

Malware Byte Stream Analysis Using Overlapped LDA

Young-Seob Jeong*, Yeong-Jin Kim**, Medard Edmund Mswahili***,
Jiyoung Woo****, Ah Reum Kang*****

*Professor, Department of Computer Engineering, Chungbuk National University, Cheongju, Korea

**Student, Department of Information Security, Pai Chai University, Daejeon, Korea

***Student, Department of Computer Engineering, Chungbuk National University, Cheongju, Korea

****Professor, AI and Bigdata Department, Soonchunhyang University, Asan, Korea

*****Professor, Department of Information Security, Pai Chai University, Daejeon, Korea

[Abstract]

More documents are appearing in online platforms, and people are vulnerable to malicious attacks in non-executable documents. Recently data-driven approaches have shown successful results in malware detection task. As they heavily rely on the dataset, it is important to make a lot of annotated data, while the annotation process is normally performed manually by domain experts. Therefore, it is necessary to develop a system or a tool that analyzes the files and help the annotation process. In this paper, we propose a new method that automatically analyzes files and generates byte-level labels using a modified version of overlapped dirichlet allocation that clusters given bytes into two (e.g., malware and benign) or more groups. By experimental results with our annotated dataset, we demonstrated that the generated byte-level labels achieved high recall (95~100%). We observed that our model suffered from low precision because the dataset is sparsely annotated, but it still has a potential to aid in finding suspicious bytes for malware analysis. We also provide sample results visualized by highlights with different colors.

▶ **Key words:** malware analysis, topic model, overlapped latent dirichlet allocation, byte stream

[요 약]

온라인 문서들이 점점 더 많이 생겨남에 따라, 문서형 파일에 대한 악의적인 공격에 취약해지고 있다. 최근 데이터 기반 접근법이 악성코드 탐지에서 성공적인 결과를 보여주고 있다. 이러한 방법은 주로 데이터셋에 크게 의존하므로, 정답이 태깅된 데이터를 충분히 많이 만드는 것이 필요하다. 하지만 태깅 작업은 도메인 전문가들이 수동으로 수행하므로, 파일을 분석하고 태깅 작업을 지원하는 도구를 개발하는 것이 필요하다. 본 논문에서는 파일을 자동으로 분석하고, 중첩 디리클레 할당 방법을 사용하여 바이트 단위의 레이블을 자동 생성하는 방법을 제안한다. 이 방법은 주어진 바이트들을 악성코드와 정상 파일 등 두 가지 이상의 그룹으로 군집화한다. 데이터셋을 이용한 실험 결과, 높은 재현율 (95~100%)을 달성함을 확인하였다. 데이터의 정답이 희소하게 태깅되어 있어 정밀도가 낮아지는 문제를 겪었지만, 악성 의심 바이트를 찾는 데 도움을 줄 가능성이 있음을 확인할 수 있었다. 찾아낸 바이트들은 다양한 색상으로 하이라이트되었으며, 이에 대한 샘플 결과를 시각화하여 제공하였다.

▶ **주제어:** 멀웨어 분석, 토픽 모델, 중첩된 잠재 디리클레 할당, 바이트 스트림

- First Author: Young-Seob Jeong, Corresponding Author: Ah Reum Kang
- *Young-Seob Jeong (ysjay@chungbuk.ac.kr), Department of Computer Engineering, Chungbuk National University
- **Yeong-Jin Kim (jin971001@naver.com), Department of Information Security, Pai Chai University
- ***Medard Edmund Mswahili (medardedmund25@chungbuk.ac.kr), Department of Computer Engineering, Chungbuk National University
- ****Jiyoung Woo (jywoo@sch.ac.kr), AI and Bigdata Department, Soonchunhyang University
- *****Ah Reum Kang (armk@pcu.ac.kr), Department of Information Security, Pai Chai University
- Received: 2024. 10. 15, Revised: 2024. 11. 25, Accepted: 2024. 11. 27.

I. Introduction

Malware detection is becoming more important as we use more data exchange services (e.g., e-mail) in daily life. The malware may cause serious damage to private database, business documents of companies, even classified documents of governmental institutions. There have been many studies on the task of malware detection, and recently deep-learning (DL) techniques brought significant performance improvements on the task[1-6]. Such DL techniques strongly rely on the data, so they give better results only if a plenty of well-annotated dataset is given: this is why the DL techniques belong to 'data-driven' approach. Many errors in the annotation may cause significant performance degradation of the data-driven approach, so it is important to construct the well-annotated dataset that is normally obtained by manual analysis by domain experts.

Several recent studies proposed DL models that exploit byte streams for malware detection task. These studies are promising as the DL models analyze the byte streams without having to define hand-crafted features: the DL models using byte streams are preferable because they extract arbitrary patterns from the byte streams so that we can easily apply them to newly appeared malware variants[7]. Of course, for training and evaluating such models, it is firstly required to annotate the byte streams and it is often done by domain experts. The domain experts mostly use one or more analysis tools such as structured storage viewer (SSV) and hex editor (HxD). Although these tools help the experts by displaying the raw bytes in different formats (e.g., HEX, Text, HTML), they commonly lack of functions that automatically analyzes byte streams for the domain experts. When a tool provides a function of byte-level analysis that provides a list of suspicious bytes, then it will be of enormous assistance to the experts by saving time and cost.

