International Journal of Advanced Smart Convergence Vol.12 No.1 149-156 (2023)
http://dx.doi.org/10.7236/IJASC.2023.12.1.149

IJASC 23-1-18

Analysis and Detection of Malicious Data Hidden in Slack Space on OOXML-based Corrupted MS-Office Digital Files

Sangwon Na[1], Hyung-Woo Lee[2]

[1]*Bachelor student, Div. of Computer Engineering, Hanshin University, Korea*
[2]*Professor, Div. of Computer Engineering, Hanshin University, Korea*

E-mail: [1]yournsw@naver.com, [2]hwlee@hs.ac.kr

Abstract

OOXML-based MS-Office digital files are extensively utilized by businesses and organizations worldwide. However, OOXML-based MS-Office digital files are vulnerable to forgery and corruption attack by including hidden suspicious information, which can lead to activating malware or shell code being hidden in the file. Such malicious code can cause a computer system to malfunction or become infected with ransomware. To prevent such attacks, it is necessary to analyze and detect the corruption of OOXML-based MS-Office files. In this paper, we examine the weaknesses of the existing OOXML-based MS-Office file structure and analyzes how concealment and forgery are performed on MS-Office digital files. As a result, we propose a system to detect hidden data effectively and proactively respond to ransomware attacks exploiting MS-Office security vulnerabilities. Proposed system is designed to provide reliable and efficient detection of hidden data in OOXML-based MS-Office files, which can help organizations protect against potential security threats.

Keywords: *Corrupted Digital File, OOXML-based MS-Office, Slack Space Detection, Digital Forensics, Ransomware.*

1. Introduction

To determine if a digital file stored on a computer system has been modified, a cryptographic hash function like SHA256 is commonly used to generate one-way hash values from the original file and the modified version, respectively. By comparing these values, it is possible to determine whether the original digital file has been forged or altered. However, if there is no original file available for comparison, it becomes difficult to determine whether the corresponding digital file has been corrupted. In such cases, checking whether a file is damaged or not may be a more appropriate approach. The internal structure of a digital file may be changed in an illegal and malicious form, or used to hide malicious code within the file[1,2,3]. With the increasing use of MS-Office files, security vulnerabilities that exploit the internal structural problems of MS Office series are being discovered. In particular, malicious ransomware is now being distributed through corrupted MS-Office files with malicious codes hidden in image files such as JPG/PNG/BMP. Therefore, it is necessary to develop

Manuscript Received: February. 8, 2023 / Revised: February. 14, 2023 / Accepted: February. 18, 2023
Corresponding Author: hwlee@hs.ac.kr
Tel: +82-31-379-0642
Professor, Divison of Computer Engineering, Hanshin University, Korea

a mechanism to efficiently determine whether a file is damaged or maliciously forged or altered by analyzing the abnormality of the internal structure of MS-Office files.

Existing digital forensic tools may not be able to detect additional digital information or malicious code stored in a digital file that exploits the vulnerability of the digital file storage method and internal structure. MS-Office files are particularly vulnerable to this type of attack, as they are stored in the form of a ZIP compressed file based on the OOXML format, and various elements in the document are managed in the form of XML[4,5,6]. While the OOXML format provides efficiency in the saving process, it can also be exploited to hide malicious scripts in MS-Office files. Through the OOXML internal structure analysis of MS-Office digital files, it is possible to detect malicious scripts, check for malicious forgery and file damage, and identify abnormal MS-Office files. This process can also detect malicious files hidden inside the digital files. Therefore, a mechanism that can efficiently determine whether a file is damaged or maliciously forged or altered by analyzing the abnormality of the internal structure of MS-Office files is proposed in this study.

As a result, the proposed mechanism can be used to determine whether a digital file is damaged or has been maliciously forged or altered. By verifying and determining whether the files are damaged through internal structure analysis of MS Office files, it is possible to check for malicious scripts, detect malicious files hidden inside, and identify abnormal MS-Office files. In the digital forensic process, the proposed mechanism can be useful for efficiently detecting and discriminating if a file is damaged or corrupted by exploiting the vulnerability of the file's internal structure to hide a specific message.

