JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 30 No. 1, pp. 87-97, January 2025
https://doi.org/10.9708/jksci.2025.30.01.087

HWP Malware Detection using Transformer

Gati Lother Martin*, Young-Seob Jeong**, Ah Reum Kang***, Jiyoung Woo****

*PhD. candidate, Department of Future Convergence Technology, Soonchunhyang University, Asan, Korea
**Professor, Department of Computer Engineering, Chungbuk National University, Cheongju, Korea
***Professor, Department of Information Security, Pai Chai University, Daejeon, Korea
****Professor, AI and Bigdata Department, Soonchunhyang University, Asan, Korea

[Abstract]

In this study, we perform static analysis on Hangul document-type malware files, including the extraction of scripts and shellcode, and carry out dynamic analysis based on the type of malware to identify characteristics of Hangul document-type malware files. Based on these characteristics, we learn information about Hangul document-type malware files and develop a deep learning-based detection model for them. Decoding streams in HWP files typically generate readable text composed mainly of JavaScript. We have proposed an effective document-based malware detection model using a transformer model that leverages an attention mechanism to capture dependencies between input and output. In this research, we differentiated Hangul-type malware provided by antivirus companies by document components (streams) and, after training on 17,265 instances, achieved an accuracy of over 96%.

▸ **Key words**: Malware, Document, Hangul word processor, Transformer model, Deep learning

[요 약]

본 연구에서는 한글 문서형 악성 파일을 대상으로 스크립트 및 쉘코드 추출 등의 정적 분석을 수행하고 악성코드 유형에 따른 동적 분석을 수행하여 한글 문서형 악성 파일의 특징을 추출한다. 도출된 특징을 기반으로 한글 문서형 악성 파일의 정보를 학습하고 딥러닝 기반의 한글 문서형 악성 파일 탐지 모델을 구축한다. HWP 파일의 스트림을 디코딩하면 주로 JavaScript로 구성된 읽을 수 있는 텍스트가 일부 생성된다. 본 디코딩된 텍스트를 입력과 출력 간의 의존성을 포착하는 주의 메커니즘(attention mechanism)을 활용한 transformer 모델을 통해 효과적인 문서 기반 악성코드 탐지 모델을 제안하였다. 본 연구에서는 백신사에서 제공된 한글형 악성코드를 문서 구성요소별(스트림) 구분하고 17,265 개를 학습한 결과 96%이상의 정확도를 달성하였다.

▸ **주제어**: 악성 코드, 문서, 한글 워드 프로세서, 트랜스포머 모델, 딥러닝

• First Author: Gati Lother Martin, Corresponding Author: Jiyoung Woo
*Gati Lother Martin (gatimartin1@gmail.com), Department of Future Convergence Technology, Soonchunhyang University
**Young-Seob Jeong (ysjay@chungbuk.ac.kr), Department of Computer Engineering, Chungbuk National University
***Ah Reum Kang (armk@pcu.ac.kr), Department of Information Security, Pai Chai University
****Jiyoung Woo (jywoo@sch.ac.kr), AI and Bigdata Department, Soonchunhyang University
• Received: 2024. 12. 27, Revised: 2025. 01. 20, Accepted: 2025. 01. 20.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

Malicious codes embedded in various document formats, such as Hangul, Word, Excel, and PDF files, continue to be widely disseminated, targeting users across major public institutions and corporations [1]. These malicious documents often impersonate legitimate applications, including events like the 2020 Peace and Unification Contest, research papers, resumes, and self-introductions. In Korea, several hacking incidents have been linked to such files. Examples include malicious Excel documents targeting the Black Friday season in 2021, and document files discussing the outcomes of the 2nd US-DPRK summit in February 2019. Additionally, malicious Hangul Word Processor (HWP) files were discovered in March 2021, masquerading as domestic portal ID transaction contracts, while Word files disguised as payment requests and export gold bar sales contracts were spread in the same year. There were disguised malicious Word files impersonating recent disaster funds related to Corona that were widely spread [2,3]. These document-based malware hides in the documents and becomes difficult to detect. If the malware has an exe extension, it gets suspicious to the receiver. Unlike executable (.exe) files, which often raise suspicion, users are more likely to trust and open documents or images, such as Hangul or Excel files. This vulnerability facilitates the rapid spread of document-type malware [4]. The existing security solutions designed for Portable Executable (PE) files have limitations in detecting and mitigating threats posed by document-based malware.

