904	0.807	0.000	0.997	0.903	0.823
$\mathrm{OCS}_{n=2,c=6}$	5-NN	0.984	0.987	0.019	0.993	0.984	0.968
	SVM	0.938	0.878	0.002	0.997	0.938	0.882
	DT	0.914	0.837	0.009	0.915	0.914	0.839
	RF	0.929	0.859	0.000	0.999	0.929	0.867
$\mathrm{OCS}_{n=2,c=11}$	5-NN	0.987	0.989	0.015	0.998	0.987	0.974
	SVM	0.985	0.976	0.006	0.996	0.985	0.970
	DT	0.997	0.998	0.004	0.997	0.997	0.994
	RF	0.997	0.997	0.000	1.000	0.997	0.994
$\mathrm{EHOC}_{l=6,c=6}$	5-NN	0.970	0.976	0.036	0.990	0.970	0.940
	SVM	0.939	0.882	0.004	0.994	0.939	0.884
	DT	0.945	0.906	0.015	0.945	0.945	0.894
	RF	0.965	0.929	0.000	0.999	0.965	0.932
$\mathrm{EHOC}_{l=6,c=11}$	5-NN	0.989	0.991	0.013	0.997	0.989	0.979
	SVM	0.972	0.944	0.000	0.997	0.972	0.946
	DT	0.955	0.921	0.011	0.991	0.955	0.912
	RF	0.982	0.964	0.000	1.000	0.982	0.964

TABLE X: Experimental results for EX3

Feature	Model	ACC	TPR	FPR	ROC	F1	MCC
MSP	5-NN	0.954	0.950	0.042	0.976	0.954	0.908
	SVM	0.966	0.962	0.030	0.982	0.966	0.932
	DT	0.959	0.966	0.048	0.963	0.959	0.918
	RF	0.961	0.934	0.012	0.988	0.961	0.923
$\mathrm{OCS}_{n=2,c=6}$	5-NN	0.970	0.978	0.038	0.984	0.970	0.940
	SVM	0.980	0.982	0.022	0.993	0.980	0.960
	DT	0.923	0.874	0.028	0.919	0.923	0.850
	RF	0.986	0.978	0.006	0.992	0.986	0.972
$\mathrm{OCS}_{n=2,c=11}$	5-NN	0.977	0.976	0.022	0.986	0.977	0.954
	SVM	0.984	0.980	0.012	0.997	0.984	0.964
	DT	0.982	0.970	0.006	0.982	0.982	0.964
	RF	0.986	0.974	0.002	0.989	0.986	0.972
$\mathrm{EHOC}_{l=6,c=6}$	5-NN	0.937	0.940	0.066	0.970	0.937	0.874
	SVM	0.965	0.966	0.036	0.991	0.965	0.930
	DT	0.931	0.904	0.042	0.964	0.931	0.863
	RF	0.968	0.942	0.006	0.984	0.968	0.937
$\mathrm{EHOC}_{l=6,c=11}$	5-NN	0.974	0.970	0.022	0.989	0.986	0.972
	SVM	0.981	0.976	0.014	0.993	0.981	0.962
	DT	0.979	0.974	0.016	0.975	0.979	0.958
	RF	0.985	0.974	0.004	0.985	0.985	0.970

Table X shows the experimental results of EX3. In both DT and RF, $\mathrm{OCS}_{n=2,c=11}$ has an ACC of 98.0%. Overall, the results of this experiment show that `Mirai` can be detected using a dataset of `Tsunami` and `Gafgyt` only. To put it another way, it demonstrates that `Tsunami` or `Gafgyt` evolved into `Mirai`. In terms of FPR, the experimental result of EX3 (1.23%) is slightly better than that of EX1 (3.08%), yet it is slightly lower than that of EX2 (0.61%). The FPR mean of $\mathrm{EHOC}_{l=6,c=11}$ and $\mathrm{OCS}_{n=2,c=11}$ is 3.08% for EX1, 0.61% for EX2, and 1.23% for EX3. This argues that malware is closely related in terms of code sharing and evolutionary development. This finding is that malware is closely related in terms of code sharing. `Tsunami` and `Gafgyt` share more code than `Tsunami` and `Mirai` [7]. There are 189 functions common by `Tsunami` and `Gafgyt`, 63 functions common by `Tsunami` and `Mirai`, and 115 functions common by `Gafgyt` and `Mirai` [7]. This analysis reveals the effect of `Mirai` on the evolution of `Tsunami` and `Gafgyt`. According to the results of the EX2 and EX3 experiments, it is a verified fact that descendant malware can be detected as ancestor malware.