The boy came home.
But nobody was at home.
He searched for food, but nothing.

54 68 69 73 20 77 69 6C 6C 20
65 20 34 4E 00 03 00 00 00 00
61 79 6C 6F 61 64 00 00 05 42

Fig. 1. Examples of topic model results. (top) Results on a document, where each word has a label and some words of 'entity' label are boxed in this example, and (bottom) results on a byte stream, where each byte has a label and bytes of 'malware' (or suspicious) label are boxed in this example.

In this paper, we propose a new method of byte-level malware analysis using a probabilistic topic model. The probabilistic topic model extracts arbitrary topics from given documents, where the topics are essentially word distributions. As depicted in Fig. 1, if we assume each byte is a word, then the topic model will work on byte streams by treating them as a collection of documents. Topic models mostly work in bag-of-words (BoW) manner; they do not incorporate spatial or any sequential patterns but only care about co-occurrence of words. Overlapped latent dirichlet allocation (OLDA)[8] is known to overcome such limitation by allowing adjacent documents to be overlapped. We slightly change the structure of the OLDA model to make it work on the byte streams because it was originally designed for image analysis. We demonstrate that the OLDA model gives byte-level labels that indicate each byte is suspicious or not, and also show that the byte-level labels are consistent with malware offsets annotated by domain experts. As far as we know, this is the first study that generates byte-level results for malware analysis.

II. Preliminaries

1. Malware Byte Stream Analysis

Malicious code analysis methods include static analysis technology using signature patterns and

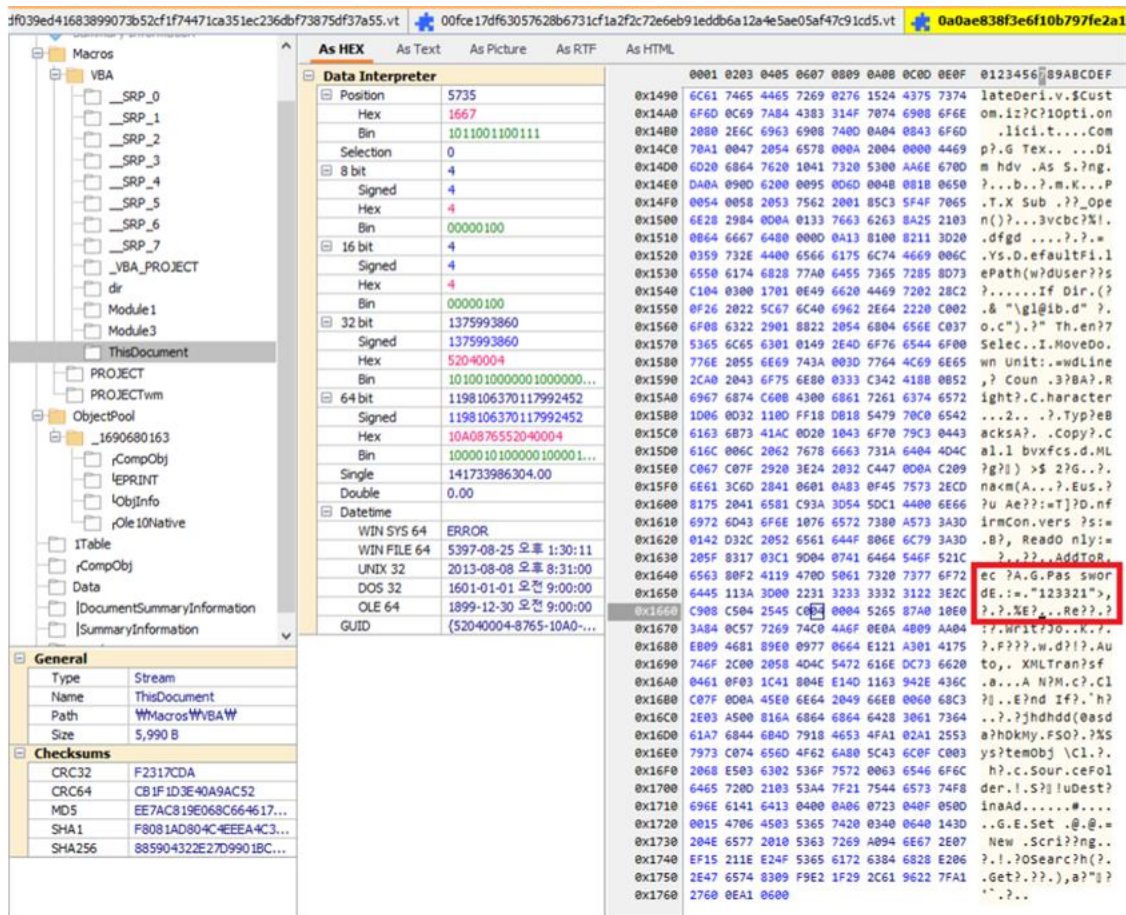


Fig. 2. Example of malware file analysis using structured storage viewer (SSV), where the left side shows a tree structure of byte streams and the right side is the stream contents in a specific format (e.g., HEX).

dynamic behavior analysis through sandbox execution. During the analysis process, domain experts often find malicious keywords via two steps. First, they analyze the file structure and identify malicious behavior by executing the file in a virtual machine. Second, they manually check the malicious code using tools such as SSV or HxD.

