black hat
EUROPE 2014

PDF ATTACK

A Journey from the Exploit Kit to the Shellcode

Jose Miguel Esparza

InTELL
BY FOX IT

Who am I?

- Jose Miguel Esparza
- Pamplona (Spain)

Who am I?

- Jose Miguel Esparza
- Lead Threat Analyst at Fox-IT InTELL
  - Malware, Botnets, C&Cs, Exploit Kits, …
- Security Researcher at Home ;p
  - PDF, NFC, Malware (again) …
- http://eternal-todo.com
- @EternalTodo on Twitter

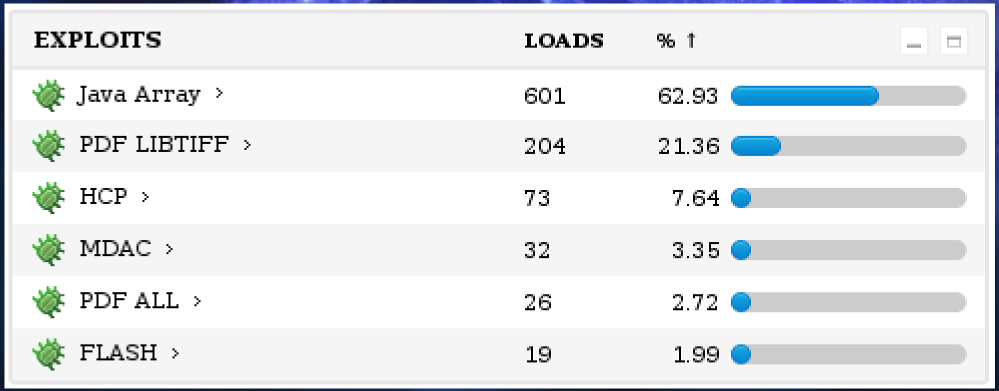
Agenda

- A Journey from the Exploit Kit to the Shellcode
  - Exploit Kits: the source of evil
  - PDF basics
  - Some basic peepdf commands
  - Analyzing PDF exploits
    - Extracting and analyzing shellcodes
  - Obfuscation of PDF files

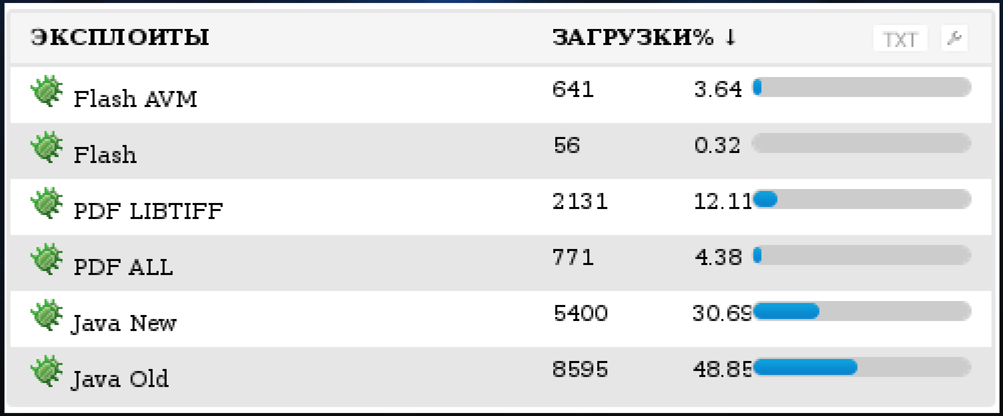
Exploit Kits: the source of evil

- Best way to infect a computer
- Effective and fresh exploits
  - IE
  - Java
  - PDF
  - Flash
  - …
- Average of 6-7 exploits

Exploit Kits: the source of evil



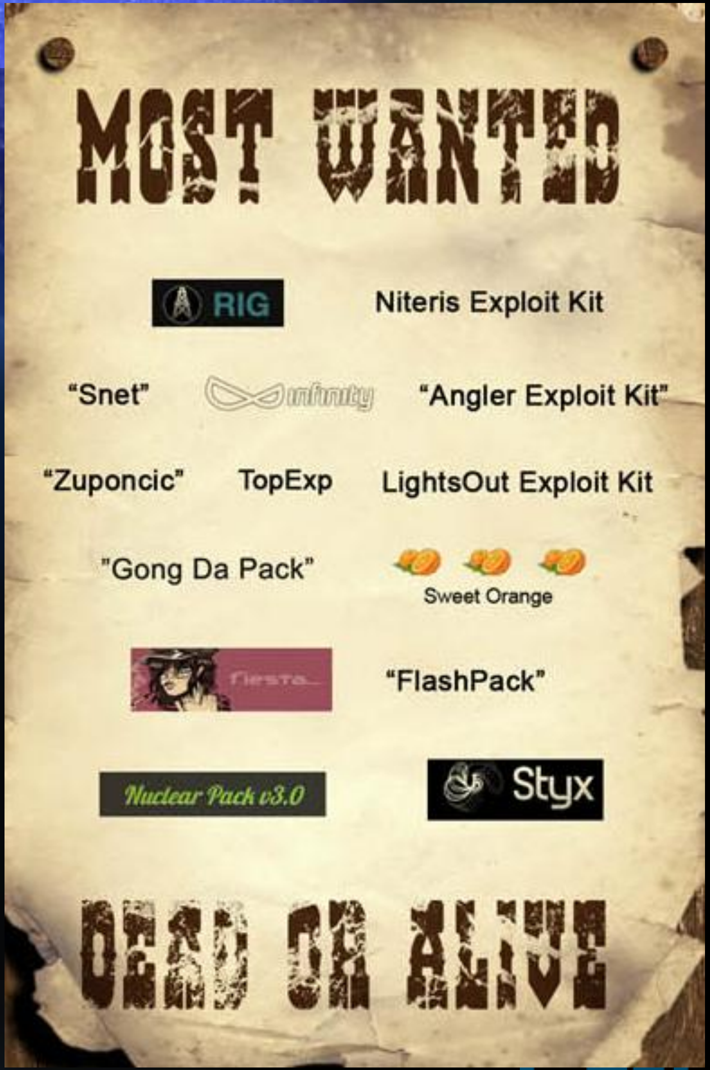
EXPLOITS	LOADS	% ↑
Java Array ›	601	62.93
PDF LIBTIFF ›	204	21.36
HCP ›	73	7.64
MDAC ›	32	3.35
PDF ALL ›	26	2.72
FLASH ›	19	1.99



