PDF Tools

I produced screencasts for my pdfid and pdf-parser tools, you can find them on Didier Stevens Labs products page.

There are translations of this page, see bottom.

pdf-parser.py

This tool will parse a PDF document to identify the fundamental elements used in the analyzed file. It will not render a PDF document. The code of the parser is quick-and-dirty, I'm not recommending this as text book case for PDF parsers, but it gets the job done.

You can see the parser in action in this screencast.

```
Usage: pdf-parser.py [options] pdf-file

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -s SEARCH, --search=SEARCH
                        string to search in indirect objects (except streams)
  -f, --filter          pass stream object through filters (FlateDecode only)
  -o OBJECT, --object=OBJECT
                        id of indirect object to select (version independent)
  -r REFERENCE, --reference=REFERENCE
                        id of indirect object being referenced (version
                        independent)
  -w, --raw             raw output for data and filters
  -a, --stats           display stats for pdf document
  -t TYPE, --type=TYPE  type of indirect object to select

  pdf-parser V0.2, use it to parse a PDF document
  Source code put in the public domain by Didier Stevens, no Copyright
  Use at your own risk
  https://DidierStevens.com
```

The stats option display statistics of the objects found in the PDF document. Use this to identify PDF documents with unusual/unexpected objects, or to classify PDF documents. For example, I generated statistics for 2 malicious PDF files, and although they were very different in content and size, the statistics were identical, proving that they used the same attack vector and shared the same origin.

The search option searches for a string in indirect objects (not inside the stream of indirect objects). The search is not case-sensitive, and is susceptible to the obfuscation techniques I documented (as I've yet to encounter these obfuscation techniques in the wild, I decided no to resort to canonicalization).

filter option applies the filter(s) to the stream. For the moment, only FlateDecode is supported (e.g. zlib decompression).

The raw option makes pdf-parser output raw data (e.g. not the printable Python representation).

objects outputs the data of the indirect object which ID was specified. This ID is not version dependent. If more than one object have the same ID (disregarding the version), all these objects will be outputted.

reference allows you to select all objects referencing the specified indirect object. This ID is not version dependent.

type allows you to select all objects of a given type. The type is a Name and as such is case-sensitive and must start with a slash-character (/).

Didier Stevens Labs

Visit my company, Didier Stevens Labs

Pages

- About
- Didier Stevens Suite
- Links
- My Software
- Professional
- Programs
  - Ariad
  - Authenticode Tools
  - Binary Tools
  - CASToggle
  - Disitool
  - EICARgen
  - ExtractScripts
  - FileGen
  - HeapLocker
  - Network Appliance Forensic Toolkit
  - Nokia Time Lapse Photography
  - oledump.py
  - OllyStepNSearch
  - PDF Tools
  - Shellcode
  - SpiderMonkey
  - Translate
  - USBVirusScan
  - UserAssist
  - VirusTotal Tools
  - XORSearch & XORStrings
  - YARA Rules

pdf-parser_V0_6_8.zip (https)
MD5: 7702EEA1C6173CB2E91AB88C5013FAF1
SHA256: 3424E6939E79CB597D32F405E2D75B2E42EF7629750D5DFB39927D5C132446EF

make-pdf tools
make-pdf-javascript.py allows one to create a simple PDF document with embedded JavaScript that will execute upon opening of the PDF document. It's essentially glue-code for the mPDF.py module which contains a class with methods to create headers, indirect objects, stream objects, trailers and XREFs.

```
C:\>make-pdf-javascript.py
Usage: make-pdf-javascript.py [options] pdf-file

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -j JAVASCRIPT, --javascript=JAVASCRIPT
                        javascript to embed (default embedded JavaScript is
                        app.alert messagebox)
  -f JAVASCRIPTFILE, --javascriptfile=JAVASCRIPTFILE
                        javascript file to embed (default embedded JavaScript
                        is app.alert messagebox)

  make-pdf-javascript, use it to create a PDF document with embedded JavaScript
that will execute automatically when the document is opened
  Source code put in the public domain by Didier Stevens, no Copyright
  Use at your own risk
  https://DidierStevens.com
```

If you execute it without options, it will generate a PDF document with JavaScript to display a message box (calling app.alert).

To provide your own JavaScript, use option –javascript for a script on the command line, or –javascriptfile for a script contained in a file.

make-pdf-embedded.py creates a PDF file with an embedded file.

```
C:\>make-pdf-embedded.py
Usage: make-pdf-embedded.py [options] file-to-embed pdf-file

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f FILTERS, --filters=FILTERS
                        filters to apply, f for FlateDecode (default), h for
                        ASCIIHexDecode
  -t, --nobinary        don't add the comment for binary format
  -a, --autoopen        open the embedded file automatically when the PDF is
                        opened
  -b, --button          add a "button" to launch the embedded file
  -s, --stego           "hide" the embedded file by replacing /EmbeddedFiles
                        with /Embeddedfiles
  -m MESSAGE, --message=MESSAGE
                        text to display in the PDF document

  tool to create a PDF document with an embedded file
  Source code put in the public domain by Didier Stevens, no Copyright
  Use at your own risk
  https://DidierStevens.com
```

Download:

make-pdf_V0_1_7.zip (https)
MD5: 73DBC0CEC9A425DE3317EB48B9A7EA81
SHA256: DCEA54C2C260152A01278C6262D82255B5944ED616663B83DC158F74F27F509E

pdfid.py
This tool is not a PDF parser, but it will scan a file to look for certain PDF keywords, allowing you to identify PDF documents that contain (for example) JavaScript or execute an action when opened. PDFiD will also handle name obfuscation.

The idea is to use this tool first to triage PDF documents, and then analyze the suspicious ones with my pdf-parser.

- ZIPEncryptFTP
- Public Drafts
- Cisco Tricks
- Reverse Engineering Mentoring
- Screencasts & Videos

Search

Top Posts

- Howto: Make Your Own Cert With OpenSSL on Windows
- PDF Tools
- Using Metasploit On Windows
- Howto: Make Your Own Cert With OpenSSL
- oledump.py

Categories

- .NET
- 010 Editor
- Announcement
- Arduino
- Bash Bunny
- Beta
- bpmtk
- Certification
- Didier Stevens Labs
- Eee PC
- Encryption
- Entertainment
- Fellow Bloggers
- Forensics
- Hacking
- Hardware
- maldoc
- Malware
- My Software
- N800
- Networking
- Nonsense
- nslu2
- OSX
- PDF
- Personal
- Physical Security
- Poll
- Puzzle
- Quickpost
- Reverse Engineering
- RFID

An important design criterium for this program is simplicity. Parsing a PDF document completely requires a very complex program, and hence it is bound to contain many (security) bugs. To avoid the risk of getting exploited, I decided to keep this program very simple (it is even simpler than pdf-parser.py).

```
# pdfid.py notepad.exe
PDFiD 0.0.2 notepad.exe
 Not a PDF document


# pdfid.py test.pdf
PDFiD 0.0.2 test.pdf
 PDF Header: %PDF-1.1
 obj                    7
 endobj                 7
 stream                 1
 endstream              1
 xref                   1
 trailer                1
 startxref              1
 /Page                  1
 /Encrypt               0
 /JS                    1
 /JavaScript            1
 /AA                    0
 /OpenAction            1
 /JBIG2Decode           0


# pdfid.py JBIG2Decode-PoC-names-obfuscation.pdf
PDFiD 0.0.2 JBIG2Decode-PoC-names-obfuscation.pdf
 PDF Header: %PDF-1.1
 obj                    6
 endobj                 6
 stream                 1
 endstream              1
 xref                   1
 trailer                1
 startxref              1
 /Page                  1
 /Encrypt               0
 /JS                    0
 /JavaScript            0
 /AA                    0
 /OpenAction            0
 /JBIG2Decode           1(1)
```

PDFiD will scan a PDF document for a given list of strings and count the occurrences (total and obfuscated) of each word:

- obj
- endobj
- stream
- endstream
- xref
- trailer
- startxref
- /Page
- /Encrypt
- /ObjStm
- /JS
- /JavaScript
- /AA
- /OpenAction
- /JBIG2Decode
- /RichMedia
- /Launch
- /XFA

Almost every PDF documents will contain the first 7 words (obj through startxref), and to a lesser extent stream and endstream. I've found a couple of PDF documents without xref or trailer, but these are rare (BTW, this is not an indication of a malicious PDF document).

/Page gives an indication of the number of pages in the PDF document. Most malicious PDF document have only one page.

/Encrypt indicates that the PDF document has DRM or needs a password to be read.

/ObjStm counts the number of object streams. An object stream is a stream object that can contain other objects, and can therefor be used to obfuscate objects (by using different filters).

/JS and /JavaScript indicate that the PDF document contains JavaScript. Almost all malicious PDF documents that I've found in the wild contain JavaScript (to exploit a JavaScript vulnerability and/or to execute a heap spray). Of course, you can also find JavaScript in PDF documents without malicious intend.

/AA and /OpenAction indicate an automatic action to be performed when the page/document is viewed. All malicious PDF documents with JavaScript I've seen in the wild had an automatic action to launch the JavaScript without user interaction.

The combination of automatic action and JavaScript makes a PDF document very suspicious.

/JBIG2Decode indicates if the PDF document uses JBIG2 compression. This is not necessarily and indication of a malicious PDF document, but requires further investigation.

/RichMedia is for embedded Flash.

/Launch counts launch actions.

/XFA is for XML Forms Architecture.

A number that appears between parentheses after the counter represents the number of obfuscated occurrences. For example, /JBIG2Decode 1(1) tells you that the PDF document contains the name /JBIG2Decode and that it was obfuscated (using hexcodes, e.g. /JBIG#32Decode).

BTW, all the counters can be skewed if the PDF document is saved with incremental updates.

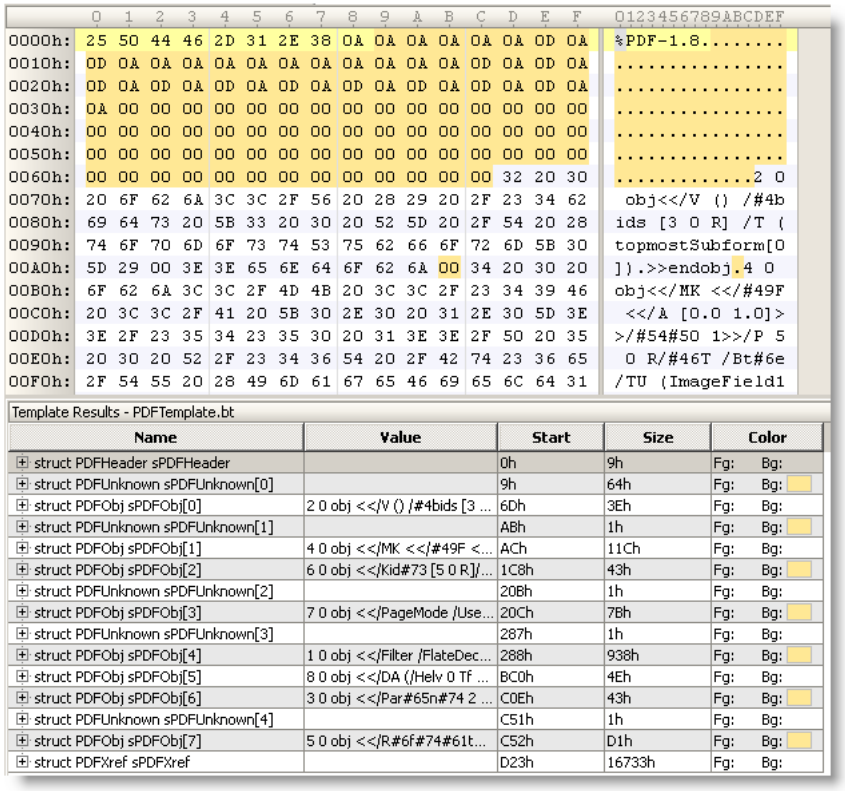
Because PDFiD is just a string scanner (supporting name obfuscation), it will also generate false positives. For example, a simple text file starting with %PDF-1.1 and containing words from the list will also be identified as a PDF document.

Download:

pdfid_v0_2_4.zip (https)
MD5: 36D5554BC881E7E21382ADA1305ED6F4
SHA256: C1DA287C9C06E3158F79CECF9C2E9A7773FC57FC92021F17B79DDD4B1E5DBB2A

PDFTemplate.bt
This is a 010 Editor template for the PDF file format.
It's particularly useful for malformed PDF files, like this example with PDFUnknown structures:

- May 2018
- April 2018
- March 2018
- February 2018
- January 2018
- December 2017
- November 2017
- October 2017
- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017
- March 2017
- February 2017
- January 2017
- December 2016
- November 2016
- October 2016
- September 2016
- August 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- February 2016
- January 2016
- December 2015
- November 2015
- October 2015
- September 2015
- August 2015
- July 2015
- June 2015
- May 2015
- April 2015
- March 2015
- February 2015
- January 2015
- December 2014
- November 2014
- October 2014
- September 2014
- August 2014
- July 2014
- June 2014
- May 2014
- April 2014
- March 2014
- February 2014
- January 2014



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	25	50	44	46	2D	31	2E	38	0A	0A	0A	0A	0A	0A	0D	0A	%PDF-1.8........
0010h:	0D	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0D	0A	0D	0A	....
0020h:	0D	0A	0D	0A	0D	0A	0D	0A	0D	0A	0D	0A	0D	0A	0D	0A	....
0030h:	0A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0040h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0050h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0060h:	00	00	00	00	00	00	00	00	00	00	00	00	00	32	20	30	............2 0
0070h:	20	6F	62	6A	3C	3C	2F	56	20	28	29	20	2F	23	34	62	obj<</V () /#4b
0080h:	69	64	73	20	5B	33	20	30	20	52	5D	20	2F	54	20	28	ids [3 0 R] /T (
0090h:	74	6F	70	6D	6F	73	74	53	75	62	66	6F	72	6D	5B	30	topmostSubform[0
00A0h:	5D	29	00	3E	3E	65	6E	64	6F	62	6A	00	34	20	30	20	]).>>endobj.4 0
00B0h:	6F	62	6A	3C	3C	2F	4D	4B	20	3C	3C	2F	23	34	39	46	obj<</MK <</#49F
00C0h:	20	3C	3C	2F	41	20	5B	30	2E	30	20	31	2E	30	5D	3E	<</A [0.0 1.0]>
00D0h:	3E	2F	23	35	34	23	35	30	20	31	3E	3E	2F	50	20	35	>/#54#50 1>>/P 5
00E0h:	20	30	20	52	2F	23	34	36	54	20	2F	42	74	23	36	65	0 R/#46T /Bt#6e
00F0h:	2F	54	55	20	28	49	6D	61	67	65	46	69	65	6C	64	31	/TU (ImageField1

