Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.2, Apr. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.2.439

CNN 기반 MS Office 악성 문서 탐지*

박 현 수,[1†] 강 아 름[2‡]
[1,2]배재대학교 (대학원생, 교수)

MS Office Malicious Document Detection Based on CNN*

Hyun-su Park,[1†] Ah Reum Kang[2‡]
[1,2]Pai Chai University (Graduate student, Professor)

요 약

웹사이트나 메일의 첨부 파일을 이용해 문서형 악성코드의 유포가 활발하게 이루어지고 있다. 문서형 악성코드는 실행 파일이 직접 실행되는 것이 아니므로 보안 프로그램의 우회가 비교적 쉽다. 따라서 문서형 악성코드는 사전에 탐지하고 예방해야 한다. 이를 탐지하기 위해 문서의 구조를 파악하고 악성으로 의심되는 키워드를 선정하였다. 문서 내의 스트림 데이터를 아스키코드값으로 변환하여 데이터셋을 만들었다. CNN 알고리즘을 이용하여 문서의 스트림 데이터 내에 존재하는 악성 키워드의 위치를 확인하고 인접 정보를 활용하여 이를 악성으로 분류했다. 파일 내의 스트림 단위로 악성코드를 탐지한 결과 0.97의 정확도를 보였고, 파일 단위로 악성코드를 탐지한 결과 0.92의 정확도를 보였다.

ABSTRACT

Document-type malicious codes are being actively distributed using attachments on websites or e-mails. Document-type malicious code is relatively easy to bypass security programs because the executable file is not executed directly. Therefore, document-type malicious code should be detected and prevented in advance. To detect document-type malicious code, we identified the document structure and selected keywords suspected of being malicious. We then created a dataset by converting the stream data in the document to ASCII code values. We specified the location of malicious keywords in the document stream data, and classified the stream as malicious by recognizing the adjacent information of the malicious keywords. As a result of detecting malicious codes by applying the CNN model, we derived accuracies of 0.97 and 0.92 in stream units and file units, respectively.

Keywords: MS Office, malicious, detection, CNN, deep learning

I. Introduction

웹사이트나 메일의 첨부 파일을 이용해 HWP, PDF, MS Office 등 문서형 악성코드의 유포가 활발하게 이루어지고 있다. HWP 문서 파일을 이용하여 공격하는 방식에는 한글 워드프로세서의 매크로(macro), postscript, 자료 연결, 배포용 문서 등 네 가지의 유형으로 분석됐다[1]. 악의적인 공격자는 의뢰서나 자기소개서 등으로 위장한 문서형 악성코드를 유포하여 금융회사, 정부 기관 관계자 등을 대상으로 기밀정보 탈취 및 금전적 이익을 취한다[2],[3]. 문서형 악성코드는 실행 파일이 직접적으로 실행되는 것이 아닌 외부(external)의 URL(Uniform Resource Locator)을 통해 C&C(Command&Control) 서버와 연결하여 악

Received(02. 17. 2022), Modified(03. 24. 2022), Accepted(03. 25. 2022)
* 이 논문은 서울시 산학연 협력사업(CY210066) 지원을 받아 수행된 연구임
† 주저자, rosemine4531@gmail.com
‡ 교신저자, armk@pcu.ac.kr(Corresponding author)

의적인 코드를 다운로드 받거나 문서 내에 악의적인 행위를 하는 VBA(Visual Basic for Application) 매크로를 담아 프로세스를 조작하고 스니핑(Sniffing)하는 유형의 악성코드이다. 이러한 이유로 문서형 악성코드는 다른 악성코드에 비해 보안 프로그램의 탐지 우회가 비교적 유용하다.

