

WHY SAFE WORKSPACE® SOLUTIONS TECHNOLOGY FAQS COMPANY RESOURCES NEWS & EVENTS CONTACT US

Blog

SUPPORT Search REQUEST DEMO

< Back

The beginners Guide To – Adobe PDF Malware Reverse Engineering Part 2

By BUFFERZONE Team, 15/06/2023

Share f in

Target: Cybersecurity specialist

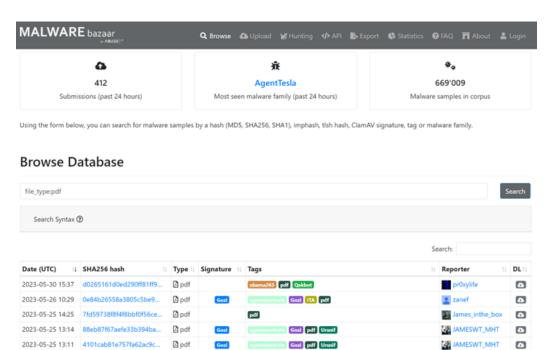
Tags: Adobe PDF, Malware, Content Disarm and Reconstruction (CDR), Reverse Engineering

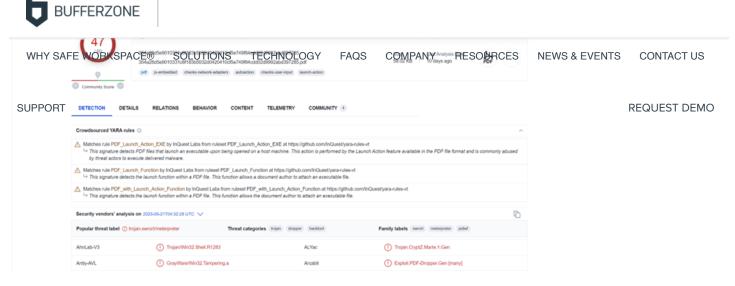
In this blog we will continue the PDF malware analysis part –1 and continue to investigate more complex malware.

Collection:

Within this blog, we shall retrieve a potentially suspicious file from MalwareBazaar and collectively examine the PDF file (remember to operate within a virtual machine). By employing the "file_type:pdf" filter, we shall acquire the most recently uploaded PDF files within the system. Let us proceed with downloading the latest file, possessing the sha256 hash:

304a28d5e9010331c8f183b5932d0420410cf5e749f84cdd02d9992abd397285. We specifically chose this file for the blog as it does not employ a phishing/luring style and possesses intriguing attributes we wish to discuss.





This indicates that the file consists of known attack vectors. To verify we will start our static analysis.

Reverse Engineering PDF File Using Static Analysis

In this blog we will focus on PDFiD [5], Pdfalyze [6], and Pdf-tool [5].

PDFiD

By running python pdfid.py <file> we will get the following output:

PDF Header: %PDF-1.3	
obj	25
endobj	25
stream	4
endstream	4
xref	2
trailer	2
startxref	2
/Page	2
/Encrypt	0
/ObjStm	0
/JS	1
/JavaScript	1
/AA	1
/OpenAction	1
/AcroForm	0
/JBIG2Decode	0
/RichMedia	0
/Launch	1
/EmbeddedFile	0
/XFA	0
/URI	0
/Colors > 2^24	0

Insights from PDfiD unveil the existence of 25 objects, 4 streams, 2 pages, along with crucial execution descriptors: /AA, /OpenAction, and /Launch. The execution procedure commonly involves /JS and / JavaScript, indicating the deployment of active scripting elements. This will assist us in prioritizing the initial search using Pdfalyze.

Pdfalyze



From this we can observe that object 23 (Action: Launch) and object (22 Action: JavaScript) are interesting and highlighted in red. We will start examining 22 and move to 23.

22./Action:JavaScript	/Root/OpenAction	Dictionary
AddressInParent /S /JS	/OpenAction /JavaScript this.exportDataObject({ cName: "form", nLaunch: 0 });	Name TextString
/Type	/Action	Name

We observe that JavaScript executes exportDataObject [9]. According to the documentation, we discover that the "cName" parameter is mandatory and indicates the desired file attachment for export. Additionally, there are three optional values for "nLaunch":

- 0: Triggers file preservation.
- 1: Triggers file opening after preservation.
- 2: Instructs Acrobat to temporarily save the file attachment and then prompt the operating system to open it (Acrobat lacks knowledge of which programs handle specific file types, whereas the OS does).

