KIPS Trans. Softw. and Data Eng.
Vol.11, No.5 pp.197~202
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2022.11.5.197

IoT Malware Detection and Family Classification Using Entropy Time Series Data Extraction and Recurrent Neural Networks

Youngho Kim† · Hyunjong Lee†† · Doosung Hwang†††

ABSTRACT

IoT (Internet of Things) devices are being attacked by malware due to many security vulnerabilities, such as the use of weak IDs/passwords and unauthenticated firmware updates. However, due to the diversity of CPU architectures, it is difficult to set up a malware analysis environment and design features. In this paper, we design time series features using the byte sequence of executable files to represent independent features of CPU architectures, and analyze them using recurrent neural networks. The proposed feature is a fixed-length time series pattern extracted from the byte sequence by calculating partial entropy and applying linear interpolation. Temporary changes in the extracted feature are analyzed by RNN and LSTM. In the experiment, the IoT malware detection showed high performance, while low performance was analyzed in the malware family classification. When the entropy patterns for each malware family were compared visually, the Tsunami and Gafgyt families showed similar patterns, resulting in low performance. LSTM is more suitable than RNN for learning temporal changes in the proposed malware features.

Keywords : Internet of Things, Machine Learning, Malware Detection, Malware Family Classification

엔트로피 시계열 데이터 추출과 순환 신경망을 이용한 IoT 악성코드 탐지와 패밀리 분류

김 영 호† · 이 현 종†† · 황 두 성†††

요 약

IoT (Internet of Things) 장치는 취약한 아이디/비밀번호 사용, 인증되지 않은 펌웨어 업데이트 등 많은 보안 취약점을 보여 악성코드의 공격 대상이 되고 있다. 그러나 CPU 구조의 다양성으로 인해 악성코드 분석 환경 설정과 특징 설계에 어려움이 있다. 본 논문에서는 CPU 구조와 독립된 악성코드의 특징 표현을 위해 실행 파일의 바이트 순서를 이용한 시계열 특징을 설계하고 순환 신경망을 통해 분석한다. 제안하는 특징은 바이트 순서의 부분 엔트로피 계산과 선형 보간을 통한 고정 길이의 시계열 패턴이다. 추출된 특징의 시계열 변화는 RNN과 LSTM으로 학습시켜 분석한다. 실험에서 IoT 악성코드 탐지는 높은 성능을 보였지만, 패밀리 분류는 비교적 성능이 낮았다. 악성코드 패밀리별 엔트로피 패턴을 시각화하여 비교했을 때 Tsunami와 Gafgyt 패밀리가 유사한 패턴을 나타내 분류 성능이 낮아진 것으로 분석되었다. 제안된 악성코드 특징의 데이터 간 시계열 변화 학습에 RNN보다 LSTM이 더 적합하다.

키워드 : 사물 인터넷, 기계학습, 악성코드 탐지, 악성코드 패밀리 분류

1. 서 론

사물 인터넷(IoT, Internet of Things)은 고유 식별자를 가진 컴퓨팅 장치들이 유무선 인터넷으로 연결되어 사람의 개입 없이 상호작용하는 시스템이다[1]. IoT 장치는 내장형 시스템(embedded system)을 사용해 데이터를 수집하고 서버로 전송하며 서버에서 분석된 정보를 바탕으로 기능을 수행한다. 스마트 홈(smart home), 헬스케어(healthcare), 스마트 팩토리(smart factory) 등 다양한 분야에서 IoT 장치가 활용되고 있다. 2030년에는 IoT 장치 사용량이 현재보다 3배 이상 증가할 것으로 예측된다[2].