문서형 악성코드로부터의 피해를 최소화하기 위하여 이를 사전에 탐지하는 모델의 개발이 필요하다. 본 연구에서는 세계적으로 가장 많이 사용되고 있는 문서인 MS Office를 대상으로 하였다. MS Office 문서의 내부 구조를 파악하기 위해 먼저 문서 구조가 비슷한 HWP 파일과 PDF 파일의 내부 구조를 참고하였다[4]. 그런 다음 MS Office의 문서 구조를 분석하고 문서 내 스트림 데이터를 수동으로 확인하여 exe나 dll과 같은 PE(Portable Executable) 파일의 확장자, 셸코드(shellcode), 파일이나 폴더의 조작 명령어, 프로세스 및 외부 URL에 접근하여 파일을 다운로드하는 코드 등의 악성으로 의심되는 키워드를 수집했다. 악성 키워드의 존재 여부를 확인하여 문서 내 스트림(stream) 데이터에 태깅(tagging)하고 CNN(Convolution Neural Network) 모델에 적용하여 악의적인 문서 파일과 정상 문서 파일을 분류하고자 한다.

2장에서는 문서형 악성코드 관련 선행 연구를 조사하였고, 3장에서는 MS Office 문서의 구조와 스트림 내 악의적인 키워드, 제안하는 모델에 대해서 기술하였다. 4장에서는 연구에서 사용한 데이터셋, 데이터 전처리 과정, 연구 방법, 결과 등을 다루고, 5장에서는 결론과 향후 연구에 대해 설명하였다.

II. Related Works

문서형 악성코드의 매크로 데이터나 Javascript, Encapsulation postscript, APT(Advanced Persistent Threat) 공격 등으로 인한 피해를 최소화하고자 머신러닝과 딥러닝을 활용하여 문서형 악성코드를 탐지하는 연구가 꾸준히 진행되고 있다. 선행 연구에서는 HWP 문서 파일과 PDF 문서 파일의 구조와 데이터를 분석하여 random forest, naïve Bayes, SVM(Support Vector Machine), CNN 알고리즘 등을 이용하여 진행하였다.

HWP, PDF, MS Office 문서형 악성코드의 구조를 분석하고 머신러닝과 딥러닝을 이용한 주요 연구를 정리하였다. Young-Seob Jeong 외[5]는 HWP 바이트 스트림을 분석하여 원시 바이트 스트림 내의 악성코드를 탐지하기 위해 새로운 CNN 모델을 설계하고 제시했다.

Ah Reum Kang 외[6]의 연구는 PDF 내에서 Javascript를 사용하는 정상적인 문서와 악의적인 문서를 수집하여 random forest, naïve Bayes, SVM 알고리즘으로 학습 및 평가하였고, random forest로 0.997의 정확도를 보였다.

Young-Seob Jeong 외[7]는 악성 및 양성 PDF 파일 내 바이트 시퀀스를 수집하고 데이터의 구조 및 데이터의 특징을 기반으로 악의적인 행위를 탐지하는 CNN 모델을 제안하였다.

Young-Seob Jeong 외[8]은 HWP 파일의 구조와 스트림에 대해 소개하였고, 스트림 데이터 중 EPS(Encapsulated PostScript)나 매크로 데이터에 악의적인 행위가 있는지 분석하였다. 이를 이용하여 사용자가 파일을 열기 전에 악성코드에 감염되는 것을 예방하는 모델을 제시하였다.

Ah Reum Kang 외[9]는 PDF 구조 분석을 통해 악의적인 PDF 문서가 stream 오브젝트를 사용한다는 것을 알아내었다. 공격자는 악성 Javascript를 난독화하거나 인코딩하여 탐지를 회피하는데 이러한 악의적인 행위를 수행하는 stream 데이터를 디코딩하여 악성 키워드를 수집하였다.

Chae-Eun Yoon 외[10]는 피싱(phishing) 메일을 통해 유포되는 악성 문서 파일 중 APT 공격에 이용되는 PDF 문서 내의 악성코드에 집중하였다. 정상적인 파일과 악의적인 파일에서 객체 사이에 존재하는 키워드의 정보와 발생 빈도 데이터를 토대로 학습을 진행하였으며, random forest 모델에서 0.9875의 정확도, SVM 모델에서 0.9833의 정확도를 도출하였다.

Deokkyu Lee와 Sangjin Lee[11]는 악성 OOXML(Office Open XML) 문서 내 악성 여부를 판단하는 요소로 개체요소 매크로, OLE object, DDE(Dynamic Data Exchange), ActiveX, EPS, External-Ref를 제시하였고, OOXML 문서 기반 악성코드 탐지 프레임워크를 구축하여 0.9845의 정확도를 보여주었다.