2. OOXML based MS-Office Series Digital File Structure

Since MS Word 2007, MS-Office files have been converted to an OOXML-based format (Office Open XML Format), which has been recognized as an international standard under ECMA-376 and ISO/IEC29500. The OOXML Architecture utilizes XML to represent the contents of various documents and their relationships, and is stored in a ZIP format file structure. Specifically, MS-Office files are stored in ZIP format based on OOXML within the MS-Office series, and information about each directory inside MS-Office is stored in the CDH (Central Directory Header) section, while each file is structured through the LFH (Local File Header). By changing the extension of a DOCX document to ZIP and unzipping it, we can view the structure and relationships, such as the MS-Office document expressed in an XML document contained in the '_rels' folder, the XML document stored in the 'docProps' folder, and the document located in the 'word /_rels' folder. The structure and format are defined, and various files, such as document.xml[7], are used to mark and save information about each page in a digital document as shown in Fig. 1.

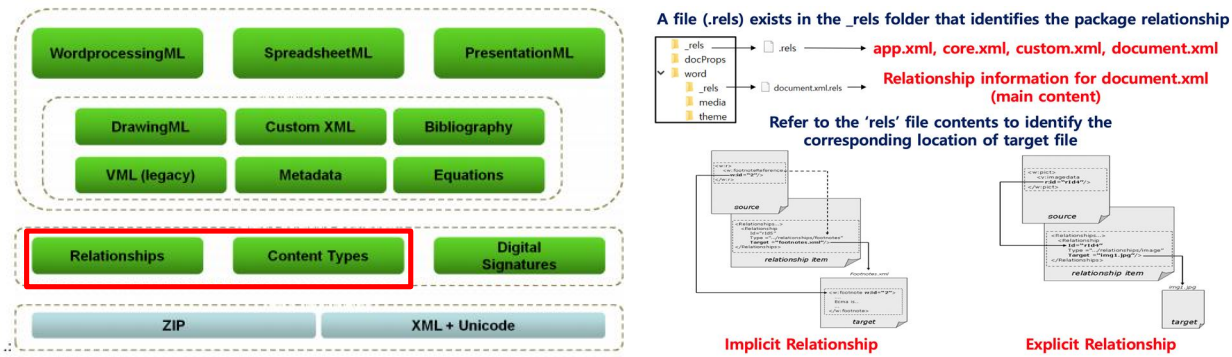

Figure 1. OOXML based MS-Office Series Digital File Structure

3. Hidden Malicious Data on OOXML-based MS-Office Digital File

3.1 Hiding Data on Slack Space of OOXML based MS-Office File

OOXML-based MS-Office files are saved in the ZIP format and their location is specified in the ECD (End of Central Directory) record field located at the end of the ZIP file. Embedded XML and image files within the MS-Word document are defined by the LFH (Local File Header), and each LFH header is referenced by its corresponding Central Directory Header (CDH). While regular ZIP files are created and stored without any internal slack space, multiple slack spaces may be present in each CDH and LFH when viewing the storage bitmap of OOXML-based MS-Office files. This activates an issue where the MS-Office Editor may not generate an error even if data is hidden in the slack space or if shell code is stored inside the MS-Office document as shown in Fig, 2. In this study, we analyzed damaged or corrupted OOXML-based MS-Office digital files and examined their abnormal file structures to solve this problem.