To mitigate the propagation of malicious code via document files targeting government agencies and corporations, it is crucial to conduct research on document-based malware analysis. Previous studies on document-based malware are insufficient to develop a generalized detection model applicable across various document types. This limitation arises because they primarily focus on extracting features from malicious files in specific formats, such as Hangul and PDF. To effectively detect novel malware and its variants, generic methods that are not specific-sample dependent are necessary.

In this study, we proposed the use of a language model that is universally applicable to texts in any language. Malware often embeds programming languages to execute its malicious functions; thus, we assume that a human language model trained on human language can also be effective for analyzing programming languages. Hangul document-type malicious codes aimed at stealing confidential information, achieving financial gains, or gathering sensitive data [5] are continuously increasing. Cyberattacks leveraging Hangul documents to target government agencies or domestic corporations pose significant risks, potentially leading to widespread domestic disruption. Malware infection using Hangul documents is specific to the software called Hangul, so there has been a lack of research on Hangul files, which are widely used in Korea. Our initial focus is on HWP files; however, the proposed model can be extended to other formats, such as PDF. Most of the attacks on documents rely on script languages for their operations. Our model is designed to classify malicious and benign scripts, including JavaScript, enabling robust detection of document-based malware.

The rest of this paper is organized as follows: Section II discusses the related research and explores different approaches. Section III describes our proposed methodology for malware detection and classification. Section IV presents the dataset and experiments. In Section V, we provide an in-depth discussion of the results. Finally, the conclusions and future research plans are discussed in Section VI.

II. Related Works

Document-type malicious code is often hidden by encoding it within the document, enabling it to bypass traditional anti-virus programs. A common method involves embedding malicious code in scripting languages, as scripting languages like JavaScript and VBScript are widely used across various programs and applications. Malware developers exploit this by embedding harmful scripts into document files [6], making these attacks both versatile and challenging to detect.

To counter the growing prevalence of these threats, various studies have been conducted to develop tools and techniques for detecting and categorizing malicious HWP files. Some techniques focus on scanning the entire file to identify anomalies [7], while others extract and analyze specific attack strategies, such as detecting embedded script-based attacks in documents [6,8]. Most of these approaches rely on machine learning methods, including support vector machines (SVM), random forest (RF), LSTM, and CNN [9-11]. The study [9] focuses on detecting malicious actions within a given byte sequence of a target HWP file by using a spatial pyramid average pooling in the CNN model. The proposed CNN takes long byte streams as input because the byte streams of HWP files are much longer than those of PDF files. They tested various pooling layer settings and proposed an appropriate pooling method for the byte stream. The best model achieved F1 scores of 84.67% and 92.27% for benign and malicious cases, respectively. Object Linking and Embedding (OLE) is a file format used in different types of documents, structured with storage and streams that resemble the File Allocation Table (FAT) file system. The stream holds all the data, including any malicious code embedded within it. [6] conducted OLE file analysis by selecting repeating keywords appearing in malicious Microsoft Office files as features. They used machine learning models to detect malware and achieved an accuracy of 99.80% with RF. The authors in [12] developed a method to extract and evaluate a set of explainable features for efficient and timely PDF malware detection. A classifier that significantly improved classification efficacy with shorter training times was constructed, deploying features of the highest significance and improving classification performance by 2%. [8] created a model for detecting malicious PDF documents containing JavaScript. They conducted analysis by examining lexical features within the JavaScript content. The model achieved a true positive rate of 85.17% on a large-scale sample collected from the VirusTotal. To detect document-type malicious code, [13] selected keywords suspected of being malicious from MS Office stream data, which were then converted to ASCII code values. A CNN model was used to classify the malicious streams and achieved 97.0% accuracy. CNN was proposed by [14] to detect malware on byte streams of PDF files. In their study, streams in the form of bytes are fed to the CNN model. Rather than feature engineering on malware documents, the proposed model is applicable to any type of document after transforming streams into byte sequences.