MinJi Choe 외[12]는 한글 문서 포맷 및 악성 행위를 식별할 수 있는 오브젝트를 분석하였고 한글 문서의 취약점과 한글 문서를 통한 공격 방식에 대하여 제시하였다.

Sung Hye Cho와 Sang Jin Lee[13]의 연구는 취약점이 많이 공개된 MS Office 97-2003 문서 파일을 수집하고 파일 내부 구조를 분석하였다. 문서 내 스트림 파일 별로 존재하는 고유 데이터 및 문서 구조를 파악하여 악성 행위의 특징을 제시하고 비정상 코드를 탐지하는 도구를 개발하였다.

본 연구는 MS Office 파일의 스트림 데이터를 분석하여 악성으로 의심되는 키워드를 수집하고 키워드 존재 여부로 스트림에 태깅하여 악성 MS Office 문서를 탐지하는 모델을 제안하고자 한다.

III. Proposed Model

3.1 MS Office structure analysis

MS Office 파일 구조는 파일 시스템 중 마이크로소프트사에서 제작한 FAT(File Allocation Table) 파일 시스템과 유사한 구조를 지닌다. 복합 문서 파일 형태의 구조를 지니는 것을 OLE(Object Linking and Embedding) 파일 포맷이라고 부른다. OLE 파일 포맷은 내부 구조에 폴더 및 파일이 존재하며 MS Office 2007 이전 버전과 HWP document file format 5.0 버전에서 사용된다.

OLE 파일은 크게 헤더 블록과 데이터 블록으로 나뉜다. 헤더 블록은 512 byte의 크기를 가지며 OLE 파일 전체의 주요 정보들을 가지고 있다. 헤더 블록 내에 Magic ID가 0x0000에서 0x0007까지의 8 byte 범위에 존재하는데 일반적으로 D0 CF 11 E0 A1 B1 1A E1이다. 이는 OLE 파일임을 나타내는 시그니처 항목이다. 데이터 블록은 512 byte 이상이며 property, 스트림 데이터, BBAT(Big Block Allocation Table), SBAT(Small Block Allocation Table)을 가지고 있는데 이 중 스트림 데이터가 가장 많은 공간을 차지한다. 따라서, 본 논문에서는 OLE 파일 데이터 블록 중 스트림 데이터를 위주로 연구를 진행하였다.

3.2 MS Office stream

OLE 파일 내의 스트림 데이터 분석을 통해 문서형 악성 파일에 exe나 dll 등의 파일 확장자, 셸코드, 파일이나 폴더의 조작, 프로세스 조작 및 외부 URL에 접근하여 파일을 다운로드하는 코드 등의 키워드를 추출하였다. 스트림 단위로 학습 및 테스트하기 위하여 데이터셋을 구축하였으며 각 데이터는 파일 이름, 스트림 이름, 사이즈, 악성 키워드 존재 여부 태깅 정보, 악성 키워드 시작 지점, 악성 키워드 종료 지점, 스트림 데이터 전체로 구성된다.

Table 1. Malicious suspect keywords

- .exe, .bax, .jar, .DLL - shell32, shellrun, shellexecute, powershell - filecopy, copyfile, savetofile, downloadfile, deletefolder, folderexists, mkdir, createdirectory - urldownloadtofile, urldownload - admin, Administrator, root, Password, Security - Crypt, Base64, Encode64, MyDecode, Convert - ThreadId, createthread, StackSize, Resource, ResourceReader - hkey_local, kenel32, auto_close, regsvr32, Runtime, stdole2, automation, Evaluate, Replace, padding, Rebuild, MELBLUEEmbedded, Delete, Unprotec, SetSuspendStats, Keylogger, manifest, assembly, requestedExecutionlevel

Table 1은 악성으로 의심되는 주요 키워드 52개를 정리한 것이다. 악성 파일에는 주로 매크로 데이터들이 포함되어 있었으며, 매크로에서 주로 쓰이는 키워드 위주로 찾았다. 수집한 키워드 중 정상 파일에도 사용될 수 있는 키워드의 경우 악성 키워드 목록에서 제외하였다.