TABLE XI: Experimental results for EX4

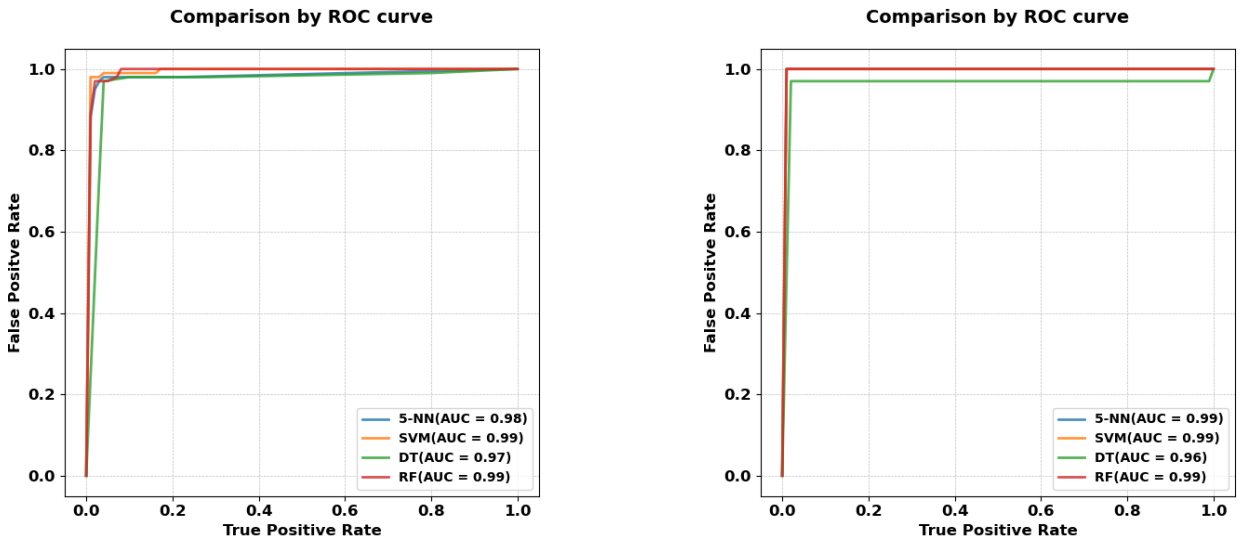
Feature	Model	ACC	TPR	FPR	ROC	F1	MCC
MSP	5-NN	0.029	0.000	0.004	0.496	0.029	-0.043
	SVM	0.029	0.000	0.000	0.910	0.029	0.000
	DT	0.029	0.000	0.004	0.498	0.029	-0.043
	RF	0.030	0.000	0.000	0.471	0.029	0.002
$\mathrm{OCS}_{n=2,c=6}$	5-NN	0.030	0.000	0.002	0.497	0.029	-0.017
	SVM	0.030	0.001	0.004	0.836	0.029	-0.017
	DT	0.030	0.000	0.000	0.500	0.029	0.002
	RF	0.030	0.000	0.000	0.473	0.029	0.002
$\mathrm{OCS}_{n=2,c=11}$	5-NN	0.030	0.000	0.004	0.494	0.029	-0.034
	SVM	0.030	0.001	0.002	0.942	0.029	-0.008
	DT	0.030	0.000	0.000	0.500	0.029	0.001
	RF	0.030	0.000	0.000	0.469	0.029	0.002
$\mathrm{EHOC}_{l=6,c=6}$	5-NN	0.030	0.000	0.004	0.498	0.029	-0.038
	SVM	0.029	0.000	0.006	0.785	0.029	-0.052
	DT	0.029	0.000	0.000	0.500	0.029	0.000
	RF	0.030	0.000	0.000	0.473	0.029	0.002
$\mathrm{EHOC}_{l=6,c=11}$	5-NN	0.030	0.000	0.002	0.495	0.029	-0.052
	SVM	0.029	0.000	0.006	0.859	0.029	-0.052
	DT	0.030	0.000	0.002	0.499	0.029	-0.024
	RF	0.030	0.000	0.000	0.477	0.029	0.002