Template Results - PDFTemplate.bt

Name	Value	Start	Size	Color
struct PDFHeader sPDFHeader		0h	9h	Fg: Bg:
struct PDFUnknown sPDFUnknown[0]		9h	64h	Fg: Bg:
struct PDFObj sPDFObj[0]	2 0 obj <</V () /#4bids [3 ...	6Dh	3Eh	Fg: Bg:
struct PDFUnknown sPDFUnknown[1]		ABh	1h	Fg: Bg:
struct PDFObj sPDFObj[1]	4 0 obj <</MK <</#49F <...	ACh	11Ch	Fg: Bg:
struct PDFObj sPDFObj[2]	6 0 obj <</Kid#73 [5 0 R]/...	1C8h	43h	Fg: Bg:
struct PDFUnknown sPDFUnknown[2]		20Bh	1h	Fg: Bg:
struct PDFObj sPDFObj[3]	7 0 obj <</PageMode /Use...	20Ch	7Bh	Fg: Bg:
struct PDFUnknown sPDFUnknown[3]		287h	1h	Fg: Bg:
struct PDFObj sPDFObj[4]	1 0 obj <</Filter /FlateDec...	288h	938h	Fg: Bg:
struct PDFObj sPDFObj[5]	8 0 obj <</DA (/Helv 0 Tf ...	BC0h	4Eh	Fg: Bg:
struct PDFObj sPDFObj[6]	3 0 obj <</Par#65n#74 2 ...	C0Eh	43h	Fg: Bg:
struct PDFUnknown sPDFUnknown[4]		C51h	1h	Fg: Bg:
struct PDFObj sPDFObj[7]	5 0 obj <</R#6f#74#61t...	C52h	D1h	Fg: Bg:
struct PDFXref sPDFXref		D23h	16733h	Fg: Bg:

Download:

PDFTemplate.zip (https)

MD5: C124200C3317ACA9C17C2AE2579FCFEB

SHA256: 24C4FEAD2CABAD82EC336DDCFD404915E164D7B48FBA7BA1295E12BBAF8EB15D

Translations

- Serbo-Croation

Comments (333)

333 Comments »

1. […] PDF, Quickpost — Didier Stevens @ 11:57 Per request, a more detailed post on how I use my pdf-parser stats […]

   Pingback by Quickpost: Fingerprinting PDF Files « Didier Stevens — Saturday 1 November 2008 @ 11:57

- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- May 2013
- April 2013
- March 2013
- February 2013
- January 2013
- December 2012
- November 2012
- October 2012
- September 2012
- August 2012
- July 2012
- June 2012
- May 2012
- April 2012
- March 2012
- February 2012
- January 2012
- December 2011
- November 2011
- October 2011
- September 2011
- August 2011
- July 2011
- June 2011
- May 2011
- April 2011
- March 2011
- February 2011
- January 2011
- December 2010
- November 2010
- October 2010
- September 2010
- August 2010
- July 2010
- June 2010
- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

2. I'd like to be able to view a scanned pdf file (with handwriting in some fields) and black out boxes on the form whose fields contain info I don't want published.

   Can that kind of thing be automated in a batch so that I don't even have to open the files ?

   That would be cool …

   Can you point me in the right direction ? I'm not looking for you to code, but sending me in the right direction for this would be useful, and it looks like you're cognizant of this kind of information.

   Comment by james — Friday 21 November 2008 @ 15:03

3. @james

   I've no experience with such tools, but you can start to look in the forum of PDF Planet.

   Comment by Didier Stevens — Saturday 22 November 2008 @ 8:46

4. […] download my Python program to generate these PoC PDF documents here, it needs the mPDF module of my PDF-tools. Quickpost info Possibly related posts: (automatically generated)PDF Stream ObjectsUsing DLE In An […]

   Pingback by Quickpost: /JBIG2Decode Essentials « Didier Stevens — Monday 2 March 2009 @ 23:12

5. […] PDF — Didier Stevens @ 7:08 I've developed a new tool to triage PDF documents, PDFiD. It helps you differentiate between PDF documents that could be malicious and those that are most […]

   Pingback by PDFiD « Didier Stevens — Tuesday 31 March 2009 @ 7:08

6. Thanks Didier for sharing _yet_again_ I appreciate it very much!

   Comment by Mitch Impey — Wednesday 1 April 2009 @ 20:17

7. […] developed a new tool to triage PDF documents, PDFiD. It helps you differentiate between PDF documents that could be malicious and those that are most […]

   Pingback by Didier Stevens posted PDFid « Betterworldforus — Friday 3 April 2009 @ 18:34

8. […] will give you statistics of some very basic elements of the PDF language. This helps you decide if a PDF could be malicious or […]

   Pingback by PDFiD On VirusTotal « Didier Stevens — Tuesday 21 April 2009 @ 16:59

9. […] We can go a number of ways with this now, but since the point of all this is the fun we can have with obfuscated scripts in Adobe PDFs we'll run it through PDFiD from Didier Stevens. […]

   Pingback by L1pht Offensive Labs » From Bloodhound to Acrobat JS — Saturday 25 April 2009 @ 2:59

10. Hello — I am using pdf-parser and python for the first time so please excuse my ignorance.

    I'm using Python 3.0.1 on Windows XP. I've copied the pdf-parser.py file into the C:\Python30 directory which contains the python executable. Below is the error I get when attempting to execute your utility:

    ```
    C:\Python30>python.exe pdf-parser.py
    File "pdf-parser.py", line 180
    print 'todo 1: %s' % (self.token[1] + self.token2[1])
    ^
    SyntaxError: invalid syntax
    ```

    ————

- July 2009
- June 2009
- May 2009
- April 2009
- March 2009
- February 2009
- January 2009
- December 2008
- November 2008
- October 2008
- September 2008
- August 2008
- July 2008
- June 2008
- May 2008
- April 2008
- March 2008
- February 2008
- January 2008
- December 2007
- November 2007
- October 2007
- September 2007
- August 2007
- July 2007
- June 2007
- May 2007
- April 2007
- March 2007
- February 2007
- January 2007
- December 2006
- November 2006
- October 2006
- September 2006
- August 2006
- July 2006
- June 2006

May 2018

M	T	W	T	F	S	S
	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			

« Apr

Any ideas? Thanks for your time.

Comment by Mike — Tuesday 5 May 2009 @ 19:02

11. @rMike No problem.

    I've not tested my Python tools on Python 3

    You should remove Python 3.0.1 and install Python 2.6. This will probably fix your problems.

    I've still to decide when I upgrade my tools to Python 3

    Comment by Didier Stevens — Tuesday 5 May 2009 @ 20:59

12. Hello — I'm now using Python 2.6.2. It appears to be working, however I am getting so much output from every pdf I examine, I wonder if I am doing something wrong.

    My syntax is

    pdf-parser.py –search javascript malware.pdf

    The utility spits out hundreds maybe thousands of lines of returned information. At the very top there appears to be useful data, however there are hundreds of lines that look like:

    todo 10: 3 'X1\x1e\x1b\x03\x12\x05X60B

    Am I doing something incorrectly here or is there a way to filter the rest of this data out?

    Thanks for your help.

    PS, I watched your video on pdf-parser and it doesn't have any audio.

    Comment by Mike — Thursday 7 May 2009 @ 20:40

13. Also, just a note I am do have two hyphens in front of search (–search)

    Comment by Mike — Thursday 7 May 2009 @ 20:41

14. No, what you're doing is correct. These todo 10 messages indicate that your PDF document is possibly corrupt. I've seen one such corrupt document before, the malware authors had inserted their payload in the PDF without respecting the PDF syntax.

    I'm sending you an e-mail to see if you can share your sample with me.

    About the video: most of my videos have no sound, it takes me much more time to produce one with audio, and I'm never satisfied with the result.

    Comment by Didier Stevens — Thursday 7 May 2009 @ 21:07

15. […] updated my PDF-tools to support […]

    Pingback by PDF Filter Abbreviations « Didier Stevens — Monday 11 May 2009 @ 0:01

16. Didier,
    Any idea whether/when Filiol, Blonce, & Frayssignes might release their 'PDF StructAzer' tool? My Google research doesn't show anything from them since their presentation at Blackhat Europe last year. According to that paper, they planned to release the tool "very soon", but maybe they've run into some red tape with their employer over it.
    John

    Comment by John McCash — Tuesday 12 May 2009 @ 13:03

17. Hi John,

    No, still no news about a release, but I'll ask him again.

Comment by Didier Stevens — Tuesday 12 May 2009 @ 13:51

18. […] added some new features to my PDF tools to handle malformed PDF […]

    Pingback by Malformed PDF Documents « Didier Stevens — Thursday 14 May 2009 @ 7:55

19. PDF Structazer is available here:
    http://www.esiea-recherche.eu/

    The tool: http://www.esiea-recherche.eu/data/PDF%20Structazer.exe
    The document (PDF): http://www.esiea-recherche.eu/data/PDF%20Structazer%20Short%20User%20Manual.pdf

    Comment by Didier Stevens — Tuesday 26 May 2009 @ 13:01

20. […] PDFiD – https://blog.didierstevens.com/2009/03/31/pdfid/ PDF Tools – https://blog.didierstevens.com/programs/pdf-tools/ Security Justice – http://securityjustice.com/ Exotic Liability – http://exoticliability.ning.com/ […]

    Pingback by SecuraBit Episode 32 PDF Love! | SecuraBit — Wednesday 27 May 2009 @ 14:33

21. […] Finding and detecting Malicious PDF's. Tyler's Snort signature. Didier Steven's fantastic PDF analysis tools. […]

    Pingback by Security Justice » Blog Archive » Security Justice - Episode 13 — Saturday 6 June 2009 @ 2:30

22. Didier,
    Does your script pdf-parser.py works on Encrypted PDF streams ?

    Comment by NeoIsOffline — Tuesday 14 July 2009 @ 12:28

23. No, I didn't add code to decrypt PDF streams. And I'm not sure if I ever will, because then it could also be considered as a copyright infringement tool, and I don't want to deal with that.

    Comment by Didier Stevens — Tuesday 14 July 2009 @ 19:13

24. Im getting errors when running the python script:

    C:\Documents and Settings\yo\Desktop\Tools\pdf>pdf-parser.py
    File "C:\Documents and Settings\yo\Desktop\Tools\pdf\pdf-parser.py", line 198
    print 'todo 1: %s' % (self.token[1] + self.token2[1])
    ^
    SyntaxError: invalid syntax

    Im getting errors when trying to run the script. Im using activepyton 3.1 on windows xp. Launching it from the commandline. Was there any recent modifications which broke the script?

    Thanks

    Comment by Dave — Friday 17 July 2009 @ 16:48

25. @Dave

    That looks the same error as in comment 10. Read comments 10 to 12 for a solution.

    Comment by Didier Stevens — Friday 17 July 2009 @ 17:12

26. wow I feel ignorant, i just breezed through the comments fast. Thanks!

    Comment by Dave — Friday 17 July 2009 @ 18:55

27. @Dave

No problem.

Comment by Didier Stevens — Friday 17 July 2009 @ 18:58

28. Hello Didier,

    This tool is very cool, I am wondering how to integrate this to the ironport (mail filter) so that all attachments like pdf will be scanned and if found that there are openaction or javascript then probably we can filter that. Also, maybe this can be integrated in the Proxy or firewall. Do you have any link to see a topic on integrating this tool.

    Thank you very much, indeed this is very helpful

    Comment by Yaggi — Tuesday 28 July 2009 @ 0:12

29. What interface options does ironport offer? Does it support ICAP? http://en.wikipedia.org/wiki/Internet_Content_Adaptation_Protocol

    Comment by Didier Stevens — Tuesday 28 July 2009 @ 18:55

30. Hello Didier,

    I talked to our notes admin and this can somehow be integrated in the sendmail. He ask me some technicalities on the integration. maybe you can also provide some link. HE told me ironport will not be used.

    Where can we possibly put this script for internet filter, is it in the proxy server or possibly to content filter boxes like from 8e6 technologies? Im hoping to finish this integration so we can maximize your script.

    Comment by Yaggi — Wednesday 29 July 2009 @ 1:30

31. @Yaggi

    How do you interface with other scanner, like AV software?

    Comment by Didier Stevens — Friday 31 July 2009 @ 12:48

32. […] Taking a quick look at the rest of the file it was clear that it is just a "simple" exploit using obfuscated javascript. So I extracted the scripts with the pdf-tools from Didier Stevens. […]

    Pingback by hong10.net » Blog Archive » Analyzing malicious PDF Documents — Monday 3 August 2009 @ 21:28

33. I was looking over your code in the pdf parser, and trying very very hard to get it to extract something from a pdf. I have tried using pdfs I've been given, as well as going directly into the files to try to mess things up, and I have yet to see the code go into the cPDFElementMalformed method. What kinds of data (objects?) do you expect to fall into this category?

    Thank you -ld

    Comment by Delucci — Tuesday 4 August 2009 @ 14:16

34. @Delucci

    It's for sometinh like this:

    %PDF-1.4

    1 0 obj
    <>
    endobj

    unexpected

    2 0 obj
    <>

endobj

Comment by Didier Stevens — Tuesday 4 August 2009 @ 15:33

35. […] the PDF analysis, I used the excellent PDF-Tools from Didier Stevens that can be located here. The main python script that was used was pdf-parser seen […]

    Pingback by PDF Malware Analysis – Part 1 | isolated-threat — Tuesday 18 August 2009 @ 21:02

36. […] can download PDFiD here. Leave a […]

    Pingback by Update: PDFiD Version 0.0.9 to Detect Another Adobe 0Day « Didier Stevens — Tuesday 13 October 2009 @ 21:24

37. […] Here is a set of tools that can embed Javascript into a PDF. […]

    Pingback by Adobe Reader 0-day exploit FINALLY fixed. | Invariable Truth — Sunday 18 October 2009 @ 9:35

38. Hi,

    Great set of tools, I have noticed while checking PDF files for embedded links (i.e. URI tags), some PDFs contain hyperlinks but do not contain the URI tag, is there an alternative method of embedding hyperlinks, should I be searching for some other keyword?

    Comment by Tye — Monday 19 October 2009 @ 15:41

39. Check if the URLs you see are in the metadata.

    Comment by Didier Stevens — Tuesday 20 October 2009 @ 16:48

40. Hi,

    I've been using your tool to decode the malicious PDF file that I have. It's using /Filter /ASCIIHexDecode /FlateDecode.