Fig. 2. depicts how to use the SSV to browse a malicious file. Domain experts browse streams by clicking a specific stream in the tree structure on the left side, and the tool will show the contents of the stream in HEX, Text, Picture, RTF, and HTML format on the right side. The experts manually check the raw bytes in line-by-line manner to find malicious code fragments; for example, in the HEX format of the 'ThisDocument' stream of VBA under Macros storage, we found the string 'Password = "123321"' as shown in the red box of Fig. 2. The HxD is another widely-used tool for malware

analysis. It is a HEX editor that can be used to edit raw bytes of large files, and it shows the raw bytes in hexadecimal and ASCII simultaneously.

Although existing tools (e.g., SSV, HxD) are widely used for malware analysis, it is still difficult for domain experts to analyze raw bytes of malware files. That is, the malicious code often uses obfuscation, dummy data, or NOP (no operation) sled to hide malicious actions, where the NOP sled is a series of NOP instructions in which the program slides the CPU's instruction execution flow to its desired destination. Such malicious actions are difficult to detect manually, so the experts may miss some of them during the analysis. In this paper, we propose a method that analyzes the given byte streams and generates byte-level labels that can assist the experts.

We believe our method will save time and cost of malware analysis and annotation process.

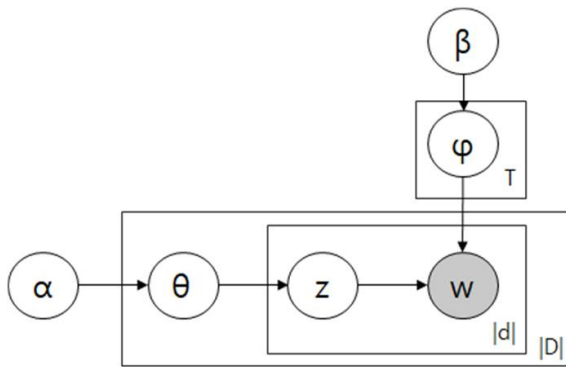


Fig. 3. Graphical representation of latent Dirichlet allocation (LDA).

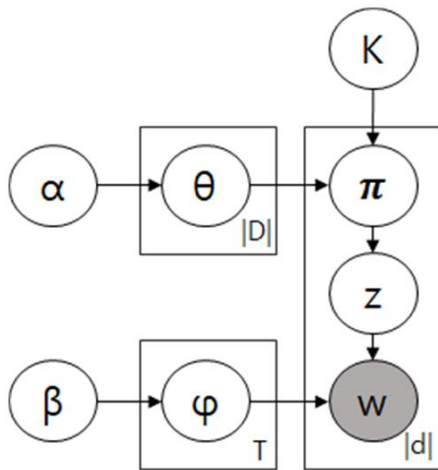


Fig. 4. Graphical representation of overlapped latent Dirichlet allocation (OLDA).

2. Malware Analysis Using Topic Models

Probabilistic topic model is a generative model that represents the underlying generation process of observations (e.g., words) with probability distributions. Fig. 3. is a graphical representation of latent Dirichlet allocation (LDA) model[9], Graphical representation of latent Dirichlet allocation (LDA), where α indicates a prior of per-document topic distribution θ on T topics, β is a prior of per-topic word distribution ϕ ; z is a per-word topic label of the word w , $|d|$ is the number of words in the document d , and $|D|$ is the number of documents in a corpus. The shaded node w means that the words are observable in the data, and other three parameters (e.g., θ , ϕ , z) are obtained via a training process. The α and β are hyper parameters. The LDA is the most well-known probabilistic topic model, that is designed on a

hypothesis that the word w (shaded node) within document d is generated from a word distribution β_t of the word's topic $z_w = t$, where the topic is drawn from a topic distribution θ_d of the document d . The key structure of LDA model is that there is a topic distribution for each document, and this might be totally different in other topic models. For example, author topic (AT) model[10], a variant of the LDA model, hypothesizes that an author has a topic distribution, and documents do not have topic distributions and are generated by one or more authors. Different topic models may discover different topics even from the same corpus, so it is necessary to carefully choose a topic model according to the data characteristics.

There have been studies that employed topic models for the malware analysis or malware detection. E. Medvet and F. Mercaldo[11] reviewed previous studies using the LDA model and k-means (KM) clustering algorithm for malware analysis in static manner. They showed that using the LDA model and KM clustering algorithm together brought performance improvements in malware family categorization. W. Stegner et al.[12] utilized the LDA model and context information (e.g., sensor data), and used k-nearest neighbor (kNN) algorithm for malware detection task. W. Hilal et al.[13] aimed at the malware detection task for executable files; they combined term-frequency features and topic distributions obtained from the LDA model, and employed several machine-learning models (e.g., logistic regression, support vector machine[14], extreme gradient boost[15]) as a classifier. The previous studies have shown a potential of the topic model in malware analysis, but there was no study that proposes a byte-level method of automatic malware analysis; the previous studies commonly analyzed opcodes within files, and mostly generated analysis results in file-level (e.g., a list of suspicious files).