Attention-based models have been proposed for feature extraction or as a malware classifier. Although CNNs have proven effective in capturing local features, attention mechanisms play a vital role in detecting structural changes in malware. Studies [15,16] demonstrated that attention-based approaches can improve performance compared to deep learning methods such as CNN or LSTM. SelAttConvLstm, a novel deep learning model, was designed by [17] to extract the temporal and spatial characteristics of network traffic data for Android malware detection. Self-attention mechanism was applied to improve the accuracy of the model. [18] proposes an oversampling-based method to detect unseen malicious JavaScript snippets with attention and Doc2Vec The proposed method was found to have shorter training time and better accuracy with a recall score of 0.72 on an imbalanced dataset.

Many studies have developed AI-based detection models based on the bytes form of malware, particularly PE files. Some use an n-gram language model that extracts n-grams from bytes and applies machine learning models. In our study, we adopted a language model on texts rather than bytes to utilize the code sequences and their associations. This study integrated the Code2Vec with the transformer model, enabling the extraction of richer and more adaptive feature representations. The proposed model can be applied to detect malware in other file formats that share the same structure, including MS Office and PDF files.

III. Methodology

1. HWP document structure

Hangul documents utilize the Compound File Binary Format (CFBF), incorporating Object Linking and Embedding (OLE) technology to facilitate data linkage (such as images, tables, etc.) among files adhering to this format. The CFBF structure encompasses directories (known as storages) and files (referred to as streams). Unlike a typical operating system directory that does not hold data, the data within a CFBF file is contained within streams located in storages. This analysis focuses specifically on streams that hold data. HWP documents are organized in a hierarchical tree structure with RootEntry at the apex. The file header, document metadata (DocInfo), text previews, and image previews are embedded in these streams. The text displayed in HWP viewers is stored in the BodyText storage, along with its stream section number. This storage section also contains components like paragraphs, tables, illustrations, and text. Each binary data set forms its own separate storage. Both the body text and binary data are organized into records, comprising a record header and the associated data. The comprehensive layout of HWP documents is depicted in Figure 1.

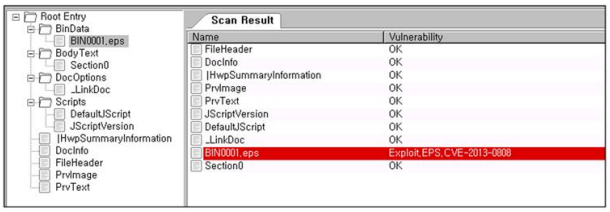

Fig. 1. Structure of HWP files.

Malware is created by embedding it as a stream in BodyText or binary in the form of shellcode and JavaScript. Malware steams that include malicious codes usually have a longer length compared to normal streams. Fig. 2 and 3 shows an example of malware. When an HWP file is opened by the OLE viewer, the encrypted text is found in the bodytext as a stream.

When we check the hex of this, we observe a string highlighted in yellow. Texts decrypted and decompressed from hex reveal that the file has shellcodes and script codes that execute these shellcodes. In this study, we examined these readable codes shown in text form to identify the differences between streams embedding malware.

2. Readable Text in scripts embedded in stream

When malicious code is created, it is obfuscated or packaged to enable the static analysis to be difficult. Obfuscation and packaging are tasks that allow codes to be written in a programming language and render file structure format difficult to read. Encrypting or compressing executable files renders it difficult to analyze the source code. Readable texts can be observed after being compressed. However, malicious codes generates more noises that are not decompressed than benign files, such as byte arrays in hexadecimal. Fig. 2 shows an example of encrypted contents stored in a stream and Fig. 3 displays an example of decrypted text from those contents.

For text-based malware detection, readable parts were extracted from the stream. One file is composed of several streams, and some streams

are compressed or encrypted depending on the type. Because the malicious code is inserted into the stream in various ways, the readable parts are extracted not only from the compressed stream but also from the decompressed stream. Not only English uppercase and lowercase letters and numbers, but also symbols such as parentheses and dots can be included in Encapsulated PostScript. VBA script or Javascript are extracted together. Fig. 3 is a sample file in which readable programming language parts such as English uppercase and lowercase letters, numbers, and some symbols are extracted. Reable texts include Windows Application Programming Interfaces (APIs) such as GetFileSize, FindFirstFileW, GetDriveTypeW, CreateDirectoryW, CreateProcessA, ReadFile, CreateFileW, FindClose, MoveFileA, lstrcatW, RegOpenKeyExW, RegCloseKey, RegSetValueExA, RegOpenKeyExA, ShellExecuteA, file names such as WSHELL32.dll, kernel32.dll, user32.dll, class names such as Object, Descriptor, Base, CONNECT, User, Agent, compatible, Sunday, Sat, Fri, Thu, Wed, Tue, Mon, Sun, and phrases such as 'not enough for stdio initialization', 'pure virtual function call'.