    I used the following command:
    pdf-parser.py -f -w malpdf.1 > mal.1

    The resulting file didn't show any JavaScript code, instead it showed "ASCIIHexDecode decompress failed".

    Wepawet is able to decode it though (http://wepawet.cs.ucsb.edu/view.php?hash=c9aad1ecee10ddcf1985ae4961e18fbf&type=js).

    Are my parameters for the tool incorrect? Or doesn't the tool support this?

    Thanks in advance.

    Comment by anima — Friday 30 October 2009 @ 5:40

41. @anima

    I've e-mailed you a request for the sample.

    Comment by Didier Stevens — Thursday 5 November 2009 @ 17:45

42. […] my method: Use the tools from here. First of all pdfid can tell you if a pdf has Javascript included as well as autorun functionality […]

    Pingback by PDF file check question - Remote Exploit Forums — Sunday 6 December 2009 @ 22:25

43. […] first tool we'll be using is pdf-parser.py from the PDF Tools suite. This script will search through a PDF file's sections, display raw data in the sections, […]

    Pingback by Reversing the Adobe 0-day APSA09-07 Exploit – Part 1 | Missouri S&T ACM SIG-SEC|Reversing — Wednesday 16 December 2009 @ 3:55

44. […] Countermeasures _____ Either you're part of the problem or you're part of the solution or you're just part of the landscape. […]

Pingback by Using-an-adobe-exploit-in-a-email-attack - Remote Exploit Forums — Tuesday 22 December 2009 @ 15:46

45. […] pdf-parser.py https://blog.didierstevens.com/programs/pdf-tools/ Lets decompression some of the zlib compressed code inside of the PDF and send the raw output to a […]

    Pingback by Reversing MerryChristmas.pdf - Sp8sCorp — Thursday 31 December 2009 @ 5:01

46. […] pdf-parser.py or PDF Structazer to analyze PDF files […]

    Pingback by Can You Trust That File? « Aggressive Virus Defense — Thursday 31 December 2009 @ 22:40

47. […] […]

    Pingback by How to encode a PDF payload in metasploit? - Remote Exploit Forums — Tuesday 5 January 2010 @ 14:10

48. […] thanks to Didier Stevens for his free PDF tools and for providing some […]

    Pingback by PDF file loader to extract and analyse shellcode « c0llateral Blog — Wednesday 6 January 2010 @ 23:19

49. Hey Didier,
    Thanks for excellent tool and great PDF analysis blog. I enjoyed every minute and in addition I have become much more paranoid when it comes to carelessly downloading tons of PDF material. Now I run all my PDFs through your "pdfid" tool, if I have downloaded anything from a suspicious site…

    But I can't help thinking that this should be implemented as an automatic plug-in/add-on to Firefox? You know, when you click on PDFs, they usually automatically open in the browser, which is nice if it was safe. But in the cyber-war era of today it is simply very bad, at it's best!

    Comment by E:V:A — Saturday 16 January 2010 @ 18:40

50. I'm looking into this, but the problem is to prevent the download PDF from being opened after it's downloaded and before it's scanned. I talked to the developer of the Fireclam add-on and he has the same issue.

    Comment by Didier Stevens — Tuesday 19 January 2010 @ 9:29

51. Only thing i get are syntax error!

    C:\pdfid_v0_0_10>pdfid.py
    File "C:\pdfid_v0_0_10\pdfid.py", line 271
    print '/%s -> /%s' % (HexcodeName2String(wordExact), wordExactSwapped)

    Comment by sheldor — Sunday 31 January 2010 @ 22:18

52. Are you using Python 3? Haven't tested PDFiD on Python 3. Use Python 2.

    Comment by Didier Stevens — Sunday 31 January 2010 @ 22:20

53. got it!! just read the comments!

    Comment by sheldor — Sunday 31 January 2010 @ 22:35

54. whow just noticed your quick response! thank you didier! great tool!

    Comment by sheldor — Sunday 31 January 2010 @ 22:36

55. Is the File Size limited? Everytime i scan larger PDF files i get exceptions like this:

    Error occured
    Traceback (most recent call last):
    File "C:\PDFtools\pdfid.py", line 363, in PDFiD

(bytesHeader, pdfHeader) = FindPDFHeaderRelaxed(oBinaryFile)
File "C:\PDFtools\pdfid.py", line 218, in FindPDFHeaderRelaxed
bytes = oBinaryFile.bytes(1024)
File "C:\PDFtools\pdfid.py", line 70, in bytes
inbytes = self.infile.read(size – len(self.ungetted))
IOError: [Errno 9] Bad file descriptor

Comment by sheldor — Monday 8 February 2010 @ 13:50

56. @sheldor: No, I didn't code an explicit file size limit. I tried on PDF files up to 41MB without problems. How large is your PDF file?

    Comment by Didier Stevens — Monday 8 February 2010 @ 14:55

57. I have this issue with PDFs 20MB and up! Well,.. then there must be another reason! Still can't figure it out!
    Anyhow, thank you!

    Comment by sheldor — Monday 8 February 2010 @ 17:27

58. @sheldor: If you can point me to an online PDF document that causes the problem you experience, I'll take a look at it.

    Comment by Didier Stevens — Monday 8 February 2010 @ 20:24

59. […] Didier Stevens has provided a fantastic resource and tools for analyzing PDF files. Some of these resources have been incorporated into VirusTotal. Didier Stevens: https://blog.didierstevens.com/programs/pdf-tools/ […]

    Pingback by PDF Exploitation & Forensic Resources « MadMark's Blog — Tuesday 16 February 2010 @ 19:00

60. […] we see in Pyew? The output of PDFId (a great tool by Didier Stevens) is shown as well as the hexadecimal output of the first block (512 […]

    Pingback by Unintended Results » Blog Archive » Analyzing PDF exploits with Pyew — Sunday 21 February 2010 @ 14:50

61. Very cool man, I tried to use PDF tools to unwind a drive-by ZeuS pdf infection. Unfortunately, it gave me some problems because I was using a newer version of Python (and it looks like the El Fiesta Exploit kit might use some kind of different zLib encoding to compress its payloads). Good stuff though!
    http://www.mdl4.com/2010/02/28/reverse-engineering-zeus/

    Comment by mdl4 — Tuesday 2 March 2010 @ 11:15

62. […] https://blog.didierstevens.com/programs/pdf-tools/ […]

    Pingback by PDF Malware Analysis Tools | Tahir's Security Blog — Wednesday 31 March 2010 @ 18:11

63. […] For the PDF analysis, I used the excellent PDF-Tools from Didier Stevens that can be located here. The main python script that was used was pdf-parser and pdfid seen […]

    Pingback by PDF Launch Command without javascript - isolated-threat — Thursday 1 April 2010 @ 10:47

64. […] the PDF with Didier Steven's pdfid.py showed that there was an OpenAction in the PDF, but no JavaScript. Interesting. Using […]

    Pingback by /Launch Malicious PDF | Portable Digital Video Recorder — Tuesday 27 April 2010 @ 22:48

65. […] @ 10:11 Now that malicious PDFs using the /Launch action become more prevalent, I release a new PDFiD version to detect (and disarm) the /Launch […]

    Pingback by Update: PDFiD Version 0.0.11 to Detect /Launch « Didier Stevens — Thursday 29 April 2010 @ 10:11

66. […] الباحث ديدر ستفينز أداة جديدة (pdfid.py)، تساعد الكشف عمّا إذا كان ملف pdf يحتوي […]

    Pingback by قبل تشغيلها | مجتمع الحماية العربي اطلاق أداة جديدة تقوم بالكشف على ملفات pdf — Thursday 29 April 2010 @ 18:48

67. […] PDFiD v0.0.11 – didierstevens.com I release a new PDFiD version to detect (and disarm) the /Launch action. […]

    Pingback by Week 17 in Review – 2010 | Portable Digital Video Recorder — Monday 3 May 2010 @ 6:41

68. […] PDFiD v0.0.11 – didierstevens.com I release a new PDFiD version to detect (and disarm) the /Launch action. […]

    Pingback by Week 17 in Review – 2010 | Infosec Events — Tuesday 4 May 2010 @ 9:40

69. I have a large volume of pdfs coming soon from a vendor, does pdfid.py handle compressed (gzip, bzip2, zip) files? If so, how. If not is it something that can be worked around or accomplished with another program?

    BTW
    really appreciate your work, your blog and website have been a treasure trove of information.

    Comment by Johnny — Tuesday 4 May 2010 @ 16:34

70. @Johnny No, but it has an option to scan all files in a folder. Unzip all PDFs to a folder and use that option.

    Comment by Didier Stevens — Tuesday 4 May 2010 @ 20:57

71. […] non fidate può essere utile eseguire un'analisi automatizzata ricorrendo al tool pdfid.py di Didier Stevens. Si tratta di uno script, funzionante su Windows, Linux e qualsiasi sistema che […]

    Pingback by Analizzare e "disinfettare" file PDF con pdfid — Tuesday 4 May 2010 @ 21:13

72. […] fonctions suspectes cachées dans le PDF (à savoir exécution de Javascript et d'exécutables) : pdfid et pdf-parser. Avant de découvrir les fonctionnalités de ces deux outils, il est important de connaitre la […]

    Pingback by Les outils d'analyse de PDF « Elevenses blog — Monday 10 May 2010 @ 14:45

73. […] Il n'a pas fallu longtemps pour que ce PoC (Proof Of Concept) ne soit utilisé par dans des PDF malicieux, permettant ainsi d'installer un trojan sur la machine cible. Didier Stevens a développé deux scripts Python permettant d'analyser les PDF pour y découvrir d'éventuelles fonctions suspectes cachées dans le PDF (entre autres exécution de Javascript et d'exécutables) : pdfid et pdf-parser. […]

    Pingback by Les outils d'analyse de PDF « ELEVENSES BLOG — Thursday 27 May 2010 @ 15:41

74. […] }; "> — Classificat com a: Eines — Comentari (0) — Lectures: 2130 abril 2010PDFID.py és una eina que analitza un fitxer PDF i mostra les característiques de les que fa ús. Per […]

    Pingback by Eina: PDFID.py | L'home dibuixat — Saturday 5 June 2010 @ 20:01

75. […] you used my pdf-parser, you've also encountered a problem. The objects lack the endobj keyword. A simple solution: […]

    Pingback by Solving the Win7 Puzzle « Didier Stevens — Friday 25 June 2010 @ 9:39

76. With PDFiD, I've noticed I get a lot of false positives on the /JS and /AA tags, since in most cases (that I've looked at) they seem to be simply text in a compressed image or something similar. I haven't seen a /JS used on it's own for Javascript, but it does seem that if there is a /JS then there is also a /S/JavaScript to go with it.

    Is this always the case, or just in the samples I've looked at so far (same applies for AA)? Finding the text JavaScript is much less likely to lead to a false positive than JS.

    Comment by Russell — Monday 28 June 2010 @ 23:28

77. @Russell Good observation, I almost always see /JavaScript together with /JS. I've seen some cases without /JavaScript, but it looks like these were non-functional.

    Comment by Didier Stevens — Tuesday 29 June 2010 @ 9:01

78. This is a complementing post. Work you have done is adorable I liked it how do you get this all in mind??? 🙂 but anyways I found this great and keep going. keep making us explore each security aspect.

thanks
Sushant

Comment by Sushant — Friday 9 July 2010 @ 10:50

79. […] plików PDF: Didier's PDF tools, Origami framework, Jsunpack-n, […]

    Pingback by » REMnux — programy do analizy złośliwego oprogramowania -- Niebezpiecznik.pl -- — Monday 12 July 2010 @ 9:15

80. […] any known viruses, when run through a total of 32 anti-virus programs. Processing the file with PDFiD reveals that the file contains no JavaScript objects, but it does contain a single JS object. […]

    Pingback by Al-Qaeda Magazine is Cupcake Recipe Book | Public Intelligence — Monday 12 July 2010 @ 21:18

81. Possible bug: PDFiD fails sometime in cPDFEOF when using –extra option for entropy, stating cntCharsAfterLastEOF doesn't exist. Defining it in init seems to fix the issue.

    Other Notes: Is it possible to use pdf-parser to parse pdf-parser output? For example, I can see a use of this when using pdf-parser to obtain contents of object streams, but then it would be nice if it were possible to use pdf-parser on THAT output to display all Launch commands, for example (similar to piping into PDFiD, but actually seeing the contents instead of just the count). Then again, object stream structure is a bit different so perhaps that's why it doesn't play nice. I haven't figured it out yet…

    Comment by Russell — Thursday 15 July 2010 @ 23:21

82. […] PDF analysis: Didier's PDF tools, Origami framework, Jsunpack-n, […]

    Pingback by Malware Analysis Tools Set Up for Linux « Wikihead's Blog — Saturday 17 July 2010 @ 9:31

83. @Russell Thanks for the feedback. I've had similar reports, and defining it in the init fixes the issue, but I also would like to understand the bug. Can you share a sample?

    Comment by Didier Stevens — Monday 19 July 2010 @ 11:30

84. […] and Flare. Furthermore, it contains several applications for analyzing malicious PDFs, such as the Didier Steven's analysis tools. The OS also provides a lot of tools for de-obfucating JavaScript, including Rhino […]

    Pingback by New Linux OS REMnux Designed For Reverse Engineering Malware « The FORWARD project blog — Tuesday 20 July 2010 @ 10:36

85. […] Il n'a pas fallu longtemps pour que ce PoC (Proof Of Concept) ne soit utilisé par dans des PDF malicieux, permettant ainsi d'installer un trojan sur la machine cible. Didier Stevens a développé deux scripts Python permettant d'analyser les PDF pour y découvrir d'éventuelles fonctions suspectes cachées dans le PDF (entre autres exécution de Javascript et d'exécutables) : pdfid et pdf-parser. […]

    Pingback by Les outils d'analyse de « ELEVENSES BLOG — Monday 2 August 2010 @ 8:29

86. […] I highly recommend any security conscious sysadmins add this tool to their toolkit, as the number of PDF exploits are likely to continue rising for the forseeable future. PDFiD can be downloaded from Didier Stevens website at https://blog.didierstevens.com/programs/pdf-tools. […]

    Pingback by PDFiD: Analyzing suspicious PDFs « Life as a cmddot — Tuesday 3 August 2010 @ 7:03

87. […] Font Format) stream that looked suspicious enough for us to decode it (thanks to the excellent pdf-parser tool from Didier Stevens). In the now clear-text stream, we could identify at least one manifest […]

    Pingback by iPhone 4 / iPad: The Keys Out Of Prison | Fortinet Security Blog — Thursday 5 August 2010 @ 8:27

88. […] – Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD: pdf-parser and […]

    Pingback by Security tools « Eikonal Blog — Monday 9 August 2010 @ 14:29

89. […] i PDF-tools di Didier Stevens si riesce ad analizzare la struttura dei file PDF, anche se tutti risultano […]

Pingback by Honeynet Project: Challenge 3/2010 (II parte) « Il non-blog di Mario Pascucci — Thursday 19 August 2010 @ 3:04

90. Is there a licensing agreement with using pdfid or pdf-parser? Can it be used as part of software that will be sold?

    Comment by Jon — Thursday 2 September 2010 @ 14:59

91. […] Here is a PDF template for the 010 Editor. It's particularly useful for malformed PDF files, like this example with PDFUnknown structures: […]

    Pingback by PDFTemplate « Didier Stevens — Friday 3 September 2010 @ 10:36

92. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and mPDF) […]

    Pingback by Python tools for penetration testers | Secondary Logic – There is always a theory !!! — Saturday 4 September 2010 @ 8:08

93. @Jon Can't contact you, you didn't provide an e-mail address.