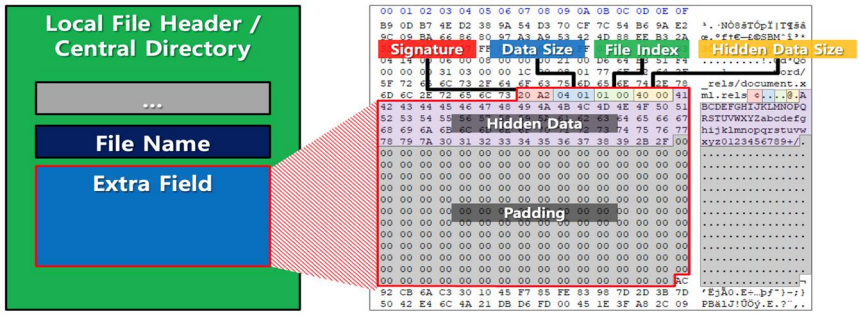

Figure 2. Data Hiding on OOXML based MS-Office Series Digital Files

3.2 Data Hiding Methods on OOXML based MS-Office File

We have described two methods for illegally hiding data by corrupting MS-Office files using an extra field (slack space) or the file comment field. Data Hiding is possible by using the slack space such as the extra field in LFH and CDH. In the ZIP header standard, it is defined so that the Extra Field part can be included in the CDH and LFH parts in consideration of extensibility. In most cases, this area is not being used for any specific purpose. However, if a malicious attacker could save data in the Extra Field, it would not be directly visible on the MS-Office editor screen. In addition, it can be abused by manipulating the Extra Field Length part, which is the size information of the Extra Field, to create a slack space, and to hide a file or data of an arbitrary length in an MS-Office file as illustrated in Fig. 3.

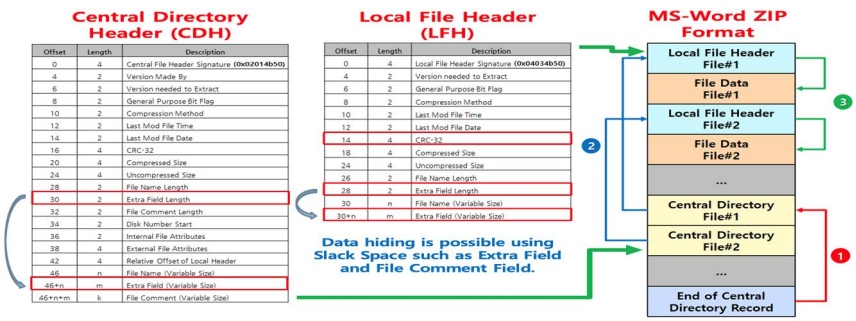

Figure 3. Data Hiding on OOXML based Digital File

As a result of hiding data in the Extra Field of the LF header for the actual MS-Office file, it was confirmed

that data can be saved in the normally allocated Extra Field part by overwriting method. Also, arbitrary data could be hidden by overwriting the Extra Field without changing the overall size of the LF header. It has been confirmed that data can be inserted into the Extra Field area even inside the CD header. In particular, in the case of the CD header, since MS-Office files are composed of several directories in the form of OOXML, it was confirmed that it is also possible to divide and store files to be hidden in multiple CDHs because there are multiple CDHs. Therefore, it was found that data can be hidden by inserting it into the Extra Field when it is divided into multiple CDHs and hidden by the partitioning method. In addition, malicious damage to MS-Office can be performed by adjusting the number of CDHs and LFHs, deleting specific CDHs, or adding LFHs.

3.3 Comparison of Data Hiding Methods on OOXML based MS-Office File

Table 1 below provides a comparison of two data hiding methods used in MS-Office files. In the first method, suspicious data can be hidden in the slack space of all Central Directory Headers (CDHs), which are of a fixed size and allow fragmented data to be concealed in CDHs. This means that a file larger than 65,535 bytes can be hidden in OOXML-based MS-Office files. Additionally, the file comment field in the CDH can also be used in the same way as the extra field. This method also increases the size of the MS-Office file, which requires adjusting the offsets in all headers. Since only one End of Central Directory (ECD) record exists in an MS Word file, only one hidden file comment can be used. Therefore, it will limit the amount of data that can be hidden to a maximum of 65,535 bytes. And the file comment field in the ECD record is located at the end of the MS-Office file structure.