IoT 장치는 취약한 아이디/비밀번호 사용, 인증되지 않은 펌웨어 업데이트, 암호화 통신(secure socket layer)의 부재 등 많은 보안 취약점으로 인해 악성코드의 공격 대상이 되고 있다[3]. 2016년 미라이 봇넷(Mirai botnet)은 10만 개 이상의 IoT 장치를 감염시켜 대규모 분산 서비스 거부 공격(distributed denial of service)을 수행했다. 미국의 DNS (Domain Name System) 관리 회사인 Dyn의 서버를 공격

※ 이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No.2019-0-00197, 스마트 퍼실러티 환경보호를 위한 신뢰기반 사이버 보안 플랫폼).
† 준 회 원 : 단국대학교 컴퓨터학과 석사과정
†† 정 회 원 : ㈜케이사인 보안기술연구소 주임연구원
††† 종신회원 : 단국대학교 소프트웨어학과 교수
Manuscript Received : September 7, 2021
First Revision : November 15, 2021
Accepted : November 17, 2021
* Corresponding Author : Doosung Hwang(dshwang@dankook.ac.kr)

해 트위터, 넷플릭스 등의 웹사이트를 다운시켰다[4]. 최근에는 공개된 미라이 봇넷 소스 코드[5]를 이용한 변종 악성코드가 유포되고 있다[6].

악성코드 탐지 기법은 서명 기반 탐지(signature-based detection)와 이상 기반 탐지(anomaly-based detection)가 있다[7]. 서명 기반 탐지는 이전에 탐지된 악성코드에서 고유 서명(signature)을 추출하고 의심스러운 파일의 서명과 대조해 악성코드 여부를 판단한다. 분석된 악성코드 정보가 많을수록 탐지율이 높아지지만, 신·변종 공격에 취약하다. 이상 기반 탐지는 정상과 악성 행위 분석을 통해 탐지 규칙을 도출한다. 신·변종 공격 탐지가 가능하지만, 행위 규칙의 범위를 정의하기 어려워 오탐률이 높다.

기존 탐지 기법의 문제를 해결하기 위해 기계학습(machine learning)을 이용한 악성코드 탐지 기법이 사용되고 있다. 파일 구조, 악성 행위 분석 등을 통해 악성코드의 특징을 정의하고 분류 모델을 학습한다. 모델은 정의된 특징에 대해 분류 규칙을 도출하거나 유사도를 기반으로 분류에 적합한 파라미터를 자동으로 찾는다.

IoT 장치는 구조의 다양성으로 인해 악성코드 분석에서 높은 복잡도를 갖는다[8]. 대부분의 개인용 컴퓨터(personal computer)는 x86 CPU 구조를 가지며 악성코드 분석 시스템도 x86 구조를 기반으로 설계된다. 그러나 IoT 장치는 MIPS, ARM, PowerPC 등 다양한 구조가 존재해 CPU 구조별로 분석 환경 구성과 악성코드 특징 정의가 요구된다. 악성코드 분석의 복잡도를 낮추기 위해 CPU 구조와 독립된 특징 추출 기법의 연구가 필요하다.

본 논문은 기계학습 기반 IoT 악성코드 탐지와 패밀리 분류를 수행한다. 실행 파일의 바이트 순서(byte sequence) 정보를 이용해 CPU 구조에 무관한 악성코드 특징 추출 방법을 제안한다. 정의된 특징을 순환 신경망(recurrent neural network) 모델로 학습해 악성코드를 분류한다. 논문의 구성은 다음과 같다. 2절에서 IoT 악성코드 분류와 관련된 연구를 서술한다. 3절에서는 악성코드 특징 추출 방법을 제안하고 분류 모델을 소개한다. 4절은 학습된 모델의 성능을 평가하며 마지막 절에서 향후 연구 방향과 결론을 제시한다.

2. 관련 연구

악성코드 특징 추출 방법은 정적 분석(static analysis)과 동적 분석(dynamic analysis)이 있다[9]. 정적 분석은 대상 파일의 실행 없이 바이트 순서, opcode(operation code) 등 실행 파일의 구조적 특징을 분석한다. 동적 분석은 VM(Virtual Machine)과 같이 제어되는 환경에서 대상 파일을 실행하여 악성 행위를 분석하며 네트워크 트래픽, 메모리 사용 정보 등의 특징을 추출한다.

T. L. WAN 외 7명은 실행 파일 진입점(entry point)의 바이트 순서를 이용해 악성코드 특징을 정의했다[10]. ELF(Executable and Linkable Format) 파일 헤더의 진입점 정보를 이용해 바이트 순서를 추출하고 n-gram을 적용했다. n-gram을 통해 부분 순서(subsequence)에 대한 빈도를 계산하여 특징을 구성했다. 8가지 CPU 구조에서 수집한 정상 파일과 악성코드를 이용해 실험을 진행했다. 악성코드 탐지에서 SVM(Support Vector Machine) 모델이 99.96%의 정확도(accuracy)를 보였다. 악성코드 패밀리 분류는 98.47%의 정확도를 얻었지만, 정밀도(precision)와 재현율(recall)이 약 91.0%로 비교적 낮았다.

