ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 10, Issue 9, September 2023

AI based Hidden Data Detection on OOXML based MS-Office File

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

[1] Div. of Com. Eng., Hanshin University, Rep. of Korea
[2] School of Computing and Artificial Intelligence, Hanshin University, Rep. of Korea
Corresponding Author Email: [1] yournsw@naver.com, [2] hwlee@hs.ac.kr

Abstract*— *In case of forgery by including hidden information in MS-Office digital document, the computer system may be malfunctioned by malware or shell code hidden in the digital document. If a malicious code is hidden in the official legal documents by corrupting the OOXML-based structural of MS-Office serial digital files, it may be possible to be infected by ransomware hidden inside of MS-Office files. Therefore, it is necessary to analyze corruption of OOXML-based MS-Office files. In this paper, we examine the weaknesses of the existing OOXML-based MS-Office file structure, and analyze how concealment and forgery are performed on MS-Office digital documents. We designed and implemented an ML based hidden data detection method for proactively responding ransomware attack on exploiting MS-Office security vulnerabilities.

***Index Terms*—OOXML, ML, Hidden data detection, Security.**

I. INTRODUCTION

In order to determine whether a digital file stored in a computer system has been forged or altered, a cryptographic hash function such as SHA256 is generally used to generate one-way hash values from the original file and the forged modified version file, respectively, and compare them to determine whether or not the original digital file has been forged or altered. There is a method (passive/passive forgery) to check whether or not it exists. However, if there is no original file to be compared, it is difficult to determine whether the corresponding digital file has been forged or altered. If it is strictly defined to check whether a file has been forged or altered, it can be said to check whether a file is damaged or not. Therefore, whether or not the digital file is damaged can be determined through the case where the internal structure of the digital file has been changed in an illegal and malicious form, or if the internal structure of the file has been changed to hide malicious code within the file[1,2,3]. Since the use of MS-Office files is rapidly increasing recently, it is necessary to select them as analysis targets, and security vulnerabilities such as exploiting internal structural problems of MS Office series are continuously being discovered. In particular, recently discovered malicious ransomware is propagated/infected through MS-Office files, and is distributed in such a way that malicious codes are hidden in image files such as JPG/PNG/BMP included in MS-Office files. There is a mechanism that can determine malicious internal structure changes and abnormal internal structures of digital files by verifying and determining whether the files are damaged through internal structure analysis of MS Office files such as MS-Office. Therefore, in this study, a mechanism that can efficiently determine whether a file is damaged or maliciously forged or altered is studied by analyzing the abnormality of the internal structure of MS Office files.

If additional digital information or malicious code is stored and hidden in a digital file by exploiting the vulnerability of the digital file storage method and internal structure, it is often not properly detected by existing digital forensic tools. This is because it is possible to insert or conceal malicious codes in the case of MS Office files by using structural vulnerabilities of the internal storage format. MS-Office is stored in the form of a ZIP compressed file based on the OOXML format, and designates, stores, and manages various elements in the document in the form of XML[4,5,6]. Although the OOXML format provides efficiency in the saving process, there are security vulnerabilities due to the structural nature of ZIP/XML, so if exploited, malicious scripts, etc. can be hidden in MS-Office files. Therefore, the information hidden in the MS-Office file through this intentional digital file corruption process is opened normally without errors being found even by the MS-Office editor. Therefore, through the OOXML internal structure verification process for MS-Office digital files, it is possible to check whether malicious scripts are included, check for malicious forgery and file damage, and identify abnormal MS-Office files. It can also detect malicious files hidden inside. In the digital forensic process, if a file is damaged or corrupted by exploiting the vulnerability of the file's internal structure to hide a specific message, a technology that can efficiently detect and discriminate must be developed. Therefore, we propose a method for hidden data detection method using 8 type of Machine Learning model to actively respond to suspicious ransomware disguising legitimate official OOXML-based MS-Office digital files.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 10, Issue 9, September 2023

II. OOXML BASED MS-OFFICE FILE

A. MS-Office File Structure

Submit your manuscript electronically for review. Since MS Word 2007, MS-Office files have been revised into an OOXML-based format (Office Open XML Format), and have been approved as XML-based international standards ECMA-376 and ISO/IEC29500. In particular, recently discovered malicious ransomware is propagated/infected through MS-Office files, and is distributed in such a way that malicious codes are hidden in image files such as JPG/PNG/BMP included in MS-Office files. to be. Therefore, there is a mechanism that can determine malicious internal structure changes and abnormal internal structures of digital files.