In this paper, for byte-level malware analysis, we propose a new method based on a topic model. Most topic models work in bag-of-words (BoW) manner and

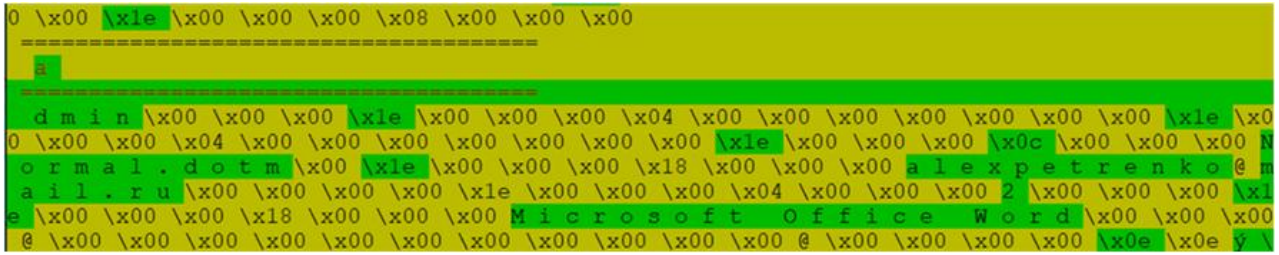


Fig. 5. Byte-level result of the OLDA model when the number of topics $T=2$, where each byte is represented as a hexadecimal (e.g., $\backslash\text{Wx01}$) or a character (e.g., 'a', 'd'), and suspicious bytes and benign (normal) bytes are highlighted with green and yellow colors, respectively.

do not incorporate spatial patterns or sequential patterns, so we exploit overlapped latent dirichlet allocation (OLDA) model. The OLDA model was originally designed for image segmentation, and it groups the image pixels (or patches) into different classes (e.g., person, car). It assumed that there are multiple documents in each image, and topic distribution of each pixel (or patch) is a mixture (i.e., average) of topic distributions of surrounding documents. Especially, the OLDA model allows the documents to have overlapped boundaries, so it better captures spatial patterns unlike other ordinary topic models. Fig. 4 depicts the graphical representation of the OLDA model, where α indicates a prior of per-document topic distribution θ on T topics, β is a prior of per-topic word distribution ϕ , π is a per-word topic distribution that is a mixture (i.e., average) of θ of K overlapped documents, z is a per-word topic label of the word w , $|d|$ is the number of words in the document d , and $|D|$ is the number of documents in a corpus. The shaded node w means that the words are observable in the data, and other three parameters (e.g., θ , ϕ , z) are obtained via a training process. The α , β , and K are hyper parameters. There were other topic models capable of dealing with spatial information; for example, Spatial LDA (SLDA)[16] used Gaussian kernel for modeling spatial patterns of given images. We chose the OLDA model in this paper as it requires a small computational load, so it will be able to quickly deliver analysis results to the domain experts. As the OLDA is originally designed for image analysis, we slightly revise the structure of the OLDA to adapt it for byte-level malware analysis.

III. Method

To aid in malware annotation or signature analysis by human experts, we propose a new method that provides byte maliciousness detection by employing the OLDA model. The OLDA model basically groups the given bytes into T clusters, so we expect that we get two clusters (i.e., malware and benign) when $T=2$, of course, T clusters may correspond to malware families when $T > 2$. In this paper, we set $T=2$ and Fig. 5 describes a sample result, where the two clusters are highlighted with green and yellow colors. Double-line of consecutive '=' symbol indicates that domain experts annotated the character 'a' (followed by 'dmin') between the lines as a suspicious byte that is beginning of a malicious action. Interestingly, some consecutive suspicious bytes, highlighted with yellow, form meaningful sequences such as 'admin', 'Normal.dotm', and 'Microsoft'. Domain experts manually analyzed this file, and annotated that the word 'admin' is the beginning of malicious action. It is worth noting that the OLDA model captures semantic meaning of the bytes. That is, it grasps the underlying patterns of clusters (e.g., malware and benign) and may give different byte-level labels even for the same values; for example, some hexadecimals of $\backslash\text{x1e}$ are yellow while others of the same value are green in Fig. 5. This implies that the OLDA model interprets the byte stream and generates byte-level labels based on their context (i.e., nearby bytes) and semantics (i.e., topic distributions).