가독 텍스트 중 .exe, .bax, .jar, .DLL은 사용자 컴퓨터 혹은 악성 문서 내에 존재할 수 있는 응용프로그램의 확장자를 통해 해당 프로그램을 실행시킬 수 있다. 가독 텍스트 중 shell32, shellrun, shellexecute, powershell의 경우 작은 크기의 코드로 소프트웨어 취약점을 이용할 수 있는 셸코드 관련 키워드로 분류했다. 가독 텍스트 중 filecopy, copyfile, savetofile, downloadfile, deletefolder, folderexists, mkdir, createdirectory의 경우 파일이나 폴더를 다운로드, 복제, 변경, 삭제 등을 수행할 수 있는 키워드이다.

외부 URL에 접속하여 파일을 다운로드 하는 urldownloadtofile, urldownload를 URL 관련 악성코드 키워드로 분류했다. 가독 텍스트 중 admin, Administrator, root, Password, Security를 사용자의 계정이나 패스워드를 탈취하

거나 추측 가능한 키워드로 분류했다. 악성 행위를 은닉하기 위한 목적으로 악성코드를 Crypt, Base64, Encode64, MyDecode, Convert로 암호화하거나 인코딩하여 삽입하고 문서 열람 시 복호화나 디코딩 과정을 거쳐 악성코드를 실행시키는 키워드로 분류했다. ThreadId, createthread, StackSize, Resource, ResourceReader의 경우 프로세스를 조작하거나 메모리의 사이즈를 알아내 메모리 버퍼 오버플로우(memory buffer overflow) 등의 공격이 가능한 키워드로 분류했다.

스트림은 OLE 파일 중 가장 큰 비중을 차지하는 데이터이며 스트림 데이터를 읽기 위해 최상위 스토리지를 통해 모든 디렉터리를 읽어온다. 스트림은 문서의 데이터가 저장되는 곳으로 스트림 데이터 중 악의적으로 실행될 수 있는 악성 키워드를 정확하게 찾아내고 이러한 악성 키워드를 포함한 스트림 파일을 해당 문서가 가지고 있다면 악성으로 분류하였다.

3.3 Model

본 연구에서는 딥러닝 알고리즘 중 CNN을 사용했다. CNN은 데이터의 특징을 추출하여 특징들의 패턴을 파악하는 구조로 Fig. 1.과 같이 표현할 수 있다. Fig. 1.의 첫 번째 5 x 5 행렬이 데이터, 두 번째 3 x 3 행렬이 필터, 세 번째 행렬이 데이터가 필터를 거치는 과정, 네 번째 3 x 3 행렬이 필터를 거치면서 생성된 convolution layer이다.

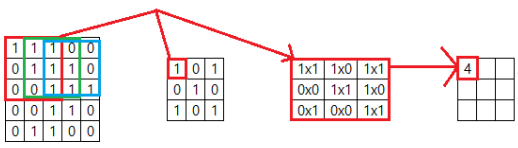

Fig. 1. Feature filtering

데이터에서 필터의 크기와 동일하게 필터가 적용될 영역을 한 칸씩 우측으로, 우측 끝에 도달하면 한 칸 하단으로 내려가 좌측부터 우측으로 옮겨 가며 연산을 수행한다. 이 과정은 하나의 압축 과정이라고 볼 수 있으며 파라미터의 개수를 효과적으로 줄여주는 역할을 한다.

CNN은 신경망에서 학습을 통해 자동으로 데이터에 대한 필터를 생성해주는 특징을 가진다. 이미지의 크기가 작다면 패딩 과정을 통해 이미지의 크기를 임의로 유지할 수 있다. 하지만 이미지의 크기를 계속 유지한 채로 다음 레이어로 가게 된다면 연산량이 기하급수적으로 늘어나기 때문에 적당히 크기도 줄이고 특정 feature를 강조할 수 있는 pooling 방식을 사용한다. pooling은 convolution layer의 사이즈를 줄여주는 과정으로 단순히 데이터의 사이즈를 줄여주고 노이즈를 상쇄시키고 미세한 부분에서 일관적인 특징을 제공한다. CNN에서는 주로 max pooling을 사용한다.