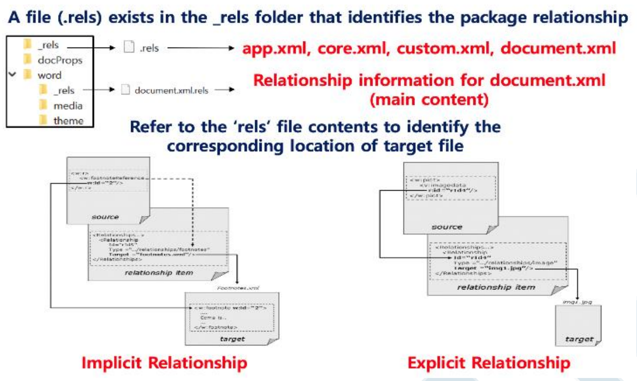

A file (.rels) exists in the _rels folder that identifies the package relationship
_rels
docProps
word
_rels
media
theme
.rels
app.xml, core.xml, custom.xml, document.xml
document.xml.rels
Relationship information for document.xml (main content)
Refer to the 'rels' file contents to identify the corresponding location of target file
source
relationship item
target
source
relationship item
target
Implicit Relationship
Explicit Relationship

Fig. 1 MS-Office File Structure

B. ML-based Hidden data Detection

MS-Office is stored in the form of a ZIP compressed file based on the OOXML format, and designates, stores, and manages various elements in the document in the form of XML[1]. Although the OOXML format provides efficiency in the saving process, there are security vulnerabilities due to the structural nature of ZIP/XML, malicious scripts can be hidden in MS-Office files. Therefore, through the OOXML internal structure verification process for MS-Office digital files, it is possible to check whether malicious scripts are included, and identify abnormal MS-Office files. Therefore, we proposed and implemented an AI-based malicious hidden data detection system.

III. DATA HIDING ON OOXML-BASED MS-OFFICE FILE

OOXML-based MS-Office files are stored in ZIP format, and are referenced from the ECD(End of Central Directory) record field at the end of the ZIP file. And embedded XML and image files in the MS-Word document are defined in LFH (Local File Header) and each LFH header is referenced by each Central Directory Header (CDH) respectively. General ZIP files are created and stored uniformly without any internal slack space. However, several slack spaces can be found in the each of CDH and LFH respectively when displaying the storage bitmap of OOXML based MS-Office files. Therefore, there is a problem that MS-Office Editor does not generate an error even if the data is concealed in the slack space or the shell code is stored inside the MS-Office document. Therefore, in this study, damaged or corrupted OOXML-based MS Office series digital files was analyzed, and abnormal file structures were analyzed and verified[7].

