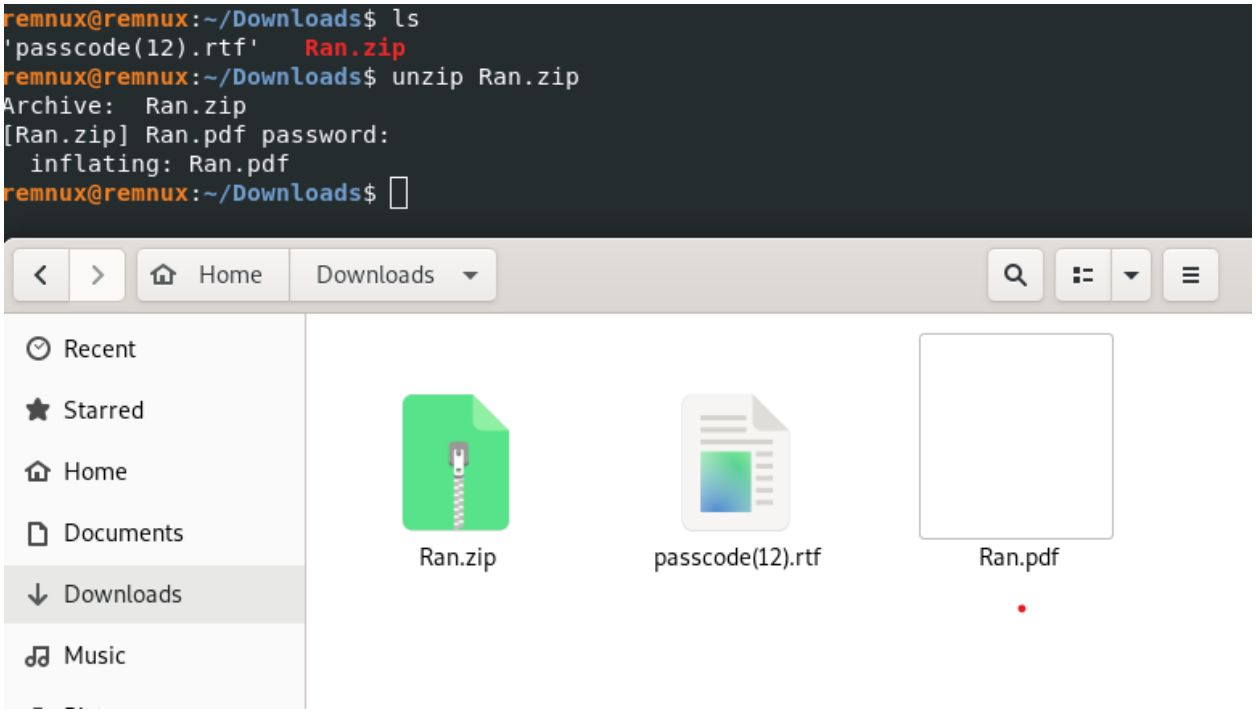
Malicious PDF File Analysis - No. 13

Steps and Screenshots:

The password provided along with zip file was incorrect. The password provided was ‘passw0rd’, however the correct password was ‘passw)rd’ . It took us a few hours to figure that out.

To analyze the malicious pdf, we used Remnux operating system since it already has various tools pre installed needed for analyzing. We were needed to analyze the pdf of group 13. Here is the screenshot of unzipping the pdf of group 13.



1) Report the number of objects in the file.