Table 1. Comparison of Corruption Methods for OOXML-based MS-Office Files

MS-Office File Corruption Method	*Description*	*Size of Hidden Data*	*Detection*
Corrupting File Header	The file header is altered to prevent the file from opening.	*N/A*	*Obvious*
Hidden data in LFH extra field	*Suspected data is hidden in the slack space of each Local File Headers (LFHs).*	*m (m<65,535) bytes*	*Not Obvious*
Hidden data in CDH extra field	*Suspected data is hidden in the slack space of all Central Directory Headers (CDHs), allowing for fragmented data to be concealed in CDHs.*	*65,535 bytes*	*Not Obvious*
Hidden data in CDH comment field	*Data is hidden in the file comment field of the CDH, which is used in the same way as the extra field. A NULL must be included at the beginning of the hidden data to prevent it from being displayed in MS-Office software.*	*65,535 bytes*	*Not Obvious*
Hidden data in ECD comment field	*Data is hidden in the file comment field of the End of Central Directory (ECD) record. Only one hidden file comment can be used, limiting the amount of data that can be hidden.*	*65,535 bytes*	*Not Obvious*
Encrypted data	*The data within the file is encrypted, making it unreadable without a decryption key.*	*N/A*	*Not Obvious*

To summarize the process of hiding abnormal data by analyzing the internal structure of the MS-Office file, it involves (1) directly modifying or changing various files included in the MS-Office file, such as XML or multimedia objects, (2) concealing arbitrary messages by using the Extra Field part in the CD header and LF header, which are MS-Office storage methods, and (3) analyzing the internal structure of the MS-Office file to determine if the file has been damaged or maliciously tampered with. In these cases, the MS-Office editor recognizes those forged file as a normal file and opens it without a specific error expression process, so it is essential to take steps to identify potential abnormalities in the internal structure of the MS-Office file.

4. Hidden Malicious Data Analysis and Detection on Corrupted MS-Office Files

4.1 Slack Space, Hidden Data and Corrupted CDH Detection

The MS-Office File Analyzer is a tool designed to detect abnormalities and corruption in MS-Office files. It features a user-friendly interface implemented in Python, and can be used without requiring any additional modules to be installed. When given an MS-Office file for inspection, the proposed analyzer automatically checks the internal structure of the OOXML-based ZIP file format to determine whether the file is damaged or not. And it also includes a feature to automatically detect hidden data as shown in Fig. 4.

To achieve this, the analyzer analyzes the internal structure of the MS-Office file saved in the OOXML-based ZIP format, carefully examining the LFH and CDH header contents to efficiently detect any suspicious hidden data. The implemented system consists of four modules: a ZIP file identification module, a ZIP file structure analysis module, a ZIP file validation module, and a GUI module. With these modules, the MS-Office File Analyzer can effectively detect and report any abnormalities or hidden data present in an input MS-Office digital file.