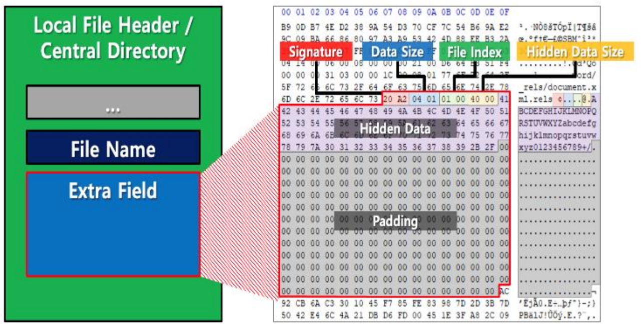
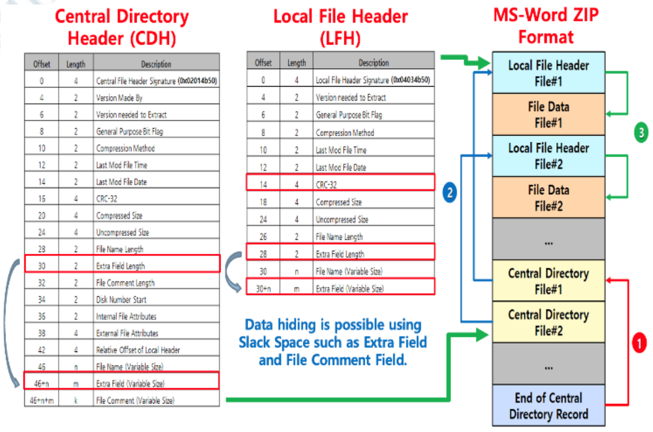


Fig. 2 Data Hiding on MS-Office File

We have described two ways to corrupt for hiding data by using extra field (slack space) or file comment field on inside of MS-Office files illegally. 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, it is confirmed that the area is not being used for any particular purpose. However, if a malicious attacker saves data in the Extra Field, it is not directly displayed 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.



Central Directory Header (CDH)

Offset	Length	Description
0	4	Central File Header Signature (0x02014b50)
4	2	Version Made By
6	2	Version needed to Extract
8	2	General Purpose Bit Flag
10	2	Compression Method
12	2	Last Mod File Time
14	2	Last Mod File Date
16	4	CRC-32
20	4	Compressed Size
24	4	Uncompressed Size
28	2	File Name Length
30	2	Extra Field Length
32	2	File Comment Length
34	2	Disk Number Start
36	2	Internal File Attributes
38	4	External File Attributes
42	4	Relative Offset of Local Header
46	n	File Name (Variable Size)
46+n	m	Extra Field (Variable Size)
46+n+m	k	File Comment (Variable Size)

Local File Header (LFH)

Offset	Length	Description
0	4	Local File Header Signature (0x04034b50)
4	2	Version needed to Extract
6	2	General Purpose Bit Flag
8	2	Compression Method
10	2	Last Mod File Time
12	2	Last Mod File Date
14	4	CRC-32
18	4	Compressed Size
24	4	Uncompressed Size
26	2	File Name Length
28	2	Extra Field Length
30	n	File Name (Variable Size)
30+n	m	Extra Field (Variable Size)

Fig. 3 Data Hiding on OOXML-based MS-Office 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.

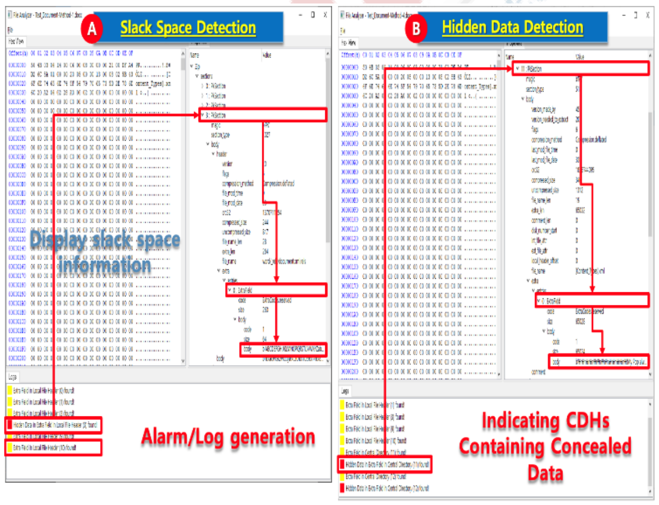
In particular, in the case of the CD header, since MS-Office files are composed of several directories in the form of OOXML, it was confirmed that it is also possible to divide and store files to be hidden in multiple CDHs because there are multiple CDHs. Therefore, it was found that data can be hidden 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.