IV. Experiment Result

4.1 Dataset

본 연구에 사용된 데이터는 총 1100개의 파일을 사용하였고 악성 파일 550개, 정상 파일 550개로 진행했다. 악성 파일의 경우 국내 백신 사가 제공한 악성 파일 550개를 사용하였다. 악성 파일은 PPT 파일 3개, XLS 파일 264개, DOC 파일 283개이다. 정상 파일은 공공데이터 포털을 이용하여 PPT, XLS, DOC 파일을 수집하였는데 PPT 파일 5개, XLS 파일 542개, DOC 파일 3개로 XLS 파일이 주를 이루었다. 공공데이터 포털은 주로 공공기관에서 사용하는 파일 등이 업로드되는데 MS Office 파일 중 XLS 파일이 대다수이고, PPT 파일과 DOC 파일의 비중이 적어 다양한 문서의 스트림 데이터를 수집하지 못하였다. 각 파일의 스트림 추출을 진행하면서 파일별로 스트림을 여러 개 가지고 있어 데이터셋은 총 10244개의 스트림 데이터를 가진다. 데이터셋의 악성 파일에는 주로 매크로 데이터들이 포함되어 있었으며, 매크로에서 주로 쓰이는 키워드들을 찾았다. 정상 파일의 매크로에서 존재할 수 있는 키워드들은 악성 키워드 목록에서 제외하였다.

4.2 Data preprocessing

악성 파일 중 악성 키워드가 존재하는 스트림, 정상 파일 중 악성 키워드가 존재하지 않는 스트림에서 랜덤하게 학습용 데이터로 추출하였다.

전제 데이터셋은 악성 파일 중 악성 키워드가 존재하는 스트림, 악성 파일 중 악성 키워드가 존재하지 않는 스트림, 정상 파일 중 악성 키워드가 존재하는 스트림, 정상 파일 중 악성 키워드가 존재하지 않는 스트림으로 구성되는데, 학습용 데이터로 사용하

지 않은 데이터를 테스트용 데이터로 사용하였다. MS Office 악성 파일을 수동으로 분석해 본 결과 악성코드는 주로 스크립트 형태로 존재했다. 악성 행위가 포함된 스크립트가 충분히 포함된 인풋 데이터를 생성하려고 하였다. 악성 스크립트에는 간단한 악성 행위도 탐지를 회피하기 위해 난독화하고, 더미데이터가 많이 포함되어 있었다. 또한 하나의 스트림에도 악성코드가 분산되어 있는 경우도 있어 악성코드 부분이 아닌 부분까지 학습이 되지 않도록 악성 키워드가 존재하는 구간 1000 byte를 기준으로 샘플링하여 학습 데이터셋을 만들었다.

학습용 데이터의 경우 악성 키워드의 위치를 태깅하여 키워드 리스트 중 스트림 데이터에서 가장 먼저 등장하는 키워드의 위치와 가장 마지막에 등장하는 키워드의 위치를 추출하여 가장 먼저 등장하는 키워드를 포함하거나 가장 마지막에 등장하는 키워드를 포함하는 스트림 데이터를 랜덤하게 1000 byte 씩 추출하였다.

4.3 Model

본 연구에서 CNN 모델을 채택한 이유는 악성코드에 자주 등장하는 키워드가 없으면 일반으로 분류하게 된다. 그러나 키워드가 포함된 raw byte 스트림 구간을 인풋으로 사용하게 되면 키워드뿐만 아니라 키워드 앞, 뒤로 등장하는 악성코드의 다른 특성들도 함께 학습하게 된다. 따라서, 연속적인 데이터를 학습할 수 있는 CNN 모델을 채택하였다.

전체 1100개의 파일에서 파일 내 스트림 데이터별로 악성 파일이면서 악성 키워드가 존재하는 스트림, 정상 파일이면서 악성 키워드가 존재하지 않는 스트림 3694개를 추출하였다. 이 중 학습 데이터로 2585개의 스트림을 랜덤하게 선정하였다.

