Basic Static Malware Analysis on Linux

WE Innovate X Zero$ploit

Supervised by : Zeyad Waleed

Prepared by : Omar Hassan

Required Task

Two suspicious PDF hashes to investigate & analyze using ***peepdf pdfid pdfparser***

SHA-256
cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577
5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e

Prerequisites : Ubuntu Machine

Setting	Recommended
RAM	3-4 GB
Disk	15-20 GB
CPU	1-2 Cores
Network	NAT

Phase 1: Environment Preparation

```
$ sudo apt update && sudo apt upgrade -y
$ sudo apt install python2 -y
$ curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-
pip.py
$ sudo python2 get-pip.py
```

Installing the tools

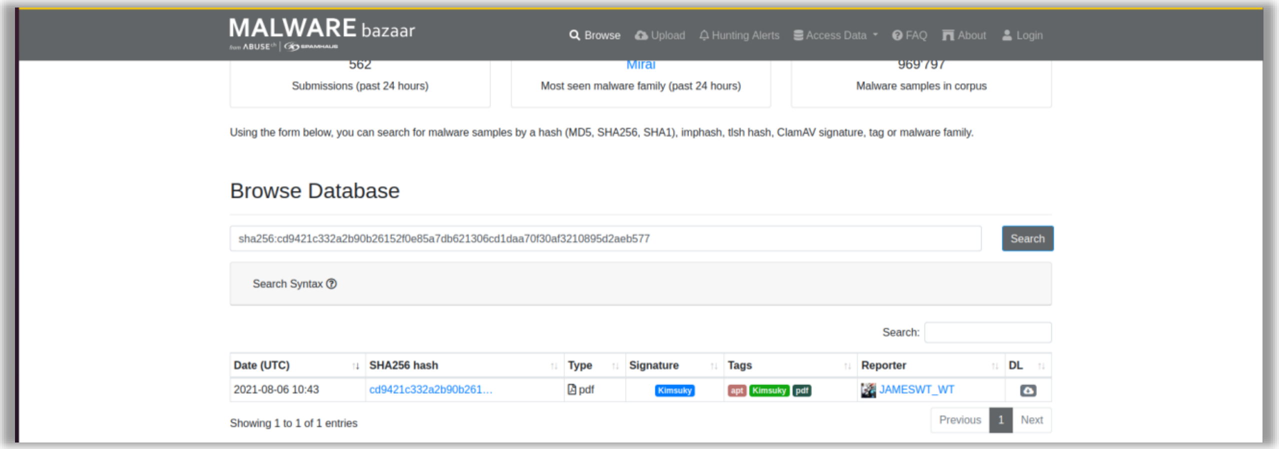
Pdfid & pdfparser are both available on **DidierStevensSuite.git** while **peepdf** is on **peepdf.git**

```
$ cd Downloads/
$ git clone https://github.com/DidierStevens/DidierStevensSuite.git
$ git clone https://github.com/jesparza/peepdf.git
```

Phase 2: Collecting the Samples

MAKE SURE TO ISOLATE YOU VIRTUAL MACHINE AFTER THE DOWNLOAD

Go to Malware Bazaar to download the samples, after that it is recommended to change the network adapter to host only.



562
Submissions (past 24 hours)

Mirai
Most seen malware family (past 24 hours)

969'797
Malware samples in corpus

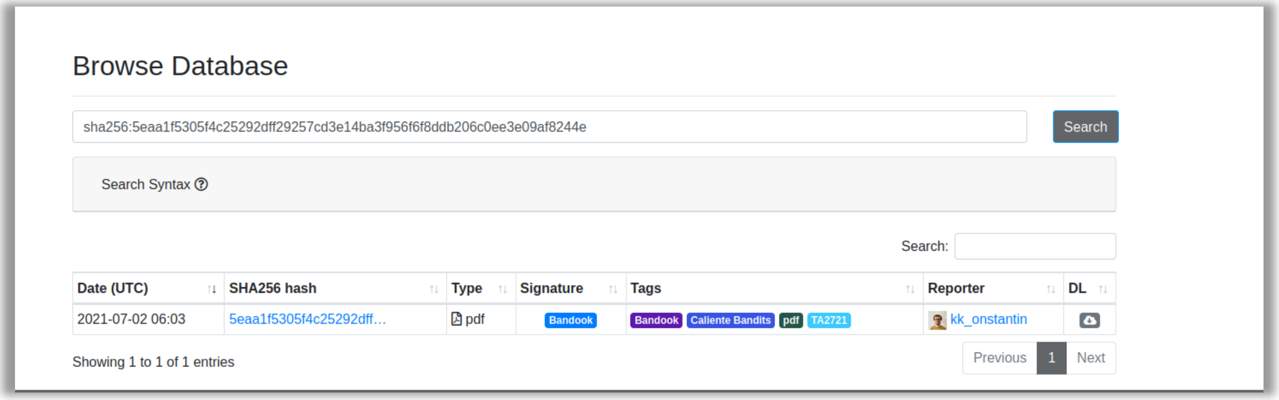
Using the form below, you can search for malware samples by a hash (MD5, SHA256, SHA1), imphash, tlsh hash, ClamAV signature, tag or malware family.

Browse Database

sha256:cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577

Date (UTC)	SHA256 hash	Type	Signature	Tags	Reporter	DL
2021-08-06 10:43	cd9421c332a2b90b261...	pdf	Kimsuky	apt Kimsuky pdf	JAMESWT_WT	