ЭКСПЛОИТЫ	ЗАГРУЗКИ% ↓	
Flash AVM	641	3.64
Flash	56	0.32
PDF LIBTIFF	2131	12.11
PDF ALL	771	4.38
Java New	5400	30.69
Java Old	8595	48.85

Exploit Kits: the source of evil

- Most used nowadays
  - RIG
  - Angler
  - Sweet Orange
  - Magnitude (TopExp)
  - Fiesta
  - Infinity (Goon/RedKit v2)
  - Styx
  - Nuclear
  - …



Jose Miguel Esparza
@EternalTodo

KahuSecurity

InTELL BY FOX IT

Exploit Kits: the source of evil

- Infection steps
  - Visit injected website / Click SPAM link
  - Redirection (maybe more than one)
  - Obfuscated Javascript
  - Plugin detection
  - Trying exploits
  - Done!

Exploit Kits: the source of evil

- Traffic Distribution Systems (TDS)
  - Country specific attacks
  - TDS + Exploit Kits = WIN!

Exploit Kits: the source of evil

- Analyzing exploit kits
  - Avoiding researchers
    - Filtering by *User-Agent / Referer / Cookies*
    - Blocking IPs
    - One-time infections
    - Country filters

Exploit Kits: the source of evil

- Analyzing obfuscated Javascript code
  - The “easy” way
    - Automatic tools
      - Online services
        - » Wepawet
        - » JSUNPACK
      - Low-interaction honeyclient
        - » Thug
    - You can miss some info

Exploit Kits: the source of evil

- Analyzing obfuscated Javascript code
  - The “easy” way
    - Automatic tools
      - Online services
        - » Wepawet
        - » JSUNPACK
      - Low-interaction honeyclient
        - » Thug
          - Recently added PDF analysis with peepdf!! ;)
    - You can miss some info

Exploit Kits: the source of evil

- Analyzing obfuscated Javascript code
  - The “easy” way
    - Virtual Machine
      - Visit the link in your controlled system
        - Good to obtain the payload/exe (if it exists)
    - You can miss some info too
      - Which exploit is being used?

Exploit Kits: the source of evil

- Analyzing obfuscated Javascript code
  - The traditional (manual) way
    - Executing different stages of JS code
      - Beautify the code
      - Understand the code
      - Looking for the *eval* function
        - » s/*eval*/*print*/
      - Hooking the *eval* function with Javascript engines
    - Looking for exploits / shellcodes
    - You cannot miss any detail

Exploit Kits: the source of evil

- Analyzing obfuscated Javascript code
  - The traditional way
    - Let’s play ;)
      - Beautify
      - Review the code
      - Remove HTML / Garbage
      - Execute it
      - Fix errors
      - …

Exploit Kits: the source of evil

- Analyzing obfuscated Javascript code
  - The traditional way
    - Let's play ;)
      - Beautify again
      - Find the shellcode
      - Find the exploit URLs
      - Download the exploits

Exploit Kits: the source of evil

- Analyzing obfuscated Javascript code
  - The traditional way
    - Let’s play ;)
      - Your own scripts