The OLDA model was originally designed for image segmentation, so we slightly modify its key structure that document boundaries are overlapped. The original OLDA model assumed that there are multiple documents having topic distributions in each image. The 'overlap' of OLDA model implies that each pixel or patch belongs to K nearby documents, where the nearby documents are determined via a distance function (e.g., Euclidean distance). With a greater K , the OLDA model will consider more regions or spatial information within the image. Unlike the image data (i.g., a form of two-dimensional matrix), the byte stream is an one-dimensional sequence of bytes. The sequential patterns that the OLDA will learn from the byte streams heavily relies on how many documents are overlapped, so it may need greater value of K as the byte streams are one dimensional data. To control the 'overlap' mechanism in a convenient way, we redefined the K as a 'stride' that is the number of bytes between the beginnings of adjacent documents. Fig. 6 depicts how the OLDA model works on the byte streams within files. Assume that the document size (i.e., the number of bytes in a document) $|d|$ is 100. For the the first file in the figure, if the stride $K = 50$, then the first document boundary is $[0, 99]$ and the second document boundary will be $[50, 149]$ because $0 + K = 50$. Note that a byte stream may have multiple documents. For example, a byte stream s is 200 bytes in length (i.e., $|s| = 200$), then it will have three documents (e.g., $[0, 99]$, $[50, 149]$, and $[100, 199]$) if $K = 50$ and $|d| = 100$. The stride K should be $1 \leq K \leq |d|$ because $K > |d|$ will make the OLDA model to omit some bytes in the stream. If the stream length $|s|$ is not a multiple of K , then we take the last document of a boundary $[|s| - K, |s| - 1]$; for example, if $|s| = 120$, then the stream has two documents (e.g., $[0, 99]$ and $[20, 119]$). If the stream length is smaller than K , then the stream has only one document padded with $\backslashx00$. It is worth noting that smaller K makes the OLDA to better capture sequential patterns, and the biggest stride (i.e., $K = |d|$) will make the OLDA model to work like the LDA model that does not incorporate sequential patterns. The

formal generative process of OLDA with our modification is described as below.

1. For each topic t ,
 - (a) Draw a word distribution Φ_t from Dirichlet(β).
2. For each document d ,
 - (a) Draw a topic distribution θ_d from Dirichlet(α).
 - (b) For each n -th word w ,
 - (i) A mixture topic distribution π_{dn} is computed by averaging the topic distributions of overlapped documents with the stride K .
 - (ii) Choose a topic z_{dn} from Multinomial(π_{dn}).
 - (iii) Generate a word w from Multinomial($\Phi_{z_{dn}}$).

The stride K is a concept that is often used in convolution operations of convolutional neural networks (CNN)[17]. When the CNN is used for image analysis, its convolutional filters walk from the top-left corner to the bottom-right, where the stride is the step size. With smaller stride values, the CNN allows the convolutional filters learn from more 'overlapped' image regions, which is similar to the OLDA model mechanism: the OLDA model makes the document boundaries are 'overlapped', and smaller stride K will allow more documents to be overlapped. The approximate equation for θ of the document d is

$$\theta_{dt} = \frac{\alpha_t + C_{dt}}{\sum_{t'} \alpha_{t'} + C_{d t'}} \quad (1)$$

where α_t is the prior value for the topic t , T is the total number of topics, and C_{dt} indicates the frequency of the topic t in the document d . The document size $|d|$ (i.e., the number of bytes in a document) is given by users, and we assume that all documents have the same size. The mixture of topic distributions π_n of n -th byte is

$$\pi_{nt} = \frac{1}{O} \sum_{d \in O} \frac{\alpha_t + C_{dt}}{\sum_{t'} \alpha_{t'} + C_{d t'}} \quad (2)$$

where O is a set of overlapped documents of the n -th byte, and C_{dt} indicates the frequency of the topic t in the document d . The size of set O (i.e.,

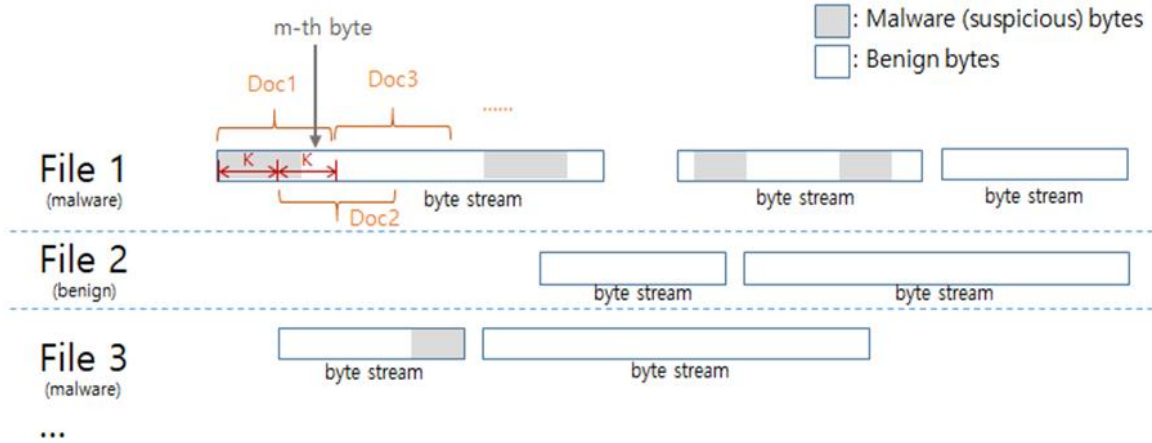


Fig. 6. Example of how the OLDA model works on the byte streams. The OLDA model assumes that there are multiple documents in a file, and allows boundaries of adjacent documents are overlapped.

the number of documents in the set) relies on the stride K . The approximate equation for the word w with the topic t is

$$\Phi_{tw} = \frac{\beta_w + C_{tw}}{\sum_{w'}^V \beta_{w'} + C_{d't'}} \quad (3)$$

where β_w is the prior value for the word w , V is the vocabulary size, and C_{tw} indicates the frequency of the word w with the topic t . As we treat the bytes as words, the vocabulary size is 256 because a byte is 8 bits (i.e., $2^8 = 256$).