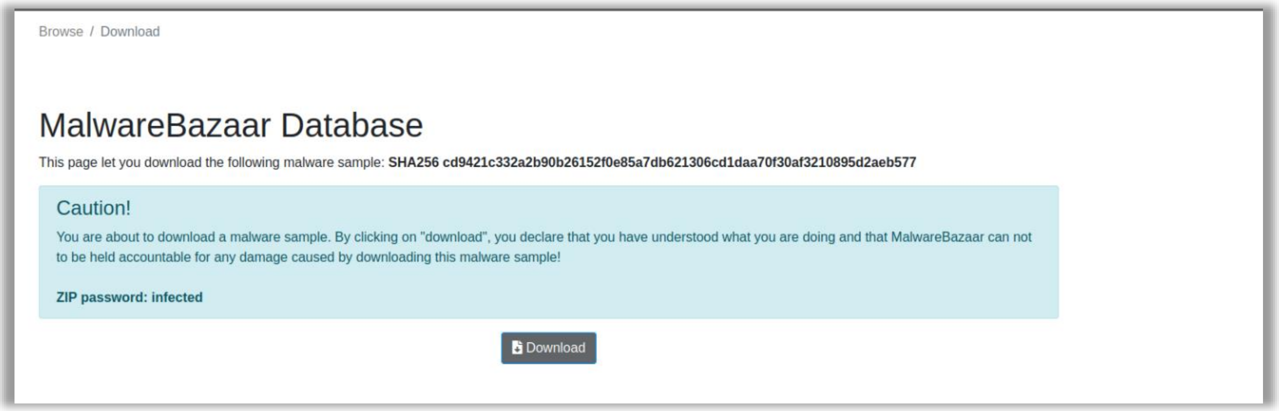
Showing 1 to 1 of 1 entries


Browse Database

sha256:5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e

Date (UTC)	SHA256 hash	Type	Signature	Tags	Reporter	DL
2021-07-02 06:03	5eaa1f5305f4c25292dff...	pdf	Bandook	Bandook Caliente Bandits pdf TA2721	kk_onstantin	

Showing 1 to 1 of 1 entries



Browse / Download

MalwareBazaar Database

This page let you download the following malware sample: **SHA256 cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577**

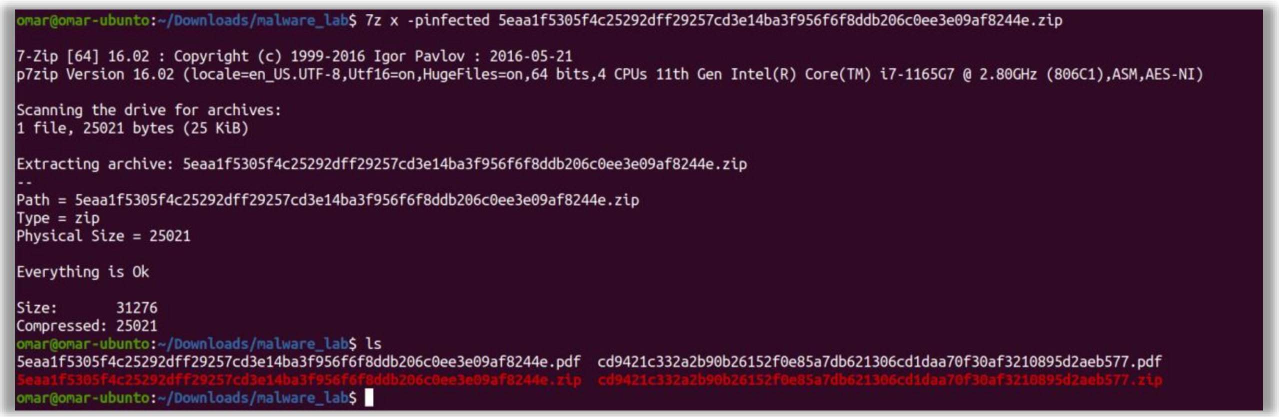
Caution!

You are about to download a malware sample. By clicking on "download", you declare that you have understood what you are doing and that MalwareBazaar can not to be held accountable for any damage caused by downloading this malware sample!

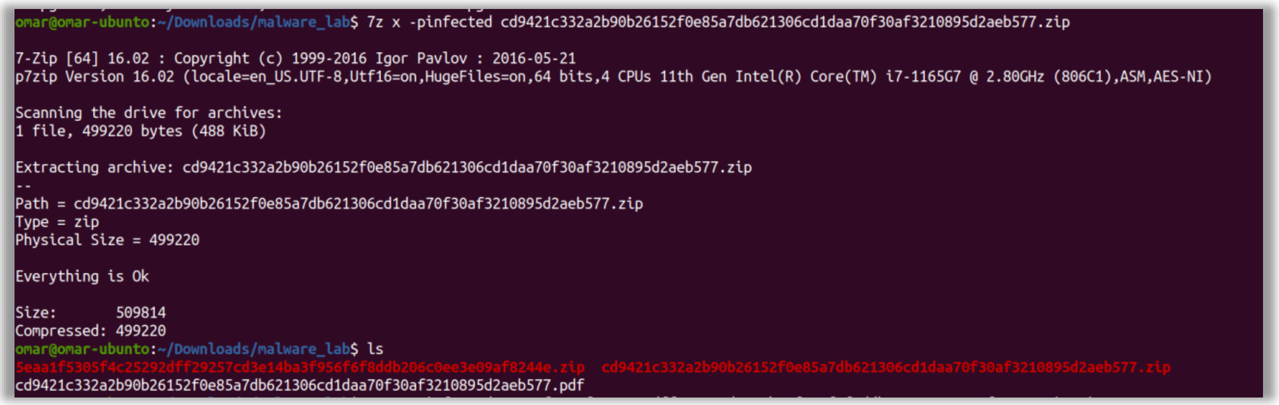
ZIP password: infected

I created a directory inside `Downloads/` to extract the zip file inside , you can skip this process if you want and do as the following :

```
  $ cd Downloads/

  $ 7z x -pinfected
cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.zip

  $ 7z x -pinfected
5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.zip
```