    Comment by Didier Stevens — Sunday 5 September 2010 @ 21:31

94. […] & pdftools – Two frameworks for analysing malicious PDF […]

    Pingback by Mercury – Live Honeypot DVD « Infosanity's Blog — Wednesday 22 September 2010 @ 14:26

95. […] https://blog.didierstevens.com/programs/pdf-tools/ […]

    Pingback by BruCON 2010 : Day 0×2 | Peter Van Eeckhoutte's Blog — Saturday 25 September 2010 @ 20:54

96. […] Font Format) stream that looked suspicious enough for us to decode it (thanks to the excellent pdf-parser tool from Didier Stevens). In the now clear-text stream, we could identify at least one manifest […]

    Pingback by » iPhone 4 / iPad: The Keys Out Of Prison — Saturday 25 September 2010 @ 22:53

97. […] Analyse verdächtiger Dateien hält Stevens verschiedene selbstentwickelte Tools auf seiner Website vorrätig, deren Nutzung für technisch unversierte Lesefreunde allerdings wenig praktikabel ist. Weil schon […]

    Pingback by Schadhafte pdf-Dateien identifizieren » Software » lesen.net — Monday 27 September 2010 @ 17:55

98. […] Didier Stevens' PDF tools Over the weekend, I was reading Didier Stevens' chapter on malicious PDF analysis and I have to give credit to him to break down the technical part of a PDF into something simple and easy to understand (er … maybe I am the only one who is coming to term with PDF for the first time). Reading the article brought me to his PDF-tools. pdfid and pdf-parser is definitely a must try if you really want to get your hands-on on PDF analysis. […]

    Pingback by Hunger 4 Knowledge #10 « David Koepi — Sunday 3 October 2010 @ 1:28

99. […] and wonder where to start. Get a Linux distro, install Python, and use Didier Stevens PDF parser [Didier Stevens]. This is a script that will structure all the objects for you, making them more readable. This is […]

    Pingback by Analyzing malicious PDFs — Monday 11 October 2010 @ 19:03

100. […] and dump the zipped sections of a PDF file. In my opinion, the best are Didier Steven's PDF Tools. Unfortunately, in this case, none of them worked for me, so I had to do it manually. I selected […]

    Pingback by Reverse engineering a Facebook ZeuS infection — Monday 25 October 2010 @ 2:24

101. […] was about malicious PDF analysis, given by "Mr PDF" himself, Didier Stevens. Using his toolbox, several malicious PDF files were analyzed with a growing complexity. Very interesting and this […]

    Pingback by Hack.lu Day #1 Wrap-up « /dev/random — Wednesday 27 October 2010 @ 21:51

102. […] pdfid.py and pdf-parser.py. Get them from from Didier Stevens PDF Tools page. […]

Pingback by Analysing a Malicious PDF Document — Saturday 6 November 2010 @ 12:08

103. […] Download: click here […]

Pingback by Malware Analysis: Handy tools for analysing PDF files « Brainfold's blog — Tuesday 16 November 2010 @ 3:00

104. […] I ran pdf-parser.py against the pdf file. The output indicated that there were 2 "interesting" objects […]

Pingback by Malicious pdf analysis : from price.zip to flashplayer.exe | Peter Van Eeckhoutte's Blog — Thursday 18 November 2010 @ 13:50

105. Didier,
Is there a way to embed a .exe in a pdf and have it automatically execute when the pdf is opened? I have tried to use your .py tool but it does not run the .exe after being opened.
Thanks,
Willie

Comment by Willie — Saturday 20 November 2010 @ 6:37

106. @Willie That's normal, Adobe Reader doesn't allow you to extract executable files. I found one way to deliver executable files: https://blog.didierstevens.com/2010/03/29/escape-from-pdf/
But Adobe has updated their reader to prevent this /Launch action.

Comment by Didier Stevens — Saturday 20 November 2010 @ 8:49

107. […] obvious choice were the pdftools from Didier Stevens. What […]

Pingback by Malware PDF. Analysis of a very simple sample. | Brundle Lab — Tuesday 23 November 2010 @ 18:35

108. […] Didier's own pdf-parser.py, the PDF's meta information for the creation date is as […]

Pingback by Praetorian Prefect | The Anonymous PR Guy and a Greece Connection — Sunday 12 December 2010 @ 0:57

109. […] Il n'a pas fallu longtemps pour que ce PoC (Proof Of Concept) ne soit utilisé par dans des PDF malicieux, permettant ainsi d'installer un trojan sur la machine cible. Didier Stevens a développé deux scripts Python permettant d'analyser les PDF pour y découvrir d'éventuelles fonctions suspectes cachées dans le PDF (entre autres exécution de Javascript et d'exécutables) : pdfid et pdf-parser. […]

Pingback by Secur-IT — Thursday 6 January 2011 @ 13:28

110. […] second Didier Steven's PDF Tools. PDF Tools includes pdf-parser.py, make-pdf-javascript.py, and pdfid.py. Pdf-parser and pdfid are […]

Pingback by Tools — Saturday 29 January 2011 @ 18:34

111. […] PDF-Parser (https://blog.didierstevens.com/programs/pdf-tools/) […]

Pingback by Attributes of a Zero Dollar Malware Analysis Environment « SecAnalysis — Tuesday 8 February 2011 @ 3:07

112. hi!
i tried using your make-pdf-javascript.py. i gave it a javascript file which executes notepad, but though it got embedded(i checked it with pdf-parser.py), it did not run.
wen i run the js file directly it executes, but when i embed it , it does not run.

Comment by pret — Tuesday 15 February 2011 @ 11:41

113. @pret And how do you start Notepad?

Comment by Didier Stevens — Tuesday 15 February 2011 @ 17:08

114. i ran notepad directly from js file using ws.run command , but wen i run the script outside pdf, it runs, wen i embed it in pdf and run, it gets embedded but does not run. pls tell how can i make it run.

Comment by pret — Thursday 17 February 2011 @ 5:28

115. @pret You are using a Windows JavaScript feature, that's not supported by Adobe's JavaScript. There is no feature to run arbitrary programs.

Comment by Didier Stevens — Thursday 17 February 2011 @ 7:06

116. I am new to Python. I have installed Python 27 and have tried running pdfid.py with no success.
The syntax >>>pdfid.py MidtermChazaraQuestions.pdf returns Invalid Syntax error in the input.
What am I doing wrong? It is extremely important I analyze this file. It may be the key to the identity theft that is destroying me. Please, help!

Comment by Joseph Ainbinder — Friday 18 February 2011 @ 19:33

117. @joseph You need to use 2.6, a module in 2.7 was deprecated.

Comment by Didier Stevens — Friday 18 February 2011 @ 20:07

118. […] pdf-parser.py – https://blog.didierstevens.com/programs/pdf-tools/ (éditer le source pour modifier la version maximale de python acceptée)- pdfid.py – […]

Pingback by escape from PDF | Linux-backtrack.com — Saturday 19 February 2011 @ 21:20

119. […] – Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD: pdf-parser and […]

Pingback by Malware analysis « Eikonal Blog — Monday 28 February 2011 @ 16:33

120. […] pdf-parser.py […]

Pingback by PDF Analysis for Humans « P4r4n0id Reversing Lab — Friday 18 March 2011 @ 15:28

121. i had a problem with "make-pdf-javascript"

first use with the original package:

C:\Documents and Settings\abdelmoumen bacetti\mpdf1>python make-pdf-javascript.py test.pdf
File "make-pdf-javascript.py", line 29
print "
^
SyntaxError: invalid syntax
##
so i changed the lines 29,30,31,32,33,55,61 in "make-pdf-javascript.py" and line 110 in "mPDF.py" because the "prints" are without parenthesis
##
after fixing the prints problem:

C:\Documents and Settings\abdelmoumen bacetti\mpdf>python make-pdf-javascript.py down.pdf
Traceback (most recent call last):
File "make-pdf-javascript.py", line 71, in
Main()
File "make-pdf-javascript.py", line 44, in Main
oPDF.stream(5, 0, 'BT /F1 12 Tf 100 700 Td 15 TL (JavaScript example) Tj ET')
File "C:\Documents and Settings\abdelmoumen bacetti\mpdf\mPDF.py", line 69, in stream
self.appendBinary(streamdata)
File "C:\Documents and Settings\abdelmoumen bacetti\mpdf\mPDF.py", line 39, in appendBinary
fPDF.write(str)
TypeError: 'str' does not support the buffer interface

##
config:

Windows XP SP2
Python 3.2

Comment by bmoumen — Sunday 10 April 2011 @ 13:37

122. @bmoumen Yes, my Python programs are not designed for Python 3. Neither do most of my programs work on 2.7, because of a deprecated module I use to parse command lines. It's something I hope to solve in a near future (i.e. make my Python programs compatible with Python 2.5, 2.6, 2.7 and 3.x).

    Comment by Didier Stevens — Monday 11 April 2011 @ 7:05

123. […] of python tools which can be used for analysing PDFs. I downloaded two of his tools from this page https://blog.didierstevens.com/programs/pdf-tools/, pdf-parser.py and […]

    Pingback by Solving the Security BSides London Challenge – Number 2 | 4armed — Thursday 21 April 2011 @ 14:39

124. […] a look at my Analyzing Malicious Documents Cheat Sheet. From the tools perspective, Didier Steven's pdf-parser is an all-time favorite. Another excellent tool, which sports a user-friendly GUI, is PDF Stream […]

    Pingback by How to Extract Flash Objects from Malicious PDF Files — Wednesday 4 May 2011 @ 15:18

125. […] PDF Tools by Didier Stevens is the classic toolkit that established the foundation for our understanding of the PDF analysis process. It includes pdfid.py to quickly scan the PDF for risky objects and, most usefully, pdf-parser.py to examine their contents. […]

    Pingback by 6 Free Tools for Analyzing Malicious PDF Files « AfterShell.com — Wednesday 11 May 2011 @ 17:46

126. […] Signatures work with a few open source tools. The first one is pdf-parser.py which is part of the PDF Tools by Didier […]

    Pingback by The Anatomy of a PDF Signature < experiment nr.: 1598 — Wednesday 11 May 2011 @ 19:39

127. […] But did you notice the inclusion of my PDFiD and pdf-parser tools? […]

    Pingback by BackTrack 5 Includes PDFiD and pdf-parser « Didier Stevens — Thursday 12 May 2011 @ 21:13

128. […] my PDF tools […]

    Pingback by Malicious PDF Analysis Workshop Screencasts « Didier Stevens — Wednesday 25 May 2011 @ 15:59

129. […] Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and […]

    Pingback by 基于python渗透测试工具 — Sunday 29 May 2011 @ 3:00

130. […] here. In the past I have also used […]

    Pingback by Checking a PDF for exploits Drija — Thursday 9 June 2011 @ 4:14

131. […] encodings to name like JBIG2Decode and DCTDecode. FlateDecode usually can be decoded by using pdf-parser […]

    Pingback by Analyzing malicious PDF « lab69 — Thursday 23 June 2011 @ 17:08

132. […] suo interno l'exploit vero e proprio. Sinceramente non sono riuscito a decomprimerlo né con pdf-parser di Didier Stevens, né con PDF Stream Dumper, né con Ghostscript come spiegato qui. Diciamo che […]

    Pingback by Jailbreakme: ecco come funziona il jailbreak per iPad 2 — Wednesday 6 July 2011 @ 21:28

133. Hi Didier,

May I ask you which tools are you using for Python (debuggers,..)

Thanks

Comment by zudqg — Wednesday 20 July 2011 @ 13:59

134. […] primero que nos interesa es determinar el contenido del PDF y para ello utilizamos las PDFtools que nos permiten analizar PDF. Ejecutamos la herramienta pdfid para ver el contenido del fichero y […]

    Pingback by Reconstructing JavaScript Exploit « Simon Roses Femerling – Blog — Wednesday 20 July 2011 @ 20:32

135. @zudqg I'm going to disappoint you, for Python, I just use a text editor.

    Comment by Didier Stevens — Thursday 21 July 2011 @ 6:31

136. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and mPDF) […]

    Pingback by Repost:Lista de ferramentas de segurança feitas em Python. « VSLA – Virtual Security Labs Anywhere — Monday 1 August 2011 @ 15:51

137. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and mPDF) […]

    Pingback by Attack Attack » Python tools for penetration testers — Monday 8 August 2011 @ 4:17

138. […] javascript heap overflow in PDF. More info to come. I used Didier Steven's pdfid and pdf-parser to extract the javascript. The Javascript which is called when the document is opened creates a […]

    Pingback by The Spy Hunter, Part II – Solution « wirewatcher — Sunday 14 August 2011 @ 20:55

139. Just a "wowie" comment – thanks for sharing these tools, they're fantastic.

    Comment by B. Oceander — Monday 26 September 2011 @ 14:55

140. Hi Didier,

    Do you have a tool, or know of a tool, that can take an existing PDF and add JS to it? I would like the ability to add javascript to multiple existing files. It would basically have the same functionality as your current make-pdf.py script, but you'd provide it an existing PDF, as well as a JS file that it would be merged with.

    Thx for your help!

    Comment by Sagui — Thursday 13 October 2011 @ 12:35

141. @Sagui Look for phptk, it can merge 2 PDF files.

    Comment by Didier Stevens — Friday 14 October 2011 @ 20:51

142. Hi Didier, thanks for providing these tools, would you have any objection to me adding them to a public github repo so people can contribute any fixes/extensions they have?