테스트용 데이터는 악성 파일이면서 악성 키워드가 존재하는 스트림, 악성 파일이면서 악성 키워드가 존재하지 않는 스트림, 정상 파일이면서 악성 키워드가 존재하는 스트림, 정상 파일이면서 악성 키워드가 존재하지 않는 스트림을 모두 사용하였다. 전체 1100개의 파일에서 10244개의 스트림을 추출하였으며, 이 중 학습에 사용된 데이터 파일명과 동일한 데이터를 제외하고 나머지 4770개의 스트림이 이에 해당한다.

학습용 데이터, 테스트 데이터의 input size를 맞춰주기 위하여 테스트 데이터 스트림에서 랜덤으로 1000 byte만큼 추출하여 사용하였다. Fig. 2.와 같이 모델을 구성하였고 epochs는 25, batch_size는 30으로 진행하였다.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 59, 17, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 59, 17, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 59, 17, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 59, 17, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 59, 17, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 59, 17, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 64192)             0
_________________________________________________________________
dense (Dense)                (None, 32)                2054176
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 66
=================================================================
Total params: 2,074,114
Trainable params: 2,074,114
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2. Model summary

모델 학습은 악성, 정상을 태깅된 스트림 단위로 진행하였고, 악성 문서 탐지는 스트림 단위로 진행한 후, 파일 내의 스트림 중 하나라도 악성 스트림이 존재하는 경우 악성으로, 모든 스트림이 정상 스트림인 경우 정상으로 판단하였다. 단순히 악성 키워드의 등장 횟수로 악성을 탐지하는 방식이 아닌 악성 키워드가 포함된 1000 byte 스트림 구간 전체를 학습하기 때문에 유사한 특징을 가지는 새로운 악성코드도 탐지가 가능하다.

4.4 Result

CNN 알고리즘을 이용하여 스트림 단위로 악성 키워드 존재 여부를 파악하여 정확도와 정밀도, 재현율 등을 도출하였고, 파일 단위로 스트림 데이터를 평가하여 파일 자체가 악의적인 문서인지 정상적인 문서인지 탐지하는 모델을 제안하고자 한다.

학습과 테스트의 손실률은 Fig. 3.과 같이 Epoch 10에서 loss는 0.0174, Epoch 10에서 validation loss는 0.029로 증가하였다. 학습과 테스트의 정확도는 Fig. 4.와 같이 accuracy가

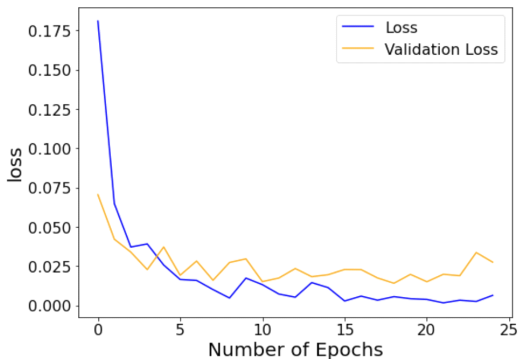

Fig. 3. Loss and validation loss

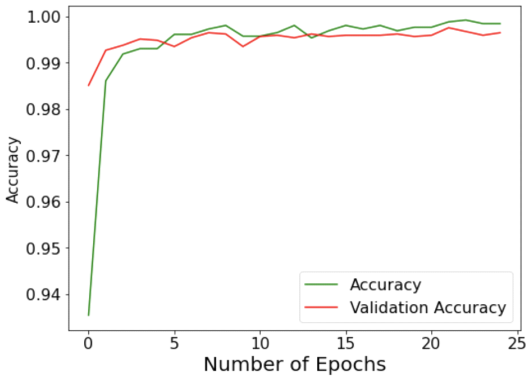

Fig. 4. Accuracy and validation accuracy

Epoch 10에서 0.9957, Epoch 10에서 validation accuracy가 0.9935로 감소하였다.

테스트 데이터로 평가 시 정확도는 Table 2.와 같이 loss는 0.2543, accuracy는 0.9717로 나타났다.