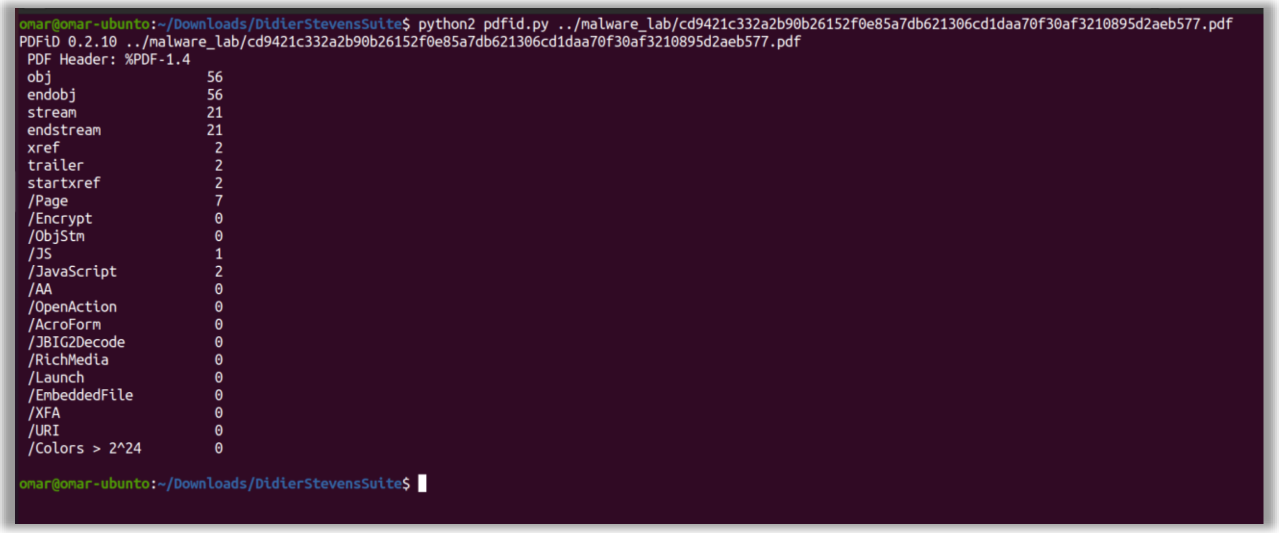
```
omar@omar-ubunto:~/Downloads/malware_lab$ 7z x -pinfected 5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,4 CPUs 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (806C1),ASM,AES-NI)

Scanning the drive for archives:
1 file, 25021 bytes (25 KiB)

Extracting archive: 5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.zip
--
Path = 5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.zip
Type = zip
Physical Size = 25021

Everything is Ok

Size:       31276
Compressed: 25021
omar@omar-ubunto:~/Downloads/malware_lab$ ls
5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.pdf  cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.pdf
5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.zip  cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.zip
omar@omar-ubunto:~/Downloads/malware_lab$
```



```
omar@omar-ubunto:~/Downloads/malware_lab$ 7z x -pinfected cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,4 CPUs 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (806C1),ASM,AES-NI)

Scanning the drive for archives:
1 file, 499220 bytes (488 KiB)

Extracting archive: cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.zip
--
Path = cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.zip
Type = zip
Physical Size = 499220

Everything is Ok

Size:       509814
Compressed: 499220
omar@omar-ubunto:~/Downloads/malware_lab$ ls
5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.zip  cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.zip
cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.pdf
```

Phase 3: Basic Static Analysis with pdfid

```
$ cd DidierStevensSuite/

$ python2 pdfid.py
../malware_lab/cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210
895d2aeb577.pdf
```



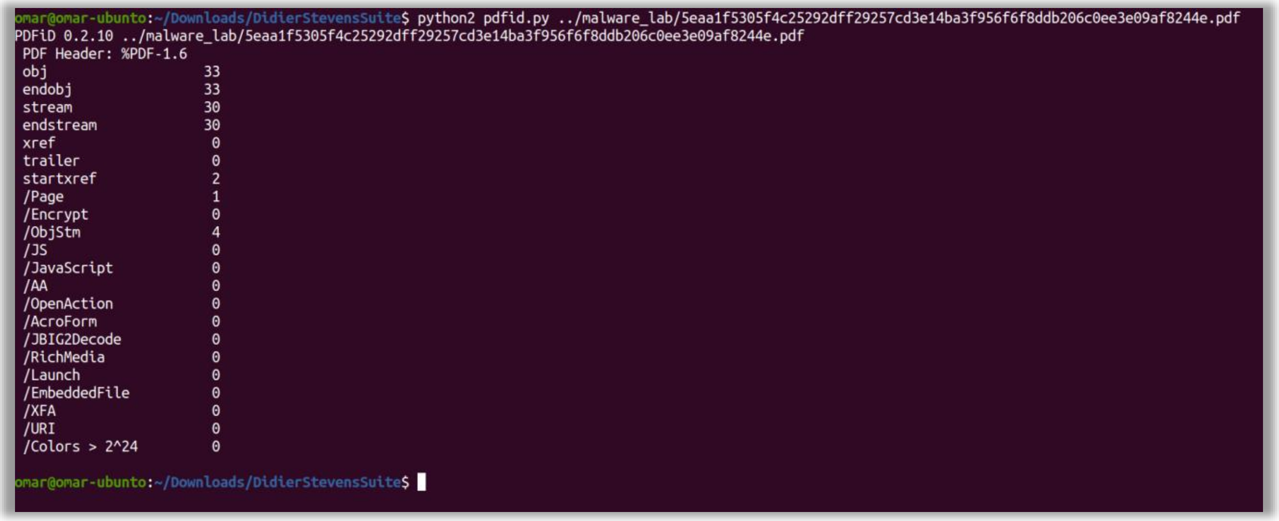
```
omar@omar-ubunto:~/Downloads/DidierStevensSuite$ python2 pdfid.py ../malware_lab/cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.pdf
PDFiD 0.2.10 ../malware_lab/cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210895d2aeb577.pdf
 PDF Header: %PDF-1.4
 obj                   56
 endobj                56
 stream                21
 endstream             21
 xref                   2
 trailer                2
 startxref              2
 /Page                  7
 /Encrypt               0
 /ObjStm                0
 /JS                    1
 /JavaScript            2
 /AA                    0
 /OpenAction            0
 /AcroForm              0
 /JBIG2Decode           0
 /RichMedia             0
 /Launch                0
 /EmbeddedFile          0
 /XFA                   0
 /URI                   0
 /Colors > 2^24         0

omar@omar-ubunto:~/Downloads/DidierStevensSuite$
```

Suspicious Indicators

- **/JS: 1**
- **/JavaScript: 2**
  *This PDF has **JavaScript objects**. That's a red flag because many malicious PDFs use embedded JavaScript for exploits.*

Now onto the 2nd Sample

```
$ cd DidierStevensSuite/

$ python2 pdfid.py
../malware_lab/cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210
895d2aeb577.pdf
```