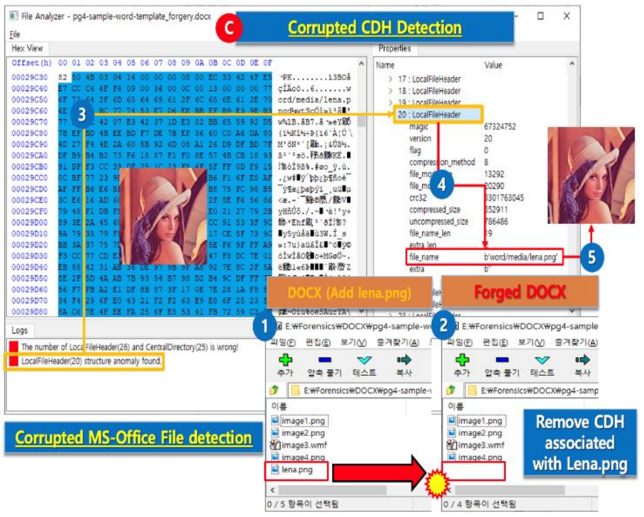
IV. AI-BASED HIDDEN DATA DETECTION ON MS-OFFICE FILES

A. Hidden Data Detection

The MS-Office File Analyzer was designed and implemented to check the abnormality of the MS-Office file and to automatically determine its corruption by using ML modules. It is implemented in a GUI form using the Python language, and implemented to operate without a separate additional module installation process. In the case of the SW developed in this study, if we input a MS-Office file to be inspected, it performs an analysis process on the internal structure of OOXML to automatically detect whether the file is damaged or not and provides a function to detect hidden data.it analyzes the internal structure of MS-Office file saved as OOXML-based ZIP file and examines LFH and CDH header contents in detail and efficiently detects abnormally hidden data. The internal module of the system implemented in this study consists of ZIP File Identification Module, ZIP File Structure Analysis Module, ZIP File Validation Module, and GUI Module.

It was confirmed that all places where the slack area exists are in the Extra Field part of the LFH and CFH. Therefore, if the slack area that does not appear in the local file header of a general ZIP file is exploited, malicious code or any additional information can be hidden. Therefore, as shown in the figure below, five types of damaged files were analyzed and a function to automatically detect them was implemented.


A
Slack Space Detection
Display slack space information
Alarm/Log generation
B
Hidden Data Detection
Indicating CDHs Containing Concealed Data


C
Corrupted CDH Detection
3
4
5
1
2
DOCX (Add lena.png)
Forged DOCX
Corrupted MS-Office File detection
Remove CDH associated with Lena.png

File Analyzer - pg4-sample-word-template_forgery.docx

The number of LocalFileHeader(26) and CentralDirectory(25) is wrong!
LocalFileHeader(20) structure anomaly found.

Fig. 4 Hidden Data Detection from Corrupted MS-Office Files

When the implemented SW is executed, the OOXML structure analysis process is performed on input MS-Office files to check whether the slack space is included in the CDH and LFH headers. If hidden data exists, an alarm is displayed. Also, as shown in the figure below, data of a specific length can be added by changing/modifying and manipulating the internal structure of the OOXML-based MS-Office file. If specific data is hidden in MS-Office series digital files after changing the data size, a module that can automatically detect it has been developed. When a file disguised as a legitimate digital file is executed, damage from encryption/infection of digital files of general users by malicious ransomware code hidden inside the file can be prevented in advance by simply executing proposed SW.