Exploit Kits: the source of evil

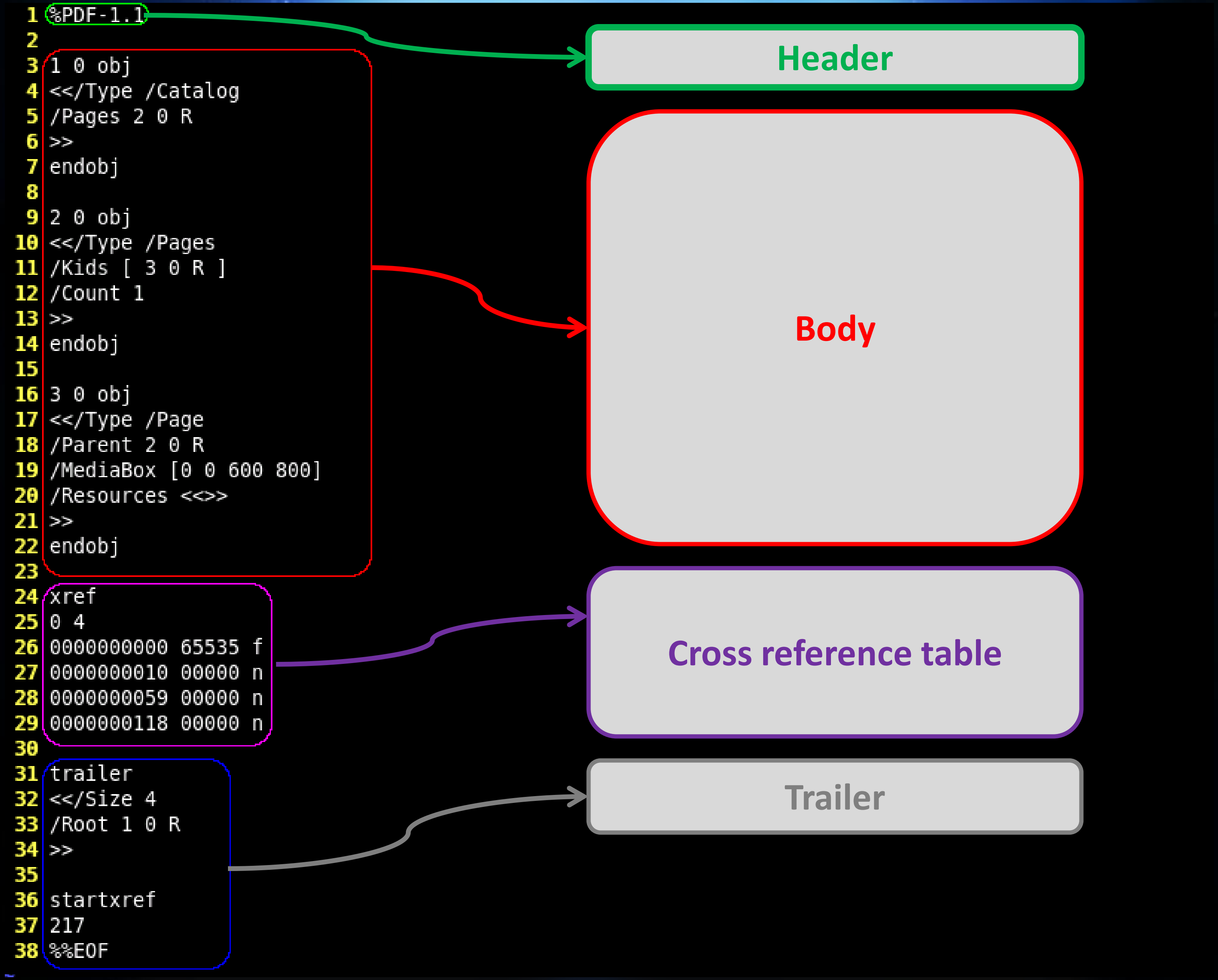
- Analyzing obfuscated Javascript code
  - The traditional way
    - Let’s play ;)

Exploit Kits: the source of evil

- Analyzing exploits
  - Java
    - Decompiler + Code review
    - IDE?
  - PDF
    - I will show you in some minutes… ;)

PDF basics

- PDF format?
- PDF structure?
- Objects?
- Filters?

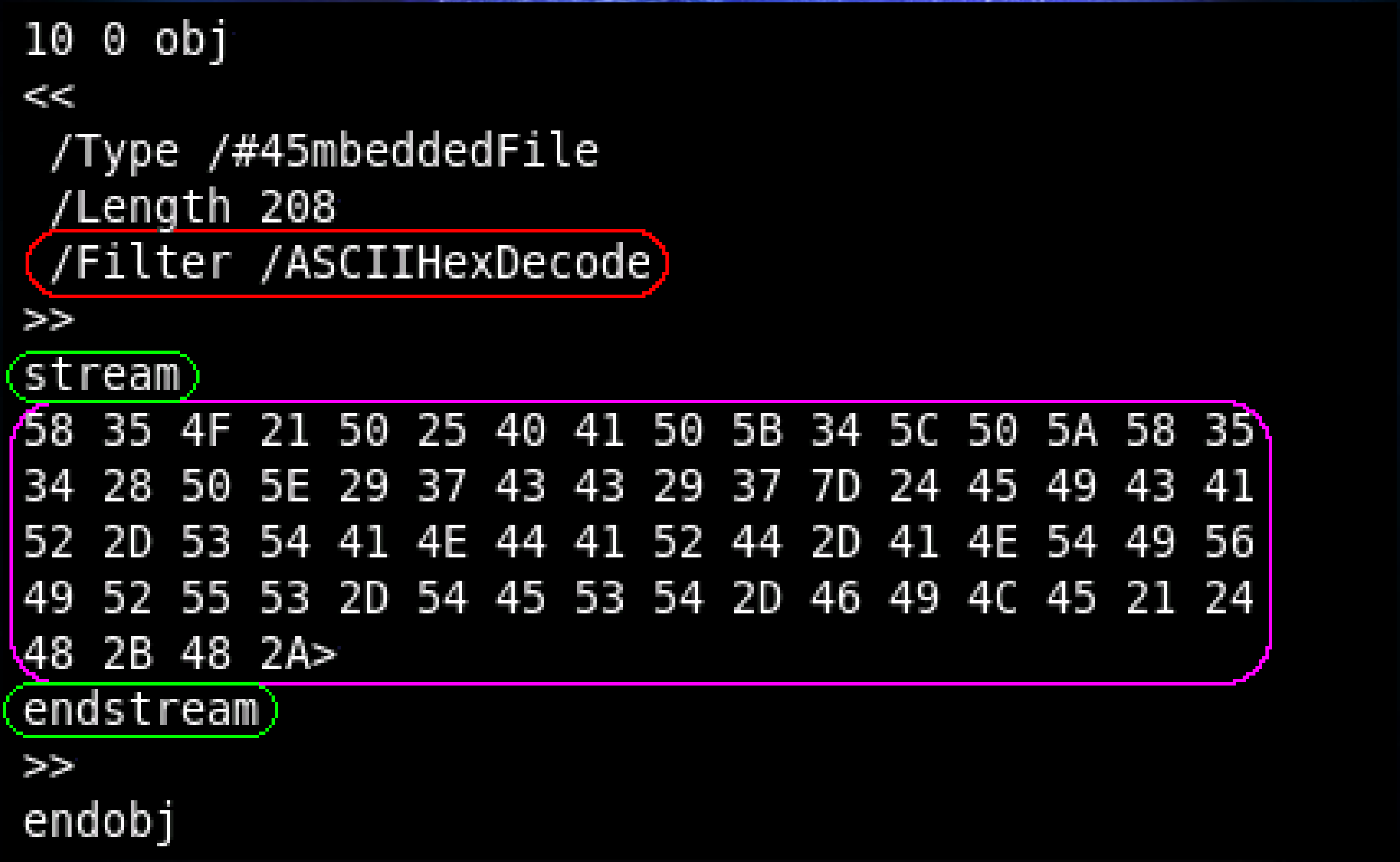


```
%PDF-1.1

1 0 obj
<</Type /Catalog
/Pages 2 0 R
>>
endobj

2 0 obj
<</Type /Pages
/Kids [ 3 0 R ]
/Count 1
>>
endobj

3 0 obj
<</Type /Page
/Parent 2 0 R
/MediaBox [0 0 600 800]
/Resources <<>>
>>
endobj

xref
0 4
0000000000 65535 f
0000000010 00000 n
0000000059 00000 n
0000000118 00000 n

trailer
<</Size 4
/Root 1 0 R
>>

startxref
217
%%EOF
```