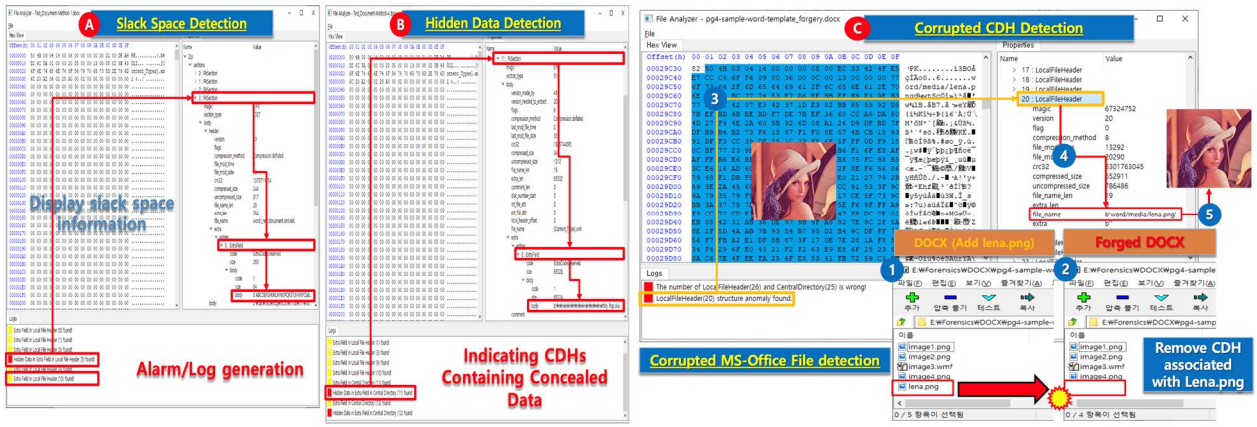

Figure 4. Slack Space, Hidden Data and Corrupted CDH Detection Results

We discovered that slack areas in OOXML-based MS-Office files only exist in the Extra Field part of the Local File Header (LFH) and Central File Header (CFH). Attackers can exploit the slack area to hide malicious code or additional information in the file without it appearing in the local file header of a typical ZIP file. This means that if an attacker exploits the slack area, they can hide malicious code or additional information in the file without it appearing in the LFH of a typical MS-Office file. To resolve this issue, we developed a function to automatically detect five types of damaged files as shown in Fig. 5.

Proposed software includes an automatic search function for slack space. When executed, it performs an OOXML structure analysis on the input MS-Office files to check whether slack space exists in the CDH and LFH headers. If hidden data is found, an alarm is displayed to notify the user. We also discovered that attackers can add a specific length of data by modifying the internal structure of an OOXML-based MS-Office file. To address this issue, we developed a module that can automatically detect any specific data that has been hidden

in an MS-Office file after changing its size. By executing proposed software, users can prevent damage from encryption or infection of digital files by malicious ransomware code that may be hidden in a file disguised as a legitimate digital file. This provides a simple solution for general users to protect their files without requiring specialized knowledge or training.

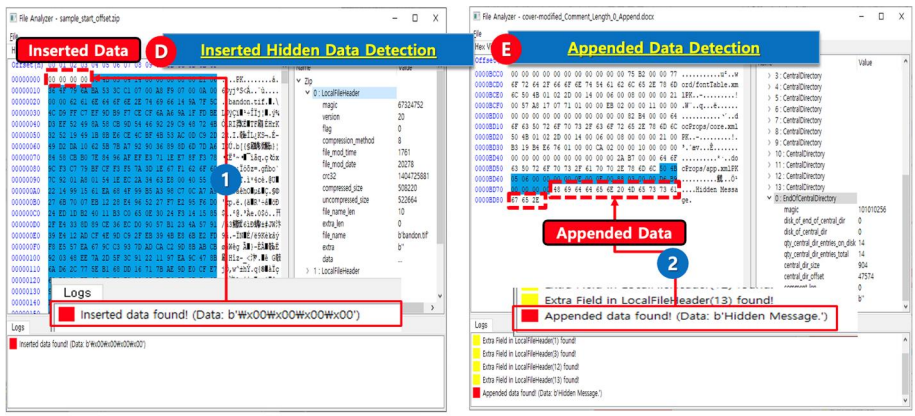

Figure 5. Inserted and Appended Hidden Data Detection Results

4.2 Implementation and Analysis Results

The proposed solution was implemented using the Python language, with the PyQT5 package used to provide a graphical user interface. The tool is designed to analyze OOXML-based MS-Office files that have been intentionally altered with malicious intent or are suspected to contain malicious code. Upon selecting a file, the tool conducts an analysis, and the result is presented in a JSON file format. An example of this is shown in Fig. 6 below. The implementation results indicate the practicality of the proposed solution and its potential to enhance the security of digital documents.

To evaluate the effectiveness of the tool, we conducted an experimental test using ten malicious MS-Office files obtained from the publicly available repository "Malicious File DB" provided by https://bazaar.abuse.ch/. The analysis showed that the tool accurately detected 100% of the manipulated malicious files in terms of their internal structure and slack space. These results demonstrate the reliability and robustness of the proposed solution in detecting concealed ransomware code hidden in OOXML-based MS-Office files. However, further testing and evaluation are needed to ensure the effectiveness and efficiency of the tool in real-world scenarios. Continued research and development in this area are necessary to enhance the security of digital documents and protect users from cyber-attacks.

