## Title

**Extract Me If You Can Abusing PDF Parsers Malware Detectors**

## Authors

**Curtis Carmony** et.al. from **Syracuse University**

## Abstract

Owing to **the popularity of the PDF format** and **the continued exploitation of Adobe Reader, the detection of malicious PDFs remains a concern.** All existing detection techniques **rely on the PDF parser to a certain extent**, while **the complexity of the PDF format** leaves an abundant space for **parser confusion**. To **quantify** the difference between these parsers and Adobe Reader, we create **a reference JavaScript extractor** by directly tapping into **Adobe Reader** at locations identified through **a mostly automatic binary analysis technique**. By comparing the output of **this reference extractor** against that of several open-source JavaScript extractors on **a large data set** obtained from **VirusTotal**, we are able to **identify hundreds of samples** which **existing extractors fail to extract JavaScript from**. By **analyzing these samples** we are able to identify **several weaknesses** in each of these extractors. Based on these lessons, we apply **several obfuscations on a malicious PDF sample, which can successfully evade all the malware detectors tested**. We call this **evasion technique a *PDF parser confusion attack***. Lastly, we demonstrate that **the reference JavaScript extractor improves the accuracy of existing JavaScript-based classifiers** and **how it can be used to mitigate these parser limitations in a real-world setting.**

## Introduction

Even though Adobe’s Acrobat Reader, more commonly known as Adobe Reader, has become increasingly secure through the addition of **advanced security mechanisms such as a sandbox** [5], new exploits continue to be found with **44 CVEs published in 2014** [1] and **128 published in 2015** at the time of writing [2]. Due to the continued exploitation of Adobe Reader along with the ubiquity of the PDF format, **the detection of malicious PDF files remains a concern**, with Kaspersky reporting that Adobe Reader was **the third most exploited target in 2014 and attracted 5% of the overall attacks** [18].

**Malicious PDF detection in commercial anti-virus products relies heavily on signature detection and is insufficient to detect PDFs containing zero-day exploits or advanced persistent threats.** To address this limitation, two classes of systems have been proposed to **detect malicious PDF files** specifically: 1) **structure and metadata based detectors** [29], [32], [38] and 2) **JavaScript-based classifiers** [23], [25], [37], [26].

**Structure and metadata based detection methods** distinguish benign and malicious PDFs by determining which structural features and metadata are most associated with each class. However, the essential malice of PDF exploits **does not originate in file structures** but **rather in embedded payloads** (e.g., JavaScript code) **that bear malicious intent**. Therefore, these detectors can be **easily evaded by the *mimicry attack***[39], [38] and the *reverse mimicry attack* [28], which hide harmful code in PDF files that exhibit structural features and metadata associated with benign files.

To **fundamentally address PDF exploits**, it is necessary to **analyze the contents of documents and search for malicious payloads**. Prior work [23] reveals that **JavaScript is the most common malicious content** in PDF exploits for **two major reasons**: 1) **the implementation of the Adobe JavaScript APIs exposes vulnerabilities** and 2) **JavaScript code is used to enable advanced exploitation techniques**, such as heap spraying. Almost all of the malicious PDF documents in our sample set collected **from VirusTotal contain JavaScript, indicating that the extraction and analysis of embedded JavaScript is essential to malicious PDF detection**.

To this end, **prior JavaScript-based classifiers** [23], [25], [37] have attempted to parse PDF documents, extract JavaScript from them, and then **analyze this JavaScript to classify it as benign or malicious**. These works all depend on **their ability to accurately extract JavaScript from PDFs**. With the exception of MPScan[26], which uses a modified version of Adobe Reader similar to the one presented in this work, each of these works rely either on **open-source parsers or their own home-grown parsers**. Because all of these parsers are incomplete and have oversimplified assumptions in regards to where JavaScript can be embedded, **these detection methods are not accurate or robust**.

In this paper, we aim to conduct a systematic study on **a new evasion technique** called the *PDF parser confusion attack*, which aims to confuse **the PDF parsers in malware detectors in order to evade detection**. In essence, this evasion attack exemplifies the *chameleon* and *werewolf* attacks that deliberately abuse **file processing in malware detectors** [22]. However, compared to other file types (e.g., ZIP, ELF and PE) that have been investigated in this previous work, the combination of the complexity of the PDF format and Adobe Reader’s leniency in parsing these files **potentially offers a much larger attack space**. Unfortunately, this attack space has not been studied sufficiently in the security community.

To enable a systematic study we have developed **a *reference JavaScript extractor***by directly tapping into Adobe Reader, which is arguably the most popular and most targeted PDF viewer [19]. To develop this reference extractor, we present a mostly automatic dynamic binary analysis technique that can quickly identify a small number of candidate tap points, which can be further refined by simple manual analysis. We then perform a differential analysis on this reference extractor and several popular extractors, using over 160,000 PDFs collected from VirusTotal. For each extractor we identify hundreds of samples which it cannot correctly process, but that contain JavaScript according to the reference extractor.

