Malicious PDF Document Detection Methods

Subjects: Computer Science, Theory & Methods

Contributor: Enzhou Song , Tao Hu , Peng Yi , Wenbo Wang

In network attacks based on malicious documents, the PDF document type accounts for a large proportion. Traditional PDF document detection technology usually builds a rule or feature library for specific vulnerabilities and therefore is only fit for single detection targets and lacks anti-detection ability.

PDF document detection multiple features

1. Introduction

In recent years, the number of network attacks through malicious documents has increased dramatically. Such attacks are often accompanied by serious harm, such as phishing, organization monitoring, and denial of service attacks. In network attacks based on malicious documents, the PDF document type accounts for a large proportion. According to the statistics of F-Secure Security, in 2020, malicious document attacks related to Adobe Reader accounted for 60% of total document attacks. Attackers can use embedded scripts, remote links and other means to carry out attacks through the plentiful functions of Adobe related products. The detection of such concealed attacks is difficult [1].

Among PDF-related attacks, constructing document vulnerabilities by exploiting defects of Adobe software is extremely harmful [2][3][4][5]. Through exploiting the vulnerabilities of document readers or parsers, such attacks can cause various types of harm, including downloading malicious programs remotely, implementing backdoor implantation, and executing malicious code directly. In 2020, Adobe released a security update bulletin to disclose the vulnerability CVE-2020-24432, the principle of which is that Adobe Acrobat Reader lacks strictness while censoring input validation. Attackers can execute arbitrary code in the context of the current user and cause serious damage.

Since 2018, Adobe Acrobat Reader has released more than 30 security update bulletins and disclosed more than 200 CVE vulnerabilities. Among them, there are more than 50 document vulnerabilities which are destructive and widely disseminated. **Table 1** presents the typical document vulnerability information disclosed by Adobe in recent years. It can be seen that document vulnerabilities are usually accompanied by harmful attacks, such as arbitrary code execution.

Table 1. Adobe typical document vulnerability information.

Number	Vulnerability	Dangerous Effects
CVE-2022-27787	Buffer overflow	Arbitrary code execution
CVE-2021-44709	Buffer overflow	Arbitrary code execution
CVE-2021-28564	Buffer overflow	Arbitrary code execution
CVE-2021-21017	Buffer overflow	Arbitrary code execution
CVE-2021-21045	Improper access control	Privilege escalation attack
CVE-2020-9704	Buffer overflow	Arbitrary code execution
CVE-2019-8249	Logical flaw	Arbitrary code execution
CVE-2019-8066	Buffer overflow	Arbitrary code execution
CVE-2018-19716	Buffer overflow	Arbitrary code execution

In recent years, researchers have presented various methods for detecting malicious documents. The detection types are mainly divided into two categories: static and dynamic detection methods. The static method usually determines the document's nature through analyzing the content, basic attributes, basic structure, metadata, and other document features without running them [6][7][8][9], whereas the dynamic method usually achieves detection through analyzing the system calls, operation behavior, and other features in a virtual environment's running process [10][11][12]. Traditional static and dynamic detection methods have advantages and limitations. Static detection does not need to execute actual samples and is thus relatively secure with high detection efficiency, fast speed, and low cost, but it ignores the malicious code extraction from documents [13]. Dynamic detection does not need to learn samples and can intuitively find the purpose of attack via running behavior, leading to a strong robustness. However, it faces such challenges as low efficiency, low speed, enormous cost, and sometimes threats from the anti-virtual machine and anti-sandbox technology [14][15][16].

2. PDF Background

The newest format of PDF was published as ISO 32000-1:2020 [17]. According to the standard, the basic structure of PDF documents is mainly divided into four parts: objects, physical structure, logical structure, and content stream [18][19], as shown in **Figure 1**.

Figure 1. The basic structure of PDFs.

The details of each part are as follows. (a) Objects. As the main part of PDF documents, objects carry various content, such as text information, fonts, embedded pictures, embedded videos, hyperlinks, and bookmarks. But the basic structure of different objects is similar regardless of the content classification. As shown in **Figure 1**, the first line in objects is the identifier, which consists of two numbers. The first is the serial number of objects. The second is the generation number of objects and is used to indicate whether the object has been modified. (b) Physical structure. This is mainly composed of four parts: file header, file body, cross-reference table, and file tail. The file header with a simple and fixed format is used to indicate the PDF version. The file body composed of document objects is the core part of the PDF. The cross-reference table is used to index document objects. The file tail is mainly used to save the summary, location, and other related information of the cross-reference table. (c) Logical structure. In the actual parsing process, PDFs are not parsed through physical structure but through logical structure. Parsing begins with the root node indicated by the file tail. The node indicates the directory, which contains pages, outlines, and other types of information. Each type of information is also organized in a tree structure. (d) Content stream. This is a common form of objects in PDFs and plays a key role in storing data. Stream objects are composed of three parts. The first part is a dictionary, which mainly stores the length and encoding method. The second part is the keyword, which is unified in different stream objects. It usually starts with “stream” and ends with “endstream”. The third part is the data between keywords.