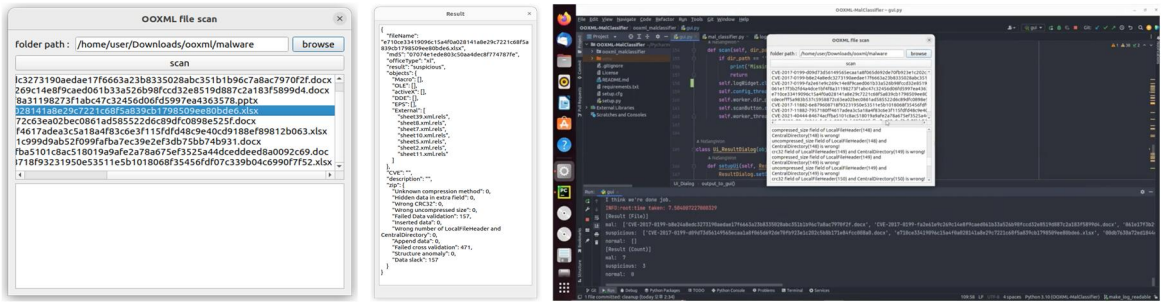

Figure 6. Implementation and Analysis Results for MS-Office Digital File

5. Conclusions

In this paper, we analyzed the structure of OOXML-based MS-Office files, identifying how attackers can exploit slack space to conceal malicious data and perform illegal corruption processes. We also proposed an advanced mechanism for detecting hidden data in MS-Office files, even when the CDH header has been intentionally deleted. To address these security risks, we designed and implemented a software tool for detecting corruption attacks on OOXML-based MS-Office files. As a result, we designed and implemented a software tool that can effectively detect corruption attacks on OOXML-based MS-Office files, enabling the development of advanced systems or mechanisms for automatically detecting concealed ransomware code hidden in these files.

The proposed mechanism and software represent significant contributions to the field of digital security and have the potential to enhance the security of public MS-Office digital document. By providing a practical solution for detecting them, we can enhance the security of public MS-Office digital documents and better protect users from malicious attacks. However, further research and development are needed to improve the robustness and efficiency of the proposed solution. This could involve exploring alternative approaches for detecting hidden data or developing more sophisticated algorithms for analyzing the file structure. Additionally, it may be necessary to evaluate the proposed solution in different scenarios and against more advanced attack techniques to ensure its effectiveness and reliability.

Acknowledgment

This work is an extended version of tha paper published at the IJCC2023 conference [7]. And this work is partially supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) (No. 2021R1F1A1046954).

References

[1] Anghel, Catalin. “Digital Forensics – A Literature Review.” EEA Journal of Information and Communication Technologies, vol. 2019, no. 1, 2019, pp. 23-27. DOI: https://doi.org/10.35219/eeaci.2019.1.05.

[2] Hassannataj Joloudari, J., Haderbadi, M., Mashmool, A., GhasemiGol, M., Shahab, S., and Mosavi, A. “Early Detection of the Advanced Persistent Threat Attack Using Performance Analysis of Deep Learning.” arXiv e-prints, 2020.

[3] Alenezi, A., Atlam, H., Alsagri, R., Alassafi, M., and Wills, G. “IoT Forensics: A State-of-the-Art Review, Challenges and Future Directions.” Proceedings of the 4th International Conference on Complexity, Future Information Systems and Risk (COMPLEXIS 2019), 2019, pp. 106-115.

[4] "Office Open XML." Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc. 11 September 2019. Web. 20 January 2023.

[5] "Zip (file format)." Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc. 24 September 2019. Web. 20 January 2023.

[6] "Document File Formats: Microsoft Word Document (DOCX/DOC)." LEAD Technologies, Inc. Web. 20 January 2023.

[7] Na, S., and Lee, H.W. “Implementation of Malicious Data Analysis and Detection System Hidden in the Slack Space of Corrupted OOXML-based MS-Office Digital Files”, Advanced and Applied Convergence Letters AACL 21 (9th International Joint Conference on Convergence, IJCC2023), pp.97-103, 2023.