    Comment by Tom — Sunday 16 October 2011 @ 13:03

143. @Tom No problem, let me know where.

    Comment by Didier Stevens — Sunday 16 October 2011 @ 13:23

144. All done https://github.com/thomcarver/pdf-tools

    Comment by Tom — Sunday 16 October 2011 @ 15:21

145. Hello,

Can some one help me to figure out how to use this pdfid tool. I have python inerpretor installed but would like to know how I can specify which file or directory I want this tool to parse.

I am new to Python.

Comment by Ishwar — Tuesday 18 October 2011 @ 12:20

146. @Ishwar: I assume you're running Windows? Then you install Python 2.X (not version 3), open a command line (cmd.exe) and type pdfid.py test.pdf where test.pdf is the file you want to check.

Comment by Didier Stevens — Wednesday 19 October 2011 @ 16:52

147. […] purpose, or write a custom tool ourselves. For the sake of this tutorial, I'll stick with Didier Steven's excellent "make-pdf" python script (which uses the mPDF […]

Pingback by Exploit writing tutorial part 11 : Heap Spraying Demystified | Corelan Team — Saturday 31 December 2011 @ 23:32

148. Hello Didier,

Thank you for providing these tools.

I have scanned a PDF I suspect may be malicious with your pdfid script, and it returned 0 for everything but " /AcroForm 1". I see above that acroform is not described in the pdfid summary. Could you please tell the meaning of this, and how to tell whether it is harmful?

Comment by Inkblots — Wednesday 11 January 2012 @ 19:40

149. @InkBlots Take a look at my PDF workshop, I've an exercise for AcroForm. AcroForm can contain JavaScript that is executed when a document is opened.

Comment by Didier Stevens — Wednesday 11 January 2012 @ 20:15

150. […] PDF-Parser (https://blog.didierstevens.com/programs/pdf-tools/) […]

Pingback by Attributes of a Zero Dollar Malware Analysis System « secanalysis.com — Monday 16 January 2012 @ 17:21

151. […] Related great tools: https://blog.didierstevens.com/programs/pdf-tools/ […]

Pingback by Re: pdf attacks vectors | Net Cleaner — Saturday 21 January 2012 @ 18:29

152. […] First we use a new version of my PDF tools to create a PDF file with embedded file: […]

Pingback by Teensy PDF Dropper Part 2 « Didier Stevens — Monday 27 February 2012 @ 0:00

153. […] https://blog.didierstevens.com/programs/pdf-tools/ http://www.mozilla.org/js/spidermonkey/ https://code.google.com/p/jsunpack-n/ http://malzilla.sourceforge.net/ […]

Pingback by | web güvenlik , SKaracan.com , Web Güvenlik , Sistem Güvenliği ve Kişisel Güvenliğe Dair Herşey | — Friday 9 March 2012 @ 17:19

154. […] PDF Tools – https://blog.didierstevens.com/programs/pdf-tools/ […]

Pingback by SecuraBit Episode 32: PDF Love! « SecuraBit — Tuesday 13 March 2012 @ 15:31

155. […] can find these tools on the PDF Tools page. Like this:LikeBe the first to like this post. Leave a […]

Pingback by Update: PDFid And pdf-parser « Didier Stevens — Wednesday 14 March 2012 @ 9:15

156. […] PDF-parser Wieloplatformowy, konsolowy program do przetwarzania i analizy dokumentów PDF. Potrafi wyodrębnić surowe dane z dokumentu takie jak skompresowane obrazy. Dobrze radzi sobie z uszkodzonymi oraz zaciemnionymi plikami. […]

Pingback by Edytory PDF 2 | Linuxiarze.pl — Saturday 24 March 2012 @ 0:16

157. […] тематику анализа PDF файлов. Getting Owned By Malicious PDF – Analysis PDF Tools от Didier […]

Pingback by Информация по анализу PDF файлов. « clickf1 web log. — Monday 2 April 2012 @ 8:39

158. […] – look forcat_open_xml.pl; other tools available, as well Skype Extractor – PDF Tools – from Didier Stevens; some of Didier's tools have been incorporated into the VirusTotal […]

Pingback by Herouxapps (Home of Freeware) — Wednesday 18 April 2012 @ 23:06

159. Fantastic tools! Many thanks, Didier. I had a pdf send to me by would-be fraudsters. It was a great relief to find that the document itself was not malicious.

Comment by John — Wednesday 9 May 2012 @ 19:02

160. […] Dider Stevens的PDFiD.pf和pdf-parser.py (https://blog.didierstevens.com/programs/pdf-tools/)写的界面。PDFiD.py和pdf- […]

Pingback by PDF恶意文档分析-PDFScope- FreebuF.COM — Tuesday 5 June 2012 @ 2:19

161. […] PDF Tools "Didier Stevens" – Didier tiene una gran colección de herramientas locales. […]

Pingback by Securización de lectores PDF « marian1105 — Wednesday 6 June 2012 @ 16:16

162. I receive this message when trying to use pdf-parser, can you help?

C:\Program Files\IronPython 2.6>ipy pdf-parser.py -help
Traceback (most recent call last):
File "pdf-parser.py", line 50, in
ImportError: No module named zlib
C:\Program Files\IronPython 2.6>

same with any file i try to scan

Comment by dannybpcr — Tuesday 3 July 2012 @ 22:41

163. @dannybpcr My tools are not developed for IronPython. You must use Python.

Comment by Didier Stevens — Tuesday 3 July 2012 @ 22:51

164. thnx for sharing

Comment by raef — Tuesday 28 August 2012 @ 11:43

165. […] can be learned from this data. Didier has published a pdf parsing tool written in python called pdf-parser.py, which looks to be very promising in analyzing pdf files. I just started playing with the tool […]

Pingback by sudosecure.net » Blog Archive » Analyzing PDF files and Shellcode — Thursday 18 October 2012 @ 16:38

166. […] PDF ise bu defa amaç, zararlı kod içerebilecek Javascript kodunu tespit etmektir. Bunun için de pdf-parser.py, peepdf ve Origami gibi araçlardan […]

Pingback by Zararlı PDF Analizi | Hack 4 Career — Thursday 6 December 2012 @ 20:01

167. A couple of observations about pdf-parser.py.

First, it is very slow on files which have large images embedded in them. I think this comes from the tokenizer code which contains lines such as
self.token = self.token + chr(self.byte)
There is a good analysis of the speed of this compared to other methods at http://www.skymind.com/~ocrow/python_string/. When I changed it so that self.token is a StringIO buffer, I got an huge increase in speed. In particular, one file which has not completed parsing after 30 minutes was now processed in a few seconds.

Secondly, I noticed that Decompress was not called on some stream data. This turned out to be because the stream was ASCII85 encoded and ended like this:
T.5*QV#Ts4I~>endstream
Note that there is no end of line character between the ASCII85 end marker (~>) and the endstream keyword. According to the PDF 1.7 specification, this is not approved of but is

allowed:
"It is recommended that there be an end-of-line marker after the data and before endstream; this marker is not included in the stream length." on page 61 of http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_1-7.pdf. The file in question was generated by reportlab.

My first thought for fixing this was to change
if self.content[i][0] == CHAR_REGULAR and self.content[i][1] == 'endstream':
to
if self.content[i][0] == CHAR_REGULAR and self.content[i][1].endswith('endstream')
and then trimming the keyword off the data. However, this does not work, as self.content[i][1] actually ends with a newline character, and self.content[i][0] has the value CHAR_DELIMITER. Something like
if self.content[i][1].strip().endswith('endstream'):
end = self.content[i][1].rindex('endstream')
data += self.content[i][1][:end]
might do the job, though it's ugly. The ideal solution would really be to use the length attribute from the dictionary, though this seems to be a bigger change.

Otherwise, the code looks great, and is really helping me with a project I am working on.

Comment by David Elworthy — Tuesday 1 January 2013 @ 19:21

168. @David Interesting, thanks for the observations. Will do some profiling.
What I'm curious about: how come you are parsing PDF files that require so much time? Are these malicious?

Comment by Didier Stevens — Tuesday 1 January 2013 @ 22:54

169. My end goal is writing a scanner application which will build archive versions of documents from photographs of pages. I'm a long way off this, and so was using a PDF build from some photos of landscapes, but even so the files were only a few megabytes. Eventually I want to generate my own PDFs, as I don't much like reportlab and pyPDF, but for now reportlab is what I am using. I was looking at your code as a way of understanding the file format. As a shorter term project, I also want to write something which willtake files with 600 dpi images from a flatbed scanner and either downsample them to a lower dpi or increase the JPEG compression, as I sometimes find the 600 dpi scans (which are meant to be archive quality) are a bit large for emailing when there's a lot of pages. Of course there are plenty of applications which allow you to manipulate PDFs interactively, but I'm a command line kind of guy, so a python script would be ideal.

Comment by David Elworthy — Tuesday 1 January 2013 @ 23:10

170. Hi there, thanks for the amazing script, it makes the life easier. I have a PDF with a postscript-type image embedded (an eps actually). I am reading the PDF reference and I think for that kind of image, it will be stored as postscript command in the stream so I am wondering if it is possible to extract the postscript in the stream directly. Thanks

Comment by Anonymous — Sunday 6 January 2013 @ 6:39

171. @Anonymous Yes, you can extract it.

Comment by Didier Stevens — Tuesday 8 January 2013 @ 9:16

172. @David pdf-parser is designed to parse malicious PDF documents, so I assume that the PDF document contains wrong information. For example, that's why I don't rely on the /Length value to parse a stream.

Comment by Didier Stevens — Tuesday 8 January 2013 @ 9:17

173. […] A month before my PDF training at HITB, it's time to release new versions of my pdf tools. […]

Pingback by Update: PDFiD Version 0.1.0 | Didier Stevens — Thursday 7 March 2013 @ 5:01

174. Hi Didier
Great work you are doing with the PDF format. One quick question about the browsers supporting pdf documents. Is it a good idea to think of browsers as better pdf readers because they are supposed to have sealed most javascript vulnerablities ?? would love to hear your opinion on that

Thanks
Jiss

Comment by Jiss — Tuesday 12 March 2013 @ 17:37

175. @Jiss The idea of PDF readers in beowers like Firefox's pdf.js, is that they are written in a higher language than standard readers (hence not in C), and thus that bugs can't be exploited like in C.

    pdf.js is written in JavaScript. Say you find a bug in pdf.js and that you try to develop an exploit for it. The best you'll be able to do, is execute arbitrary JavaScript.

    Comment by Didier Stevens — Tuesday 12 March 2013 @ 23:23

176. Using Reader 10.1.6 on MacOSX 10.7, get the following error when embedding an EXE:
    Acrobat EScript Built-in Functions Version 10.0
    Acrobat SOAP 10.0

    TypeError: Invalid argument type.
    Doc.exportDataObject:1:Doc undefined:Open
    ===> Parameter cName.
    TypeError: Invalid argument type.
    Doc.exportDataObject:1:Doc undefined:Open
    ===> Parameter cName.

    Comment by Phil — Wednesday 13 March 2013 @ 15:30

177. @Phil What options did you use to create this document?

    Comment by Didier Stevens — Wednesday 13 March 2013 @ 21:08

178. Didier, this was my third attempt. This one used -a -m.

    Comment by Phil — Wednesday 13 March 2013 @ 21:26

179. @Phil OK, I was sure you used option -a. You have to know that PDF readers like Adobe Reader do not allow you to extract executable files. To determine if a file is executable or not, Adobe Reader looks at the extension. So you can't extract .exe files (unless you change the extension to something that is not executable, like .txt).
    Option -a instructs my tool to add JavaScript to the PDF document to extract the embedded file automatically. But since this is not allowed for an .exe file, the script fails, and that is what you see in the error messages.

    FYI because you are doing this on OSX: Python (.py) is allowed as executable file type.

    Comment by Didier Stevens — Wednesday 13 March 2013 @ 21:33

180. Thanks. Suspected that much. Was able to unpackage Acrobat to determine the list of disallowed extensions. For everyone else, that list is:
    .ade:3|.adp:3|.app:3|.arc:3|.arj:3|.asp:3|.bas:3|.bat:3|.bz:3|.bz2:3|.cab:3|.chm:3|.class:3|.cmd:3|.com:3|.command:3|.cpl:3|.crt:3|.csh:3|.desktop:3|.dll:3|.dylib:3|.exe:3|.fxp:3|.gz:3|.hex:3|.hlp:3|.hqx:3|.hta:3|.inf:3|.ini

    Comment by Phil — Wednesday 13 March 2013 @ 21:48

181. @Phil IIRC, the number following the extension indicates what is allowed or not. Look at the end of the list: .pdf and .fdf have number 2.

    Comment by Didier Stevens — Wednesday 13 March 2013 @ 21:53

182. Hello. Sir
    When we look for tags like /JS, I think they should be seen when an object starts.

    Consider this file: http://www.mcafee.com/in/resources/white…/wp-new-era-of-botnets.pdf
    pdfid.py shows /JS in this file but this /js is actually written as part of text.

can you please help me on this. is this really a javascript into this document or not. I try this pdf file with pdfextract and this also could not extract any javascript.

please help
i will be very grateful to you on this.

Comment by himanshu — Wednesday 20 March 2013 @ 9:52

183. @Himanshu

analyze the file with pdf-parser and search for /JS.
If pdf-parser can't find it, then it is not a name in a dictionary but most likely a string in a stream.

Comment by Didier Stevens — Wednesday 20 March 2013 @ 10:15

184. hello mr stevens.

i closely follow your post and i am facing a problem when i m trying to run .exe embedded in a pdf through make-pdf-embedded.py . it is not running on the windows 7 machine. also it is not supported by adobe x and above. is there a way out for this prob

yours sao zumin

Comment by sao zumin — Friday 22 March 2013 @ 5:50

185. @sao That doesn't work. Please take a look at comments 105 and 106.

Comment by Didier Stevens — Friday 22 March 2013 @ 7:45

186. Hi Didier, Great info and tools.
I noticed that extension .py was missing from the list of disallowed extensions. Is it possible to use python to assist in launching an executable?

Thanks
Matahachi

Comment by Matahachi — Friday 12 April 2013 @ 8:40

187. @Matahachi Yes, Python is allowed, I used it as an example in my training class.