3. Static Detection Method

Currently, the common static detection methods are mainly divided into three categories. The first category tends to detect the content features of files, mainly to extract suspicious JavaScript code fragments, shellcode data fragments, and metadata content in PDF documents. According to Tzermias et al. [11], more than 90% of malicious PDF document attacks need to be implemented with JavaScript and other codes. The detection model proposed by Laskov et al. [20] extracts JavaScript and uses lexical tagging to build an OCSVM classification. However, such methods are insufficient for PDF documents that do not rely on JavaScript code. Some document vulnerabilities build attack chains with the help of the document format. The second category tends to detect the structural

features of documents. It achieves detection mainly through extracting document structure and combining features such as metadata. Šrndić et al. [21] extracted vast features of basic structure for PDF but also had limitations in extracting malicious features. Cohen et al. [7] adopted the SFEM method to extract features from the document structure. Chandran et al. [22] scanned the structure of the PDFs through PeePDF and used the GRU model to employ classification. Srndic et al. [23] processed metadata in a similar way to structural paths and then substituted the data into classification models to achieve detection. But such methods have limited abstraction of features and are not comprehensive enough to detect content features. The static methods above are not comprehensive in feature extraction, resulting in insufficient detection for various attack methods. The third category tends to build the feature library and use multiple features. Wen Weiping et al. [24] designed a feature library for document vulnerabilities. The malicious document can be identified when it matches the relevant features of the feature library. However, such methods only apply to malicious PDF documents with disclosed vulnerabilities and have no detection effect on 0-day vulnerabilities. Falah et al. [25] used feature engineering to evaluate multiple features and detect malicious PDFs, but they ignored malicious features in JavaScript code, and the feature evaluation method deserves improvement.

4. Dynamic Detection Method

Dynamic detection methods mainly focus on JavaScript code and shellcode data fragments embedded in the document. The MDScan method [14] mainly executes the extracted JavaScript code. It extracts the relevant operation performance of memory as a sequence and performs subsequent detection. But these similar matching methods have limitation in detecting new type of attacks. Iwamoto et al. [26] used the simulation method to execute the document shellcode. It is mainly based on the entropy of the byte sequence, which can solve the problem of difficult vulnerability triggering to some extent. However, this detection is insufficient for some malicious codes that can be only triggered in a specific situation. Xu et al. [27] proposed opening the PDF document with the same reader in the heterogeneous operating system and identified malicious documents through the similarity performance of system calls and process tracking. However, such methods have an excessive overhead and low detection efficiency. Liu et al. [28] executed JavaScript code in PDF documents through their own built-in execution environment and monitored common malicious behaviors. But such methods can only detect traditional and common attack methods. It is insufficient to detect the document using new anti-detection method. In summary, the dynamic detection method is expensive and consumes large amounts of resources and memory space. It is not suitable for situations with large numbers of samples, short time requirements, and low resource requirements [29].

References

1. Yu, M.; Jiang, J.G.; Li, G.; Liu, C.; Huang, W.Q.; Song, N. A Survey of Research on Malicious Document Detection. J. Cyber Secur. 2021, 6, 54–76.
2. Nissim, N.; Cohen, A.; Moskovitch, R.; Shabtai, A.; Edry, M.; Bar-Ad, O.; Elovici, Y. ALPD: Active Learning Framework for Enhancing the Detection of Malicious PDF Files. In Proceedings of the

2014 IEEE Joint Intelligence and Security Informatics Conference, The Hague, The Netherlands, 24–26 September 2014; pp. 91–98.

3. Wang, Y. The De-Obfuscation Method in the Static Detection of Malicious PDF Documents. In Proceedings of the 7th Annual International Conference on Network and Information Systems for Computers, Guiyang, China, 23–25 July 2021; pp. 44–47.
4. Lei, J.; Yi, P.; Chen, X.; Wang, L.; Mao, M. PDF Document Detection Model Based on System Calls and Data Provenance. J. Comput. Appl. 2022, 42, 3831.
5. Lu, X.; Wang, F.; Jiang, C.; Lio, P. A Universal Malicious Documents Static Detection Framework Based on Feature Generalization. Appl. Sci. 2021, 11, 12134.
6. Maiorca, D.; Giacinto, G.; Corona, I. A pattern recognition system for malicious PDF files detection. In International Conference on Machine Learning and Data Mining in Pattern Recognition; Springer: Berlin/Heidelberg, Germany, 2012; pp. 510–524.
7. Cohen, A.; Nissim, N.; Rokach, L.; Elovici, Y. SFEM: Structural Feature Extraction Methodology for the Detection of Malicious Office Documents Using Machine Learning Methods. Expert Syst. Appl. 2016, 63, 324–343.
8. Wang, L.N.; Tan, C.; Yu, R.W. The Malware Detection Based on Data Breach Actions. J. Comput. Res. Dev. 2017, 54, 1537–1548.
9. Feng, D.; Yu, M.; Wang, Y. Detecting Malicious PDF Files Using Semi-Supervised Learning Method. In Proceedings of the International Conference on Advanced Computer Science Applications and Technologies, Beijing, China, 25–26 March 2017.
10. Corona, I.; Maiorca, D.; Ariu, D.; Giacinto, G. Lux0R: Detection of malicious PDF-embedded JavaScript code through discriminant analysis of API references. In Proceedings of the Workshop on Artificial Intelligent and Security Workshop, Scottsdale, AZ, USA, 3–7 November 2014; pp. 47–57.
11. Tzermias, Z.; Sykiotakis, G.; Polychronakis, M.; Markatos, E.P. Combining static and dynamic analysis for the detection of malicious documents. In Proceedings of the Fourth European Workshop on System Security; ACM: New York, NY, USA, 2011; pp. 1–6.
12. Maiorca, D.; Ariu, D.; Corona, I.; Giacinto, G. An evasion resilient approach to the detection of malicious PDF files. In Proceedings of the 2015 International Conference on Information Systems Security and Privacy, Angers, France, 9–11 February 2015; Springer: Cham, Switzerland, 2015; pp. 68–85.
13. Du, X.; Lin, Y.; Sun, Y. Malicious PDF document detection based on mixed feature. J. Commun. 2019, 40, 118–128.