23./Action:Launch	/Root/Pages/Kids[0]/AA[/0]	Dictionary
AddressInParent /S /Type /Win	/AAI/O] /Laumch /Action /Action /Action /Action /Action /Comparison of the comparison of the compariso	Name Name Dictionary

Launch a cmd.exe with the file that was saved.

The form is found in object $20 \rightarrow to 21$:





Pdf-parser

Object 21 appears to be a flat Decode stream. To analyze it, we will follow these steps:

- 1. Execute the pdf-parser [5] command and save the file:
- > Python pdf-parser.py -f -o 21 -d extract_21 <file>
 - 2. The file will be saved as extract_21.
 - 3. Next, we will employ the File Linux command (a file-type detector) to determine the object's nature. It is identified as a PE32 executable rather than a PDF in object 22.
- > File extract_21

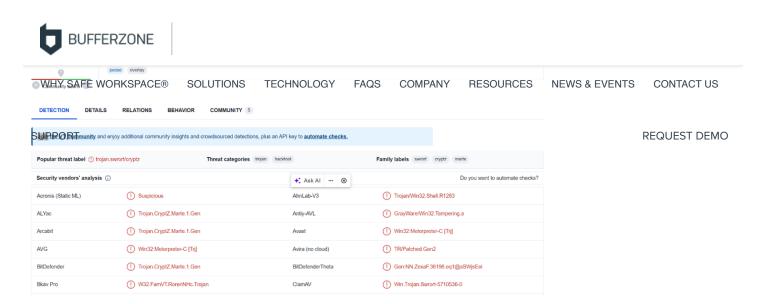
The result will indicate: PE32 executable (GUI) Intel 80386, for MS Windows.

- 4. Now, let us calculate the MD5 hash of the exported file:
- > md5 extract_21

5f00d238716e3f337786f4355b2b9787

The next step is to search the executable md5 in VirusTotal: 5f00d238716e3f337786f4355b2b9787

The beginners Guide To – Adobe PDF Malware Reverse Engineering Pa...



We can conclude that the file contains a malicious embedded object (detected by 57 engines).

Summary

In this blog, we expand upon the initial blog and investigate a more intricate PDF malware assault. Attack patterns may vary, but the research approach remains consistent. We trust you found the novice's PDF guide Part -2 to be informative. Kindly visit our website for upcoming blog entries.

References

- [1] Adobe PDF, https://www.adobe.com/acrobat/about-adobe-pdf.html
- [2] Common Crawl data statistics, https://commoncrawl.github.io/cc-crawl-statistics/plots/mimetypes.
- [3] Dubin, Ran. "Content Disarm and Reconstruction of PDF Files." IEEE Access (2023).
- [4] MalwareBazaar, Public Malware Repository, https://bazaar.abuse.ch/
- [5] Didier Stevens, PDF tools, https://blog.didierstevens.com/programs/pdf-tools/
- [6] Pdfalyzer, https://github.com/michelcrypt4d4mus/pdfalyzer
- $\label{thm:com/gui/file/do265161d0ed290ff81ff99e4571de9b709b357c9e663ad2b4519b68497705f5 \cite{Section 1} \cite{Section 1} Yara, https://virustotal.github.io/yara/$
- [9] Adobe PDF importing and exporting attachments, https://acrobatusers.com/tutorials/print/importing-and-exporting-pdf-file-attachments-acrobat-javascript/

PRODUCT COMPANY RESOURCES NEWS & EVENTS CONTACT BLOG

WHY SAFE WORKSPACE® ABOUT CASE STUDIES PRESS RELEASES CONTACT US Stop Worrying and Start Isolating – Combining Phishing Detection and Zero-Trust Prevention



© 2024 BUFFERZONE® Security Ltd. All rights reserved. BUFFERZONE® and SafeBridge are registered trademarks of BUFFERZONE® Security Ltd.

TERMS OF USE | PRIVACY POLICY | GDPR STATEMENT Design: DVIVO / Code: Alice

6 of 6