Table 2. Evaluate loss and accuracy

loss	0.2543
accuracy	0.9717

학습에 사용된 파일을 제외한 나머지 4770개의 스트림 데이터로 테스트를 진행하였다. 4770개의 스트림은 547개의 파일로부터 추출된 스트림이다. Table 3.은 학습된 모델에 테스트 데이터를 적용하고 결과를 confusion matrix를 이용하여 표현한 것이다. Accuracy는 0.9717, precision은 0.972, recall은 0.879, f1-score는 0.923의 결과가 도출되었다.

Table 3. Confusion matrix by stream

		actual	
		malware	normal
predicted	malware	3823	23
	normal	112	812

파일 단위로 진행한 평가는 악성 파일이면 malware 값이 1, 정상 파일이면 malware 값을 0으로 하고 비교 데이터로 모델 평가 시 예측한 데이터를 토대로 파일 내의 스트림 데이터에 악성 키워드가 하나라도 존재하면 malware 값은 1, 아니면 0으로 적용하여 정확도를 평가하였다. Table 4.는 탐지 결과를 파일 단위로 평가하여 confusion matrix로 표현한 것이다. 547개의 파일 중 504개의 파일을 정답으로 분류하였고 0.9213의 정확도를 보여주었다. 악성 파일을 정상 파일로 분류한 41개의 악성 파일은 사이즈가 다른 악성 파일 대비 작은 사이즈를 가지고, 카운팅 된 키워드의 수도 정상 파일과 비슷하였다. 정상 파일을 악성 파일로 분류한 2개의 정상파일은 Security와 .DLL 이라는 키워드를 포함하고 있었다.

Table 4. Confusion matrix by file

		actual	
		malware	normal
predicted	malware	278	2
	normal	41	226

4.5 Discussion

데이터를 학습하여 스트림 단위로 평가를 진행했을 경우 0.9717의 정확도를 보여주었고, 파일 단위로 평가를 진행했을 경우 0.9214의 정확도를 보여주었다. 정상 파일에는 없고 악성 파일에만 존재하는 키워드를 더 찾아서 데이터를 추출하고 학습을 시킨다면 더 높은 정확도를 보여줄 것으로 예상한다.

V. Conclusions

웹사이트나 메일의 첨부 파일을 통해 다운로드되는 문서형 악성코드의 유포가 활발하게 이루어지고 있다. 이는 사용자가 첨부 파일의 제목만 보고 다운로드하여 파일을 열람할 가능성이 높고, 실행 파일이 직접적으로 실행되는 것이 아닌 external의 URL을 통한 C&C 서버와의 연결, VBA 매크로를 담아

프로세스를 조작하거나 스니핑하는 유형의 악성코드이기 때문에 다른 악성코드에 비해 보안 프로그램의 탐지 우회가 비교적 유용하다. 이를 탐지하기 위해 문서형 악성코드 내 인코딩, 디코딩, 암호화 등으로 숨겨져 있는 악성코드를 찾고 외부로의 연결을 시도하는 키워드를 찾아내기 위해 스트림 데이터를 수동으로 분석하였다.

정상 파일 550개, 악성 파일 550개를 수집하여 해당 파일의 스트림 데이터를 저장하여 순차적으로 악성 키워드가 있는지 검사하여 악성 키워드 위치를 태깅하고 악성 키워드 존재 여부를 표시하여 데이터셋을 만들었다.

딥러닝 알고리즘 중 하나인 CNN 알고리즘을 이용하기 위해 키워드를 아스키코드로 변환하여 저장하였다. 하나의 바이트에 0x00부터 0xFF까지 표현되는 데이터를 아스키코드로 변환하여 저장하면 0에서 255까지로 표현된다. CNN 알고리즘이 이미지 데이터의 공간 정보를 유지하면서 인접 이미지와의 특징을 효과적으로 인식하고 강조하는 방식이다. 본 연구에서 만든 데이터셋도 이미지와 마찬가지로 값이 0에서 255까지 분포되어있고, 악성 키워드가 존재하는 위치를 특정하여 인접 정보를 인식하고 강조하기 위하여 CNN 알고리즘을 채택하였다. CNN 알고리즘을 이용하여 스트림 단위로 테스트를 진행한 결과로 0.9717의 정확도를, 파일 단위로 테스트를 진행한 결과로 0.9213의 정확도를 보여주었다.