B. AI-based Hidden Data Detection

A method for detecting hidden files in MS-Office files was derived by applying machine learning techniques. As shown in the figure below, files containing malicious code were collected and static and dynamic analysis were performed using Cuckoo sandbox[8]. Training and test data were collected from *MalwareBazaar*, a free malicious software database, and normal files were tested on files disclosed to Digital Corpora. About 150,000 normal files and about 3,800 malicious files with doc, docx, xls, xlsx, ppt, and pptx extensions were collected. Within the *oletools* module, a Python library, there is a library called *olevba*. Additionally, we analyze the internal source code for VBA macros embedded in OLE or OpenXML files. Therefore, the Cuckoo sandbox automatically executed macros of MS-Office file on Windows OS, performed static and dynamic sandbox analysis, and suspicious keywords that can interfere with virtualization were derived and created as IoC(Indicators of Compromise).

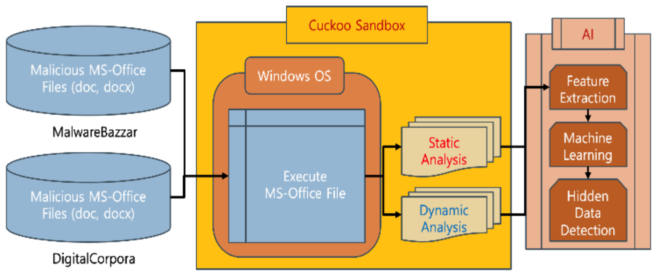


Fig. 5 AI based Hidden Data Detection

The detailed analysis process is diagrammed as shown in the figure below. After installing Windows OS and MS-Office SW in the Cuckoo sandbox, MS-Office files containing malicious codes were executed.

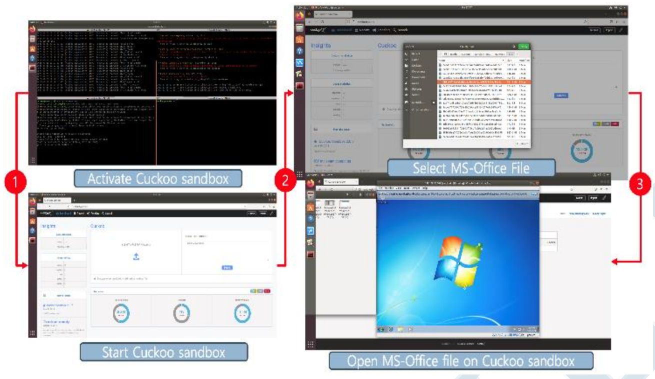


Fig. 6 Static/Dynamic Analysis Using Cuckoo sandbox (1)

When MS-Word file is executed using Cuckoo sandbox, static and dynamic analysis results and action-oriented event information can be analyzed.

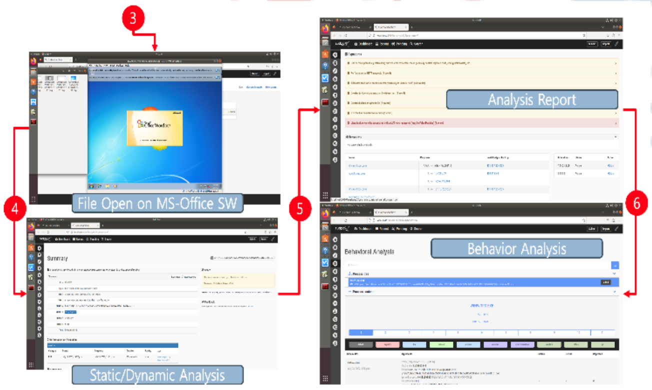
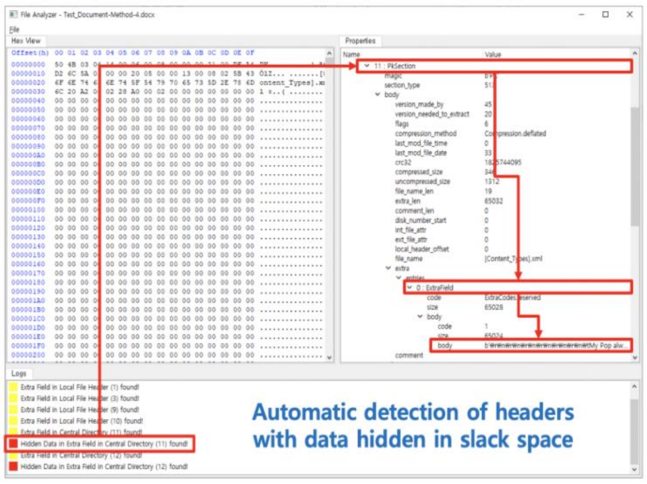