Topic model parameters (e.g., θ, Φ of the LDA) are commonly intractable because of couplings between the variables, as explained in [9]. There are some approximation algorithms such as variational approximation, Laplace approximation, or Markov chain Monte Carlo (MCMC). In this paper, we utilize a collapsed Gibbs sampling [18] that is one of the MCMC algorithm. It works in iterative manner, and every step of the Markov chain draws the latent variable z_w of each word w , so that the parameters (e.g., θ, Φ) can be updated.

IV. Experiment

1. Experimental Setup

To demonstrate the usefulness of our method, we firstly trained the OLDA model on our annotated dataset. The trained OLDA model gives byte-level labels

Table 1. The number of data samples.

	Malware	Benign	Total
File	526	536	1,062
Stream	8,433	1,812	10,245

Is that indicate the byte is suspicious or not, so we check if the byte-level labels are consistent with annotation of domain experts by computing recall scores. Additionally, we visualize the obtained labels by highlighting them with different colors.

Our dataset has malware and benign files of Microsoft (MS) office, where the malware files were provided from a Korean anti-virus company, and the benign files were collected from public portal site of the Korean Ministry of the Interior and Safety. All streams of malware and benign files are extracted; compressed streams are firstly decompressed and thereafter extracted. Every stream is examined manually by domain experts, and a malware stream is annotated with an offset (i.e., an index) of a byte that is a beginning of malicious action. That is, the domain experts just found the beginning byte of obvious malicious actions, so other nearby suspicious bytes are not annotated. The reason of this is that it is difficult to determine exact boundary of malicious byte blocks because there can be various ways of malicious actions. The number of samples is described in Table 1, and the stream data is available online[19]. For experiments, we used a

machine of 256GB RAM and Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz 48 cores. We utilized Python3 language to implement our model.

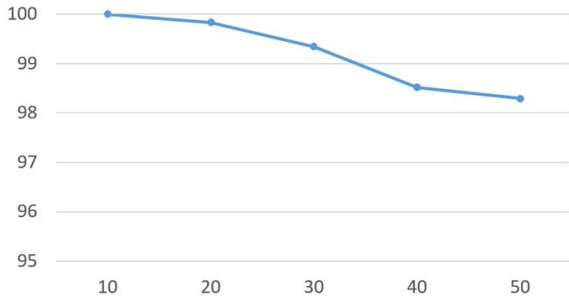


Fig. 7. Experimental results of malware offset detection given $d=100$, where the horizontal axis is the stride K , and the vertical axis is the recall in percent

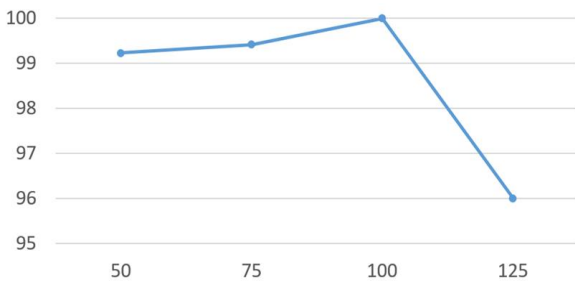


Fig. 8. Experimental results of malware offset detection given $K=19$, where the horizontal axis is the document size $|d|$, and the vertical axis is the recall in percent.

In all experiments, we set the number of topics $T=2$, so we expected that the OLDA model clusters the given bytes into two groups: malware and benign. The prior parameters α and β were 0.1 and 0.01, respectively. We assume that a document has 100 bytes (i.e., $|d|=100$), and varied the stride K from 10 to 50. The number of iterations was 75. All hyper-parameter settings were obtained via a grid-searching.

2. Result

Once the OLDA model is trained, the latent variable z of every byte is labelled with either of 0 (for benign) or 1 (for malware); we set $T=2$, therefore $z \in \{0,1\}$. We compared the labels with the offsets annotated by domain experts; we denote this task as malware offset detection in this paper. We firstly divided the set S of byte streams into two

sets: a malware stream set S_m and a benign stream set S_b . Let z_{sw} is a label of the word w (i.e., w -th byte) in the stream s , and o_s is an annotated offset of the stream s , where $s \in S_m$. For each stream s , we checked the value of z_{so_s} and computed the recall by

$$\frac{\sum_{s \in S_m} z_{so_s}}{|S_m|} \times 100 \quad (4)$$

where $z_{so_s} = 1$ if the corresponding byte is malware, otherwise 0. High recall indicates that the OLDA model discovers more suspicious bytes, which in turn helps human experts more effectively in malware analysis. The reason of computing only recall in this paper is related to the difficulty of annotation process of the dataset. When the expert annotators find malicious actions of arbitrary length (i.e., arbitrary number of bytes), it will be difficult to annotate per-byte labels because some bytes may be ambiguous in determining whether they are part of malware. Instead, only a single byte within the malicious byte sequence is labelled as malware, implying that the precision is meaningless.