We analyze whether a model trained on `Dofloo` can detect `Tsunami`, `Gafgyt`, and `Mirai` (Table XII). The detection rates, on the other hand, were close to 0%. The reason is that `Dofloo` belonging to the `Spike` family has no correlation with `Tsunami`, `Gafgyt`, and `Mirai` [18]. Although the malware `Dofloo` is unrelated to `Tsunami`, `Gafgyt`, or `Mirai`, In 5-NN, $\mathrm{OCS}_{n=2,c=11}$ and $\mathrm{EHOC}_{n=2,c=11}$ have an ACC of more than 91%.

TABLE XII: EX5 dataset

Dataset	Type	Family	No. of Data
Training Set	Benign	-	2,093
	Malware	Tsunami	467
		Gafgyt	8,582
		Mirai	7,404
Test Set	Benign	-	500
	Malware	Dofloo	844

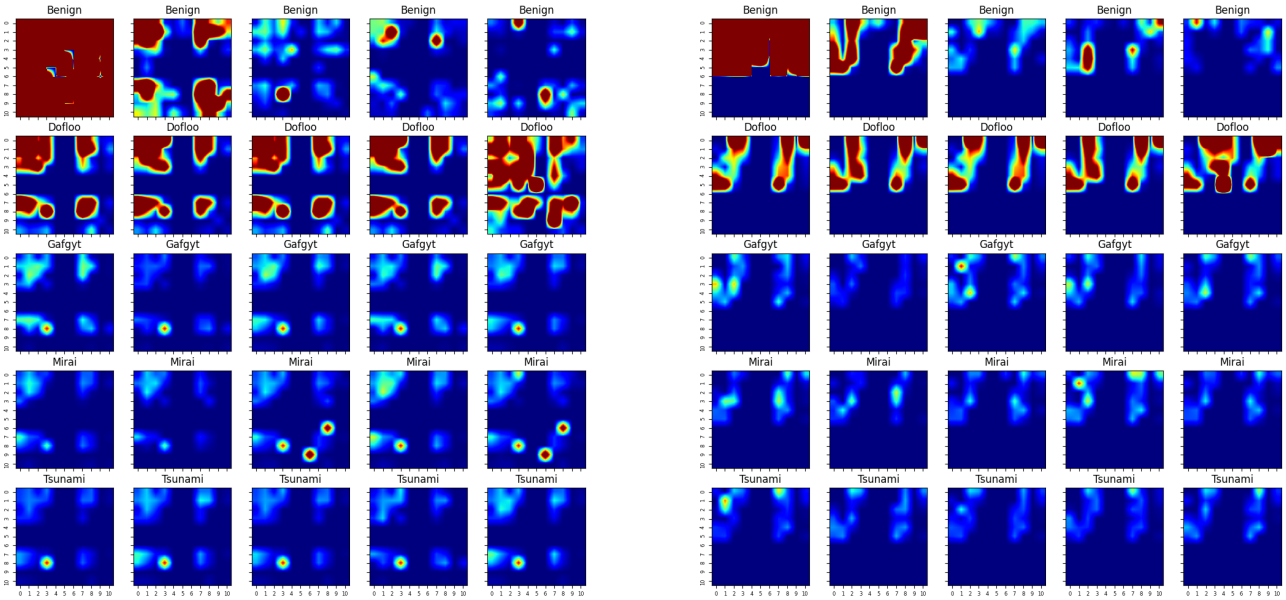
ROC (Receiver Operating Characteristic) graphs examine