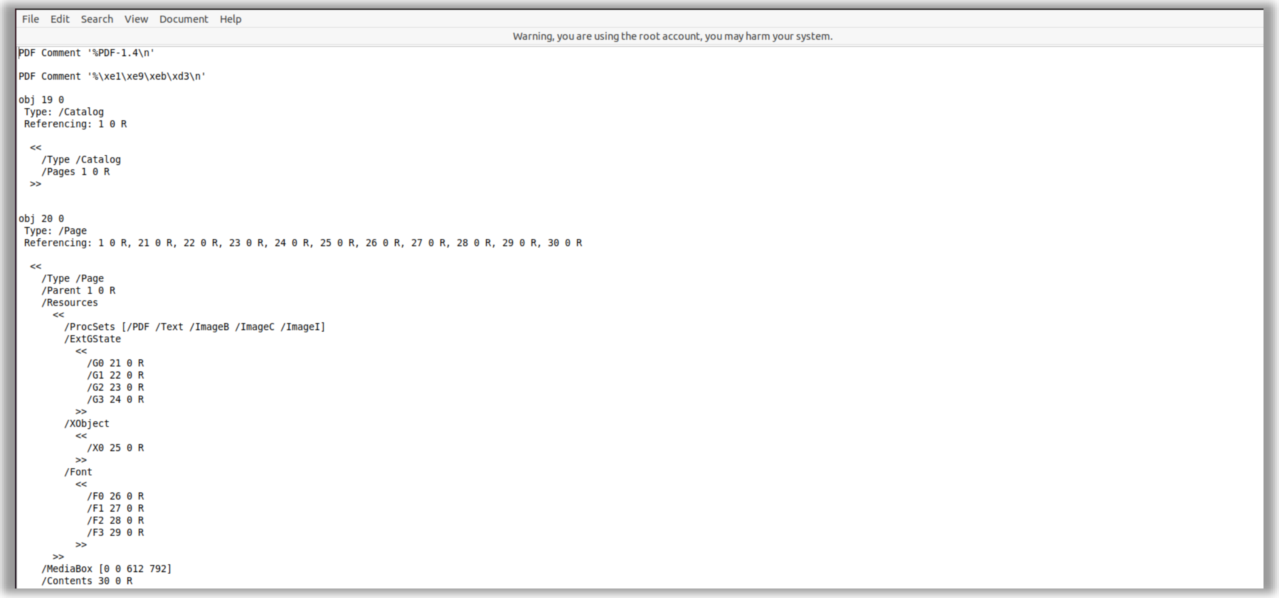


```
omar@omar-ubunto:~/Downloads/DidierStevensSuite$ python2 pdfid.py ../malware_lab/5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.pdf
PDFiD 0.2.10 ../malware_lab/5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.pdf
 PDF Header: %PDF-1.6
 obj                   33
 endobj                33
 stream                30
 endstream             30
 xref                   0
 trailer                0
 startxref              2
 /Page                  1
 /Encrypt               0
 /ObjStm                4
 /JS                    0
 /JavaScript            0
 /AA                    0
 /OpenAction            0
 /AcroForm              0
 /JBIG2Decode           0
 /RichMedia             0
 /Launch                0
 /EmbeddedFile          0
 /XFA                   0
 /URI                   0
 /Colors > 2^24         0

omar@omar-ubunto:~/Downloads/DidierStevensSuite$ 
```

No suspicious Indicator was found

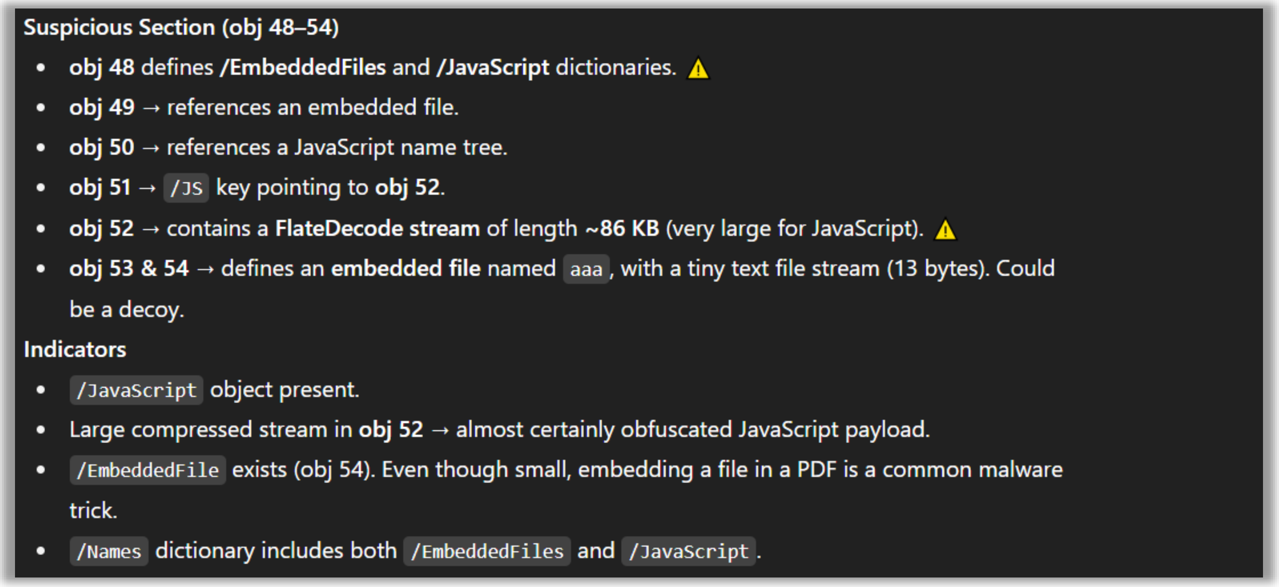
Phase 4: Inspecting Objects pdfparser

```
$ python2 pdf-parser.py
../malware_lab/cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210
895d2aeb577.pdf >> pdfparser_firstSample.txt
```


File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

```
PDF Comment '%PDF-1.4\n'

PDF Comment '%\xe1\xe9\xeb\xd3\n'

obj 19 0
 Type: /Catalog
 Referencing: 1 0 R

  <<
    /Type /Catalog
    /Pages 1 0 R
  >>


obj 20 0
 Type: /Page
 Referencing: 1 0 R, 21 0 R, 22 0 R, 23 0 R, 24 0 R, 25 0 R, 26 0 R, 27 0 R, 28 0 R, 29 0 R, 30 0 R

  <<
    /Type /Page
    /Parent 1 0 R
    /Resources
      <<
        /ProcSets [/PDF /Text /ImageB /ImageC /ImageI]
        /ExtGState
          <<
            /G0 21 0 R
            /G1 22 0 R
            /G2 23 0 R
            /G3 24 0 R
          >>
        /XObject
          <<
            /X0 25 0 R
          >>
        /Font
          <<
            /F0 26 0 R
            /F1 27 0 R
            /F2 28 0 R
            /F3 29 0 R
          >>
      >>
    /MediaBox [0 0 612 792]
    /Contents 30 0 R
```

Instead of analyzing everything manually you can facilitate this process using AI and getting the result.