Comment by Didier Stevens — Friday 12 April 2013 @ 19:56

188. […] Didier Stevens PDF tool kit to the rescue! Didier has created some great forensic tools for working with PDF […]

Pingback by BSides 2013 – Challenge 4 – CSCUK Challenge | TabChalk - Beware the devil inside! — Saturday 27 April 2013 @ 12:34

189. […] the PDF using Didier Stevens' PDFiD tool shows that the two PDFs are very similar. They may not be identical, but the similarities […]

Pingback by Malicious PDFs On The Rise | Security Intelligence Blog | Trend Micro — Tuesday 30 April 2013 @ 9:54

190. […] Ok, we have our PDF now and we are ready to begin our analysis. We need a tool for inspection, in our case we'll use Didier Steven's pdf-parser.py […]

Pingback by Analysis of CVE-2010-0188 PDF from RedKit ExploitKit — Friday 10 May 2013 @ 20:11

191. […] PDFiD will give you false positives for /JS and /AA. This happens with files of a couple of MBs or […]

Pingback by PDFiD: False Positives | Didier Stevens — Monday 10 June 2013 @ 8:49

192. Can I Decompress file in Mac-OS(Macintosh)??

Comment by Sandeep Vasoya — Monday 8 July 2013 @ 7:31

193. @Sandeep My Python programs work on OSX too.

    Comment by Didier Stevens — Monday 8 July 2013 @ 18:07

194. Thanks Didier…

    Comment by Sandeep — Tuesday 9 July 2013 @ 4:59

195. […] PDF Parser […]

    Pingback by Tools » Damul's Blog — Thursday 29 August 2013 @ 2:11

196. Hi,
    I have a PDF that only has about 500 pages, but your pdfid shows /Page to be around 1100. Any ideas what's going on?

    PS. Thanks again for keeping these tools up to date. I've been following your work since early versions.

    Comment by CurlyBird — Wednesday 4 September 2013 @ 16:14

197. @CurlyBird This counter counts the number of instances of the /Page name found in the document. There could be several reasons why pdfid finds more /Page instances than there are pages.
    Without having the document, it's hard to tell. But one reason can be that your document is made with incremental updates (e.g. that the document contains previous versions of the PDF document, and thus that pdfid counts these too).

    Comment by Didier Stevens — Wednesday 4 September 2013 @ 17:23

198. @Didier: Great! Very quick answer. That would make sense. I noticed some discrepancies in the document regarding file size vs content and suspected missing information, which is why I decided to inspect it closer in the first place. Is there any way to retrieve this previous version? (Or revert these incremental updates?)

    Comment by CurlyBird — Wednesday 4 September 2013 @ 19:44

199. @CurlyBird Yes, it's something I teach in my training. Search for %%EOF not at the end of the file.

    Comment by Didier Stevens — Wednesday 4 September 2013 @ 22:08

200. @Didier: Great! Sure enough there were some missing stuff in there, but there were 6 counts of "EOF%%" but I can only tell any obvious difference between the 1st and 2nd versions. The later ones "look" the same. I wish there were some kind of more visual PDF diffing utility…

    BTW. I got the offsets by:
    strings -n 4 -t x -e s weird.pdf |grep -i -E "%%EOF"

    Then extracted the versions with:
    dd if=weird.pdf of=weird_a.pdf bs= count=1

    Thanks again.

    PS. Sorry, I can't attend your training as I live very far away.

    Comment by CurlyBird — Thursday 5 September 2013 @ 12:47

201. Web Magic above: "bs=" should be "bs=offset+6"

    Comment by CurlyBird — Thursday 5 September 2013 @ 12:53

202. Hello Didier
.I don't know anything in Python and in pdf security…
Today i downloaded a pdf file. When i opened it, "cmd" appeared and many lines displayed quickly inside…
Soi have to check this strange pdf file…
So i installed Python 3.3.2 (my pc's OS is Windows 7) but i have a problem when i test:

C:\Python33>python.exe pdf-parser.py
File "pdf-parser.py", line 486
except zlib.error, e:
^
SyntaxError: invalid syntax

Thanks for your help. It's very importan and urgent for me to check this file and to know if my pc has a problem.
Mathias

Comment by Mathias Rollet — Saturday 7 September 2013 @ 22:21

203. @Mathias Try with Python 2.7 But if you don't know anything about PDF, my tools will not help you. In your case it's better to upload the PDF file to VirusTotal and see if it is detected by AV.

Comment by Didier Stevens — Sunday 8 September 2013 @ 8:45

204. […] utilizada: https://blog.didierstevens.com/programs/pdf-tools (script […]

Pingback by Uso de PDF para Exploração de Vulnerabilidades | Yuri Diogenes — Wednesday 9 October 2013 @ 18:20

205. Hi Didier,

I find the PFDiD.py interesting but I am having a difficult time trying to get it to work. I have a VM with Windows XP installed and python 2.6.6 installed and appears to be working fine. If I enter PDFiD MyFile.pdf I get a syntax error. although your example shows PDFiD 0.0.2 test.pdf I just don't understand what significance the number has and what the correct syntax is to get my example to work. Another example I'm having difficulties with is pdf-parser.py that should work by implementing pdf-parser.py MyFile.pdf –search=javascript. Any help would be greatly appreciated

Thanks,

Michael

Comment by Michael — Sunday 26 January 2014 @ 3:36

206. @Michael Can you report the syntax error?

Comment by Didier Stevens — Sunday 26 January 2014 @ 16:58

207. Didier,

Thanks for getting back to me so soon! Below will be the syntax errors:

Example 1: PDFiD

Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> PDFiD TheFlyv3_EN4Rdr.pdf
File "", line 1
PDFiD TheFlyv3_EN4Rdr.pdf

^
SyntaxError: invalid syntax
>>>

Example 2: pdf-parser.py

>>> pdf-parser.py TheFlyv3_EN4Rdr.pdf –search=javascript
File "", line 1
pdf-parser.py TheFlyv3_EN4Rdr.pdf –search=javascript
^
SyntaxError: invalid syntax
>>>

Thanks,

Michael

Comment by Michael — Sunday 26 January 2014 @ 19:43

208. OK, I see what's wrong. You start Python and then launch pdfid or pdf-parser.
That's not how you should do it, you should launch these tools from the command line (cmd.exe).

Start cmd.exe and type pdfid.py test.pdf, where test.pdf is one of your PDFs, and make sure pdfid is in the working folder c:\test:
C:\Test\>pdfid.py test.pdf

Comment by Didier Stevens — Sunday 26 January 2014 @ 22:25

209. Didier,

I think another problem with this is that I don't have pdfid install properly where do I get that?

Thanks,

Michael

Comment by Michael — Monday 27 January 2014 @ 3:48

210. @Michael. There is no install program, it's just a Python program.
Maybe you're not familiar with the command-line in Windows. I suggest you save pdfid.py and your pdf in the same directory, and then open a command-line in that directory.

Comment by Didier Stevens — Monday 27 January 2014 @ 19:34

211. Didier,

Awesome! Thank you for all your support it worked perfectly.

Michael

Comment by Anonymous — Tuesday 28 January 2014 @ 3:32

212. Quick question – I am trying to use PDFid to detect malicious flash inside a PDF. When I run PDFid it shows zeros for RichMedia, EmbeddedFile, OpenAction, and AA. What output from PDFid should indicate the flash?

A sample PDF with the flash can be found here: http://contagiodump.blogspot.com/2011/04/apr-22-cve-2011-0611-pdf-swf-marshall.html

Thanks.

Comment by Todd — Monday 10 February 2014 @ 0:45

213. @Todd Can you post the pdfid analysis?

Comment by Didier Stevens — Wednesday 12 February 2014 @ 23:39

214. Sure can..

C:\malwaresandbox>pdfid.py "Marshall Plan for the North Africa.pdf"
PDFiD 0.1.2 Marshall Plan for the North Africa.pdf
PDF Header: %PDF-1.7
obj 35
endobj 35
stream 24
endstream 24
xref 1
trailer 1
startxref 5
/Page 3
/Encrypt 0
/ObjStm 8
/JS 0
/JavaScript 0
/AA 0
/OpenAction 0
/AcroForm 0
/JBIG2Decode 0
/RichMedia 0
/Launch 0
/EmbeddedFile 0
/XFA 0
/Colors > 2^24 0

Comment by Todd — Thursday 13 February 2014 @ 0:38

215. @Todd OK, I see, you have 8 object streams (/ObjStm). These are streams that contain object, that's where the Flash code is hiding. You need to inflate these streams and pipe that to pdfid. That is something I explain in my PDF Analysis workshop: http://DidierStevensLabs.com

Comment by Didier Stevens — Thursday 13 February 2014 @ 23:20

216. […] The shellcode contains the URL that the exploit will contact to download the malicious payload. We can extract it using pdf-parser: […]

Pingback by Email-borne exploits: the not-so innocuous killers targeting small business | Malwarebytes Unpacked — Monday 12 May 2014 @ 18:24

217. I ran make-pdf-javascript.py on my windows platform but nothing happened … C:\Users\Suleiman JK\Desktop>Python make-pdf-javascript.py "C:\Users\Suleiman JK\Desktop\suleiman.pdf" … I suppose a pdf file with a name "suleiman" to be created on desktop but nothing happened. do I ran the tool correctly ?//

Comment by Suleiman Kheetan — Tuesday 1 July 2014 @ 21:31

218. Yes. But make sure you use Python 2 for the make tools.

Comment by Didier Stevens — Thursday 3 July 2014 @ 15:11

219. do you mean to use any version of python 2 till 2.77 ?

Comment by Suleiman Kheetan — Friday 4 July 2014 @ 16:07

220. Yes, Python 2 is 2.x.x. I recommend the latest.

Comment by Didier Stevens — Friday 4 July 2014 @ 16:31

221. I knew where was my mistake that I'm using windows 8. I tried it in windows xp and worked fine. but how could we make it works on windows 8 ?

Comment by Suleiman Kheetan — Friday 4 July 2014 @ 22:21

222. […] PDF Tools de Didier Stevens. PDFStreamDumper – Esta es una herramienta gratuita para el análisis PDFs maliciosos. SWF Mastah – Programa en Python que extrae stream SWF de ficheros PDF. […]

Pingback by Listado de Herramientas Forenses | ROOTAGAINSTTHEMACHINE — Monday 7 July 2014 @ 12:14

223. […] PDFiD […]

Pingback by بررسی پرونده‌های آلوده - ایمن وب — Wednesday 9 July 2014 @ 18:49

224. […] embedf Create a blank PDF document with an embedded file. This is for research purposes to show how files can be embedded in PDFs. This command imports Didier Stevens Make-pdf-embedded.py script as a module. (https://blog.didierstevens.com/programs/pdf-tools/) […]

Pingback by ParanoiDF - PDF Analysis Tool — Sunday 17 August 2014 @ 5:01

225. Didier, I need some help. I have recently upgraded to Adobe XI and some of my previously OK adobe pdf files now cannot be read. Adobe indicate that they have increased "security" and enforced some compliance in their document headers. I have gone back though my useful bits of software that may be able to help me and come across your 010 editor. I have compared a couple of files that are ok and still working but I am not able to spot any significant differences (other than length, width height etc.).

    Have you com across this problem before ? Any ideas as to how to resolve the problem. BTW the file is quite large 25MB and my programming skills are now quite poor – used to be an assembler / c programmer about 20 years ago!!!

    Comment by Paul Kirikal — Sunday 24 August 2014 @ 15:19

226. @Paul Can you read the files with Sumatra PDF?

Comment by Didier Stevens — Sunday 24 August 2014 @ 20:36

227. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parserand make-pdf and mPDF) […]

Pingback by Python：渗透测试开源项目 – Arschloch — Friday 5 September 2014 @ 19:55

228. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and mPDF) […]

Pingback by Lufsec – Python tools for penetration testers — Saturday 6 September 2014 @ 4:39

229. […] technique is using Didier Stevens suite of tools to analyze the content of the PDF and look for suspicious elements. One of those tools is Pdfid […]

Pingback by Malicious Documents – PDF Analysis in 5 steps | Count Upon Security — Monday 22 September 2014 @ 10:02

230. […] this PDF was another red herring considering the image in the PDF, but I persisted and fired up pdf-parser and got into the internals of the PDF file (Figure 8). Admittedly I am still learning PDF […]

Pingback by CSAW 2014 walkthrough – Fluffy No More | Overflow Security — Thursday 25 September 2014 @ 1:24

231. […] PDF ise bu defa amaç, zararlı kod içerebilecek Javascript kodunu tespit etmektir. Bunun için de pdf-parser.py, peepdf ve Origami gibi araçlardan […]

Pingback by Zararlı PDF Dosyalarının Adım Adım Analizi | caglar's space — Tuesday 30 September 2014 @ 13:45

232. Dear mr. steven
I'm trying to use pdfid in windows 8 with python 2.7.8 when I ran it by cmd.exe I have this error :
C:\Users\Test\Desktop>pdfid.py MultiplePages.pdf
Traceback (most recent call last):
File "C:\Users\Test\Desktop\pdfid.py", line 25, in
import urllib.request
File "C:\Python27\lib\urllib.py", line 33, in
from urlparse import urljoin as basejoin
File "C:\Python27\lib\urlparse.py", line 119, in
from collections import namedtuple
File "C:\Python27\lib\collections.py", line 12, in
import heapq as _heapq
ImportError: No module named heapq

could you help me please

Comment by Suleiman Khitan — Friday 3 October 2014 @ 21:21

233. regarding to the 232 question i have this error

C:\Users\Test\Desktop>pdfid.py MultiplePages.pdf
Traceback (most recent call last):
File "C:\Users\Test\Desktop\pdfid.py", line 20, in
import zipfile
File "C:\Python27\lib\zipfile.py", line 4, in
import struct, os, time, sys, shutil
File "C:\Python27\lib\shutil.py", line 12, in
import collections
File "C:\Python27\lib\collections.py", line 12, in
import heapq as _heapq
ImportError: No module named heapq

Comment by Suleiman Khitan — Friday 3 October 2014 @ 21:26

234. @Suleiman I can not reproduce your problem, it works fine with 2.7.8. Check pdfid.py, because the reported line numbers (20 and 25) are not normal.
The first 50 lines are pdfid are comment, so they can not create an import error.

Comment by Didier Stevens — Saturday 4 October 2014 @ 8:08

235. […] DE MALWARE PDF Tools de Didier Stevens. PDFStreamDumper – Esta es una herramienta gratuita para el análisis PDFs […]

Pingback by HERRAMIENTAS USADAS EN LA COMPUTACIÓN FORENSE | RECOLECCIÓN DE EVIDENCIA DIGITAL — Wednesday 19 November 2014 @ 14:07

236. Hi Planning to build a PDF tester application, my job involved testing Pdf for formats issues and check values in each cell of the report for format and length.
Is there a tool to check all the objects in an present it in tree view.

