정 보 보 호 학 회 지
제27권 제3호, 2017. 6

딥러닝을 이용한 악성코드탐지 연구동향

최 선 오*, 김 영 수*, 김 종 현*, 김 익 균*

요 약

인터넷의 발달로 인류가 많은 유익을 얻었지만 동시에 악성코드와 같은 또다른 문제를 겪고 있다. 이러한 악성코드를 막기 위해 시그니처 기반의 안티바이러스 프로그램이 많이 사용되고 있지만 악성코드의 변종이나 제로데이 악성코드를 막는데 한계를 가지고 있다. 이러한 문제를 해결하기 위하여 본 논문에서는 딥러닝을 이용하여 악성코드를 탐지하고 분류하는 연구동향에 대해 소개한다.

Ⅰ. 서 론

인터넷의 발달로 인류가 많은 유익을 얻었지만 동시에 악성코드와 같은 또다른 문제를 겪고 있다. 악성코드는 컴퓨터를 파괴하기도 하고 컴퓨터의 정보를 유출하거나 랜섬웨어[17]와 같이 사용자의 파일을 암호화하고 금전적인 손해를 끼치기도 한다.

이러한 악성코드의 문제를 해결하기 위하여 다양한 종류의 안티바이러스 프로그램이 사용되고 있다. 대부분의 안티바이러스 프로그램은 시그니처나 휴리스틱 기반으로 동작하고 있다. 악성코드 분석가들이 악성코드를 분석하고 특별한 패턴을 시그니처로 안티바이러스 엔진에 등록하면 안티바이러스 엔진이 사용자의 컴퓨터에서 해당 패턴을 인식하여 악성코드를 탐지하게 된다. 그러나 시그니처에 기반한 악성코드 탐지는 악성코드의 변종이나 제로데이 공격에 대응하기 어렵다. 이러한 문제를 해결하기 위하여 최근 머신러닝과 딥러닝을 이용하여 악성코드를 탐지하고자 하는 노력들이 행해지고 있다. [1,3,4,5,6,7]

최근 머신러닝과 딥러닝과 같은 인공지능 기술이 많은 관심을 받고 있다. 강화학습과 딥러닝을 이용한 구글 딥마인드의 알파고가 유명하고 이미지 처리나 자연어 처리 분야에서 인공지능 기술이 놀라운 성능을 보여주고 있다. 한편으로 악성코드 탐지 및 분류 분야에서도 머신러닝과 딥러닝과 같은 인공지능 기술이 적용되고 있다.

[1]은 악성코드의 DLL정보들을 데이터로 사용하고 Naive Bayes 알고리즘 등을 적용하여 악성코드를 탐지하였고 [5]는 악성코드의 API를 추출한후 Decision Tree, Support Vector Machine, Hidden Markov Model을 적용하여 악성코드를 탐지하였다. [7,3,4,6] 은 CNN과 RNN 등과 같은 딥러닝 모델을 적용하여 악성코드를 탐지하고 분류하였다.

딥러닝을 이용한 악성코드 탐지는 크게 두 부분으로 이루어진다. 첫째는 악성코드로부터 특징 데이터를 추출하는 것이다. 이것을 위하여 악성코드를 실행하여 API를 추출하기도 하고 정적분석을 통하여 opcode와 같은 어셈블리코드를 사용하기도 한다. 또는 악성코드를 이미지 파일로 간주하여 특징데이터를 추출하기도 한다. 두 번째는 앞에서 추출한 특징데이터를 가지고 딥러닝 모델을 적용하여 트레이닝하는 것이다. 여러 악성코드의 특징데이터를 사용하여 딥러닝 모델을 트레이닝하고 이후에 탐지 대상이 되는 악성코드를 딥러닝 모델에 적용하여 악성유무 및 해당 악성코드가 어떤 악성코드 패밀리에 속하는지 분류할 수 있다.