To find the basic details about the pdf file, we used *pdfid.py* command. As we can see from screenshot below, 11 objects were found in the pdf. Also it seems like no file is embedded in the malicious pdf since value for /EmbeddedFile is 0.

```
remnux@remnux:~/Downloads$ pdfid.py Ran.pdf
PDFiD 0.2.8 Ran.pdf
 PDF Header: %PDF-1.0
 obj                    11
 endobj                 11
 stream                  1
 endstream               1
 xref                    2
 trailer                 2
 startxref               2
 /Page                   2
 /Encrypt                0
 /ObjStm                 0
 /JS                     1
 /JavaScript             1
 /AA                     1
 /OpenAction             1
 /AcroForm               0
 /JBIG2Decode            0
 /RichMedia              0
 /Launch                 1
 /EmbeddedFile           0
 /XFA                    0
 /URI                    0
 /Colors > 2^24          0

remnux@remnux:~/Downloads$
```

2) **Determine whether the file is compressed or not.**

We can use *pdf-parser.py --content Ran.pdf* command to see all the content information of the pdf. From the screenshots below we can see that for 11 objects that are in the pdf, none of them have /Filter key in it. As we can see from the screenshots below of 11 objects there is no /Filter key in them. Hence, it can be concluded that the file is not compressed.

```
remnux@remnux:~/Downloads$ pdf-parser.py --content Ran.pdf
PDF Comment '%PDF-1.0\r\n'

obj 1 0
 Type: /Catalog
 Referencing: 2 0 R

  <<
    /Pages 2 0 R
    /Type /Catalog
  >>


<<
        /Pages 2 0 R
        /Type /Catalog
>>
```

```
obj 2 0
 Type: /Pages
 Referencing: 3 0 R

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


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

```
obj 3 0
 Type: /Page
 Referencing: 4 0 R, 2 0 R

  <<
    /Contents 4 0 R
    /Parent 2 0 R
    /Resources
      <<
        /Font
          <<
            /F1
              <<
                /Type /Font
                /Subtype /Type1
                /BaseFont /Helvetica
                /Name /F1
              >>
          >>
      >>
    /Type /Page
    /MediaBox [ 0 0 795 842 ]
  >>


<<
        /Contents 4 0 R
        /Parent 2 0 R
        /Resources <<
                /Font <<
                        /F1 <<
                                /Type /Font
                                /Subtype /Type1
                                /BaseFont /Helvetica
                                /Name /F1
                        >>
                >>
        >>
        /Type /Page
        /MediaBox [ 0 0 795 842 ]
>>
```

```
obj 4 0
 Type:
 Referencing:
 Contains stream

  <<
    /Length 0
  >>

 '\r\n'

xref

trailer
  <<
    /Root 1 0 R
    /Size 5
    /Info 0 0 R
  >>

startxref 429

PDF Comment '%%EOF\r\n'

obj 5 0
 Type:
 Referencing: 6 0 R

  <<
    /EmbeddedFiles 6 0 R
  >>

<</EmbeddedFiles 6 0 R>>

obj 6 0
 Type:
 Referencing: 7 0 R

  <<
    /Names [(template)7 0 R]
  >>

<</Names[(template)7 0 R]>>
```

```
obj 7 0
 Type: /Filespec
 Referencing: 8 0 R

  <<
    /UF (template.pdf)
    /F (template.pdf)
    /EF
      <<
        /F 8 0 R
      >>
    /Desc (template)
    /Type /Filespec
  >>

<</UF(template.pdf)/F(template.pdf)/EF<</F 8 0 R>>/Desc(template)/Type/Filespec>>

obj 8 0
 Type: /Action
 Referencing:

  <<
    /S /JavaScript
    /JS (this.exportDataObject({ cName: "template", nLaunch: 0 });)
    /Type /Action
  >>

<</S/JavaScript/JS(this.exportDataObject({ cName: "template", nLaunch: 0 });)/Type/Ac
tion>>

obj 10 0
 Type: /Action
 Referencing:

  <<
    /S /Launch
    /Type /Action
    /Win
      <<
        /F (cmd.exe)
        /D '(c:\\\\windows\\\\system32)'
        /P '(/Q /C %HOMEDRIVE% & cd %HOMEPATH%&(#546865205365637265742063 6f6465206973
3a523131373936343136 echo @echo "The secret code is hiding"> text.bat)&(start text.ba
t)\n\n\n\n\n\n\n\n\n\nThis is the Launch Message)'
```