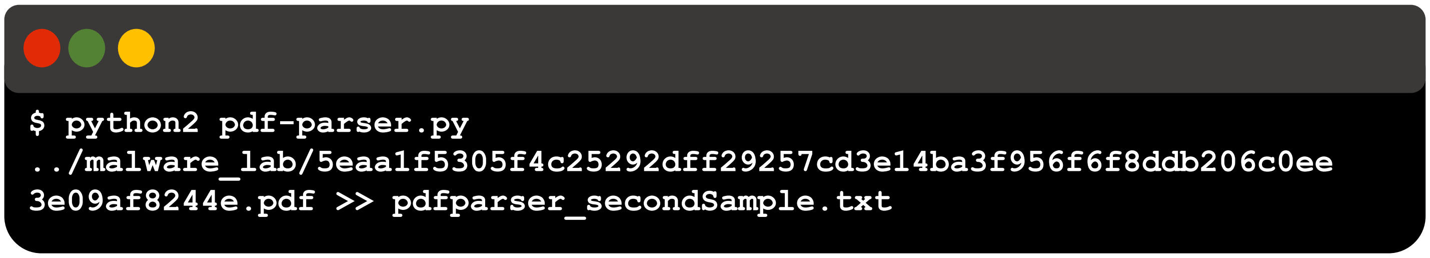
Suspicious Section (obj 48–54)

- **obj 48** defines **/EmbeddedFiles** and **/JavaScript** dictionaries. ⚠️
- **obj 49** → references an embedded file.
- **obj 50** → references a JavaScript name tree.
- **obj 51** → `/JS` key pointing to **obj 52**.
- **obj 52** → contains a **FlateDecode stream** of length **~86 KB** (very large for JavaScript). ⚠️
- **obj 53 & 54** → defines an **embedded file** named `aaa`, with a tiny text file stream (13 bytes). Could be a decoy.

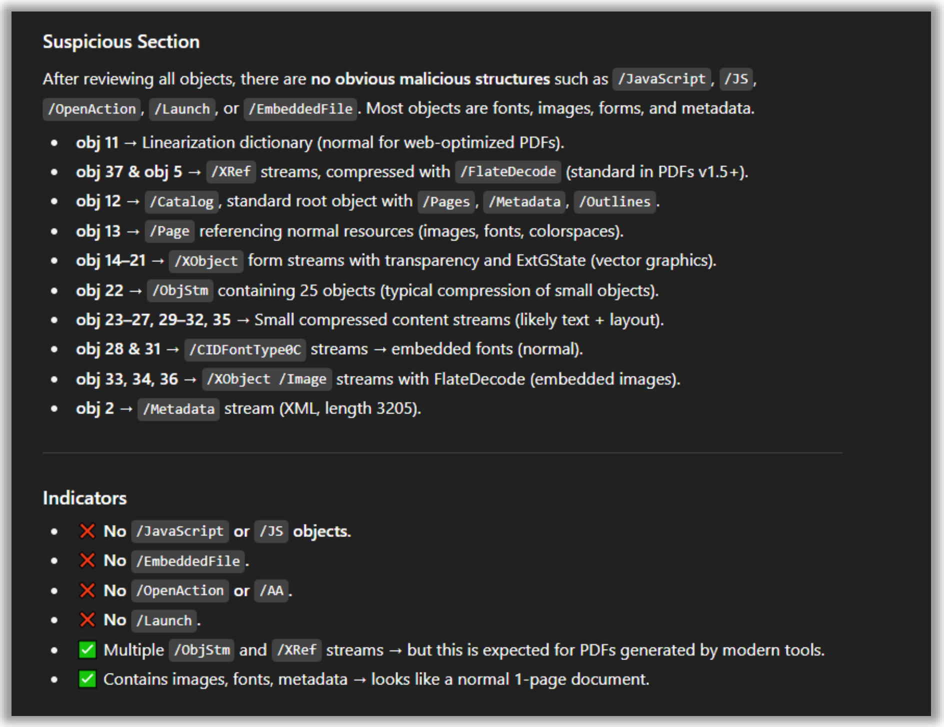
Indicators

- `/JavaScript` object present.
- Large compressed stream in **obj 52** → almost certainly obfuscated JavaScript payload.
- `/EmbeddedFile` exists (obj 54). Even though small, embedding a file in a PDF is a common malware trick.
- `/Names` dictionary includes both `/EmbeddedFiles` and `/JavaScript`.

Same goes in with the second sample



```
$ python2 pdf-parser.py
../malware_lab/5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee
3e09af8244e.pdf >> pdfparser_secondSample.txt
```



Suspicious Section

After reviewing all objects, there are **no obvious malicious structures** such as `/JavaScript`, `/JS`, `/OpenAction`, `/Launch`, or `/EmbeddedFile`. Most objects are fonts, images, forms, and metadata.