PDF basics

- Body
  - Sequence of objects
  - Object types
    - Boolean: *true false*
    - Numbers: 123 -98 4. -.002 123.6
    - Strings: *(hola) <686f6c61>*
      - *68 (h) 6f (o) 6c (l) 61 (a)*
    - Names: */Type /Filter*
    - Dictionaries: *<< /Type /Catalog /Root 1 0 R >>*
    - Arrays: *[1.0 (test) <</Length 273>>]*
    - Streams

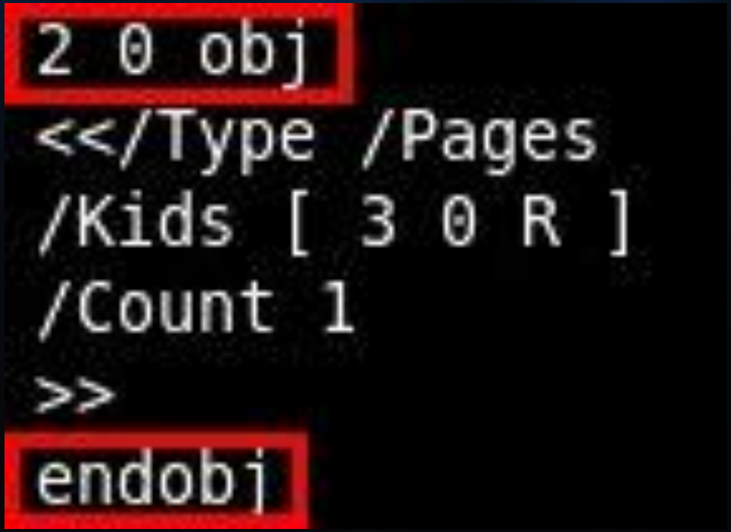
PDF basics



```
10 0 obj
<<
 /Type /#45mbeddedFile
 /Length 208
 /Filter /ASCIIHexDecode
>>
stream
58 35 4F 21 50 25 40 41 50 5B 34 5C 50 5A 58 35
34 28 50 5E 29 37 43 43 29 37 7D 24 45 49 43 41
52 2D 53 54 41 4E 44 41 52 44 2D 41 4E 54 49 56
49 52 55 53 2D 54 45 53 54 2D 46 49 4C 45 21 24
48 2B 48 2A>
endstream
>>
endobj
```

PDF basics

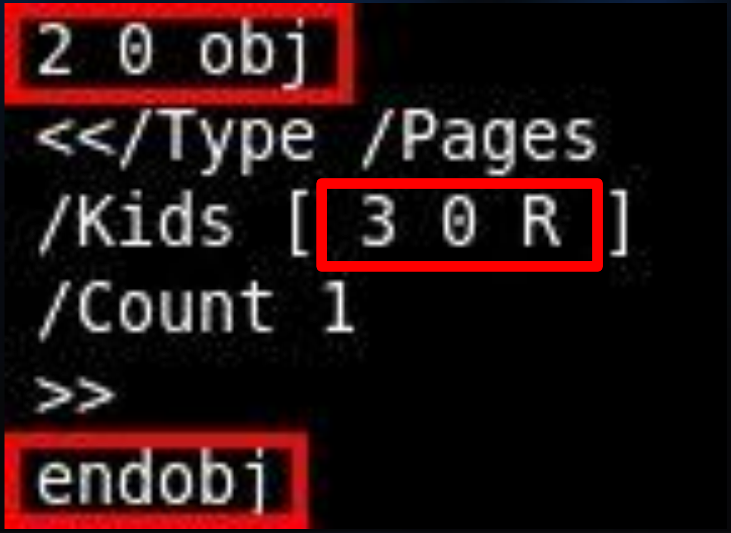
- Object types
  - Indirect objects
    - Reference: “object_id generation_number R”



```
2 0 obj
<</Type /Pages
/Kids [ 3 0 R ]
/Count 1
>>
endobj
```