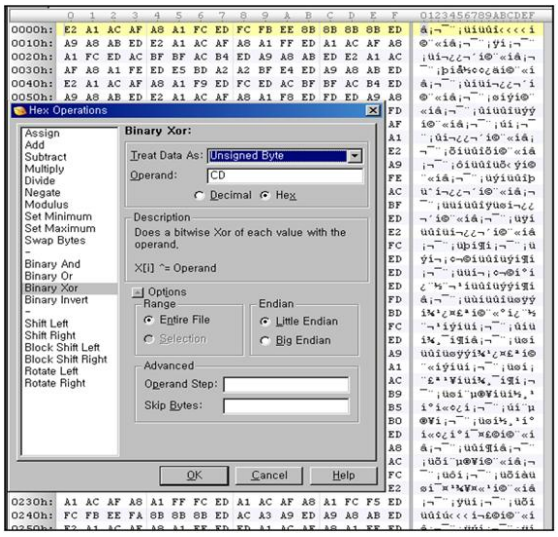

Fig. 2. Encrypted content in stream

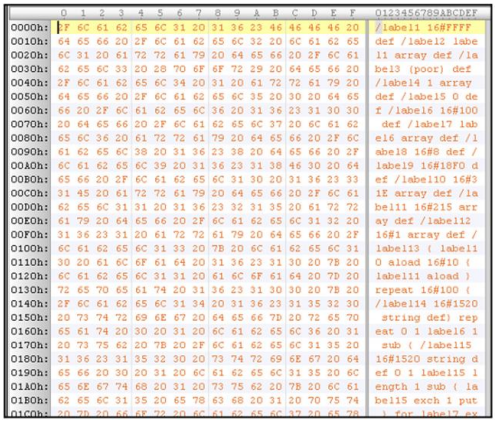

Fig. 3. Decrypted text from stream.

3. Proposed Method

To detect the malicious actions in code sequences of HWP files, we use a transformer based model. Transformer is a neural network architecture that uses attention [19] to handle sequential input data. Recurrent neural networks are one of the best ways for processing sequential data, however, sequential computation precludes parallelization which becomes critical for longer sequence lengths. Transformer introduces the attention mechanism to effectively handle sequence-based input without relying on recurrence. The goal of the attention mechanism is to learn the dependencies between the words in the sentence and perceive the context.

The transformer architecture has two parts; an encoder and a decoder. The encoder has the role of abstracting and representing the input; while the decoder has the role of generating the output by receiving the abstract of the input.

In this study, we used only the encoder part of the transformer to generate the abstracts that contain the information of the input sequence. The encoder layer converts the input sequences into continuous representations that hold the learned information for the entire sequence. Encoder is implemented by integrating multi-head attention, layer normalization [20] for regularization, and a fully connected network. Initially, the extracted streams were tokenized into tokens (i.e., characters or words) using a tokenizer algorithm. Then, the

sequence of these tokens was fed to the input embedding layer and position encoding.

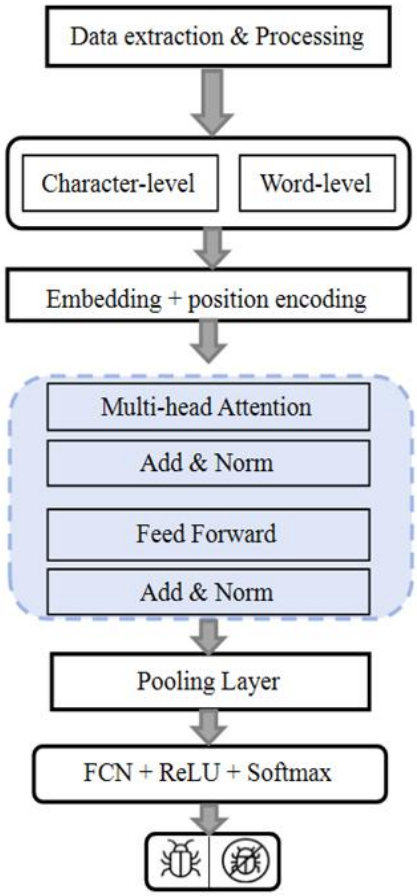

Fig. 4. Overview of the proposed method.