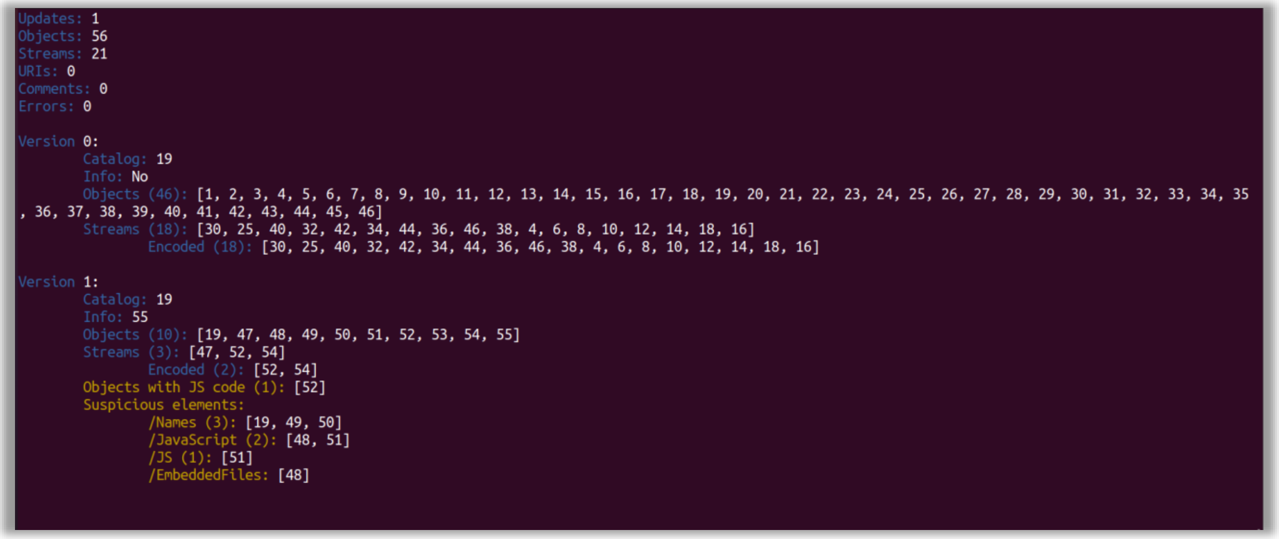
- **obj 11** → Linearization dictionary (normal for web-optimized PDFs).
- **obj 37 & obj 5** → `/XRef` streams, compressed with `/FlateDecode` (standard in PDFs v1.5+).
- **obj 12** → `/Catalog`, standard root object with `/Pages`, `/Metadata`, `/Outlines`.
- **obj 13** → `/Page` referencing normal resources (images, fonts, colorspaces).
- **obj 14–21** → `/XObject` form streams with transparency and ExtGState (vector graphics).
- **obj 22** → `/ObjStm` containing 25 objects (typical compression of small objects).
- **obj 23–27, 29–32, 35** → Small compressed content streams (likely text + layout).
- **obj 28 & 31** → `/CIDFontType0C` streams → embedded fonts (normal).
- **obj 33, 34, 36** → `/XObject /Image` streams with FlateDecode (embedded images).
- **obj 2** → `/Metadata` stream (XML, length 3205).

Indicators

- ❌ **No** `/JavaScript` **or** `/JS` **objects.**
- ❌ **No** `/EmbeddedFile`.
- ❌ **No** `/OpenAction` **or** `/AA`.
- ❌ **No** `/Launch`.
- ✅ Multiple `/ObjStm` and `/XRef` streams → but this is expected for PDFs generated by modern tools.
- ✅ Contains images, fonts, metadata → looks like a normal 1-page document.

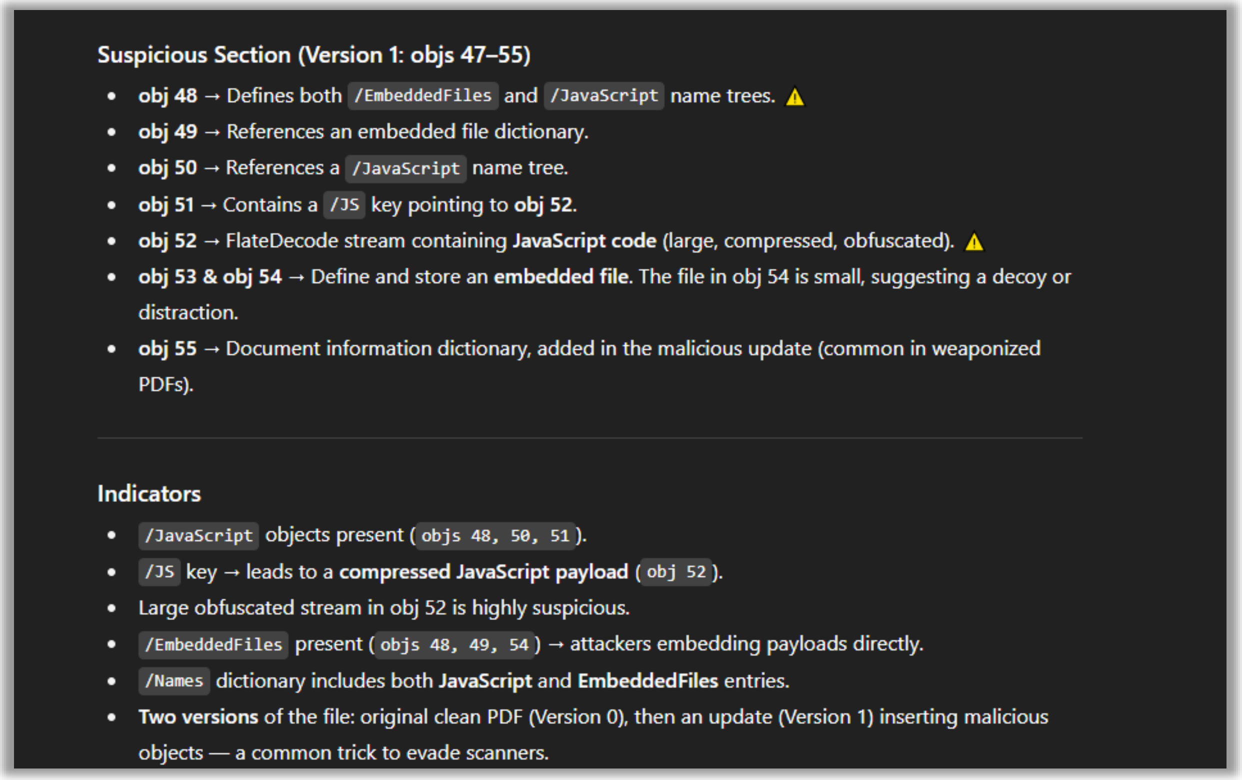
Phase 5: Interactive Deep dive with peepdf

```
$ cd peepdf/

$ python2 peepdf.py
../malware_lab/cd9421c332a2b90b26152f0e85a7db621306cd1daa70f30af3210
895d2aeb577.pdf
```



```
Updates: 1
Objects: 56
Streams: 21
URIs: 0
Comments: 0
Errors: 0

Version 0:
        Catalog: 19
        Info: No
        Objects (46): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35
, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46]
        Streams (18): [30, 25, 40, 32, 42, 34, 44, 36, 46, 38, 4, 6, 8, 10, 12, 14, 18, 16]
                Encoded (18): [30, 25, 40, 32, 42, 34, 44, 36, 46, 38, 4, 6, 8, 10, 12, 14, 18, 16]

Version 1:
        Catalog: 19
        Info: 55
        Objects (10): [19, 47, 48, 49, 50, 51, 52, 53, 54, 55]
        Streams (3): [47, 52, 54]
                Encoded (2): [52, 54]
        Objects with JS code (1): [52]
        Suspicious elements:
                /Names (3): [19, 49, 50]
                /JavaScript (2): [48, 51]
                /JS (1): [51]
                /EmbeddedFiles: [48]
```