H. Darabian 외 4명은 opcode의 최대 순서 패턴(maximal sequential pattern)을 이용해 ARM 구조의 다형성(polymorphic) 악성코드 탐지를 수행했다[11]. 맵리듀스(MapReduce) 기반 MG-FSM[12]을 사용하여 최대 순서 패턴을 추출했다. Opcode를 기능(functionality)에 따라 6개의 범주(category)로 분류하고 최대 순서 패턴에서 나타나는 범주 변화에 대한 빈도를 계산해 특징을 구성했다. 기존 악성코드 탐지에서 k-NN(k-Nearest Neighbor)이 99.81%의 정확도를 보였으며 다형성 악성코드 탐지는 SVM 모델이 약 82.0% 이상의 정확도를 보고했다.

J. Jeon 외 2명은 중첩된 클라우드 기반 가상 환경(nested cloud-based virtual environment)에서 동적 분석을 수행했다[13]. 메모리, 시스템 호출(system call), 네트워크, 파일 시스템, 프로세스 정보를 통합하여 하나의 이미지 채널(channel)을 구성했다. 악성 행위 빈도, 메모리 사용 정보와 함께 3차원 이미지를 표현하고 이중 선형 보간법(bilinear interpolation)을 이용해 고정 크기로 변환했다. 512×512 크기의 이미지를 ZFNet[14] 모델로 학습해 99.28%의 악성코드 탐지 정확도를 얻었다.

동적 분석은 분석 환경 구축이 복잡하며 연산 자원의 소모가 크다. Opcode를 이용한 정적 분석은 특정 CPU 구조에 국한되어 있어 IoT 악성코드 분석에서 높은 복잡도를 갖는다. 실행 파일의 바이트 순서를 이용한 특징은 CPU 구조와 독립된 특징 표현이 가능하다. 그러나 n-gram을 이용한 특징은 $n=2$일 때 최대 65,536의 크기를 갖는다. 기계학습에서 특징 집합의 크기가 클수록 분류 모델의 복잡도가 증가하며 과적합(overfitting) 현상이 발생할 수 있다.

3. 제안 방법

제안하는 특징은 바이트 순서의 엔트로피(entropy) 정보를 이용한다. 엔트로피는 바이트 순서에서 나타나는 값들의 불확실성(uncertainty)을 의미한다[15]. 같은 바이트 값이 많을수록 엔트로피가 낮아지며 무작위성(randomness)이 클수록 높은 엔트로피 값을 갖는다. 실행 파일은 .text, .data 등 다수의 섹션(section)으로 구성되며 각 섹션은 파일마다 크기와 엔트로피 수준(level)에서 차이를 나타낸다[16]. 바이트 순서에 대한 부분 엔트로피를 계산하여 1차원의 엔트로피 순서 정보를 추출하고 선형 보간법(linear interpolation)을 적용해 고정 길이의 패턴을 정의한다. 선형 보간을 통해 특징

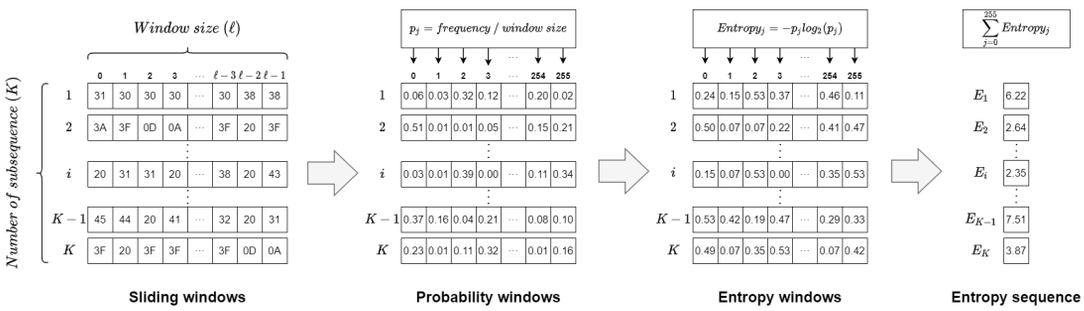


Fig. 1. Process of Entropy Sequence Extraction

집합의 크기를 조절한다. 정의된 엔트로피 패턴의 시계열 분석을 위해 순환 신경망 모델을 이용한다.