```
<</S/Launch/Type/Action/Win<</F(cmd.exe)/D(c:\\windows\\system32)/P(/Q /C %HOMEDRIVE%
 & cd %HOMEPATH%&(#546865205365637265742063 6f64652069733a523131373936343136 echo @ech
o "The secret code is hiding"> text.bat)&(start text.bat)









This is the Launch Message)>>>>

obj 1 0
 Type: /Catalog
 Referencing: 2 0 R, 5 0 R, 9 0 R

  <<
    /Pages 2 0 R
    /Names 5 0 R
    /OpenAction 9 0 R
    /Type /Catalog
  >>


<<
        /Pages 2 0 R/Names 5 0 R/OpenAction 9 0 R
        /Type /Catalog
>>
```

```
obj 3 0
 Type: /Page
 Referencing: 4 0 R, 2 0 R, 10 0 R

  <<
    /Contents 4 0 R
    /Parent 2 0 R
    /Resources
      <<
        /Font
          <<
            /F1
              <<
                /Type /Font
                /Subtype /Type1
                /BaseFont /Helvetica
                /Name /F1
              >>
          >>
      >>
    /Type /Page
    /MediaBox [ 0 0 795 842 ]
    /AA
      <<
        /O 10 0 R
      >>
  >>


<<
        /Contents 4 0 R
        /Parent 2 0 R
        /Resources <<
                /Font <<
                        /F1 <<
                                /Type /Font
                                /Subtype /Type1
                                /BaseFont /Helvetica
                                /Name /F1
                        >>
                >>
        >>
        /Type /Page
        /MediaBox [ 0 0 795 842 ]
/AA<<</O 10 0 R>>>>
```

```
xref

trailer
  <<
    /Size 11
    /Prev 429
    /Root 1 0 R
    /Info 0 0 R
  >>

startxref 46023

PDF Comment '%%EOF\r\n'
```

3) Determine whether the file is obfuscated or not.

We can use *peepdf Ran.pdf* command to check if the pdf is obfuscated or not. For the screenshot below it can be seen that the Encoded field is set to 0 meaning that it is not obfuscated.

```
remnux@remnux:~/Downloads$ peepdf Ran.pdf
Warning: PyV8 is not installed!!

File: Ran.pdf
MD5: c52604b608f97967712796e4f25a6ca0
SHA1: 330407f1526bb5a8516418304375fe6ae01c6803
SHA256: a0c07e0a81c4cb3dbe3fe9191b8316e45b211e4dbdfe8e63db2eb726e02e1390
Size: 1788 bytes
Version: 1.0
Binary: False
Linearized: False
Encrypted: False
Updates: 1
Objects: 10
Streams: 1
URIs: 0
Comments: 0
Errors: 0

Version 0:
        Catalog: 1
        Info: 0
        Objects (4): [1, 2, 3, 4]
        Streams (1): [4]
                Encoded (0): []

Version 1:
        Catalog: 1
        Info: 0
        Objects (6): [1, 3, 5, 6, 7, 10]
        Streams (0): []
        Suspicious elements:
                /OpenAction (1): [1]
                /Names (2): [6, 1]
                /AA (1): [3]
                /Launch (1): [10]
                /EmbeddedFiles: [5]
```

To confirm that the pdf is not obfuscated we can use another command, *pdfinfo Ran.pdf* . As we can see from the screenshot below, the encrypted field is no. Hence, it is confirmed.

```
remnux@remnux:~/Downloads$ pdfinfo Ran.pdf
Tagged:         no
UserProperties: no
Suspects:       no
Form:           none
Syntax Warning: Bad launch-type link action
JavaScript:     no
Pages:          1
Encrypted:      no
Page size:      795 x 842 pts
Page rot:       0
File size:      1788 bytes
Optimized:      no
PDF version:    1.0
```

4) **Find and Extract JavaScript.**

From step 3 it was concluded that the pdf is not encrypted and likewise it does not contain any encrypted javascript code since it is missing 'Objects with JS code' field as well below 'Encoded (0)' field. The only javascript code that is in pdf was found inside object 8 as seen in screenshot below.