As the initial value of the latent variable z is randomly chosen, the parameters (e.g., θ, Φ) will be different even if we use the same data for training. Therefore, we trained the OLDA model three times independently, and got averaged recalls from the three trained models. The results of malware offset detection are summarized in Fig. 7, where the document size $|d|=100$. Surprisingly, we got 100% recall when $K=10$; this means that our method has a potential to deliver trust-worthy analysis results. We also observed that the recall degrades as K increases. The $K > 10$ makes less documents to be overlapped, so it might hinder the OLDA model from learning sequential patterns.

With a fixed $K=10$, we varied the document size $|d|$ from 50 to 125, and the results are summarized in Fig. 8 where the recall values are obtained via the equation 4. The performance gap was not big between different $|d|$, but found that the best setting is $|d|=100$.

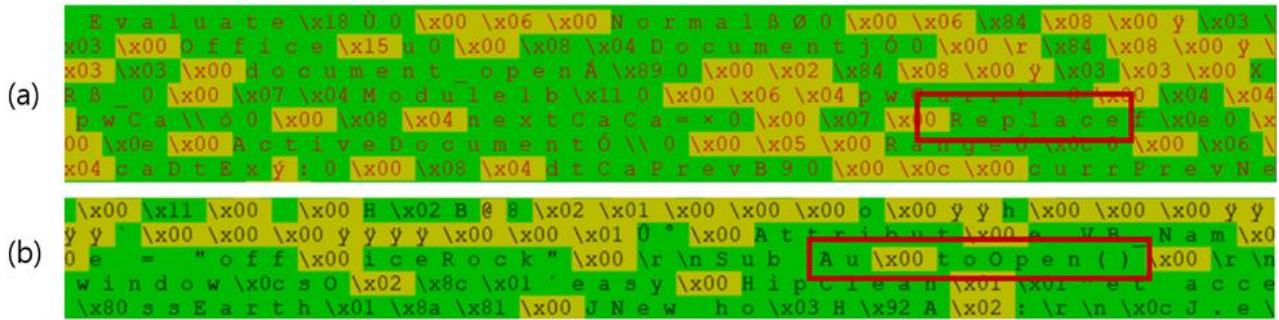


Fig. 9. Visualized sample results of byte-level labels, where malware (suspicious) and benign bytes are highlighted with green and yellow, respectively. (a) 'Replace' is often used to obfuscate and hide malicious code. (b) 'AutoOpen' might be used to execute other malicious programs.

We examined the byte-level results with the best setting (i.e., $K = 10$, $|d| = 100$). Fig. 9 shows visualized samples by highlighting the byte-level labels with two colors, where yellow and green are for benign and malware (suspicious), respectively. We manually check the byte-level results, and found two positive aspects that we experienced during the examination. First, we observed that the bytes highlighted with green (i.e., suspicious bytes) seem reasonable; for example, in (a) of Fig. 9, 'Replace' is often used to obfuscate and hide malicious actions, and 'AutoOpen' in (b) can be used to execute other malicious programs. This allows the domain experts to more focus on suspicious bytes, so it will make the analysis or annotation process more accurate. Second, the results are easy to examine in color, improving the efficiency of analysis; this will allow the analysis or annotation process to be faster. As data-driven approaches heavily rely on the amount of data and annotation quality, our method will eventually contribute to the data-driven approaches.

3. Limitation

In this paper, we provide recall scores of malware offset detection as shown in Fig. 7 and Fig. 8. We do not use other metrics such as precision and F1 scores because false negatives cause much worse outcome than false positives. Assume that we have an analysis tool that often fails to recognize malicious bytes, then nobody wants to use such tools as a single malicious byte

may cause severe damage. Of course, it is also necessary to reduce false positives, but we focus on the false negatives in this paper because our proposed method is not designed for fully-automatic detection of malicious bytes, but a tool that helps human experts. In future work, we will further examine to improve our method by reducing the false positives.

V. Conclusion

The malware annotation process requires much time and effort of domain experts. We propose a new method that uses a slightly modified version of overlapped latent dirichlet allocation (OLDA) to generate byte-level labels indicating whether each byte is suspicious or not. Given a set of byte streams with malware offsets annotated by domain experts, we demonstrated that the generated byte-level labels are consistent with the offsets. We furthermore examined cases by visualizing the byte-level labels with different colors. We believe that our method will greatly contribute to the malware annotation process, so eventually it will assist many related works (e.g., malware detection using data-driven models). With regard to that we are motivated to conduct more experiments with byte streams of other file formats (e.g., malware PDF), and also perform with greater values of T for three or more different malware families.

ACKNOWLEDGEMENT

Following are results of a study on the "Leaders in INdustry-university Cooperation 3.0" Project, supported by the Ministry of Education and National Research Foundation of Korea. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2020R111A3053015).