By delving into these discrepancies between the reference extractor and the existing extractors we have identified several new obfuscations, and further quantified their impact when used in parser-confusion attacks on JavaScript extractors and malware detectors. By combining several of these obfuscations, **we demonstrate that a malicious PDF can successfully evade *all* the malware detectors evaluated, including signature- based, structure/metadata-based, and JavaScript-based detectors**.

These findings suggest that the key to effective countermeasures is a **high-fidelity parser** that closely mimics the parsing logic of Adobe Reader. One possible solution is to directly deploy our reference JavaScript extractor for JavaScript-based detectors. Our experiment shows that this deployment scheme not only incurs acceptable runtime overhead, but also produces **much higher detection accuracy**. Our experiments show that after replacing the original parser with our reference extractor, the detection rate of PJScan [23] increases from **68% to 96%** for a specific version of Adobe Reader, based on a fairly rudimentary classifier.

Paper Contributions. In summary, this paper makes the following contributions:

* We propose a mostly-automatic, platform independent **tap point** identification technique to correctly identify tap points related to **JavaScript parsing** and execution in Adobe Reader which are used to develop a reference JavaScript extractor.
* Using our reference extractor we systematically evaluate the shortcomings of existing JavaScript extraction tools. We have identified hundreds of PDF samples (both benign and malicious), which existing extractors failed to extract JavaScript from. We manually investigate many of them, and identify their root causes.
* We construct several PDF parser confusion attacks by combining several of the obfuscations identified in our analysis. These evasions have proved to be effective in successfully evading all of the malware detectors we tested.
* We discuss **several mitigation techniques**. In particular, we demonstrate that with **our reference JavaScript extractor**, the detection rate of an existing classifier increases significantly from 68% to 96% on our sample set, and present a possible deployment scenario for the reference extractor.

We plan to **release the complete data set** and also **launch a public service for our reference JavaScript extractor**, to help security researchers conduct further research on this problem. A list of MD5 hash values are available for part of the data set and can be found at **https://goo.gl/qtbuOC**.

## Background

## Reference JavaScript Extractor

## Differential Analysis

## Parser Confusion Attacks

## Mitigations

## Limitations

## Conclusion

## References

[1] “Adobe acrobat reader : Security vulnerabilities published in 2014,” http://www.cvedetails.com/vulnerability-list/vendor id-53/product id- 497/year- 2014/Adobe- Acrobat- Reader.html.