본 논문의 구조는 다음과 같다. 2장에서는 딥러닝의 개요를 소개하고 3장에서는 악성코드탐지를 위한 딥러닝모델에서 사용되는 악성코드 특징 추출방법에 대해서 소개한다. 그리고 4장에서는 악성코드탐지를 위한 여러 딥러닝모델과 그것들의 성능을 소개한다. 그리고 마지

본 연구는 2017년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행되었습니다. (No.2016-0-00078, 맞춤형 보안서비스 제공을 위한 클라우드 기반 지능형 보안기술개발)
* 한국전자통신연구원 정보보호연구본부 지능보안연구그룹 (suno@etri.re.kr)

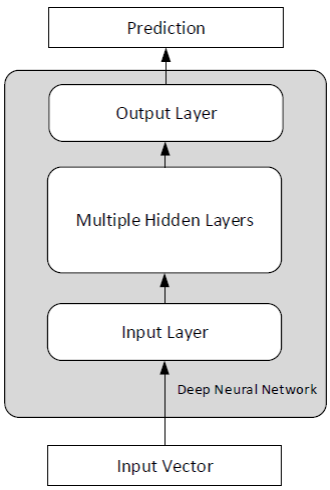
막으로 5장에서는 결론을 맺는다.

Ⅱ. 딥러닝 소개

그림 1은 기본적인 딥러닝의 구조를 보여주고 있다 [4]. 신경망(neural network)은 입력벡터(input vector)를 입력으로 받고 이것은 입력층(input layer)에 주어진다. 그리고 이것은 몇 개의 은닉층(hidden layer)에서 학습이 이루어지고 최종적으로 출력층(output layer)에서 결과를 얻게 된다.

입력 특징 벡터가 $x = [x_1, x_2, ..., x_n]$ 라고 하면 각 은닉층의 각 뉴런 j에서 $y_j = f(\sum_{i=1}^{n} w_{ji} x_i + b_j)$ 가 출력된다. w_{ji}는 각 계층에서의 가중치 매트릭스의 요소이고 b_j는 편향(bias)이고 $f(\bullet)$는 활성화함수(activation function)로서 뉴런의 출력값에 비연속성을 부여한다.

그리고 최종적으로 출력층은 우리가 풀고자 하는 문제를 해결한다. 예를 들어 악성코드가 어떤 종류의 악성코드 패밀리에 속하는지 분류(classification)하는 문제에서 출력층은 마지막 은닉층의 결과를 각 입력 샘플의 클래스를 예측하는 확률분포로 변환시킨다. 따라서 트레이닝 데이터를 가지고 딥러닝모델을 학습시킨 이후에 테스트 데이터를 딥러닝모델에 적용하면 데스트 데이터가 어떤 악성코드 패밀리에 해당하는지 예측할 수 있게 된다.



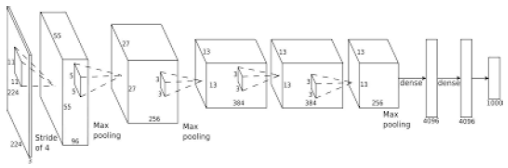
〔그림 1〕 딥러닝 구조

딥러닝 모델의 장점은 데이터로부터 특징을 추출할 필요를 줄여준다는 것이다. 딥러닝은 간단한 로우레벨의 특징으로부터 더 복잡한 상위레벨의 특징을 자동적으로 추출할 수 있는 방법을 제공한다. 예를 들어 이미지 인식의 경우에는 이미지 데이터가 제공되었을 때 첫 번째 은닉층은 이미지의 테두리(edge) 특징을 추출할 수 있고 두 번째 은닉층은 첫 번째 은닉층의 테두리 특징으로부터 개체(object) 특징을 추출할 수 있게 된다. 최종적으로 출력층에서는 입력데이터로 제공된 이미지가 어떤 사물에 해당하는지 구별할 수 있게 된다.