3.1 특징 추출

바이트 부분 순서의 엔트로피 값을 계산해 특징을 표현한다. 실행 파일 16진수 바이트 순서 $B=<b_1, b_2, .., b_k, .., b_N>$가 있다. b_k는 $0 \sim 255$범위의 값을 갖는 k번째 바이트이며 N은 실행 파일의 크기와 같다. 슬라이딩 윈도우(sliding window) 기법을 통해 l크기의 윈도우를 b_1부터 b_{N-l+1}까지 일정 간격으로 이동하면서 부분 순서 정보를 추출한다. 부분 순서에서 나타나는 바이트별 빈도율 $p_j (j=0, 1, .., 255)$을 이용하여 엔트로피 E를 계산한다[Equation (1)]. Fig. 1은 부분 바이트 순서에서 엔트로피를 계산하는 과정이다. K개의 엔트로피 $<E_1, E_2, .., E_K>$는 실행 파일 구성에 대한 1차원의 엔트로피 순서(sequence) 정보를 나타낸다.

$$E = -\sum_{j=0}^{255} p_j \log_2(p_j) \tag{1}$$

엔트로피 순서 정보의 길이 K는 슬라이딩 윈도우 크기와 적용 간격, 실행 파일의 크기에 따라 가변적으로 나타난다. 고정 길이의 엔트로피 패턴을 정의하기 위해 선형 보간법을 적용한다. 선형 보간법은 두 점 (x_0, y_0), (x_1, y_1) 사이의 값 (x, y)을 추정할 때 주어진 x에 대한 추정값 y를 두 점의 직선 거리에 따라 선형적으로 결정한다[Equation (2)].

$$y = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0} \tag{2}$$

$1 \sim K$ 범위에서 균일한 간격으로 L개의 x값을 선택하고 선형 보간법을 통해 해당 지점의 엔트로피 값을 계산하여 L 길이의 엔트로피 패턴을 구성한다(Fig. 2).

3.2 순환 신경망

순환 신경망 모델은 순환 구조를 통해 현재 데이터를 학습할 때 이전 시간에 대한 정보를 활용한다[17]. 시간 t의 출력 h_t은 입력값 x_t와 시간 $t-1$의 출력 h_{t-1}에 의해서 결정되며 입력 순서 정보 $(x_0, x_1, .., x_t, ..)$에 대응하는 같은 길이의 출력

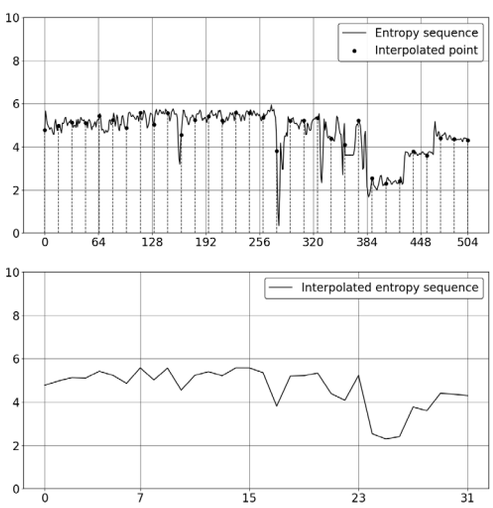


Fig. 2. Example of Linear Interpolation ($L=32$)

순서 정보 $(h_0, h_1, .., h_t, ..)$가 생성된다. 순환 신경망 모델은 RNN과 LSTM(Long Short-Term Memory)이 있다. RNN은 현재 입력값에 대한 가중치 W_x와 이전 시간 출력의 가중치 W_h, 편향 b를 학습한다[Equation (3)].