14. Vatamanu, C.; Gavriluţ, D.; Benchea, R. A practical approach on clustering malicious PDF documents. J. Comput. Virol. 2012, 8, 151–163.
15. Maiorca, D.; Ariu, D.; Corona, I.; Giacinto, G. A Structural and Content-Based Approach for a Precise and Robust Detection of Malicious PDF Files. In Proceedings of the 1st International Conference on Information Systems Security and Privacy, Angers, France, 9–11 February 2015; pp. 27–36.
16. Lu, X.; Zhuge, J.; Wang, R.; Cao, Y.; Chen, Y. De-Obfuscation and Detection of Malicious PDF Files with High Accuracy. In Proceedings of the 46th Hawaii International Conference on System Sciences, Wailea, HI, USA, 7–10 January 2013; pp. 4890–4899.
17. ISO 32000-2:2020. Available online: https://www.pdfa.org/resource/iso-32000-pdf/ (accessed on 1 December 2022).
18. Šrndic, N.; Laskov, P. Detection of malicious pdf files based on hierarchical document structure. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 25–27 February 2013.
19. Jose, T.S.; Santos, D.L. Malicious PDF Documents Detection using Machine Learning Techniques. In Proceedings of the 4th International Conference on Information Systems Security and Privacy, Madeira, Portugal, 22–24 January 2018.
20. Laskov, P. Static detection of malicious JavaScript-bearing PDF documents. In Proceedings of the Twenty-Seventh Computer Security Applications Conference, Orlando, FL, USA, 5–9 December 2011; pp. 373–382.
21. Nedim, Š.; Pavel, L. Practical Evasion of a Learning-Based Classifier: A Case Study. In Proceedings of the 2014 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–18 May 2014; pp. 197–211.
22. Chandran, P.P.; Jeyakarthic, M. Jeyakarthic: Intelligent Optimal Gated Recurrent Unit based Malicious PDF Detection and Classification Model. In Proceedings of the 2022 International Conference on Applied Artificial Intelligence and Computing, Salem, India, 9–11 May 2022; pp. 1273–1279.
23. Šrndić, N.; Laskov, P. Hidost: A Static Machine-Learning-Based Detector of Malicious Files. EURASIP J. Inf. Secur. 2016, 2016, 22.
24. Wen, W.; Wang, Y.; Meng, Z. PDF file vulnerability detection. J. Tsinghua Univ. (Sci. Technol.) 2017, 57, 33–38.
25. Falah, A.; Pan, L.; Huda, S.; Pokhrel, S.R.; Anwar, A. Improving malicious PDF classifier with feature engineering: A data-driven approach. Future Gener. Comput. Syst. 2021, 115, 314–326.

26. Iwamoto, K.; Wasaki, K. A Method for Shellcode Extraction from Malicious Document Files Using Entropy and Emulation. Int. J. Eng. Technol. 2016, 8, 101–106.
27. Xu, M.; Kim, T. Plat Pal: Detecting malicious documents with platform diversity. In Proceedings of the USENIX Security Symposium, Vancouver, BC, Canada, 16–18 August 2017; pp. 271–287.
28. Liu, D.; Wang, H.; Stavrou, A. Detecting malicious javascript in pdf through document instrumentation. In Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Atlanta, GA, USA, 23–26 June 2014; pp. 100–111.
29. Yu, M.; Jiang, J.; Li, G.; Li, J.; Lou, C.; Liu, C.; Huang, W.; Wang, Y. A Unified Malicious Documents Detection Model Based on Two Layers of Abstraction. In Proceedings of the IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th In-ternational Conference on Smart City; IEEE 5th International Conference on Data Science and Systems, Zhangjiajie, China, 10–12 August 2019; pp. 2317–2323.