Here, the vector representation for each token was produced with the position information to capture the meaning and position of each token. These vector representations are then fed into three parameters: query, key, and value in the encoder self-attention. The query contains the vector of one word to pay attention to, whereas the keys are vector representation of all words in the sequence. The self-attention is calculated as softmax-normalized dot products between the query and key vectors. The output is the weighted sum of the value vectors.

In malware static analysis, the byte sequences are converted to an encoding scheme that is human-readable; however, it is still not easy to learn the difference in pattern between malicious and normal scripts. Therefore, learning to distinguish between malware and normal scripts requires the ability to pay attention to each part of the sequence. To capture multiple complex relationships, the transformer computes multiple attentions for the given sequence. Multi-head attention is the extension of the attention mechanism to jointly attend multiple positions in parallel. The attention module splits its query, key, and value parameters and passes each independently through a separate head. The final score is calculated by combining all attention scores. In our proposed model, on the top of the transformer layer, we used a feed-forward network with a pooling layer and two fully connected layers, as shown in Fig. 4.

IV. Experiments

1. Dataset

The data were collected from approximately 1,200 malicious HWP files and 360 benign files. Benign files were collected from websites, usually created by government agencies. Malware files were provided by a Korean antivirus company. These files included a number of stream objects, of which one or more contain malware. Selected samples were analyzed based on known signatures to recognize the malicious actions. Then, malicious actions were run in a virtual environment to validate. Based on sample analysis, we could build the rule-based system to check whether the stream in a file is malicious or not. Fig 6. is an example of a case where encrypted codes exists in a HWP file a. When the file is opened with the HWP OLE viewer, the codes are decrypted and shown.

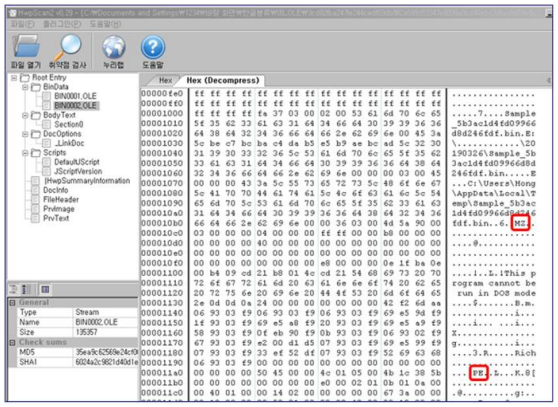

Fig. 6. Malware document including portable executable.

When it is opened with a text editor to analyze the code in detail, we can see the GhostScript program (gbb.exe) execution part for executing PostScript, as well as the PE execution part. Fig. 7 shows the part that executes the GhostScript program, separated by version. Fig. 8 shows the part that reads and executes PE binary.

```
/gswin32c (C:\\Program Files\\Hnc\\Common80\\ImgFilters\\GS\\gs8.60\\bin\\gswin32c.exe) def
gswin32c status
{
        /path gswin32c def
}if
/gswin32c (C:\\Program Files (x86)\\Hnc\\Common80\\ImgFilters\\GS\\gs8.60\\bin\\gswin332c.exe) def
gswin32c status
{
        /path gswin32c def
}if
/gswin32c (C:\\Program Files\\Hnc\\HOffice9\\Bin\\ImgFilters\\GS\\gs8.60\\bin\\gswin332c.exe) def
gswin32c status
{
        /path gswin32c def
}if
/gswin32c (C:\\Program Files (x86)\\Hnc\\HOffice9\\Bin\\ImgFilters\\GS\\gs8.60\\bin\\gswin332c.exe) def
gswin32c status
{
```

Fig. 7. Codes executing ghost script program.

```
}if
/datastring 1024 string def
{
        path (w) file / out exch def
        {
                current file datastring readhexstring
                {
                        out exch writestring
                }
                {
                        dup length 0 gt
                        {out exch writestring} {pop} ifelse
                        exit
                }ifelse
        }loop
        out closefile
}bind
exec
4D5A90003000000040000000FFFF0000B8000000000000004000000000000000
```

Fig. 8. Malware document including PE.