$$h_t = \tanh(h_{t-1} W_h + x_t W_x + b) \tag{3}$$

LSTM은 순서 정보의 장기 의존성 학습을 위해 RNN 구조에 기억 셀(memory cell)과 게이트(gate)를 함께 구성한다. 기억 셀 c_t는 처음부터 시간 t까지의 데이터에서 나타난 정보를 의미한다. 게이트는 입력값의 활성화를 제어하며 입력, 출력, 망각 게이트가 있다. 망각 게이트 f는 기억 셀 c_{t-1}에서 가중치에 따라 불필요한 정보를 제거하며 입력 게이트 i는 기억 셀 g의 활성화를 제어한다. 출력 게이트 o는 활성화 함수가 적용된 출력 h_t의 활성도를 조절한다[Equation (4-9)].

$$f = \sigma(h_{t-1} W_h^{(f)} + x_t W_x^{(f)} + b^{(f)}) \tag{4}$$

$$g = \tanh(h_{t-1} W_h^{(g)} + x_t W_x^{(g)} + b^{(g)}) \tag{5}$$

$$i = \sigma(h_{t-1} W_h^{(i)} + x_t W_x^{(i)} + b^{(i)}) \tag{6}$$

$$o = \sigma(h_{t-1} W_h^{(o)} + x_t W_x^{(o)} + b^{(o)}) \quad (7)$$

$$c_t = f \odot c_{t-1} + g \odot i \quad (8)$$

$$h_t = o \odot \tanh(c_t) \quad (9)$$

σ : *Sigmoid function*, $\odot$: *Hadamard product*

두 모델의 성능을 비교 분석하여 악성코드 분류 모델을 선택한다. 실험에 사용된 RNN과 LSTM 모델은 1개의 순환 계층과 2개의 완전 연결 층(FC, Fully Connected layer)으로 구성했다(Table 1). IoT 장치는 저사양 하드웨어로 구성되어 연산 자원의 제약이 존재한다[8]. 메모리 자원 소모와 연산량을 줄이기 위해 순환 계층과 완전 연결 층 개수를 최소화했다. 순환 계층에서 엔트로피 패턴의 길이만큼 연산이 반복되며 최종 출력이 완전 연결 층을 통해 클래스별 예측 확률값으로 변환된다.

Table 1. RNN/LSTM Model Architecture

Layer	Hyper-parameter	
	Malware detection	Family classification
RNN/LSTM	Units=8, Activation=tanh	
FC	Units=6, Activation=relu	
FC (output)	Units=2 Activation=softmax	Units=4 Activation=softmax
Optimizer	Adam(learning rate=0.001)	

4. 실험 및 결과

악성코드 탐지와 패밀리 분류 실험은 멀웨어스닷컴[18]에서 수집된 데이터 집합을 이용했다. ARM, MIPS 등 6가지 CPU 구조에서 실행되는 ELF 파일이며 악성코드는 4개의 패밀리가 존재한다. 정상 파일과 악성코드 패밀리는 안티바이러스 소프트웨어 Kaspersky[19]의 분석 결과를 기반으로 구분했다. Table 2는 수집된 데이터 집합의 구성이다.

모델 평가는 5식 교차 검증(5-way cross validation)을 수행했으며 성능 지표는 정확도, 정밀도, 재현율, F1-score를 사용했다. 실험에 사용된 데이터 집합은 클래스 불균형이 존재하여 다수 데이터를 갖는 클래스에 대해 모델 성능이 편향되어 계산될 수 있다. 따라서 클래스별 성능을 평가하고 이에 대한 평균 성능을 계산한다.

특징 추출을 위한 슬라이딩 윈도우 크기는 사전 실험에서 가장 높은 분류 성능을 보인 512로 설정했으며 적용 간격은 윈도우 크기의 반으로 선택했다. 엔트로피 패턴의 길이에 따른 모델의 분류 성능을 비교 분석한다. Fig. 3은 엔트로피 패턴 길이(L)별 악성코드 탐지와 패밀리 분류 성능 변화 비교이다. 악성코드 탐지에서 RNN은 $L=8$일 때 정확도 97.6%, F1-score 94.1%의 성능을 얻었다. LSTM의 경우 가장 높은 탐지 성능을 보인 패턴 길이는 128로 97.8%의 정확도와 94.5%의 F1-score를 보였다. RNN은 엔트로피 패턴의 길이