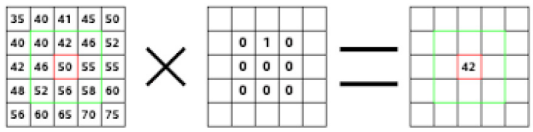
CNN (Convolutional Neural Network)은 신경망의 한 종류이다 [6]. CNN은 세 종류의 계층으로 구성된다. 첫 번째는 완전연결계층(fully- connected) 이고 두 번째는 합성곱계층(convolu- tional) 이고 세 번째는 풀링계층(pooling)이다. 그림 2는 이미지 분류 컨테스트에 사용된 CNN 모델의 예제를 보여준다 [14]. 이 구조는 8개의 계층을 가지는데 처음 다섯 개는 합성곱계층이고 나머지는 완전연결계층으로 구성되어 있다. 참고로 합성곱계층과 풀링계층은 주로 함께 사용되기 때문에 하나의 계층으로 보기도 한다.

이미지와 같은 고차원 입력 데이터를 다룰 때 현재 계층의 뉴런을 이전 계층의 모든 뉴런으로 연결하는 것은 적절하지 않다. 이러한 문제를 해결하기 위하여 CNN이 사용되는데 CNN에서 각 뉴런은 입력 뉴런의 일부분에만 연결된다. 즉 그림 3과 같이 CNN에서는 인접계층의 뉴런들 간의 공간적 연관관계를 고려할 수 있다.

또한 CNN에서는 풀링계층 (pooling layer)을 사용한다. 풀링계층은 그림 4과 같이 각 이미지를 겹치지 않는 정사각형 형태로 나누고 각 서브영역에서 최대값을 출

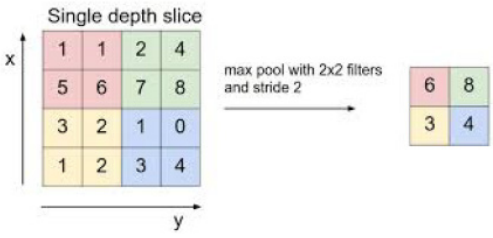


〔그림 2〕 CNN 모델 예제 (AlexNet 구조)



〔그림 3〕 Convolution

력한다. 이것을 max-pooling 이라고 한다. 풀링계층은 계산량을 줄일 수 있고 입력데이터의 변화에 풀링계층의 출력값이 크게 좌우되지 않는 장점을 가진다.



〔그림 4〕 Max Pooling

Ⅲ. 딥러닝을 위한 악성코드 특징추출방법

악성코드 탐지용으로 딥러닝을 사용하기 위해서는 악성코드들로부터 특징을 추출하는 전처리 작업이 필요하다. 이러한 전처리 작업을 위하여 다양한 방법이 사용되고 있다.

[4,7]의 경우에는 기본적으로 악성코드의 API 콜을 사용한다. 동적분석을 위하여 악성코드에서 사용되는 API와 파라미터를 추출한다. API와 파라미터는 알려지지 않은 파일의 동적행위를 분석하는데 사용된다 [10,11]. [4,7]에서 114개의 하이레벨 API가 사용되고 있다. 그리고 이것을 두 가지 데이터 셋으로 사용한다. 첫 번째 데이터 셋에서는 하이레벨 API 이벤트와 한 개의 입력 파라미터를 조합해서 사용한다. 그리고 두 번째 데이터 셋은 다음과 같이 API 이벤트들의 trigram을 사용한다.

{ PUSH, CALL, MOV }

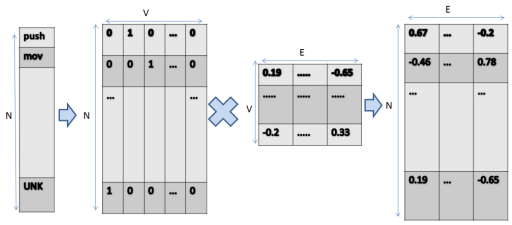
즉 세 개의 연속적인 API 이벤트의 조합을 사용한다. 그러나 이러한 두 개의 데이터 셋은 수백만 개의 잠재적인 특징을 포함할 수 있기 때문에 각 클래스를 가장 잘 나타낼 수 있는 특징을 추출하기 위해 상호 정보(mutual information) [2]를 사용하여 50,000개의 특징을 선택한다.