Fig. 7 Static/Dynamic Analysis Using Cuckoo sandbox (2)

C. Feature Extraction for AI-based Hidden Data Detection

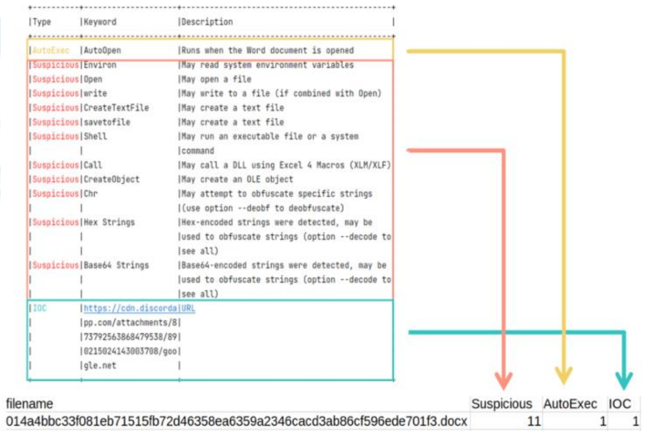
As a result of the experiment, malicious VBA macros included in malicious doc files could be extracted as follows. As a result of the analysis, it was confirmed that malicious data was classified into AutoExec, Suspicious, and IoC by type.


File Analyzer - Test_Document-Method-4.docx
Hex View
Properties
Logs
Automatic detection of headers with data hidden in slack space

```
Extra Field in Local File Header (1) found!
Extra Field in Local File Header (3) found!
Extra Field in Local File Header (9) found!
Extra Field in Local File Header (10) found!
Extra Field in Central Directory (11) found!
Hidden Data in Extra Field in Central Directory (11) found!
Extra Field in Central Directory (12) found!
Hidden Data in Extra Field in Central Directory (12) found!
```

Fig. 8 Hidden Data Detection SW

We conducted an experiment using a total of 3,206 MS-Office series files (doc, docx) collected from the internet. Among them, 2,565 files were used as training data, and 641 files were used as test data. For testing the AI-based automatic detection feature, we utilized 16 feature information, including the number of suspicious information confirmations, automatic execution counts, and IoCs, as shown in the figure below. In this regard, we employed a total of eight models, namely Random Forest, SVM, K-NN, Gaussian Naïve Bayes, Logistic Regression, Decision Tree, Ada Boost, and Gradient Boosting.



```
+----------+-------------------+---------------------------------------------+
|Type      |Keyword            |Description                                  |
+----------+-------------------+---------------------------------------------+
|AutoExec  |AutoOpen           |Runs when the Word document is opened
|Suspicious|Environ            |May read system environment variables
|Suspicious|Open               |May open a file
|Suspicious|write              |May write to a file (if combined with Open)
|Suspicious|CreateTextFile     |May create a text file
|Suspicious|savetofile         |May create a text file
|Suspicious|Shell              |May run an executable file or a system
|          |                   |command
|Suspicious|Call               |May call a DLL using Excel 4 Macros (XLM/XLF)
|Suspicious|CreateObject       |May create an OLE object
|Suspicious|Chr                |May attempt to obfuscate specific strings
|          |                   |(use option --deobf to deobfuscate)
|Suspicious|Hex Strings        |Hex-encoded strings were detected, may be
|          |                   |used to obfuscate strings (option --decode to
|          |                   |see all)
|Suspicious|Base64 Strings     |Base64-encoded strings were detected, may be
|          |                   |used to obfuscate strings (option --decode to
|          |                   |see all)
|IOC       |https://cdn.discorda|URL
|          |pp.com/attachments/8|
|          |73792563868479538/89|
|          |0215024143003708/goo|
|          |gle.net             |
```

filename	Suspicious	AutoExec	IOC
014a4bbc33f081eb71515fb72d46358ea6359a2346cacd3ab86cf596ede701f3.docx	11	1	1