Preprocessing involved removing null contents and assigning a unique number to the streams that originated from the same files. Finally we collected a total of 17,265 samples (benign: 15,530, malicious: 1,735). The data was then split into train and test sets with a ratio of 85% and 15%, respectively.

2. Model

The experiments were performed in two phases: stream and file level. At the stream level, the streams were classified as either malicious or benign. In each phase, character and word-level embedding was used to represent the features. Fig. 5 shows the overview of the proposed classification approach. After preprocessing, the streams were tokenized using a tokenization algorithm. The vectorization on both character and word level. For the character-level, the vocabularies were built with only characters, whereas at the word-level, the vocabularies were built with words (including characters also). The generated representations were the input to the transformer, which learns the representations and adds the positional information to each token in the sequence. The transformer received these representations and used the self-attention mechanism to capture the relevance between each pair of tokens.

Table 1. Parameter settings for the transformer model.

Parameter	Value
Embedding dimension	100
Attention heads	3
Dropout rate	0.2
Hidden layer units (transformer)	32
Hidden layer units (neural network)	(32, 16, 10)

The proposed architecture was developed by performing different experiments with different parameter settings to obtain the optimal ones.

Table 1 summarizes the optimal parameters for the classification model. To transform input codes into numerical values (integer vector), the embedding size was 100, the number of attention

Table 2. Experimental results where the two values of performance metrics correspond to benign and malicious cases, respectively.

Embedding	Model accuracy		Performance metrics			File Accuracy	
	Accuracy	F1-score	Precision	Recall	F1-score	Accuracy	F1-score
Char-level	95.39	72.85	96.88/79.03	98.06/69.67	97.47/73.99	90.06	75.28
Word-level	96.76	82.90	98.28/82.35	98.14/83.47	98.20/82.91	94.00	86.03

heads was three, and the hidden layer size for the feed-forward network inside the transformer was 32. In addition, we use GlobalMaxPooling1D() pooling layer followed by two or three fully connected layers with rectifier linear unit (ReLU) [21] as the activation function. We adopted the Adam optimizer [22] with an initial learning rate of 0.001, and a dropout [23] probability of 0.2 for the fully connected layers. The parameters were optimized using the early-stopping algorithm. The sequence length was set to 5,000 and 210 for character-level and word-level respectively. We use accuracy, precision, recall, and F1 score as metrics to evaluate the performance for both phases, as shown in Table 2.

V. Results and Discussion

Table 2 lists the results of the two embedding methods, respectively. As discussed in Section 3, the ability of the transformer to attend to each part of the sequence and capture the main pattern of the malware scripts has an impact on the performance of the model. The performance on word-level embedding provides better accuracy compared to character level. This is because, compared to characters, words carry more meaning that may aid in learning the difference between patterns (malicious or benign). This suggests that the model can perform well on semantic (i.e., compare similar scripts) as well as syntactic (i.e., learn sequence patterns) as discussed by [24].

Because our goal is to detect the malware attack in the document file, a low false negative rate is highly important because it represents the ratio of missed malicious cases which may be considered benign cases. On that basis, the higher recall (i.e. sensitivity) of malicious cases should be preferable. Overall, the best detection model, with high recall and best accuracy use word-level embedding. The effectiveness of the model is quantified using area under the curve (AUC) scores. Table 3 shows the AUC scores of three experiments with different parameters through the grid search for word-level embedding. Integer vector scored the false positives rates of (42∼45).

Table 3. AUC scores for three experiments with input sequence in word-level embedding.

Model	AUC	FN/FP
1	97.38	39/44
2	97.01	41/42
3	96.82	41/45

The current study developed a model by analyzing samples provided by antivirus companies. Document-based malware is not widely disclosed because it is decryptable, making it susceptible to misuse. While some PDF malware samples are available on public sites, there is no publicly available data for Hangul documents, limiting our ability to conduct experiments with other data sources.

Malware infections using Hangul documents are specific to the Hangul software and therefore attract relatively little global attention. Similar to other document-based malware infections, malware can be facilitated using various features of Hangul, and the presence of lure documents makes it easy to deceive users. Moreover, like other office documents, it uses an OLE structure, but its data

structure is complex, making it more challenging to analyze compared to other office documents.