PDF basics

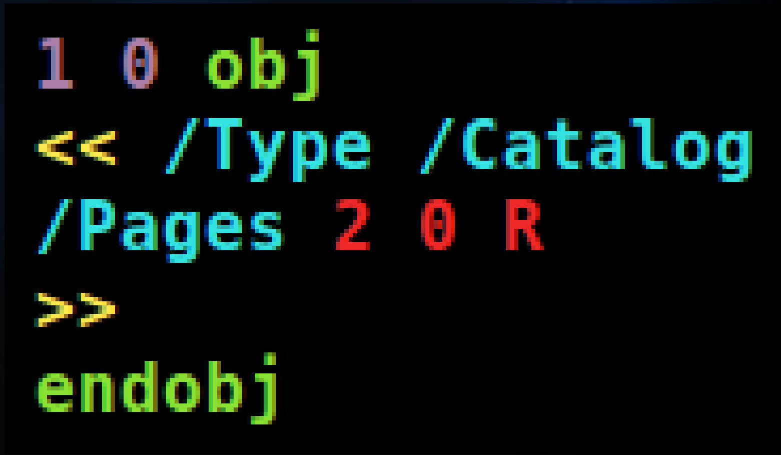
- Object types
  - Indirect objects
    - Reference: “object_id generation_number R”



```
2 0 obj
<</Type /Pages
/Kids [ 3 0 R ]
/Count 1
>>
endobj
```

PDF basics

- Tree structure → References
- Root node
  - /Catalog
- If an element isn't in the downward path from the /Catalog **DOES NOT EXIST**



```
1 0 obj
<< /Type /Catalog
/Pages 2 0 R
>>
endobj
```

PDF basics

- You can use just a text editor!!

peepdf

“peepdf sounds like the Swiss army knife of PDF security apps”

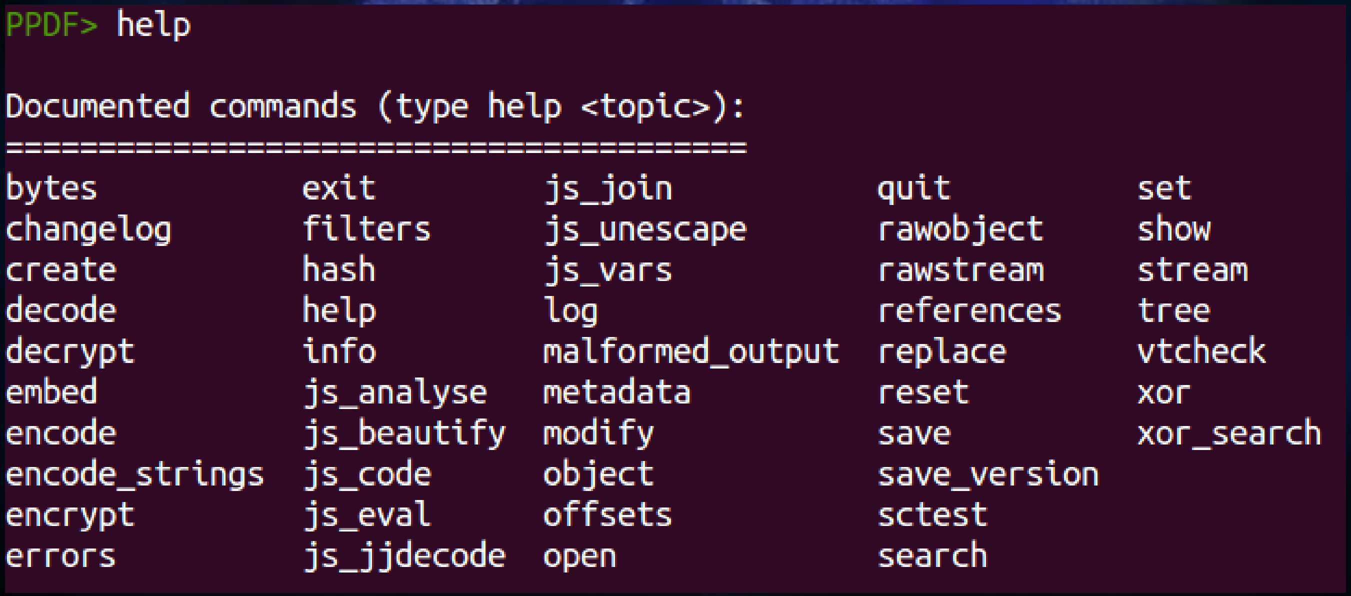
http://peepdf.eternal-todo.com

peepdf

- Characteristics
  - Python
  - Command line
  - Interactive console (colorized)
  - Included in REMnux and BackTrack / Kali Linux