악성코드의 특징을 추출하는 다른 방법은 파일의 어셈블리 코드로부터 추출된 opcode를 사용하는 것이다 [6,13]. 자연어처리 시스템에서는 단어(word)를 심볼로 간주할 수 있다. 그림 5와 같이 하나의 악성파일에서 각 opcode들은 사이즈 V의 벡터로 변환된다. [6]에서는 하나의 파일에 대하여 최대 10,000개의 opcode를 사용한다. N*V의 매트릭스는 V*E의 매트릭스와의 연산을 통하여 저차원의 N*E의 매트릭스로 변환된다.

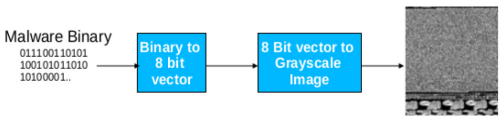
악성코드로부터 특징을 추출하는 또다른 방법은 악성코드를 이미지로 간주하는 것이다 [15]. 이 아이디어는 같은 패밀리에 속하는 악성코드의 이미지들은 유사하고 다른 패밀리에 속하는 악성코드의 이미지들은 구별된다는 것이다. 그림 6과 같이 악성코드파일을 8비트 정수의 벡터로 읽고 이것을 2차원 배열로 변환한다. 이때 해당 2차원 배열은 이미지로 시각화될 수 있다.

악성코드를 이미지로 시각화하는 것의 장점은 바이너리의 다른 섹션들이 쉽게 구별될 수 있다는 것이다. 또한 악성코드 제작자들은 악성코드의 변종을 만들기 위해 기존 악성코드의 일부분 만을 고치기 때문에 이미지는 전체적인 구조는 유지하면서 작은 변화를 탐지하는데 유용하다. 결과적으로 변종에 속하는 악성코드의 이미지는 같은 패밀리에 속하는 원본 악성코드의 이미지와 매우 유사하고 다른 패밀리에 속하는 악성코드의 이미지와는 쉽게 구별될 수 있다.

그 다음으로 이미지로부터 texture 특징을 추출하기 위하여 [6]에서는 GIST[16]를 사용하였고 그 이후에 KNN 분류기를 사용하였다.



〔그림 5〕 Opcode를 사용한 CNN Embedding Layer



〔그림 6〕 악성코드를 이미지로의 시각화

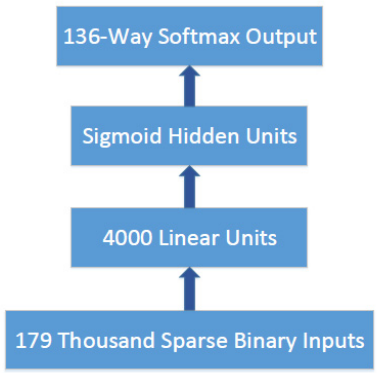
Ⅳ. 딥러닝을 이용한 악성코드 탐지방법

[7]에서 제안한 딥러닝 모델은 그림 7과 같다. [7]에서는 API trigram과 API 파라미터를 사용하여 179,000 개의 특징을 사용한다. 이것을 Random projection [8]을 통하여 4000개의 특징으로 줄이고 은닉층으로 Sigmoid 함수를 사용한다. Random projection에서는 sparse random projection matrix R을 사용하고 R의 오직 0.22% 값만이 1로 세팅되고 또 다른 0.22%의 값이 -1로 세팅되고 나머지 99.56%의 값들은 0으로 세팅된다. 따라서 50,000개의 특징들은 약 4,000개로 줄어들게 된다. 이렇게 random projection을 사용하여 계산량을 줄일 수 있다. 그리고 출력층에서는 136개의 악성코드 패밀리로 분류하기 위하여 Softmax 함수를 사용한다.