(a) ROC for dataset EX2

(b) ROC for dataset for EX3

Fig. 2: ROC analysis for EX2 and EX3



(a) 2-gram: $\mathrm{OCS}_{n=2,c=11}$

(b) Entropy histogram: $\mathrm{EHOC}_{l=6,c=11}$

Fig. 3: Feature visualization over 11 opcode category

the trade-off between FPR and TPR in terms of malware and benign instances in the test datasets. In Figure 2, the ROC analysis for EX2 and EX3 reveals that the proposed approach is effective for detection analysis because it does not cause overfitting and the effect of class imbalance. The AUC value (close to 0.99) of 5-NN, SVM and RF is higher than that of DT: 0.97 for EX2 and 0.96 for EX3. In both graphs, the AUC value of EX3 approaches 1.0 as the similar trends occur for the chosen classifiers except for DT. This result indicates that the proposed feature method is suitable for IoT malware analysis.

C. Feature visualization and Analysis

To demonstrate the relationship between the detection result and the proposed feature, a 2D graph of the proposed feature is employed. Figure 3 depicts the feature patterns of benign and malware. Figure 3(a) represents the image of 2-grams while Figure 3(b) shows those of entropy histograms for opcode sequence. It clearly visualizes the similarities and differences between IoT malware. Malware and benign can be visually distinguished in 3(a) and (b). Each malware has a similar pattern. `Gafgyt`and `Tsunami` are more similar than `Mirai` as in EX2 in Table IX. `Gafgyt`, `Mirai`, and

Tsunami are similar as the experimental results in EX3 in Table X. Dofloo and other malware (i.e., Gafgyt, Mirai, and Tsunami) can easily be separated as the experimental results in EX4 in Table XI.

V. Conclusion

This paper performed IoT malware detection and malware family detection according to evolution. The proposed feature is opcode category sequences derived from an opcode sequence, which allows for simple comparison of two malware files. However, for machine learning analysis, the training feature of malware with varying sizes must be configured with a fixed length. A training feature pattern was constructed by adopting an entropy histogram for a fixed-length feature from a variable-length category sequence. The entropy histogram feature can include a partial change within the sliding window, and provides a fixed-length training feature according to a predefined level. The 2D visualization of the entropy histogram, on the other hand, has the advantage of allowing for malware family association. The proposed training feature outperformed the 2-gram feature and MSPs in the IoT malware detection experiment.

IoT malware detection was conducted in accordance with the malware's evolutionary development. Various classification problems were constructed based on the evolutionary order from Dofloo, Gafgyt, Mirai and Tsunami. As a result of the experiment, the model trained with the ancestor family was able to detect descendant malware families. However, the model trained only with benign and Dofloo performed poorly in detecting other malware. These results showed that malware that has evolved from prior malware can be detected while emerging malware is still difficult to detect. Thus, it is necessary to learn malware according to the evolution system in order to build a robust IoT malware detection system.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00197, Trust-based cyber security platform for smart facility environment).

References

[1] Sachchidanand Singh and Nirmala Singh. Internet of things (iot): Security challenges, business opportunities & reference architecture for e-commerce. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1577–1581. Ieee, 2015.

[2] Xingbin Jiang, Michele Lora, and Sudipta Chattopadhyay. An experimental analysis of security vulnerabilities in industrial iot devices. *ACM Transactions on Internet Technology (TOIT)*, 20(2):1–24, 2020.

[3] Elisa Bertino and Nayeem Islam. Botnets and internet of things security. *Computer*, 50(2):76–79, 2017.

[4] Artur Marzano, David Alexander, Osvaldo Fonseca, Elverton Fazzion, Cristine Hoepers, Klaus Steding-Jessen, Marcelo H. P. C. Chaves, Ítalo Cunha, Dorgival Guedes, and Wagner Meira. The evolution of bashlite and mirai iot botnets. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00813–00818, 2018.