Table 2. Dataset Information

No	Type	Family	CPU Architecture						Num. of data	
			ARM	x86	x86-64	MIPS	PowerPC	SPARC		
1	Benign	-	3,176	1,591	2,486	720	176	169	8,318	
2	Malware	Gafgyt	9,325	7,511	3,743	7,276	3,375	2,851	34,081	62,562
3		Mirai	9,460	3,641	296	5,301	2,543	2,049	23,290	
4		Dofloo	1,071	1,710	52	183	0	0	3,016	
5		Tsunami	543	617	301	468	167	79	2,175	
Total			23,575	15,070	6,878	13,948	6,261	5,148	70,880	

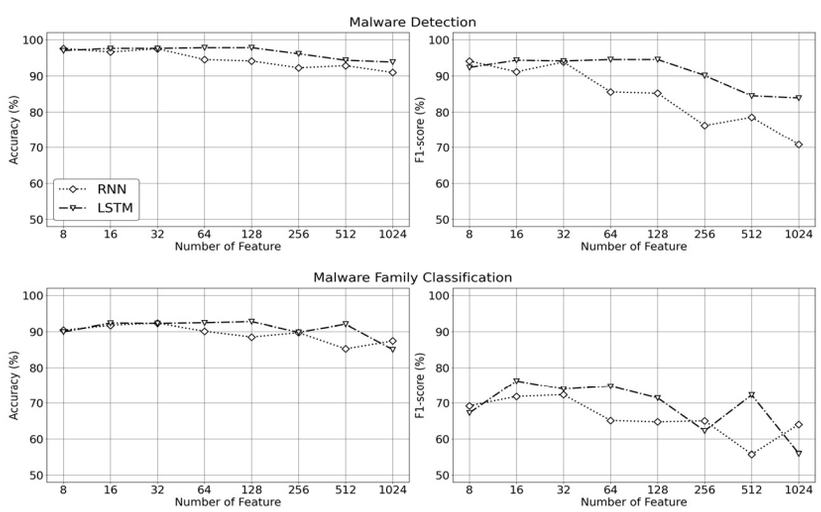


Fig. 3. Performance of Malware Detection and Family Classification

Table 3. Malware Detection Performance

Metric \ Model		RNN	LSTM
Len. of Entropy Pattern (L)		8	128
Train	Accuracy	0.977	0.978
	Precision	0.955	0.958
	Recall	0.934	0.933
	F1-score	0.944	0.945
Test	Accuracy	0.976	0.978
	Precision	0.956	0.963
	Recall	0.928	0.928
	F1-score	0.941	0.945

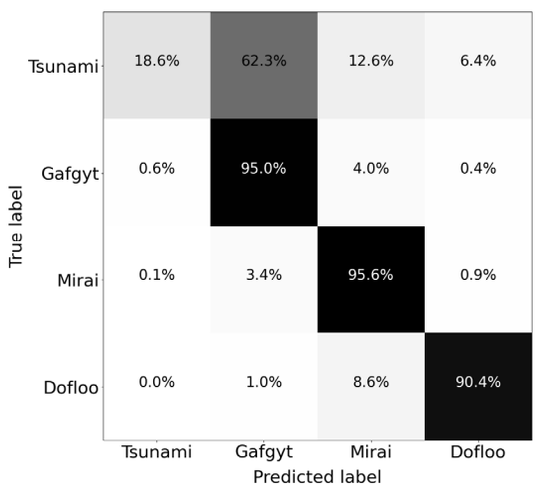


Fig. 4. Confusion Matrix for Malware Family Classification

가 길어질수록 탐지 성능이 감소했다. LSTM은 길이가 길어질수록 탐지 성능이 점차 향상되었지만, 128보다 길어지면 성능이 감소했다. Table 3은 RNN과 LSTM의 악성코드 탐지 최대 성능 비교이다.

패밀리 분류는 악성코드 탐지보다 비교적 낮은 성능을 보였다. RNN의 경우 $L=32$일 때 정확도 92.3%와 F1-score 72.3%로 가장 높은 성능을 얻었으며 L이 증가할수록 분류 성능이 점차 감소했다. LSTM은 $L=16$일 때 92.3%의 정확도와 76.2%의 F1-score 성능을 보였다.