I have plans to build a tool which has the meta data on how the object should look like in the resultant PDF. The tool will evaluate and check with the meta data for discrepancies.

Comment by udhay — Thursday 20 November 2014 @ 11:39

237. @udhay I know tools like PDF Dissector and PDF Structazer do this, but these tools are discontinued. You could take a look at the PDF template I developed for 010 Editor.

Comment by Didier Stevens — Saturday 22 November 2014 @ 17:31

238. […] nombrar algunas herramientas específicas comenzaremos por pdfid, una aplicación sencilla para explorar "de un vistazo" la estructura del documento (cabecera, […]

    Pingback by Herramientas para el análisis de documentos PDF maliciosos « DabacodLAB — Friday 28 November 2014 @ 14:28

239. In your pdf-parser.py you have:

    def CharacterClass(byte):
    if byte == 0 or byte == 9 or byte == 10 or byte == 12 or byte == 13 or byte == 32:
    return CHAR_WHITESPACE
    …
    Might be faster if you wrote:

    def CharacterClass(byte):
    if byte in b"\x00\x09\x0A\x0C\x0D\x20":
    return CHAR_WHITESPACE
    …

    Or

    def CharacterClass(byte):
    if byte in frozenset((0, 9, 10, 12, 13, 32)):
    return CHAR_WHITESPACE

    Comment by Mark Summerfield — Monday 5 January 2015 @ 11:58

240. I'm sure your last example is not faster. Each time you want to test a byte, you create a frozenset. For performance, you should create this frozenset only once (for example global variable).

    I believe more speed can be gained by not using a function., e.g. do the test inline.
    But then you loose readability and maintainability. That's why performance is less important as design requirement to me.

    Comment by Didier Stevens — Monday 5 January 2015 @ 21:05

241. […] In that case one of the best tool available is oledump.py from Didier Stevens (also known for his PDF tools…but we will talk about that in an upcoming […]

    Pingback by Word document analysis with oledump.py | SimonGaniere.ch — Monday 12 January 2015 @ 13:58

242. […] PDFs using "pdfcop", "pdf-parser", "pdfid", "pdfdecompress" and […]

    Pingback by REMnux Usage Tips for Malware Analysis on Linux — Thursday 15 January 2015 @ 3:09

243. […] The course now teaches steps for analyzing malicious Adobe PDF documents, making use of utilities such as Origami and Didier Stevens' PDF Tools. […]

    Pingback by Expansion of the SANS Reverse-Engineering Malware (REM) Course FOR610 in 2010 — Tuesday 27 January 2015 @ 18:00

244. […] PDFiD identifies PDFs that contain strings associated with scripts and actions. […]

    Pingback by Analyzing Malicious Documents Cheat Sheet — Tuesday 27 January 2015 @ 18:11

245. […] PDFs using "pdfcop", "pdf-parser", "pdfid", "pdfdecompress" and […]

    Pingback by REMnux Usage Tips for Malware Analysis on Linux — Tuesday 27 January 2015 @ 19:56

246. […] PDFiD identifies PDFs that contain strings associated with scripts and actions. […]

    Pingback by Analyzing Malicious Documents Cheat Sheet | iTeam Developers — Monday 2 February 2015 @ 7:33

247. May anyone help me get all of objects in a PDF file extracted or viewed?

Comment by udhay — Wednesday 18 March 2015 @ 9:12

248. @udhay just run pdf-parser

Comment by Didier Stevens — Wednesday 18 March 2015 @ 9:14

249. Got an error:
C:\Users\root\Downloads>python pdf-parser.py -w pagrindinis_brezinys.pdf
PDF Comment %PDF-1.5

Traceback (most recent call last):
File "pdf-parser.py", line 1201, in
Main()
File "pdf-parser.py", line 1094, in Main
print('PDF Comment %s' % FormatOutput(object.comment, options.raw))
File "C:\Python33\lib\encodings\cp775.py", line 19, in encode
return codecs.charmap_encode(input,self.errors,encoding_map)[0]
UnicodeEncodeError: 'charmap' codec can't encode characters in position 13-15: c
haracter maps to

Comment by Donatas — Thursday 19 March 2015 @ 6:31

250. @Donatas Can you share the PDF?

Comment by Didier Stevens — Thursday 19 March 2015 @ 8:30

251. […] 1℃ 0评论 While attack vectors based on Malicious PDF are a well known topic (SANS, Didier's tools), understanding how those vectors are spread up nowadays is an interesting "research" […]

Pingback by PDF Versions Malicious Content Distribution – ThinfoSec.COM关注通化信息安全 — Tuesday 24 March 2015 @ 7:21

252. […] For about half a year now, I've been adding YARA support to several of my analysis tools. Like pdf-parser. […]

Pingback by pdf-parser And YARA | Didier Stevens — Tuesday 31 March 2015 @ 21:13

253. […] Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parserand make-pdf and […]

Pingback by Python：渗透测试开源项目【源码值得精读】 - 有Bug — Saturday 11 April 2015 @ 5:53

254. […] updated and references updated/removed. To automate this process as much as possible, I updated my pdf-parser program to generate a Python program that in turn, generates the original […]

Pingback by pdf-parser: A Method To Manipulate PDFs Part 1 | Didier Stevens — Thursday 16 April 2015 @ 0:01

255. Hi!

Any chance to support utf-8 in files names? They are attached, but displayed with gibberish in Adobe Reader.

Thanks.

Comment by Anonymous — Friday 17 April 2015 @ 19:05

256. @Anonymous Can you provide more details? What did you do exactly and what is the problem?

Comment by Didier Stevens — Friday 17 April 2015 @ 19:07

257. ——-@Anonymous Can you provide more details?

    It’s me, thanks for quick answer)

    Cent OS 7 64, Python 2.7.5

    I have file with non-english characters in name and embed it with make-make-pdf-embedded.
    Try following (it’s Cyrillic letters)
    ————————————————————–
    echo > йцукенг.txt
    make-pdf-embedded.py -b йцукенг.txt embed.pdf
    ————————————————————-

    Open embed.pdf and see what happen to embedded name.

    Comment by vovodroid — Saturday 18 April 2015 @ 5:19

258. @vovodroid I tried to reproduce your problem, but I’m not making progress. I can create a file called йцукенг.txt, but I can’t issue a command with that name in cmd.exe. I changed the codepage to Cyrillic (855 if I remember correctly), but that did not help.
    I see that you used CentOS. I’ll try that when I have time to install it in a VM.

    Comment by Didier Stevens — Saturday 18 April 2015 @ 20:05

259. Hi!
    ——I can’t issue a command with that name in cmd.exe
    If you use Windows you can just open notepad.exe and paste йцукенг.txt to Save dialog.

    ——I see that you used CentOS
    I guess problem persists in any Linux distribution. It happens in Windows as well, but name is corrupted in other way, due to different file system encoding (UTF16 in Windows and UTF8 in Linux).

    Right name: йцукенг.txt
    LInux: –¹—Ü—É–º–µ–½–³.txt
    Windows: È^ÛÍÂÌ„.txt

    You see, in Linux name is twice in Length because of utf8.

    Thanks.

    Comment by vovodroid — Sunday 19 April 2015 @ 3:42

260. […] some searching, I found Didier’s Steven’s work, realizing I should have looked there first. Didier has PDFid.py for summary analysis and mPDF.py […]

    Pingback by PowerShell | computer security and system designs — Friday 24 April 2015 @ 0:22

261. […] and decompressing a stream (for example containing a JavaScript script) is easy with pdf-parser. You select the object that contains the stream (example object 5: -o 5) and you “filter” the […]

    Pingback by pdf-parser: A Method To Manipulate PDFs Part 2 | Didier Stevens — Wednesday 29 April 2015 @ 0:01

262. Thanks for the great tool.
    I recently tested a 38 pages PDF that after all is not malicious and has been created with PDF-XChange 4.0.194.0
    pdf-parser quits with an error message:
    obj 214 0
    Type: /Annot

Referencing:
Traceback (most recent call last):
File “C:\workdir\PY\PDF\pdf-parser.py”, line 1359, in
Main()
File “C:\workdir\PY\PDF\pdf-parser.py”, line 1307, in Main
PrintObject(object, options)
File “C:\workdir\PY\PDF\pdf-parser.py”, line 999, in PrintObject
PrintOutputObject(object, options)
File “C:\workdir\PY\PDF\pdf-parser.py”, line 773, in PrintOutputObject
oPDFParseDictionary = cPDFParseDictionary(object.content, options.nocanonicalizedoutput)
File “C:\workdir\PY\PDF\pdf-parser.py”, line 635, in __init__
self.parsed = self.ParseDictionary(dataTrimmed)[0]
TypeError: ‘NoneType’ object has no attribute ‘__getitem__’

pdfid states:
PDF Header: %PDF-1.4
obj 216
endobj 216
stream 114
endstream 114
xref 1
trailer 1
startxref 1
/Page 38
/Encrypt 0
/ObjStm 0
/JS 0
/JavaScript 0
/AA 0
/OpenAction 0
/AcroForm 0
/JBIG2Decode 0
/RichMedia 0
/Launch 0
/EmbeddedFile 0
/XFA 0
/Colors > 2^24 0
Thanks

Comment by Michael O — Wednesday 27 May 2015 @ 6:44

263. Can you share this document so that I can check the error?

Comment by Didier Stevens — Wednesday 27 May 2015 @ 19:02

264. Thanks for your reply. Sorry, no I can’t. But I’ll try to find the problem or generate a PDF that produces the same error.

Comment by Michael O — Thursday 28 May 2015 @ 7:12

265. […] PDF tools — поиск и выявление подозрительных объектов в PDF документах, анализ элементов PDF. […]

Pingback by Хакерский дистрибутив на базе Windows. - Cryptoworld — Thursday 25 June 2015 @ 15:48

266. […] this new version of pdf-parser, option -H will now also calculate the MD5 hashes of the unfiltered and filtered stream of selected […]

    Pingback by Update: pdf-parser Version 0.6.4 | Didier Stevens — Thursday 13 August 2015 @ 0:00

267. […] AnalyzePDF, Pdfobjflow, pdfid, pdf-parser, peepdf, Origami, PDF X-RAY Lite, PDFtk, […]

    Pingback by REMnux: Distribución de Linux especializada en en el análisis de malware | Skydeep — Thursday 20 August 2015 @ 1:49

268. I'm using pdf-parser.py parsing a pdf-file and it works great, but in some cases I get the error message "FlateDecode decompress failed, unexpected compression method: fd. zlib.error Error -3 while decompressing data: incorrect header check". Sometimes instead of "fd." it says "0e.".

    As an example: http://pastebin.com/uNpMMDBP

    And I can't really understand how to get around this. Can you help me? Thanks!

    Comment by miggedy — Thursday 24 September 2015 @ 19:00

269. This happens when the Python compression module does not support thet compression method that was used, or when the compressed data is actually not compressed data, or when it is corrupt.
    I recently wrote a SANS Internet Storm Center diary entry showing how to deal with this problem:
    https://isc.sans.edu/forums/diary/Handling+Special+PDF+Compression+Methods/19597/

    Comment by Didier Stevens — Thursday 24 September 2015 @ 19:08

270. Thanks for the quick reply!

    Comment by miggedy — Thursday 24 September 2015 @ 20:56

271. Hi friend, thank you so much for share. It is wonderful tool.

    Comment by eliasbernier — Friday 9 October 2015 @ 21:15

272. Hi,
    All Fedora users can now install 'pdfid' and 'pdf-parser' from my Copr repos: fszymanski/pdfid, fszymanski/pdf-parser.

    Comment by Filip — Tuesday 3 November 2015 @ 9:47

273. […] useful programs were exiftool and peepdf and Didier Steven's pdf-tools. I also used pdfgrep, but I had to download the latest source, and then compile it with the perl […]

    Pingback by Scanning for confidential information on external web servers | The Grymoire — Saturday 6 February 2016 @ 16:51

274. […] Herramientas PDF Didier Stevens : analizar, identificar y crear archivos PDF (incluye PDFiD , pdf-parser y maquillaje pdf y MPDF) […]

    Pingback by Herramientas Python para pruebas de penetración | Blog IhackLabs - Hacking — Wednesday 2 March 2016 @ 10:50

275. […] firmware-mod-kit 固件拆包/组包工具forensics pdf-parser PDF文件挖掘工具forensics scrdec […]

    Pingback by CTF工具集合安装脚本操作姿势 | 安全渗透军火库|SHENTOU.ORG — Saturday 12 March 2016 @ 2:09

276. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and mPDF) […]

    Pingback by 渗透测试之Python工具箱 | 安全小飞侠的窝 — Sunday 13 March 2016 @ 16:49

277. […] Python 编写的PDF文件分析工具，可以帮助检测恶意的PDF文件 Didier Stevens' PDF tools: 分析，识别和创建 PDF 文件(包含PDFiD，pdf-parser，make-pdf 和 mPDF) Opaf: 开放 […]

    Pingback by Python渗透测试工具合集 - 征服者785号征服者785号 — Sunday 27 March 2016 @ 10:30

278. […] Stevens' PDF tools: analiza, identifica y crea ficheros PDF (incluye PDFiD, pdf-parser, make-pdf y […]

    Pingback by Coleccion de herramientas de hacking hechas en Python – Underc0de Blog — Thursday 21 April 2016 @ 12:59

279. […] Didier Stevens' PDF tools: analiza, identifica y crea ficheros PDF (incluye PDFiD, pdf-parser, make-pdf y mPDF) […]

    Pingback by Coleccion de herramientas de hacking hechas en Python – Underc0de Blog — Wednesday 27 April 2016 @ 12:32

280. […] Stevens' PDF tools: analiza, identifica y crea ficheros PDF (incluye PDFiD, pdf-parser, make-pdf y […]

    Pingback by Coleccion De Herramientas De Hacking Hechas En Python – A Security Breach — Saturday 7 May 2016 @ 9:35

281. hey, I love the tools I used them quite a bit.
    I wonder though if it would be difficult to add a feature to add the java script to an existing PDF

    Comment by Anonymous — Thursday 2 June 2016 @ 0:51

282. This is something I teach in my training.

    Comment by Didier Stevens — Sunday 5 June 2016 @ 16:25

283. […] Stevens' PDF tools: analiza, identifica y crea ficheros PDF (incluye PDFiD, pdf-parser,make-pdf y […]

    Pingback by Coleccion De Herramientas De Hacking Hechas En Python - TecnologosRD — Friday 8 July 2016 @ 12:21

284. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and mPDF) […]

    Pingback by Python tools for Penetration Testers | Think outside the Box — Tuesday 12 July 2016 @ 16:47

285. […] PDF Tools – pdfid, pdf-parser, and more from Didier Stevens. […]

    Pingback by Awesome Malware Analysis Lists – vulnerablelife — Wednesday 3 August 2016 @ 18:26

286. […] Didier Stevens' PDF tools: analiza, identifica y crea ficheros PDF (incluye PDFiD, pdf-parser, make-pdf y mPDF) […]

    Pingback by Computo Forense y Hacking » Coleccion de herramientas de hacking hechas en Python — Wednesday 28 September 2016 @ 2:38

287. […] Didier Stevens' PDF tools: 分析，识别和创建 PDF 文件(包含PDFiD，pdf-parser，make-pdf 和 mPDF) Opaf: 开放 PDF 分析框架，可以将 PDF 转化为 XML […]

    Pingback by Python渗透测试工具合集-技术客 — Monday 24 October 2016 @ 2:46

288. […] Python 编写的PDF文件分析工具，可以帮助检测恶意的PDF文件 Didier Stevens' PDF tools: 分析，识别和创建 PDF 文件(包含PDFiD，pdf-parser，make-pdf 和 mPDF) Opaf: 开放 […]

    Pingback by Python渗透测试工具合集 | 技术客 — Monday 7 November 2016 @ 10:23

289. […] new version of pdf-parser is a bugfix for […]

    Pingback by Update: pdf-parser Version 0.6.6 | Didier Stevens — Monday 28 November 2016 @ 0:00

290. […] Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parserand make-pdf and […]

    Pingback by Python：渗透测试开源项目 | 歪布IT笔记 — Tuesday 29 November 2016 @ 16:45

291. […] https://blog.didierstevens.com/programs/pdf-tools/ […]

    Pingback by Introduction to PDF Analysis – RIT Computing Security Blog — Monday 12 December 2016 @ 2:08

292. Didier Great Tool. I have incorporated this tool into our EnCase Integrated Threat Toolkit (EITT) and our customers absolutely love this tool. I have a need for a tool that can parse all office docs looking for embedded objects just like your PDF tools do. Do you currently have a tool that would do this or is there a way to customize this tool to include all office docs.

thanks.