[5] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[6] *DDoS attacks in Q2 2021*, (accessed December 7, 2021). https://securelist.com/ddos-attacks-in-q2-2021/103424/.

[7] Emanuele Cozzi, Pierre-Antoine Vervier, Matteo Dell'Amico, Yun Shen, Leyla Bilge, and Davide Balzarotti. The tangled genealogy of iot malware. In *Annual Computer Security Applications Conference*, pages 1–16, 2020.

[8] Ruchi Vishwakarma and Ankit Kumar Jain. A survey of ddos attacking techniques and defence mechanisms in the iot network. *Telecommunication systems*, 73(1):3–25, 2020.

[9] Andrei Costin and Jonas Zaddach. Iot malware: Comprehensive survey, analysis framework and case studies. *BlackHat USA*, 2018.

[10] BooJoong Kang, Suleiman Yerima, Sakir Sezer, and Kieran Mclaughlin. N-gram opcode analysis for android malware detection. *International Journal on Cyber Situational Awareness*, 1:231–255, 11 2016.

[11] Hamid Darabian, Ali Dehghantanha, Sattar Hashemi, Sajad Homayoun, and Kim-Kwang Raymond Choo. An opcode-based technique for polymorphic internet of things malware detection. *Concurrency and Computation: Practice and Experience*, 32(6):e5173, 2020.

[12] Huy-Trung Nguyen, Quoc-Dung Ngo, and Van-Hoang Le. Iot botnet detection approach based on psi graph and dgcnn classifier. In *2018 IEEE International Conference on Information Communication and Signal Processing (ICICSP)*, pages 118–122, 2018.

[13] Hisham Alasmary, Aminollah Khormali, Afsah Anwar, Jeman Park, Jinchun Choi, Ahmed Abusnaina, Amro Awad, Daehun Nyang, and Aziz Mohaisen. Analyzing and detecting emerging internet of things malware: A graph-based approach. *IEEE Internet of Things Journal*, 6(5):8977–8988, 2019.

[14] Tzu-Ling Wan, Tao Ban, Shin-Ming Cheng, Yen-Ting Lee, Bo Sun, Ryoichi Isawa, Takeshi Takahashi, and Daisuke Inoue. Efficient detection and classification of internet-of-things malware based on byte sequences from executable files. *IEEE Open Journal of the Computer Society*, 1:262–275, 2020.

[15] Jiawei Su, Danilo Vargas Vasconcellos, Sanjiva Prasad, Daniele Sgandurra, Yaokai Feng, and Kouichi Sakurai. Lightweight classification of iot malware based on image recognition. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 02, pages 664–669, 2018.

[16] Farrukh Shahzad and Muddassar Farooq. Elf-miner: Using structural knowledge and data mining methods to detect new (linux) malicious executables. *Knowl. Inf. Syst.*, 30:589–612, 03 2012.

[17] Quoc-Dung Ngo, Huy-Trung Nguyen, Van-Hoang Le, and Doan-Hieu Nguyen. A survey of iot malware and detection methods based on static features. *ICT Express*, 6(4):280–286, 2020.

[18] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi. Analysis of ddos-capable iot malwares. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 807–816. IEEE, 2017.

[19] Benjamin Vignau, Raphaël Khoury, Sylvain Hallé, and Abdelwahab Hamou-Lhadj. The evolution of iot malwares, from 2008 to 2019: Survey, taxonomy, process simulator and perspectives. *Journal of Systems Architecture*, page 102143, 2021.

[20] Seoungyul Euh, Hyunjong Lee, Donghoon Kim, and Doosung Hwang. Comparative analysis of low-dimensional features and tree-based ensembles for malware detection systems. *IEEE Access*, 8:76796–76808, 2020.

[21] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PLOS ONE*, 12(6):1–17, 06 2017.

[22] John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press, 2 edition, 2020.