Fig. 4는 악성코드 패밀리 분류에 대한 LSTM 모델의 오차 행렬(confusion matrix)이다. Mirai, Gafgyt, Dofloo 패밀리는 높은 분류 성능을 얻었지만, 대부분의 Tsunami 악성코드가 Gafgyt 패밀리로 오분류되어 전체적으로 낮은 성능을 보고했다. 악성코드 패밀리별 엔트로피 패턴을 시각화하여 비교했을 때 Tsunami와 Gafgyt 패밀리가 유사한 패턴으로 나타났다(Fig. 4). 모델 학습에 사용된 Gafgyt 악성코드 수는 27,264개로 1,740개의 Tsunami보다 현저히 많다. 모델 학습이 Gafgyt 패밀리의 엔트로피 패턴에 편향되어 유사한 패턴을 보이는 Tsunami 악성코드가 오분류되는 것으로 분석

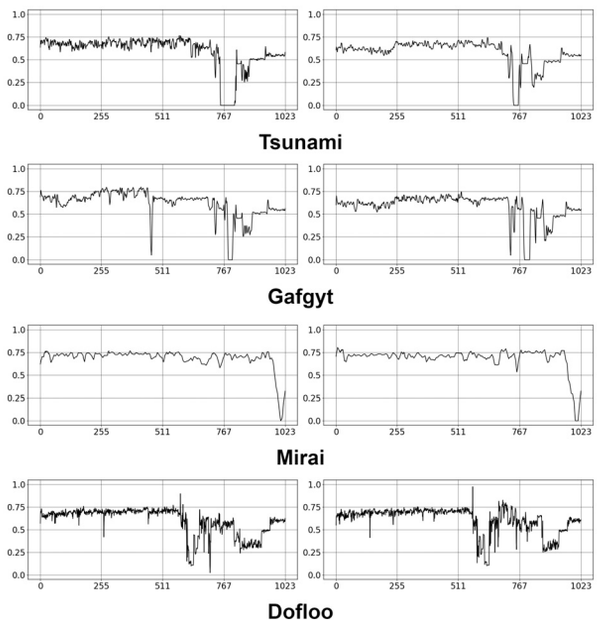


Fig. 5. Visualization of Entropy Sequence for Malware Family

된다. Dofloo 패밀리는 Tsunami와 비슷한 학습 데이터 수를 갖지만, 엔트로피 패턴이 명확하게 구분되어 높은 분류 성능을 보였다(Fig. 5).

5. 결 론

본 논문에서는 기계학습 기반 IoT 악성코드 탐지와 패밀리 분류를 수행했다. IoT 장치의 CPU 구조 다양성으로 인한 악성코드 분석의 복잡도를 낮추기 위해 실행 파일의 바이트 순서를 이용한 특징을 제안했다. 부분 바이트 순서의 엔트로피를 계산하여 1차원의 엔트로피 순서 정보를 추출하고 선형 보간법을 적용해 고정 길이의 엔트로피 패턴을 정의했다. 제안하는 특징은 CPU 구조와 독립된 특징 표현이 가능하며 특징 집합의 크기를 조절할 수 있다.

순환 신경망 모델을 이용해 엔트로피 순서 정보의 시계열 패턴을 분석하여 악성코드 탐지와 패밀리 분류를 수행했다. 악성코드 탐지에서 RNN은 엔트로피 패턴의 길이가 8일 때 94.1%의 F1-score를 얻었으며 LSTM은 128일 때 94.5%의 F1-score를 보고했다. 패밀리 분류는 비교적 낮은 성능을 나타냈다. LSTM이 16 길이의 엔트로피 패턴을 학습해 92.3%의 정확도와 76.2%의 F1-score를 나타냈으며 RNN은 엔트로피 패턴 길이가 32일 때 정확도 92.3%, F1-score 72.3%를 보고했다. 패밀리 분류에 대한 오차 행렬을 비교 시 대부분의 Tsunami 악성코드가 Gafgyt 패밀리로 오분류되었다. 모델 학습이 수집된 데이터 수가 더 많은 Gafgyt 패밀리 특징에 편향되어 유사한 패턴을 나타내는 Tsunami 악성코드가 오분류되는 것으로 분석된다.