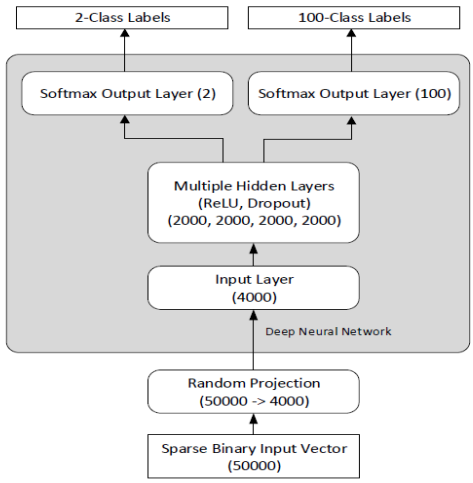
둘째로 [4]에서 제안한 딥러닝 모델은 그림 8과 같다. [4]의 딥러닝모델은 두가지 목적을 위하여 사용된다. 첫째는 파일이 악성인지 정상인지를 판단하는 것이고 두 번째는 악성파일을 100가지의 악성파일 클래스로 분류하는 것이다.

[4]에서는 50,000 개의 특징을 사용한다. 그러나 50,000개의 특징을 모두 사용하는 것은 계산량이 너무 많다. 이러한 문제를 해결하기 위하여 random projection [8]이 사용된다.

딥러닝을 위하여 제일 처음 입력 벡터들은 정규화된다. 정규화는 딥러닝 트레이닝이 더 빠르게 수렴하도록 만든다. 그리고 은닉층에서는 ReLU (rectified linear unit)을 사용한다. ReLU 함수는



〔그림 7〕 악성코드 분류를 위한 딥러닝 모델



〔그림 8〕 멀티태스크 러닝을 위한 딥러닝 모델

$$f(\gamma) = \max(0, \gamma)$$

로 정의되고 이것은 vanishing gradient problem을 해결할 수 있고 stochastic gradient descent의 수렴을 가속화시킨다. 딥러닝 모델에서는 학습을 할 때 각 가중치의 경사도를 줄여가면서 수렴하게 만드는 경사도 하강법(gradient descent)을 사용한다. 그런데 간혹 딥러닝 모델에 따라 경사도가 0이 되거나 무한대가 되는 vanishing gradient problem이 발생하기도 한다. ReLU는 이 문제를 해결하는데 도움이 되는 것으로 알려져 있다.

그리고 동시에 은닉층에서는 Dropout [9]가 사용되는데 Dropout의 아이디어는 임의로 은닉층의 뉴런 중 일부를 업데이트하지 않는 것이다. Dropout를 사용하므로써 딥러닝모델이 overfitting 되는 것을 막을 수 있다. Overfitting은 딥러닝 모델이 특정 데이터에만 맞도록 트레이닝되어서 일반적인 데이터에서 나쁜 예측 결과를 주는 것을 말한다.

표1은 [4]의 딥러닝모델과 [7]의 딥러닝모델과의 성능비교를 보여준다. [4]에서는 [7]에서 사용된 sigmoid 함수를 ReLU 함수로 대체하였고 Dropout가 사용되었다. ReLU 함수와 Dropout를 사용하는 것이 좀더 정확한 모델을 만들 수 있는 것을 보여준다. 또한 ReLU 함수를 사용하는 것이 Sigmoid 함수보다 트레이닝을 위한 반복 횟수를 줄여주는 것을 확인할 수 있다.

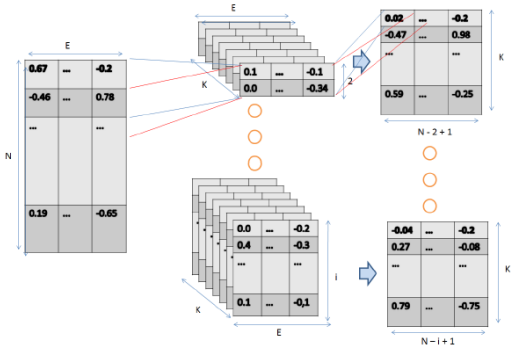
〔표 1〕 악성분류 시험결과 비교

	Baseline Model (Original Results [7])	Baseline Model (Our Data)		Single Task Model (Our Data)	
Layers	Test Error(%)	Test Error(%)	Epoch	Test Error(%)	Epoch
1	0.49	0.5906	190	0.3711	64
2	0.50	0.4882	186	0.3702	82
3	0.51	0.4845	200	0.3686	77
4		0.4934	200	**0.3683**	81