http://peepdf.eternal-todo.com

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
```

http://peepdf.eternal-todo.com

peepdf

- Characteristics
  - Command file option
    - Batch / Automation
  - XML output
  - Easily updated from repository

http://peepdf.eternal-todo.com

peepdf

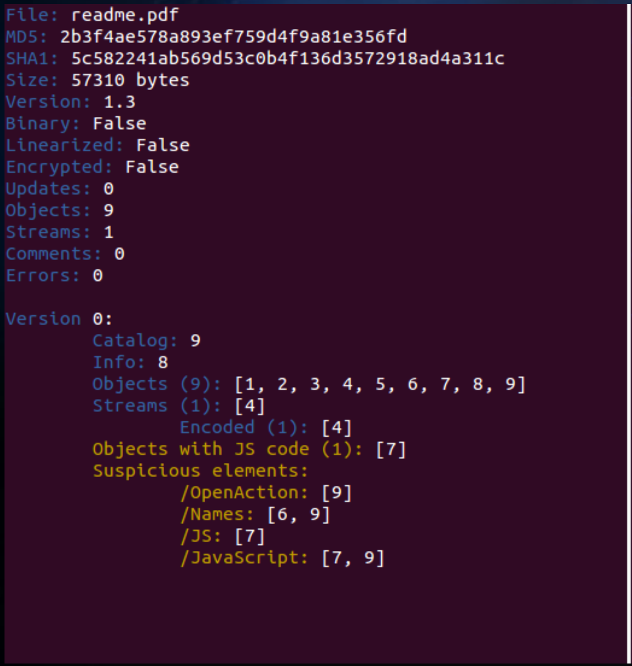
- Why peepdf?
  - Support for:
    - Encryption
    - Object Streams (compressed objects)
    - Most used filters
    - FlateDecode / LZWDecode Parameters
  - Javascript Analysis
  - Shellcode emulation

peepdf

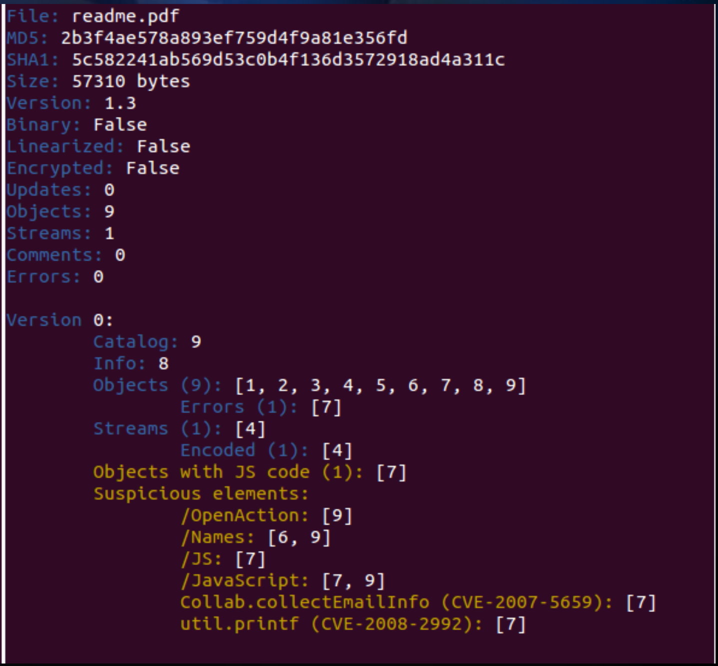
- Why peepdf?
  - Shows Suspicious Elements
  - Shows potential Vulnerabilities
  - Powerful Interactive Console
  - Easy extraction of objects / JS code / shellcode
  - PDF Obfuscation
  - Alive project!!

peepdf

- Important commits
  - s/Spidermonkey/PyV8/g



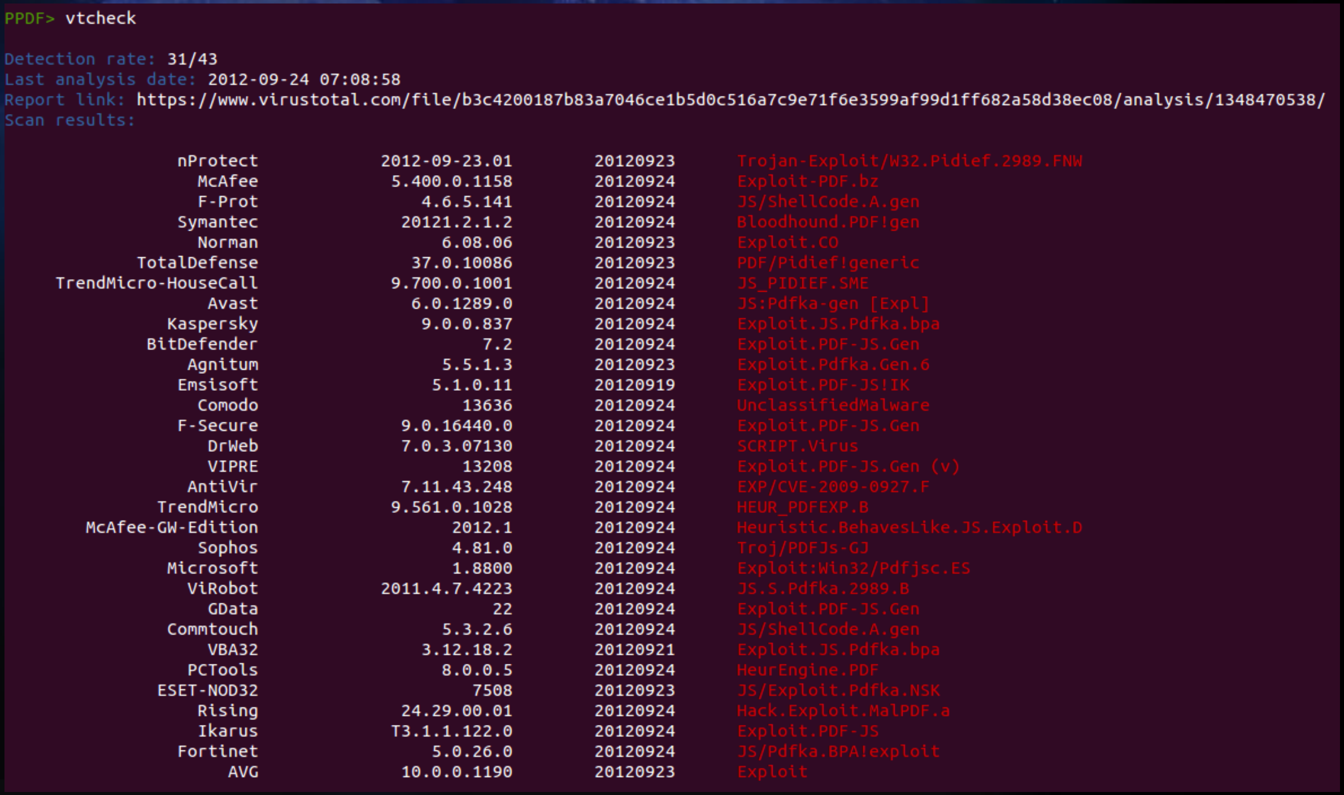
```
File: readme.pdf
MD5: 2b3f4ae578a893ef759d4f9a81e356fd
SHA1: 5c582241ab569d53c0b4f136d3572918ad4a311c
Size: 57310 bytes
Version: 1.3
Binary: False
Linearized: False
Encrypted: False
Updates: 0
Objects: 9
Streams: 1
Comments: 0
Errors: 0