FINAL VERDICT : Malicious



Suspicious Section (Version 1: objs 47–55)

- **obj 48** → Defines both `/EmbeddedFiles` and `/JavaScript` name trees. ⚠
- **obj 49** → References an embedded file dictionary.
- **obj 50** → References a `/JavaScript` name tree.
- **obj 51** → Contains a `/JS` key pointing to **obj 52**.
- **obj 52** → FlateDecode stream containing **JavaScript code** (large, compressed, obfuscated). ⚠
- **obj 53 & obj 54** → Define and store an **embedded file**. The file in obj 54 is small, suggesting a decoy or distraction.
- **obj 55** → Document information dictionary, added in the malicious update (common in weaponized PDFs).

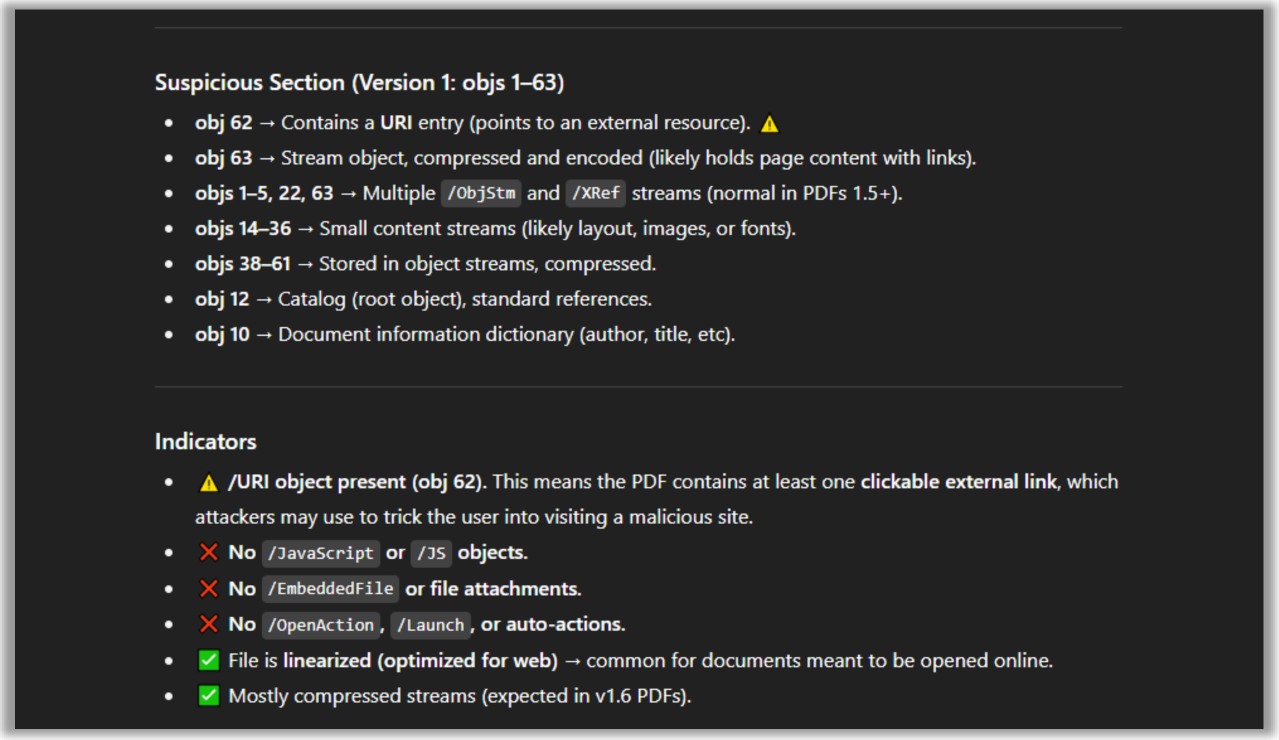
Indicators

- `/JavaScript` objects present (`objs 48, 50, 51`).
- `/JS` key → leads to a **compressed JavaScript payload** (`obj 52`).
- Large obfuscated stream in obj 52 is highly suspicious.
- `/EmbeddedFiles` present (`objs 48, 49, 54`) → attackers embedding payloads directly.
- `/Names` dictionary includes both **JavaScript** and **EmbeddedFiles** entries.
- **Two versions** of the file: original clean PDF (Version 0), then an update (Version 1) inserting malicious objects — a common trick to evade scanners.

Same goes in with the second sample

```
$ python2 peepdf.py
../malware_lab/5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee
3e09af8244e.pdf
```

But wait…



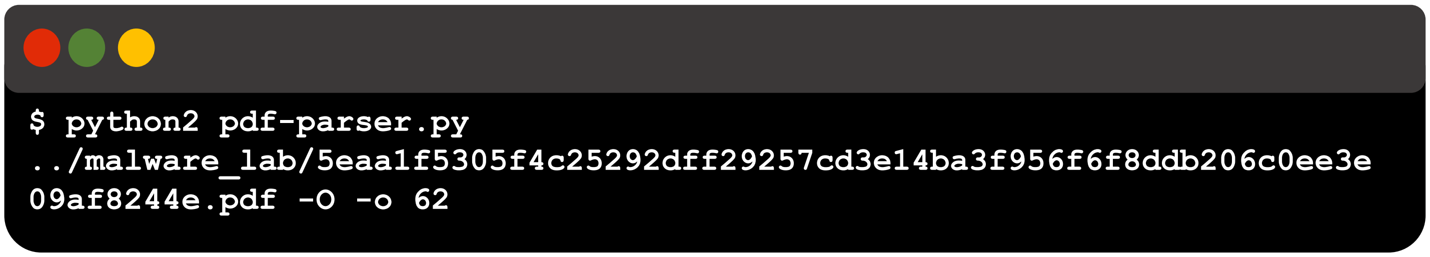
Suspicious Section (Version 1: objs 1–63)

- **obj 62** → Contains a **URI** entry (points to an external resource). ⚠
- **obj 63** → Stream object, compressed and encoded (likely holds page content with links).
- **objs 1–5, 22, 63** → Multiple `/ObjStm` and `/XRef` streams (normal in PDFs 1.5+).
- **objs 14–36** → Small content streams (likely layout, images, or fonts).
- **objs 38–61** → Stored in object streams, compressed.
- **obj 12** → Catalog (root object), standard references.
- **obj 10** → Document information dictionary (author, title, etc).