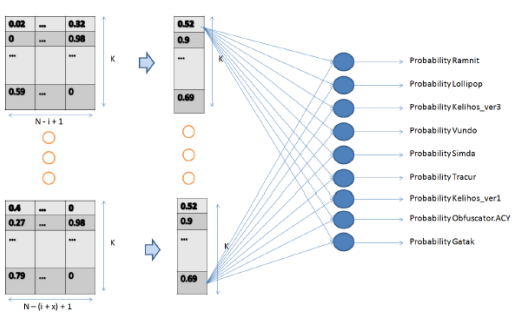
셋째로 [6]에서는 opcode를 특징으로 사용하고 CNN을 이용하여 악성코드를 분류하였다. 그림 5와 같이 opcode로부터 CNN embedding layer가 만들어지고 그 다음으로 그림 9와 같은 합성곱계층(convolutional layer)을 가진다. 해당 합성곱계층은 k 개의 필터를 가진다.

다음으로 그림 10과 같이 max-pooling 계층이 사용되고 마지막으로 악성코드가 어떤 악성코드 패밀리에 속하는지 예측을 하게 된다.

표 2는 opcode를 사용한 CNN모델에 의한 악성코드 분류 결과를 보여준다. opcode를 사용하는 것은 악성코드 이미지를 특징으로 사용하는 것[6, 15]보다 좋은 정확도를 보여준다.



〔그림 9〕 CNN Convolutional layer



〔그림 10〕 CNN Max-pooling & output layer

〔표 2〕 CNN : confusion matrix (Acc: 0.9947)

	Ramnit	Lollipop	Kelihos_ver3	Vundo	Simda	Tracur	Kelihos_ver1	Obfuscator.ACY	Gatak
Ramnit	1528	0	0	8	0	1	0	4	0
Lollipop	2	2473	0	2	0	1	0	0	0
Kelihos_ver3	0	0	2938	4	0	0	0	0	0
Vundo	0	0	0	474	0	0	0	1	0
Simda	0	0	0	0	42	0	0	0	0
Tracur	4	0	0	1	0	746	0	0	0
Kelihos_ver1	0	0	0	6	0	0	392	0	0
Obfuscator.ACY	11	0	0	9	0	1	0	1207	0
Gatak	0	0	0	1	0	2	0	0	1010

V. 결 론

본 논문에서는 딥러닝을 이용한 악성코드탐지 연구동향을 소개하였다. 딥러닝을 이용한 악성코드탐지는 크게 두 부분으로 이루어진다. 첫째는 악성코드로부터 딥러닝 모델 학습에 사용할 특징 데이터를 추출하는 것이다. 이러한 특징 데이터로 악성코드의 API나 opcode 등이 사용된다. 둘째는 추출한 특징 데이터를 사용하여 딥러닝 모델을 만들고 학습하는 것이다. 악성코드탐지를 위한 딥러닝 모델로 CNN이나 RNN 등이 사용된다. 각 딥러닝 모델은 악성코드탐지에 있어서 상당히 높은 성능을 보여준다.

그러나 최근에는 인공지능에 기반한 악성코드탐지 기술을 회피하는 방법에 관한 연구도 이루어지고 있다[18]. 이러한 상황에서 악성코드로부터 더 정확한 특징들을 추출하고 악성코드를 탐지하고 분류하기 위한 좀 더 유용한 딥러닝 모델을 연구하는 것이 필요하다.

참 고 문 헌

[1] Matthew G. Schultz, Eleazae Eskin, and Erez Zadok, “Data Mining Methods for Detection of New Malicious Executables,” IEEE International Conference on Security and Privacy, 2001.