```
remnux@remnux:~/Downloads$ pdf-parser.py -c Ran.pdf --filter --raw > object.txt
remnux@remnux:~/Downloads$
```

```
obj 8 0
 Type: /Action
 Referencing:

<</S/JavaScript/JS(this.exportDataObject({ cName: "template", nLaunch: 0 });)/Type/Action>>

  <<
    /S /JavaScript
    /JS (this.exportDataObject({ cName: "template", nLaunch: 0 });)
    /Type /Action
  >>


<</S/JavaScript/JS(this.exportDataObject({ cName: "template", nLaunch: 0 });)/Type/Action>>

<</S/JavaScript/JS(this.exportDataObject({ cName: "template", nLaunch: 0 });)/Type/Action>>
```

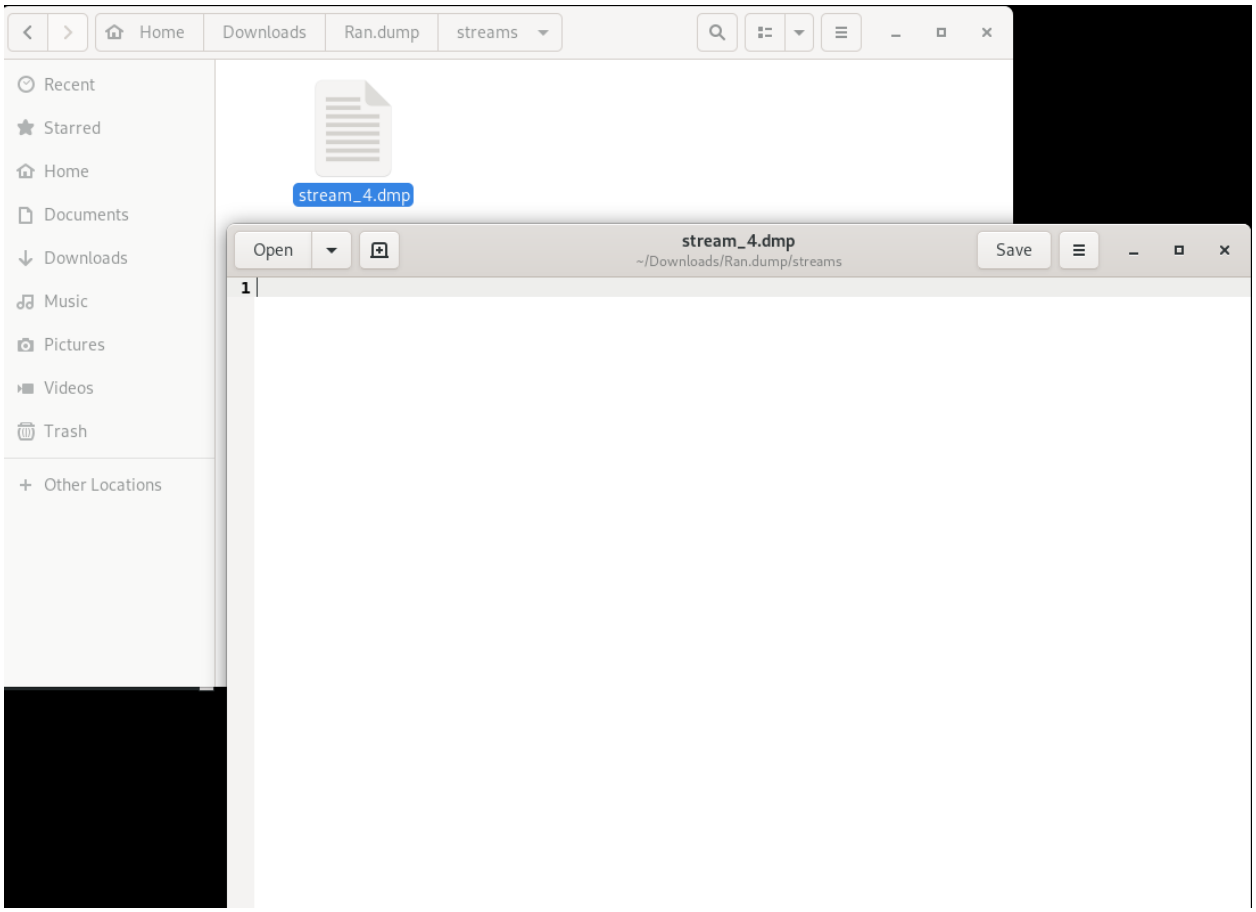
To particularly filter out javascript objects we used command *pdf-parser.py -s /javascript Ran.pdf* as seen in screenshot below.