향후 연구로는 이러한 문서형 악성코드를 정확히 탐지하는 인공지능 모델을 구현하기 위하여 본 연구에서 부족했던 악성 키워드를 추가로 수집할 예정이다. 키워드를 많이 수집하되 정상 파일에는 최대한 등장하지 않는 키워드를 찾는 것이 악의적인 문서를 찾는 데에 도움이 된다. 악성 문서를 추가로 수집하고 분석하여 키워드를 추가하고 딥러닝 모델에 적용하여 본 연구에서 진행한 모델과 비교하여 정확도를 높일 예정이다.

References

[1] "Malware that exploits HWP files has been found," Boannews, Aug. 2018. pp.1

[2] "Malicious documents disguised as reimbursement requests," AhnLab, Mar, 2021. pp.1

[3] "RTF malware disguised as a cover letter of a specific airline," AhnLab, Oct. 2021. pp.1

[4] Wikipedia, "Com Structured Storage," https://en.wikipedia.org/wiki/COM_Structured_Storage, Jan. 2022.

[5] Young-Seob Jeong, Jiyoung Woo, SangMin Lee and Ah Reum Kang, "Malware Detection of Hangul Word Processor Files Using Spatial Pyramid Average Pooling," Sensors, 20(18), pp.5265, Sep. 2020.

[6] Ah Reum Kang, Young-Seob Jeong, Se Lyeong Kim and Jiyoung Woo, "Malicious PDF Detection Model against Adversarial Attack Built from Bengin PDF Containing JavaScript," Applied Sciences, 9(22), pp. 4764, Nov. 2019.

[7] Young-Seob Jeong, Jiyoung Woo and Ah Reum Kang, "Malware detection on byte streams of pdf files using convolutional neural networks," Security and Communication Networks, 2019. Apr. 2019.

[8] Young-Seob Jeong, Jiyoung Woo and Ah Reum Kang, "Malware Detection on Byte Streams of Hangul Word Processor Files," Applied Sciences, 9(23), pp. 5178, Jan. 2019.

[9] Ah Reum Kang, Young-Seob Jeong, Se Lyeong Kim, Jonghyun Kim, Jiyoung Woo and Sunoh Choi, "Detection of malicious pdf based on document structure features and stream objects," The Korea Society of Computer and Information, 23(11), pp.85-93, Nov. 2018.

[10] Chae-Eun Yoon, Hey-hyeon Jeoung and Chang-Jin Seo, "Detection for Document-Type Malware Code using Deep Learning Model and PDF Object Analysis," The Koran Institute of Electrical Engineers, pp.44-49, Mar.

2021.
〔11〕 Dekkyu Lee and Sangjin Lee, "A Study of Office Open XML Document-Based Malicious Code Analysis and Detection Methods," Journal of the Korea Institute of Information Security & Cryptology, pp. 429-442, Jun. 2020.
〔12〕 MinJi Choe, KangSik Shin and DongJae Jung, "HWP Format Vulnerability Analysis For Document-Type Malware Detection," The Korean Institute of Information Scientists and Engineers, pp.1188-1190, Jun. 2020.
〔13〕 Sung Hye Cho and Sang Jin Lee, "A Research of Anomaly Detection Method in MS Office Document," The Korea Information Processing Socirety, 6(2), pp.87-94, 2017.

〈저자소개〉

박 현 수 (Hyun-su Park) 학생회원
2022년 2월: 배재대학교 사이버보안학과 학사
2022년 3월~현재: 배재대학교 사이버보안학과 석사과정
〈관심분야〉 정보보호, 인공지능

강 아 름 (Ah Reum Kang) 종신회원
2006년 2월: 서울여자대학교 컴퓨터공학과 학사
2012년 2월: 고려대학교 정보보호학과 석사
2016년 2월: 고려대학교 정보보호학과 박사
2021년 3월~현재: 배재대학교 정보보안학과 조교수
〈관심분야〉 정보보호, 인공지능, 악성코드