바이트 순서 정보를 이용한 특징은 CPU 구조에 무관한

특징 표현이 가능하다. 그러나 선형 보간법을 이용한 특징 구성은 전체 엔트로피 변화에 대한 정보를 표현하기 어렵다. 분류 성능 향상을 위해 정보 손실을 최소화하는 악성코드 특징 설계가 요구된다. 또한, IoT 장치는 CPU 구조 다양성 이외에 저사양 하드웨어 사용으로 인한 연산 자원의 제약이 존재한다. 연산 자원의 소모가 적은 악성코드 탐지 모델 연구가 필요하다.

References

[1] A. S. Gillis, "What is IoT (Internet of Things) and how does it work," *IoT Agenda, TechTarget*, 11, 2020.

[2] Statista, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030," 2020.

[3] M. B. Barcena and C. Wueest, "Insecurity in the internet of things," *Security Response, Symantec*, 2015.

[4] N. Woolf, "DDoS attack that disrupted internet was largest of its kind in history, experts say," *The Guardian*, 26, 2016.

[5] J. Gamblin, Mirai Source Code [Internet], https://github.com/jgamblin/Mirai-Source-Code.

[6] AhnLab, "Mirai variant malware analysis report," *ASEC Report*, Vol.100, 2020.

[7] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys (CSUR)*, Vol.50, No.3, pp.1-40, 2017.

[8] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," In *2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2018.

[9] R. Sihwail, K. Omar, and K. A. Z. Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," *International Journal on Advanced Science, Engineering and Information Technology*, Vol.8, No.4-2, pp.1662-1671, 2018.

[10] T. L. Wan et al., "Efficient detection and classification of internet-of-things malware based on byte sequences from executable files," *IEEE Open Journal of the Computer Society*, Vol.1, pp.262-275, 2020.

[11] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K. K. R. Choo, "An opcode-based technique for polymorphic Internet of Things malware detection," *Concurrency and Computation Practice and Experience*, Vol.32, No.6, pp.e5173, 2019.

[12] I. Miliaraki, K. Berberich, R. Genmulla, and S. Zoupanos, "Mind the gap: Large-scale frequent sequence mining," In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ACM, 2013.

[13] J. Jeon, J. H. Park, and Y. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, Vol.8, pp.96899-96911, 2020.

[14] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolution networks," *European Conference on Computer Vision*, Springer, Cham, 2014.

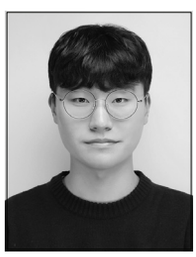
[15] R. Lyda and J. Hamrock, "Using entropy analysis to find encrpyted and packed malware," *IEEE Security & Privacy*, Vol.5, No.2, pp.40-45, 2007.

[16] I. Sorokin, "Comparing files using structural entropy," *Journal in Computer Virology*, Vol.7, No.4, pp.259-265, 2011.

[17] S. Goki, "Deep learning from scratch 2," Oreilly, 2019.

[18] Malwares.com [Internet], https://www.malwares.com/.

[19] Kaspersky [Internet], https://www.kaspersky.co.kr/.



김 영 호

https://orcid.org/0000-0003-4947-8761
e-mail : dudgh1002@naver.com
2020년 단국대학교 소프트웨어학과(학사)
2020년 ~ 현 재 단국대학교 컴퓨터학과 석사과정
관심분야 : Machine Learning, Deep Learning

이 현 종

https://orcid.org/0000-0002-2990-1545
e-mail : infin94@hotmail.com
2017년 단국대학교 컴퓨터과학과(학사)
2019년 단국대학교 소프트웨어학과(석사)
2019년 ~ 현 재 ㈜케이사인 보안기술연구소 주임연구원
관심분야 : Machine Learning, Deep Learning

황 두 성

https://orcid.org/0000-0003-1840-9296
e-mail : dshwang@dankook.ac.kr
1986년 충남대학교 계산통계학과(학사)
1990년 충남대학교 전자계산학과(석사)
2003년 Wayne State Univ. 컴퓨터학과 (박사)
2003년 ~ 현 재 단국대학교 소프트웨어학과 교수
관심분야 : Machine Learning, Parallel Processing, Computer Vision