[2] Manning C.D., Raghavan, P., Schutze, H., “An Introduction to Information Retrieval,” Cambridge University Press, 2009

[3] Razvan Pascanu, Jack W. Stokes, Hermineh Sanossian, Mandy Marinescu, and Anil Thomas, “Malware Classification with Recurrent Networks,” IEEE International Conference on Acoustics, Speech and Signal Processing, 2015

[4] Wenyi Huang and Jack W. Stokes, “MtNet: A Multi-Task Neural Network for Dynamic Malware Classification,” International

Conference on Detection of Intrusions and Malware & Vulnerability Assessment, 2016
[5] Daesung Moon, Sung Bum Pan, and Ikkyun Kim, “Host-based intrusion detection system for secure human-centric computing,” Journal of Supercomputing, 2016
[6] Daniel Gibert, “Convolutional Neural Networks for Malware Classification,” Master Thesis, Universitat de Barcelona, 2016
[7] George E. Dahl, Jack W. Stokes, Li Deng, and Dong Yu, “Large-Scale Malware Classification using Random Projections and Neural Networks,” IEEE International Conference on Acoustics, Speech and Signal Processing, 2013
[8] Li, P., Hastie, T.J., Church, K.W., “Very sparse random projections,” ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ICDM), 2006
[9] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., “Dropout: A simple way to prevent neural networks from overfitting.” J. Mach. Learn. Res., 2014
[10] Ulrich Bayer, Christopher Kruegel, and Engin Kirda, “TTAnalyze: A tool for analyzing malware,” Annual Conference of the European Institute for Computer Antivirus Research (EICAR), 2006
[11] A. Moser, C. Kruegel, and E. Kirda, “Exploring multiple execution paths for malware analysis,” IEEE Symposium on Security and Privacy, 2007
[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean, “Distributed representations of words and phrases and their compositionality,” Advamces om Neural Information Processing Systems, 2013
[13] https://www.kaggle.com/c/malware-classification
[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, 2012
[15] L. Nataraj, S. Karthikeyan, G. Jacob, B. S. Manjunath, “Malware Images: Visualization and Automatic Classification,” ACM VizSec, 2011
[16] Aude Oliva and Antonio Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” International Journal of Computer Vision, 2001
[17] WannerCry randomware attack, https://en.wikipedia.org/wiki/WannaCry_ransomware_attack
[18] W. Xu, Y. Qi, D. Evans, “Automatically Evading Classifiers,” Network and Distributed Systems Symposium, 2016

〈저자소개〉

최 선 오 (Sunoh Choi)

2005년 2월 : 고려대학교 컴퓨터학과 졸업
2008년 2월 : 고려대학교 컴퓨터학과 석사
2014년 5월 : Purdue 대학교 전자컴퓨터공학과 박사
2014년 8월~현재 : ETRI 지능보안연구그룹 선임연구원
관심분야 : 데이터보안, 네트워크보안, 딥러닝

김 영 수 (Young Soo Kim)

1998년 2월 : 성균관대학교 정보공학과 졸업
2000년 2월 : 성균관대학교 컴퓨터공학과 석사
2009년 8월 : 성균관대학교 컴퓨터공학과 박사
2012년~2015년 : 충남대학교 컴퓨터공학과 겸임교수
2000년 2월~현재 : ETRI 지능보안연구그룹 책임연구원
관심분야 : 암호학, 네트워크보안, 디지털포렌식

김 종 현 (Jonghyun Kim)

1995년~1998년 : 삼성전자 연구원
2000년 : 오클라호마 주립대학교 컴퓨터과학과 석사
2005년 : 오클라호마 주립대학교 컴퓨터과학과 박사
2005년~현재 : ETRI 지능보안연구그룹 PL/책임연구원
관심분야 : 정보보호, 네트워크보안, 네트워크 포렌식

김 익 균 (Ikkyun Kim)
정회원

1994년 2월 : 경북대학교 컴퓨터공학과 졸업
1996년 2월 : 경북대학교 컴퓨터공학과 석사
2009년 2월 : 경북대학교 컴퓨터공학과 박사
2004년~2005년 : Purdue 대학교 초빙 연구원
1996년~현재 : ETRI 지능보안연구그룹 그룹장/책임연구원
관심분야 : 네트워크보안, 컴퓨터 네트워크, 클라우드보안, 빅데이터 분석