REFERENCES

- [1] Yousefi-Azar, M., Varadharajan, V., Hamey, L., and Chen, S. "Mutual Information and Feature Importance Gradient Boosting: automatic byte n-gram feature reranking for Android malware detection." *Software: Practice and Experience* Vol. 51. No. 7. pp. 1518-1539. 2021. DOI: 10.1002/spe.2971
- [2] Huang, Wenyi, and Jack W. Stokes. "MtNet: a multi-task neural network for dynamic malware classification." *Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain*, pp. 399-418. July 7-8, 2016, DOI: 10.1007/978-3-319-40667-1_20
- [3] Kumar, Rajesh, and S. Geetha. "Malware classification using XGboost-Gradient Boosted Decision Tree." *Advances in Science Technology and Engineering Systems Journal* Vol. 5. No. 5, pp. 536-549. May, 2020. DOI: 10.25046/aj050566
- [4] Jeong, Y. S., Woo, J., Lee, S., and Kang, A. R. 2020. "Malware detection of hangul word processor files using spatial pyramid average pooling." *MDPI*. Vol. 20. No. 18. 5265. Sep, 2020. DOI: 10.3390/s20185265
- [5] Yan, Jinpei, Yong Qi, and Qifan Rao. "Detecting malware with an ensemble method based on deep neural network." *Security and Communication Networks*. Vol. 2018. March, 2018, DOI: 10.1155/2018/7247095
- [6] Jeong, Young-Seob, Jiyoung Woo, and Ah Reum Kang. "Malware detection on byte streams of hangul word processor files." *Applied Sciences* Vol. 9. No. 23. Nov, 2019, DOI: 10.3390/app9235178
- [7] Gandotra, Ekta, Divya Bansal, and Sanjeev Sofat. "Malware analysis and classification: A survey." *Journal of Information Security* 2014, Vol. 5, No. 2, 44440, March, 2014. DOI: 10.4236/jis.2014.52006
- [8] Jeong, Young-Seob, and Ho-Jin Choi. "Overlapped latent Dirichlet allocation for efficient image segmentation." *Soft Computing* 19 Vol. 19, pp 829-838. Aug, 2014, DOI: 10.1007/s00500-014-1410-x
- [9] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal of machine Learning research* 3. pp. 993-1022. Jan, 2003.
- [10] Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. "The author-topic model for authors and documents." *arXiv preprint*. 2012. DOI: 10.48550/arXiv.1207.4169
- [11] Medvet, Eric, and Francesco Mercaldo. "Exploring the usage of topic modeling for android malware static analysis." *2016 11th International Conference on Availability, Reliability and Security (ARES)*. IEEE, 2016 pp. 609-617. DOI:10.1109/ARES.2016.10
- [12] Stegner, W., Kapp, D., Kebede, T., and Jha, R. "Context-Aware Malware Detection Using Topic Modeling." In *NAECON 2021-IEEE National Aerospace and Electronics Conference*, IEEE. pp. 326-331. Aug. 2021. DOI: 10.1109/NAECON49338.2021.9696385
- [13] Hilal, W., Wilkinson, C., Alsadi, N., Surucu, O., Giuliano, A., Gadsden, S. A., and Yawney, J. "A topic modeling-based approach to executable file malware detection." In *Disruptive Technologies in Information Sciences VI*. SPIE. Vol. 12117, pp. 76-85. May, 2022. DOI: 10.1117/12.2619033
- [14] Burges, Christopher JC. "A tutorial on support vector machines for pattern recognition." *Data mining and knowledge discovery 2.2* Vol. 2. pp. 121-167. 1998. DOI: 10.1023/A:1009715923555
- [15] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. pp. 785-794. Aug. 2016. DOI: 10.1145/2939672/2939785
- [16] Wang, Xiaogang, and Eric Grimson. "Spatial latent dirichlet allocation." *Proceedings of the 20th International Conference on Neural Information Processing Systems NIPS'07*. pp. 1577-1584. Vancouver, British Columbia, Canada. Dec. 2007.
- [17] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, Vol. 86. No. 11. pp. 2278-2324. 1998. DOI: 10.1109/5.726791
- [18] Griffiths, T., Steyvers, M., Blei, D., and Tenenbaum, J. "Integrating topics and syntax." *17th International Conference on Neural Information Processing Systems*. pp. 537-544. Vancouver, British Columbia, Canada. Dec. 2004.
- [19] Ah Reum Kang, Dataset. <https://sites.google.com/arkang.net/malwarebytestreams>.

Authors



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 pharmaceuticals.



Yeong-Jin Kim received the B.S degree in cyber security from Pai Chai University, Daejeon, South Korea, in 2023. He currently pursuing M.S degree in Information Security at Pai Chai University.

He study at Pai Chai University, Daejeon, South Korea. His research interests artificial intelligence, malware, and cyber security



Medard Edmund Mswahili received his M. Sc. Eng in Big Data Engineering, ICT convergence department from Soonchunhyang University, Asan, South Korea in 2022. He's currently pursuing his Ph.D. as a research

assistant at Data Analysis and Artificial Intelligence Lab in the Computer Engineering department at Chungbuk National University, Cheong-ju city, South Korea. His research interest mainly focuses in Machine & Deep Learning in Pharmaceutical data analysis (drug discovery & development) and natural language processing.



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 business intelligence.



Ah Reum Kang received the M.S. and Ph.D. degrees in information security from Korea University, South Korea, in 2012 and 2016. She is a professor in the Department of Information Security at Pai Chai University

in Daejeon, South Korea. Her current research interests include security, artificial intelligence, malware, medical data analysis, and online game security.