Fig. 9 Feature Extraction Mechanism

D. Comparison of AI-based Hidden Data Detection

To build a machine learning model, a Python-based machine learning library (scikit-learn) was used. In addition, the following functions were implemented to calculate the accuracy and F1 score of the malicious data classification algorithm. In order to find the machine learning model that provides optimal performance, an experiment was conducted with a total of eight machine learning models. As a result of experimenting with 8 machine learning models, it was found that hidden files in MS-Office files can be detected best when using the Random Forest model as shown in the figure below.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 10, Issue 9, September 2023

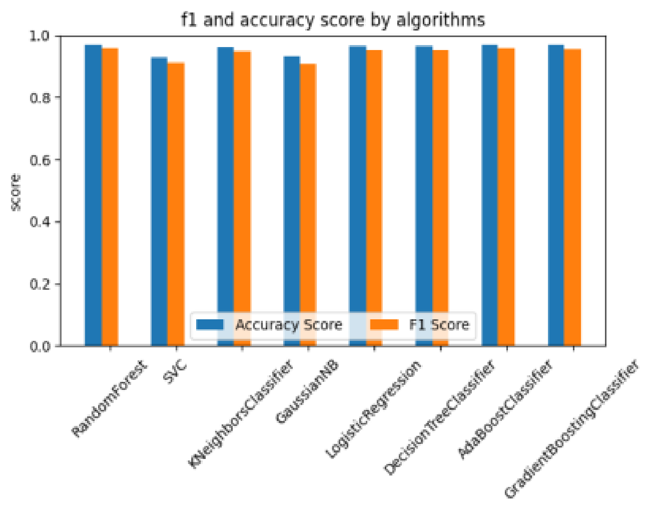


Fig. 10 F1-score on AI-based Hidden Data Detection

V. CONCLUSIONS

In this paper, we analyzed the OOXML-based MS-Office series digital file structure and proposed how illegal corruption process is possible on official public MS-Office digital documents by concealing malicious data on several slack space such as extra fields of CDH/LFH respectively. In addition, an advanced malicious data hidden detection mechanism in MS-Office files by intentionally deleting the CDH header is presented. We propose a method for malicious hidden data analysis and detection method based on 8 type of Machine Learning model to actively respond to suspicious ransomware disguising legitimate official OOXML-based MS-Office digital files. As a result, we designed and implemented a software for detecting corruption attack on OOXML-based public MS-Office files. Based on this, it is possible to provide an advanced system or mechanism for automatically detecting concealed ransomware code hidden in OOXML-based MS-Office digital files.

VI. ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2021R1F1A1046954).

REFERENCES

[1] Anghel, Catalin, “Digital Forensics – A Literature Review,” 2019. 42. 23-27. 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] A. Alenezi, H. Atlam, R. Alsagri, M. Alassafi, and G. Wills, “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), pages 106-115.
[4] Wikipedia contributors, "Office Open XML," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Office_Open_XML&oldid=917283554 (accessed January 20, 2023).
[5] Wikipedia contributors, "Zip (file format)," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Zip_(file_format)&oldid=916422219 (accessed January 20, 2023).
[6] File Formats: Microsoft Word Document (DOCX/DOC), https://www.leadtools.com/help/leadtools/v20/dh/to/document-file-formats-microsoft-word-document-docx-doc.html
[7] H. Lee, H.W. Lee, “Forgery Detection Mechanism with Abnormal Structure Analysis on Office Open XXML based MS-Word File,” IJASC 8 (4), 2019.
[8] Cuckoo, Automated Malware Analysis, https://cuckoosandbox.org.