* [2]  “Adobe acrobat reader : Security vulnerabilities published in 2015,” http://www.cvedetails.com/vulnerability- list/vendor id- 53/product id- 497/year- 2015/Adobe- Acrobat- Reader.html.
* [3]  “CVE-2013-3346 Adobe Reader Tool Button Use After Free,” http://www.rapid7.com/db/modules/exploit/windows/browser/adobe toolbutton.
* [4]  “difflib – helpers for computing deltas,” https://docs.python.org/2/ library/difflib.html.
* [5]  “Introducing adobe reader protected mode,” http://blogs.adobe.com/ security/2010/07/introducing- adobe- reader- protected- mode.html.
* [6]  “jsunpack-n,” http://code.google.com/p/jsunpack- n/.
* [7]  “libpdfjs,” http://sourceforge.net/projects/libpdfjs/.
* [8]  “origami,” https://code.google.com/p/origami- pdf/.
* [9]  “PDFrate A machine learning based classifier operating on document metadata and structure,” http://pdfrate.com/.
* [10]  “Poppler,” http://poppler.freedesktop.org/.
* [11]  “PyPDF2,” https://github.com/mstamy2/PyPDF2.
* [12]  “Vulnerability Details : CVE-2013-3346,” http://www.cvedetails.com/ cve/2013- 3346.
* [13]  J. Berkenbilt, “QPDF: A Content-Preserving PDF Transformation Sys- tem,” http://qpdf.sourceforge.net/.
* [14]  G. Delugre ́, “The undocumented password validation algorithm of adobe reader x,” http://esec-lab.sogeti.com/post/The-undocumented- password- validation- algorithm- of- Adobe- Reader- X.
* [15]  B. Dolan-Gavitt, T. Leek, J. Hodosh, and W. Lee, “Tappan zee (north) bridge: mining memory accesses for introspection,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 839–850.
* [16]  P. Fogla and W. Lee, “Evading network anomaly detection systems: Formal reasoning and practical techniques,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006.
* [17]  P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, “Poly- morphic blending attacks,” in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, 2006.
* [18]  M. Garnaeva, V. Chebyshev, D. Makrushin, R. Unuchek, and A. Ivanov, “Kaspersky security bulletin 2014.” http: //securelist.com/analysis/kaspersky- security- bulletin/68010/kaspersky- security- bulletin- 2014- overall- statistics- for- 2014/.
* [19]  D. Goodin, “It’s official: Adobe reader is world’s most-exploited app,” http://www.theregister.co.uk/2010/03/09/adobe reader attacks/, 2010.
* [20]  A. Henderson, A. Prakash, L. K. Yan, X. Hu, X. Wang, R. Zhou, and H. Yin, “Make it work, make it right, make it fast: Building a platform- neutral whole-system dynamic binary analysis platform,” in *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, ser. ISSTA 2014. New York, NY, USA: ACM, 2014, pp. 248–258. [Online]. Available: http://doi.acm.org/10.1145/2610384.2610407
* [21]  G. Hunt and D. Brubacher, “Detours: Binary interception of win32 functions,” in *Third USENIX Windows NT Symposium*. USENIX, July 1999, p. 8. [Online]. Available: http://research.microsoft.com/ apps/pubs/default.aspx?id=68568
* [22]  S. Jana and V. Shmatikov, “Abusing file processing in malware detectors for fun and profit,” in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, ser. SP ’12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 80–94. [Online]. Available: http://dx.doi.org/10.1109/SP.2012.15
* [23]  P. Laskov and N. Sˇrndic ́, “Static Detection of Malicious JavaScript- bearing PDF Documents,” in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC ’11. New York, NY, USA: ACM, 2011, pp. 373–382. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076785
* [24]  M. Lee, “GNU PDF project leaves FSF High Priority Projects list; mis- sion complete!” https://www.fsf.org/blogs/community/gnu-pdf-project- leaves- high- priority- projects- list- mission- complete.
* [25]  D. Liu, H. Wang, and A. Stavrou, “Detecting Malicious Javascript in PDF Through Document Instrumentation,” in *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, ser. DSN ’14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 100–111. [Online]. Available: http://dx.doi.org/10.1109/DSN.2014.92
* [26]  X. Lu, J. Zhuge, R. Wang, Y. Cao, and Y. Chen, “De-obfuscation and Detection of Malicious PDF Files with High Accuracy,” in *46th Hawaii International Conference on System Sciences, HICSS 2013, Wailea, HI, USA, January 7-10, 2013*, 2013, pp. 4890–4899. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2013.166
* [27]  C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, “Pin: Building customized program analysis tools with dynamic instrumentation,” in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2005.
* [28]  D. Maiorca, I. Corona, and G. Giacinto, “Looking at the Bag is Not Enough to Find the Bomb: An Evasion of Structural Methods for Malicious PDF Files Detection,” in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ser. ASIA CCS ’13. New York, NY, USA: ACM, 2013, pp. 119–130. [Online]. Available: http: //doi.acm.org/10.1145/2484313.2484327
* [29]  D. Maiorca, G. Giacinto, and I. Corona, “**A Pattern Recognition System for Malicious PDF Files Detection**,” in *Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition*, ser. MLDM’12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 510–524. [Online]. Available: http: //dx.doi.org/10.1007/978-3-642-31537-4 40
* [30]  P. Ratanaworabhan, B. Livshits, and B. Zorn, “Nozzle: A defense against heap-spraying code injection attacks,” in *Proceedings of the Usenix Security Symposium*. USENIX, 2009. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=81085
* [31]  M. Z. Shafiq, S. A. Khayam, and M. Farooq, “Embedded malware de- tection using markov n-grams,” in *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2008.
* [32]  C. Smutz and A. Stavrou, “**Malicious PDF Detection Using Metadata and Structural Features**,” in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC ’12. New York, NY, USA: ACM, 2012, pp. 239–248. [Online]. Available: http://doi.acm.org/10.1145/2420950.2420987
* [33]  K. Snow, S. Krishnan, F. Monrose, and N. Provos, “Shellos: Enabling fast detection and forensic analysis of code injection attacks,” in *USENIX Security Symposium*, 2011. [Online]. Available: http://static.usenix.org/events/sec11/tech/full papers/Snow.pdf
* [34]  Y. Song, M. E. Locasto, A. Stavrou, A. D. Keromytis, and S. J. Stolfo, “On the infeasibility of modeling polymorphic shellcode,” in *Proceed- ings of the 14th ACM Conference on Computer and Communications Security*, 2007.
* [35]  D. Stevens, “PDF Tools,” http://blog.didierstevens.com/programs/pdf- tools/.
* [36]  ——, “PDFiD On VirusTotal,” http://blog.didierstevens.com/2009/04/ 21/pdfid- on- virustotal/.
* [37]  Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. P. Markatos, “Combining static and dynamic analysis for the detection of malicious documents,” in *Proceedings of the Fourth European Workshop on System Security*, ser. EUROSEC ’11. New York, NY, USA: ACM, 2011, pp. 4:1–4:6. [Online]. Available: http: //doi.acm.org/10.1145/1972551.1972555
* [38]  N. Sˇrndic ́ and P. Laskov, “**Detection of malicious PDF files based on hierarchical document structure**,” in *In Proceedings of the Network and Distributed System Security Symposium, NDSS 2013*. The Internet Society, 2013.
* [39]  N. Sˇrndic and P. Laskov, “Practical evasion of a learning-based classifier: A case study,” in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, ser. SP ’14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 197–211. [Online]. Available: http://dx.doi.org/10.1109/SP.2014.20