Version 0:
        Catalog: 9
        Info: 8
        Objects (9): [1, 2, 3, 4, 5, 6, 7, 8, 9]
        Streams (1): [4]
                Encoded (1): [4]
        Objects with JS code (1): [7]
        Suspicious elements:
                /OpenAction: [9]
                /Names: [6, 9]
                /JS: [7]
                /JavaScript: [7, 9]
```



```
File: readme.pdf
MD5: 2b3f4ae578a893ef759d4f9a81e356fd
SHA1: 5c582241ab569d53c0b4f136d3572918ad4a311c
Size: 57310 bytes
Version: 1.3
Binary: False
Linearized: False
Encrypted: False
Updates: 0
Objects: 9
Streams: 1
Comments: 0
Errors: 0

Version 0:
        Catalog: 9
        Info: 8
        Objects (9): [1, 2, 3, 4, 5, 6, 7, 8, 9]
                Errors (1): [7]
        Streams (1): [4]
                Encoded (1): [4]
        Objects with JS code (1): [7]
        Suspicious elements:
                /OpenAction: [9]
                /Names: [6, 9]
                /JS: [7]
                /JavaScript: [7, 9]
                Collab.collectEmailInfo (CVE-2007-5659): [7]
                util.printf (CVE-2008-2992): [7]
```

peepdf

- Important commits
  - vtcheck



```
PPDF> vtcheck

Detection rate: 31/43
Last analysis date: 2012-09-24 07:08:58
Report link: https://www.virustotal.com/file/b3c4200187b83a7046ce1b5d0c516a7c9e71f6e3599af99d1ff682a58d38ec08/analysis/1348470538/
Scan results:

                 nProtect            2012-09-23.01        20120923      Trojan-Exploit/W32.Pidief.2989.FNW
                   McAfee             5.400.0.1158        20120924      Exploit-PDF.bz
                   F-Prot                4.6.5.141        20120924      JS/ShellCode.A.gen
                 Symantec              20121.2.1.2        20120924      Bloodhound.PDF!gen
                   Norman                  6.08.06        20120923      Exploit.CO
             TotalDefense               37.0.10086        20120923      PDF/Pidief!generic
     TrendMicro-HouseCall             9.700.0.1001        20120924      JS_PIDIEF.SME
                    Avast               6.0.1289.0        20120924      JS:Pdfka-gen [Expl]
                Kaspersky                9.0.0.837        20120924      Exploit.JS.Pdfka.bpa
              BitDefender                      7.2        20120924      Exploit.PDF-JS.Gen
                  Agnitum                  5.5.1.3        20120923      Exploit.Pdfka.Gen.6
                 Emsisoft                 5.1.0.11        20120919      Exploit.PDF-JS!IK
                   Comodo                    13636        20120924      UnclassifiedMalware
                 F-Secure              9.0.16440.0        20120924      Exploit.PDF-JS.Gen
                    DrWeb              7.0.3.07130        20120924      SCRIPT.Virus
                    VIPRE                    13208        20120924      Exploit.PDF-JS.Gen (v)
                  AntiVir              7.11.43.248        20120924      EXP/CVE-2009-0927.F
               TrendMicro             9.561.0.1028        20120924      HEUR_PDFEXP.B
        McAfee-GW-Edition                   2012.1        20120924      Heuristic.BehavesLike.JS.Exploit.D
                   Sophos                   4.81.0        20120924      Troj/PDFJs-GJ
                Microsoft                   1.8800        20120924      Exploit:Win32/Pdfjsc.ES
                  ViRobot            2011.4.7.4223        20120924      JS.S.Pdfka.2989.B
                    GData                       22        20120924      Exploit.PDF-JS.Gen
                Commtouch                  5.3.2.6        20120924      JS/ShellCode.A.gen
                    VBA32                3.12.18.2        20120921      Exploit.JS.Pdfka.bpa
                  PCTools                  8.0.0.5        20120924      HeurEngine.PDF
               ESET-NOD32                     7508        20120923      JS/Exploit.Pdfka.NSK
                   Rising              24.29.00.01        20120924      Hack.Exploit.MalPDF.a
                   Ikarus             T3.1.1.122.0        20120924      Exploit.PDF-JS
                 Fortinet                 5.0.26.0        20120924      JS/Pdfka.BPA!exploit
                      AVG              10.0.0.1190        20120923      Exploit
```

peepdf

- Important commits
  - js_vars
  - js_jjdecode

peepdf

- Commands
  - Console
    - help
    - log
    - open
    - reset
    - quit
    - exit

peepdf

- Commands
  - Showing information
    - Whole document
      - info
      - tree
      - offsets
      - hash
      - bytes
      - metadata
      - changelog
      - save_version
      - errors

peepdf

- Commands
  - Showing information
    - Objects
      - object
      - rawobject
      - stream
      - rawstream
      - references
      - hash

peepdf

- Commands
  - Extracting information
    - Shell redirection ;)
      - Files
        - » stream 6 > stream6_file
        - » js_code 12 >> pdf_js_code_file
      - Variables
        - » js_unescape variable myVar $> unescaped_sh
        - » rawstream 5 $>> all_my_rawstreams_var

peepdf

- Commands
  - Javascript functions
    - js_code
    - js_eval
    - js_analyse
    - js_unescape
    - js_join
    - js_vars
    - js_jjdecode

peepdf

- Commands
  - Shellcode emulation
    - sctest
      - pylibemu: libemu wrapper for Python

peepdf

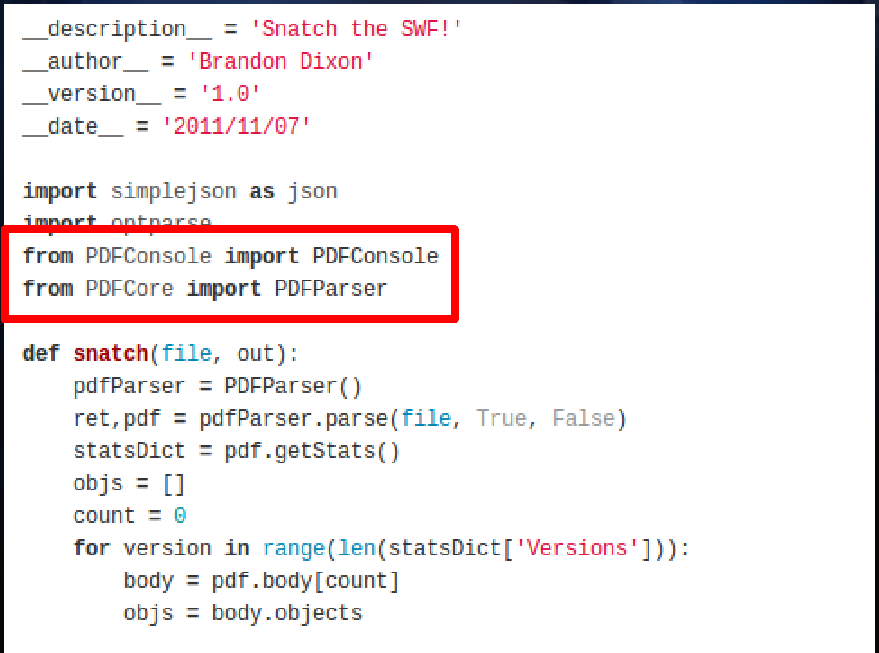
- Commands
  - Modification / Creation
    - modify
    - filters
    - decode
    - encode
    - encode_strings
    - embed
    - encrypt
    - malformed_output
    - create
    - save

peepdf

- Commands
  - Misc
    - set
    - search
    - show
    - xor
    - xor_search

Using peepdf as a library

- Some developments based on peepdf
  - SWF Mastah (Brandon Dixon)



```
__description__ = 'Snatch the SWF!'
__author__ = 'Brandon Dixon'
__version__ = '1.0'
__date__ = '2011/11/07'

import simplejson as json
import optparse
from PDFConsole import PDFConsole
from PDFCore import PDFParser

def snatch(file, out):
    pdfParser = PDFParser()
    ret,pdf = pdfParser.parse(file, True, False)
    statsDict = pdf.getStats()
    objs = []
    count = 0
    for version in range(len(statsDict['Versions'])):
        body = pdf.body[count]
        objs = body.objects
```

peepdf

- Some basic commands…

Analyzing PDF exploits

- How to identify malicious files
  - Suspicious elements
    - /Action
    - /OpenAction
    - /AA
    - /AcroForm
    - /Names
    - /JavaScript
    - /EmbeddedFile
    - Known vulnerabilities

Analyzing PDF exploits

- Most used vulnerabilities
  - LibTiff (TIFF images)
  - Collab.collectEmailInfo
  - Collab.getIcon
  - Doc.media.newPlayer
  - …

Analyzing PDF exploits

- How to identify malicious files
  - Obfuscation
    - Strange codification in objects
    - Encryption
    - Malformed objects
    - Embeded PDFs
    - Javascript

Analyzing PDF exploits

- How to identify malicious files
  - Patterns
    - One page without content
    - Big objects
    - Gaps between objects (offsets)
    - Strange structure
    - Characteristic strings
      - Metadata
      - Tools

Analyzing PDF exploits

- How to identify malicious files
  - Malformed documents
    - Headers
    - Objects Tags

Analyzing real exploits

- Practicing all the theory
- Not a sample exploit, a real one
- Extracting the interesting parts
- Extracting the shellcode
- Analyzing the shellcode

Analyzing real exploits

- Playing with real exploits

PDF obfuscation

- Remove characteristic strings
- Split up Javascript code (/Names)
- If the code is in:
  - String ⟶ octal encoding (\143\172)
  - Stream ⟶ filters (not usual, parameters)
- Compress (object streams)
- Encrypt (default password)
- Malform (endobj, header)
- Nest PDFs

Thanks

- Franc Ruiz @francruar
- Kahu Security @kahusecurity
- Black Hat
- Fox-IT
- Security research community

black hat EUROPE 2014

THANKS!!

Jose Miguel Esparza
jesparza AT eternal-todo.com
http://eternal-todo.com
@EternalTodo

InTELL BY FOX IT