Due to the complexity of the model developed for Hangul documents, which feature an intricate OLE structure, this approach is applicable to other documents with similar OLE configurations. Moving forward, there is a plan to conduct further research to analyze and develop malware detection models for different types of documents. Additionally, there is an interest in cross-verifying whether the malware detection model created for Hangul documents can be effectively adapted to other document formats. This cross-validation will help ensure the model's robustness and versatility, potentially expanding its applicability across various document-based environments and enhancing overall cybersecurity measures.

In addition, as future work, we will examine the impact of imbalanced data. For that, we will apply sampling methods: oversampling and undersampling. The oversampling method increased the minority samples with multiple copies to establish a balance between the classes. In contrast, undersampling decreases (deletes) the number of majority classes until a balanced distribution is achieved. Furthermore, more sophisticated techniques like GANs (generated adversarial networks) and SMOTE will be investigated to generate synthetic samples instead of duplicating existing ones.

VI. Conclusions

Malware attacks are becoming one of the most serious threats to information security, particularly during pandemic when most activities are performed remotely. Developers are using different techniques, such as code obfuscation to bypass detection. To counter these attacks, we must use advanced methods to learn their techniques and identify their common traits, such as codebase. In this study, we proposed a method to detect malicious cases in HWP document files using a transformer-based model. More than 1,200 files were used to create the malware dataset. The experiments were performed in two phases: stream and file-level binary classification (i.e. to detect if stream or file is malicious or benign). The efficiency was evaluated in terms of accuracy, AUC score, and F1 score. The model that utilizes Code2Vec with word-level embedding outperformed other approaches, achieving an accuracy of 96.85% on steam classification. In the future, we will focus on the reducing false negatives by increasing the number of malicious samples through the use of data generation techniques. Furthermore, since transformer-based models have significant computational complexity, we plan to explore optimization strategies to ensure that the model can be applied efficiently in real-time detection.

ACKNOWLEDGEMENT

This research was supported by "Regional Innovation Strategy (RIS)" through by the National Research Foundation of Korea (NRF) by the Ministry of Education(MOE) (2021RIS-004) and by 2022 Sabbatical Year of Soonchunhyang University.

REFERENCES

[1] Singh, P.; Tapaswi, S.; Gupta, S. Malware detection in pdf and office documents: A survey. Information Security Journal: A Global Perspective 2020, 29, 134-153. DOI: 10.1080/19393555.2020.1723747

[2] Khweiled, R.; Jazzar, M.; Eleyan, D. Cybercrimes during COVID-19 Pandemic. International Journal of Information Engineering & Electronic Business 2021, 13. DOI: 10.1016/j.dcan.2022.06.005

[3] Pranggono, B.; Arabo, A. COVID-19 pandemic cybersecurity issues. Internet Technology Letters 2021, 4, e247. DOI: 10.1002/itl2.247

[4] Mainka, C.; Mladenov, V.; Rohlmann, S. Shadow Attacks: Hiding

and Replacing Content in Signed PDFs. In Proceedings of the Proceedings 2019 Network and Distributed System Security Symposium. Internet Society, 2021. DOI: 10.14722/ndss.2021.24117

[5] Polito, C. The Evolution of North Korean Cyber Threats 2019.

[6] Choi, H.K.; Kang, A.R. OLE File Analysis and Malware Detection using Machine Learning. Journal of the Korea Society of Computer and Information 2022, 27, 149-156. DOI: 10.9708/JKCSI.2022.27.05.149

[7] Ali, M.; Shiaeles, S.; Bendiab, G.; Ghita, B. MALGRA: Machine learning and N-gram malware feature extraction and detection system. Electronics 2020, 9, 1777. DOI: 10.3390/electronics9111777

[8] Laskov, P.; Šrndi´c, N. Static detection of malicious JavaScript-bearing PDF documents. In Proceedings of the Proceedings of the 27th annual computer security applications conference, 2011, pp. 373–382. DOI: 10.1145/2076732.2076785

[9] Jeong, Y.S.; Woo, J.; Lee, S.; Kang, A.R. Malware Detection of Hangul Word Processor Files Using Spatial Pyramid Average Pooling. Sensors 2020, 20, 5265. DOI: 10.3390/s20185265