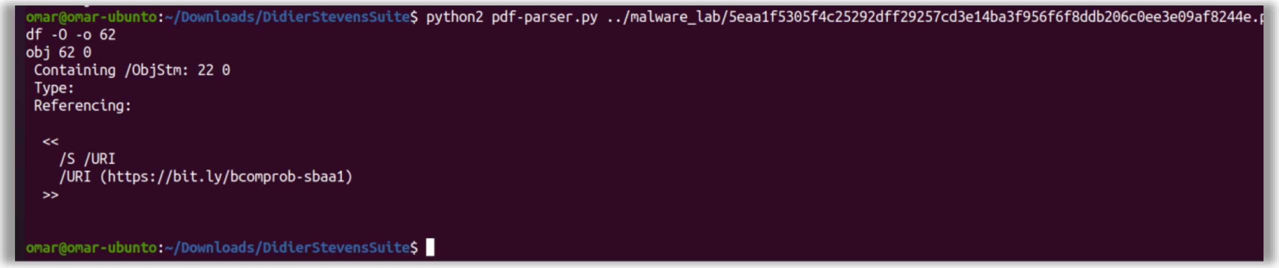
Indicators

- ⚠ **/URI object present (obj 62).** This means the PDF contains at least one **clickable external link**, which attackers may use to trick the user into visiting a malicious site.
- ❌ **No `/JavaScript` or `/JS` objects.**
- ❌ **No `/EmbeddedFile` or file attachments.**
- ❌ **No `/OpenAction`, `/Launch`, or auto-actions.**
- ✅ File is **linearized (optimized for web)** → common for documents meant to be opened online.
- ✅ Mostly compressed streams (expected in v1.6 PDFs).

Obj 62 has an URI entry that was missed by the other tools , lets check this object using pdfparser & see where the url takes us.



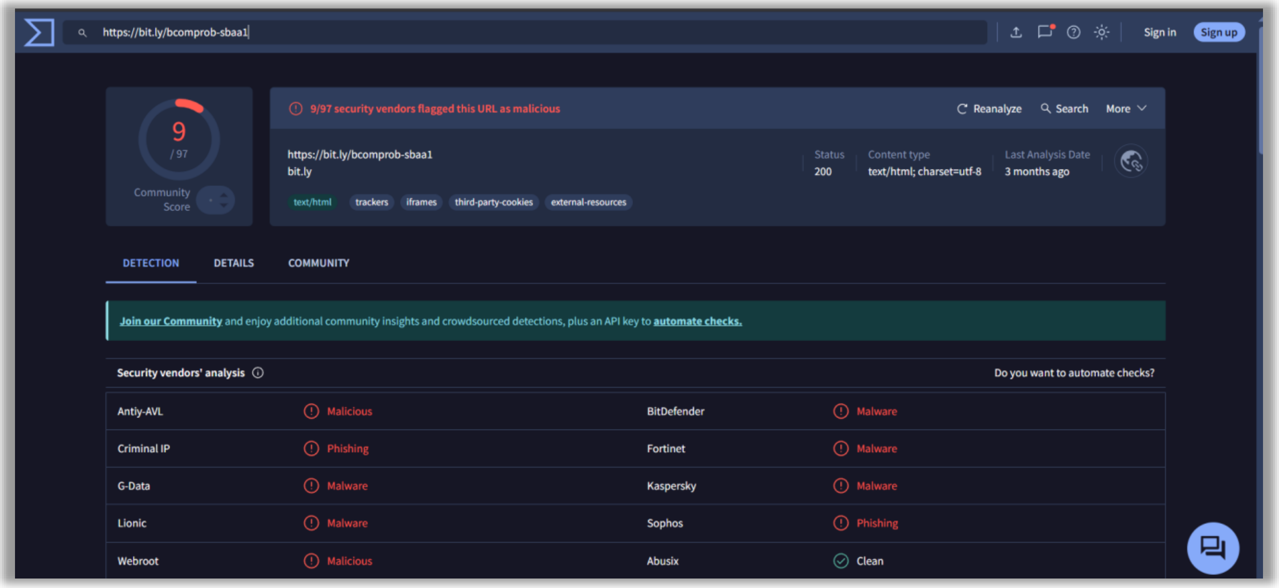
```
$ python2 pdf-parser.py
../malware_lab/5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e
09af8244e.pdf -O -o 62
```



```
omar@omar-ubunto:~/Downloads/DidierStevensSuite$ python2 pdf-parser.py ../malware_lab/5eaa1f5305f4c25292dff29257cd3e14ba3f956f6f8ddb206c0ee3e09af8244e.p
df -O -o 62
obj 62 0
 Containing /ObjStm: 22 0
 Type:
 Referencing:

  <<
    /S /URI
    /URI (https://bit.ly/bcomprob-sbaa1)
  >>


omar@omar-ubunto:~/Downloads/DidierStevensSuite$ 
```

It appears the url was shortened too , lets check it on VirusTotal.

FINAL VERDICT : Malicious



https://bit.ly/bcomprob-sbaa1

9 / 97 Community Score

9/97 security vendors flagged this URL as malicious

https://bit.ly/bcomprob-sbaa1
bit.ly

text/html trackers iframes third-party-cookies external-resources

Status	Content type	Last Analysis Date
200	text/html; charset=utf-8	3 months ago

DETECTION DETAILS COMMUNITY

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

Security vendors' analysis

Do you want to automate checks?

Antiy-AVL	Malicious	BitDefender	Malware
Criminal IP	Phishing	Fortinet	Malware
G-Data	Malware	Kaspersky	Malware
Lionic	Malware	Sophos	Phishing
Webroot	Malicious	Abusix	Clean

Both PDF’s are Malicious