Comment by Mark Morgan — Tuesday 20 December 2016 @ 22:16

293. Yes, my oledump.py tool.

Comment by Didier Stevens — Wednesday 21 December 2016 @ 19:41

294. […] When you receive a suspicious PDF these days, it could be just a scam without malicious code. Let's see how to analyze such samples with PDF Tools. […]

Pingback by PDF Analysis: Back To Basics | NVISO LABS – blog — Wednesday 28 December 2016 @ 11:28

295. Hi Didier,

I recently received a PDF document that I have attempted to analyze using your tools. When opened, it was obvious that the document has a link to a credential harvesting site that it tempts users to click on. However, using pdf-parser, I am unable to locate the URI object. I attempted to decompress the 4 object streams but received errors relating to unexpected compression method. I then attempted to follow the method you posted regarding the handling of special PDF compression methods but also to no avail. Is this a new technique or is there something I have missed? Of note, there appears to be some form of DRM/encryption also applied as there are also 2 /Encrypt objects. I have uploaded the file to VT (SHA 256: 7d2b615630efd2fa3713d97e57afb9972f43e7d4a67cc706af7c789dd1dbe47f) if you are interested in taking a look.

Tom

Comment by Tom — Thursday 5 January 2017 @ 18:54

296. You need to decrypt the PDF with a tool like qpdf.

Comment by Didier Stevens — Monday 9 January 2017 @ 20:03

297. […] Didier Stevens'ın PDF Araçları – PDF dosyalarını analiz, tanıma ve yaratma (PDFiD, pdf-parser, mPDF ve make-pdf dahil) […]

Pingback by Python ve Güvelik Modülleri - Python Türkiye — Monday 23 January 2017 @ 21:19

298. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser andmake-pdf and mPDF) […]

Pingback by Python Tools – Toor — Sunday 26 February 2017 @ 8:52

299. […] of the name /JavaScript. However it is easy to write a program that normalizes obfuscated names (pdfid does this for […]

Pingback by Developing complex Suricata rules with Lua – part 1 | NVISO LABS – blog — Friday 10 March 2017 @ 7:46

300. […] Files: PDF pdfid pdfid Locate common suspicious artifacts in a PDF file remnux-didier (APT) https://blog.didierstevens.com/programs/pdf-tools/ Examine Document Files: PDF Pdfobjflow pdf-parser.py | pdfobjflow.py Visualizes the output from […]

Pingback by Remnux-A tool for reverse engineering Malware – Infohub — Saturday 8 April 2017 @ 22:40

301. […] and reading I came accross Didier Stevens's blog, which contained information on a bunch PDF tools he had written and how to use them. After some fiddling and searching, I found some JavaScript […]

Pingback by ZonkSec - DakotaCon 2017 CTF Write Ups — Wednesday 12 April 2017 @ 17:53

302. […] document 123-148752488-reg-invoice.pdf is a PDF with an embedded file and JavaScript. Here is pdfid's […]

Pingback by Malicious Documents: The Matryoshka Edition | Didier Stevens — Thursday 20 April 2017 @ 0:02

303. […] Tools: make-pdf tools […]

Pingback by Bash Bunny Dropping PDF Via HID | Didier Stevens Videos — Saturday 22 April 2017 @ 22:32

304. […] pdf tools, oledump.py, […]

    Pingback by Malicious Documents: The Matryoshka Edition | Didier Stevens Videos — Sunday 23 April 2017 @ 19:36

305. […] I create a pure ASCII PDF file with an embedded executable using my make-pdf-embedded.py […]

    Pingback by Bash Bunny PDF Dropper | Didier Stevens — Monday 24 April 2017 @ 0:00

306. […] make-pdf 0.1.5 This tool will embed javascript inside a PDF document. https://blog.didierstevens.com/programs/pdf-tools/ […]

    Pingback by List of some Penetration Testing Tools – Doxsec — Wednesday 26 April 2017 @ 20:40

307. […] https://blog.didierstevens.com/programs/pdf-tools/ […]

    Pingback by CH Magazine | Content-Type Attack: Dark Hole in a Secure Environment — Friday 12 May 2017 @ 4:13

308. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and mPDF) […]

    Pingback by Python for penetration testers – vulnerablelife — Saturday 13 May 2017 @ 5:03

309. […] pdf-parser.py from https://blog.didierstevens.com/programs/pdf-tools/ location to be set in first line of […]

    Pingback by DATA - Credential Phish Analysis and Automation - Sapsi Security Services — Wednesday 7 June 2017 @ 17:16

310. Hi and thanks for sharing such tools.
    Unfortunately when I try pdf_parser I always get the following error even when calling it with just -h argument :
    pdf-parser.py", line 561
    except zlib.error as e:
    ^
    SyntaxError: invalid syntax

    OS : Windows10
    Python 2.6.6

    Do I have to install something else ?
    Cheers.

    Comment by AdV — Tuesday 25 July 2017 @ 16:12

311. Can you try with the latest version of Python 2.7? I.e. 2.7.13

    Comment by Didier Stevens — Tuesday 25 July 2017 @ 16:15

312. Thanks for fast reply !
    Sorry Didier I probably didn't type in the correct syntax.
    I tried again with full path to Python.exe and it worked (at least with -h).
    I'm gonna try on my pdf now.

    Sorry for that unuseful comment and you might want to delete it. No problem.
    Cheers.

    Comment by AdV — Tuesday 25 July 2017 @ 16:23

313. Hi Didier,
    I'm wondering if you've released any new versions of pdfid.py and pdf-parser.py? Kali folks just released, for free, "Kali linux revealed" and I wanted to take a look, however, pdfid.py hangs while trying to analyze this file. After ctrl-c this is the output I get:

```
$ python pdfid.py Kali_Revealed_1st_edition (1).pdf
^CTraceback (most recent call last):
File "pdfid.py", line 930, in
Main()
File "pdfid.py", line 927, in Main
PDFiDMain(filenames, options)
File "pdfid.py", line 885, in PDFiDMain
ProcessFile(filename, options, plugins)
File "pdfid.py", line 704, in ProcessFile
xmlDoc = PDFiD(filename, options.all, options.extra, options.disarm, options.force)
File "pdfid.py", line 513, in PDFiD
attErrorOccured.nodeValue = 'True'
AttributeError: 'NoneType' object has no attribute 'nodeValue'
```

I re-downloaded the file, so I don't think it's corrupt… Maybe some new features pdf format added that aren't accounted for in pdfid.py?

Comment by Gene — Thursday 27 July 2017 @ 0:34

314. Can you rename the file (delete " (1)" from the filename) and try again?

Comment by Didier Stevens — Friday 28 July 2017 @ 17:26

315. Didier,
I edited the path out of the name, when I posted, and forgot to get rid of the (1). I downloaded this twice, just to make sure, so that's why one name was changed by the browser. I ran the utility against both filenames, the original name without the (1) and the one with (1), same result. The file name was in quotes when I ran this in the shell, just to re-iterate. It is failing because of some pdf format, not because of a file name.

Comment by Gene — Wednesday 2 August 2017 @ 7:26

316. Sorry, but I can not reproduce your problem:

```
PDFiD 0.2.1 Kali_Revealed_1st_edition.pdf
PDF Header: %PDF-1.4
obj 3304
endobj 3304
stream 486
endstream 486
xref 1
trailer 1
startxref 1
/Page 344
/Encrypt 0
/ObjStm 0
/JS 0
/JavaScript 0
/AA 0
/OpenAction 0
/AcroForm 0
/JBIG2Decode 0
/RichMedia 0
/Launch 0
```

/EmbeddedFile 0
/XFA 0
/Colors > 2^24 0

The MD5 of the file is 40CD00C451F9037A32352031CBED84E5, check if you have the same file.

Comment by Didier Stevens — Wednesday 2 August 2017 @ 17:32

317. Hey Didier,
Great work, BTW. I had a suggestion for what I think would be a useful feature for pdfid. In addition to the strings you're currently counting, also count "/URI (http". I think that all of the malicious PDF files I've seen for the last couple of years have just been vehicles to get malicious links past email filtering. It would be useful as well, to actually parse out the links, as pdf-parser does, but that's probably beyond your intended scope for this tool. Another possible alternate way to do this would be to count only 'suspicious' http URI values, such as those using bare IP addresses, shortened URLs, or other criteria.
Thanks
John McCash

Comment by John McCash — Thursday 21 September 2017 @ 15:21

318. […] a tool such as pdfid.py or peepdf.py to perform some reckon on the documents (e.g. suspicious tags, potential JavaScript […]

Pingback by (Not) All She Wrote: Rigged PDFs | Security Over Simplicity — Thursday 28 September 2017 @ 17:18

319. […] Didier Stevens' PDF tools: analyse, identify and create PDF files (includes PDFiD, pdf-parser and make-pdf and mPDF) […]

Pingback by Python for penetration testers – CISO Tunisia — Sunday 22 October 2017 @ 11:23

320. Didier – I'm tying to use pdf-parser.py to extract images from some small pdf documents. So far I've been able to use –stats to find the /XObjects, and -o to save those as files. The images in the documents I am playing with are very small, and are embedded withing the text. Is there a way to expose/extract the location of the bounding box (or otherwise locate the text surrounding the images)?

Comment by Joey Quinn — Tuesday 14 November 2017 @ 23:37

321. […] PDF Tools by Didier Stevens […]

Pingback by Checking for maliciousness in Acroform objects on PDF files – Furoner.CAT — Wednesday 15 November 2017 @ 15:22

322. Yes, you have to look into the stream of the objects that put the images on the page.

Comment by Didier Stevens — Thursday 16 November 2017 @ 9:27

323. […] this new version of pdfid.py, a new option was added: […]

Pingback by Update: pdfid.py Version 0.2.3 | Didier Stevens — Monday 27 November 2017 @ 0:00

324. […] PDF Tools de Didier Stevens. PDFStreamDumper – Esta es una herramienta gratuita para el análisis PDFs maliciosos. SWF Mastah – Programa en Python que extrae stream SWF de ficheros PDF. […]

Pingback by Forensics PowerTools (Listado de herramientas forenses) – Securiza Neuquen — Wednesday 13 December 2017 @ 0:33

325. […] pdfid.py confirms the PDF is encrypted (name /Encrypt): […]

Pingback by Cracking Encrypted PDFs – Part 1 | Didier Stevens — Tuesday 26 December 2017 @ 17:15

326. […] keys, you can always check the /Encrypt dictionary of the PDF you created, for example with my pdf-parser (in this example /Length 128 tells us a 128-bit key is […]

Pingback by Cracking Encrypted PDFs – Conclusion | Didier Stevens — Friday 29 December 2017 @ 0:00

327. […] Tools: pdfid.py, pdf-parser.py […]

    Pingback by PDF's /URI – Didier Stevens Videos — Sunday 31 December 2017 @ 17:00

328. Getting these errors. I have tried doing some stuff suggested here with no luck

    C:\Python26>python pdfid.py sampls/pdf-thisCreator.file
    'module' object has no attribute 'OrderedDict'

    C:\Python26>python pdfid.py samples/pdf-thisCreator.file
    'module' object has no attribute 'OrderedDict'

    C:\Python26>pdfid.py BouncingButton.pdf
    'module' object has no attribute 'OrderedDict'

    Comment by Mick — Sunday 4 February 2018 @ 18:46

329. Try Python 2.7

    Comment by Didier Stevens — Sunday 4 February 2018 @ 19:46

330. That worked,

    C:\Python27>python pdfid.py samples/pdf-thisCreator.file

    C:\Python27>

    I have pdf files in the folder that have malicious files in them

    Comment by morrim03 — Sunday 4 February 2018 @ 23:37

331. […] Didier's PDF Tools […]

    Pingback by Most Important Tools and Resources For Security Researcher, Malware Analyst, Reverse Engineer – My (Yet Another) Cybersecurity Blog — Wednesday 28 February 2018 @ 8:13

332. Using latest version of pdfid.py (0.2.4) and noticed that it's looking for /AA and /JS inside of images within the XMP packet (). If –disarm is used, it will replace those occurrences and break the images.

    Comment by Kerri — Saturday 7 April 2018 @ 7:46

333. Please read this: https://blog.didierstevens.com/2013/06/10/pdfid-false-positives/

    Comment by Didier Stevens — Saturday 7 April 2018 @ 7:55

RSS feed for comments on this post. TrackBack URI

Leave a Reply (comments are moderated)

Enter your comment here...