[10] Azeez, N.A.; Odufuwa, O.E.; Misra, S.; Oluranti, J.; Damaševi˘cius, R. Windows PE malware detection using ensemble learning. In Proceedings of the Informatics. Multidisciplinary Digital Publishing Institute, 2021, Vol. 8, p. 10. DOI: 10.3390/informatics8010010

[11] Agrawal, P.; Trivedi, B. Machine learning classifiers for android malware detection. In Data Management, Analytics and Innovation; Springer, 2021; pp. 311–322. DOI: 10.1007/978-981-15-5616-6_22

[12] Falah, A.; Pan, L.; Huda, S.; Pokhrel, S.R.; Anwar, A. Improving malicious PDF classifier with feature engineering: A data-driven approach. Future Generation Computer Systems 2021, 115, 314-326. DOI: 10.1016/j.future.2020.09.015

[13] Park, H.s.; Kang, A.R. MS Office malicious document detection based on CNN. Journal of the Korea Institute of Information Security & Cryptology 2022, 32, 439-446. DOI: 10.13089/JKIISC.2022.32.2.439

[14] Jeong, Y.S.; Woo, J.; Kang, A.R. Malware detection on byte streams of pdf files using convolutional neural networks. Security and Communication Networks 2019, 2019. DOI: 10.1155/2019/8485365

[15] Choi, S.; Bae, J.; Lee, C.; Kim, Y.; Kim, J. Attention-based automated feature extraction for malware analysis. Sensors 2020, 20, 2893. DOI: 10.3390/s20102893

[16] Ganesan, S.; Ravi, V.; Krichen, M.; Sowmya V.; Alroobaea, R.; Soman, K. Robust malware detection using residual attention network. In Proceedings of the 2021 IEEE international conference on consumer electronics(ICCE). IEEE, 2021, pp. 1-6.

[17] Shen, L.; Feng, J.; Chen, Z.; Sun, Z.; Liang, D.; Li, H.; Wang, Y. Self-attention based convolution-LSTM for android malware detection using network traffics grayscale image. Applied Intelligence 2023, 53, 683-705. DOI: 10.1007/s10489-022-03523-2

[18] Phung, N.M.; Mimura, M. Detection of malicious javascript on an imbalanced dataset. Internet of Things 2021, 13, 100357. DOI: 10.1016/j.iot.2021.100357

[19] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. Advances in neural information processing systems 2017, 30.

[20] Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. arXiv preprint arXiv:1607.06450 2016. DOI: 10.48550/arXiv.1607.06450

[21] Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the Icml, 2010.

[22] Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 2014.

[23] Wager, S.; Wang, S.; Liang, P. Dropout training as adaptive regularization. arXiv preprint arXiv:1307.1493 2013.

[24] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 2013. DOI: 10.48550/arXiv.1301.3781

Authors

She received her M. Sc. Eng in Big Data Engineering, ICT convergence department from Soonchunhyang University, Asan, South Korea in 2022. She's currently pursuing her Ph.D. as a research assistant at Advance Data Analytics at Soonchunhyang University, Asan city, South Korea. Her research interest mainly focuses in Machine & Deep Learning in language processing.

Young-Seob Jeong received the M.S., Ph.D degree in computer science from KAIST, Daejeon, Korea, in 2012 and 2016, respectively. He is a faculty member of the department of computer engineering, Chungbuk National university, Cheong-ju city, Korea. His current research topics include malware detection using deep learning techniques, language models, healthcare system for patients, and pharmaceutics.

Ah Reum Kang received the M.S. and Ph.D degree in Information Security from in Seoul, South Korea in 2012 and 2016, respectively. She is a currently professor at Paichai University. From 2016 to 2018, she was a researcher at the Department of Computer Science and Engineering at the University at Buffalo, State University of New York, USA. Her research interest includes computer security, privacy-preserving data mining and data science in general.

Jiyoung Woo received the B.S., M.S., Ph.D. degree in Industrial Engineering from KAIST in 2000, 2002, and 2006. She is currently a Professor in the Department of Big Data Engineering, Soonchunhyang University. From 2008 to 2010, She was a researcher with AI lab of Arizona University, USA. She was a research professor at Graduate School of Information Security, Korea University from 2010 to 2016. Her research interest includes data mining and analytics.