```
remnux@remnux:~/Downloads$ pdf-parser.py -s /javascript Ran.pdf
obj 8 0
 Type: /Action
 Referencing:

  <<
    /S /JavaScript
    /JS (this.exportDataObject({ cName: "template", nLaunch: 0 });)
    /Type /Action
  >>


remnux@remnux:~/Downloads$
```

To further analyze and check for javascript code we used another command *pdfextract -s Ran.pdf* which finds the streams in the pdf and puts them in streams folder.

```
remnux@remnux:~/Downloads$ pdfextract -s Ran.pdf
Extracted 1 PDF streams to 'Ran.dump/streams'.
remnux@remnux:~/Downloads$
```

Now checking inside the streams folder, there is *stream_4.dmp* file which is empty and hence it is confirmed that the stream is empty as seen in the screenshot below.



5) De-obfuscate JavaScript.

The pdf does not have any obfuscated javascript that can be de-obfuscated.

6) Extract the shell code.

Now to extract the shell code, we can filter the objects that contains /action as type. We can use command *pdf-parse.py -s /action Ran.pdf* as the result is shown in the screenshot below. The object 10 is the shell code since it is opening the cmd to run the code. We can also see from the screenshot below that the shell code has not been encrypted.

```
remnux@remnux:~/Downloads$ pdf-parser.py -s /action Ran.pdf
obj 8 0
 Type: /Action
 Referencing:

  <<
    /S /JavaScript
    /JS (this.exportDataObject({ cName: "template", nLaunch: 0 });)
    /Type /Action
  >>


obj 10 0
 Type: /Action
 Referencing:

  <<
    /S /Launch
    /Type /Action
    /Win
      <<
        /F (cmd.exe)
        /D '(c:\\\\windows\\\\system32)'
        /P '(/Q /C %HOMEDRIVE% & cd %HOMEPATH%&(#546865205365637265742063 6f6465206973
3a523131373936343136 echo @echo "The secret code is hiding"> text.bat)&(start text.ba
t)\n\n\n\n\n\n\n\n\n\nThis is the Launch Message)'
      >>
  >>
```

Shell code

7) **Create a shell code executable**

As we can see while launching command *pdf-parser.py --content Ran.pdf* , the object 4 contains some stream.

```
obj 4 0
 Type:
 Referencing:
 Contains stream

  <<
    /Length 0
  >>

 '\r\n'

xref

trailer
  <<
    /Root 1 0 R
    /Size 5
    /Info 0 0 R
  >>

startxref 429

PDF Comment '%%EOF\r\n'
```

But the length of the stream is 0 meaning that neither it calls any javascript or shellcode when the pdf file is opened.

Hence it is not possible to create shellcode executable from the given pdf since it has an empty stream.

However we tried making shellcode executable using the object 10 since it has some code related to powershell or cmd on windows.

```
remnux@remnux:~/Downloads$ pdf-parser.py -s /action Ran.pdf
obj 8 0
 Type: /Action
 Referencing:

  <<
    /S /JavaScript
    /JS (this.exportDataObject({ cName: "template", nLaunch: 0 });)
    /Type /Action
  >>


obj 10 0
 Type: /Action
 Referencing:

  <<
    /S /Launch
    /Type /Action
    /Win
      <<
        /F (cmd.exe)
        /D '(c:\\\\windows\\\\system32)'
        /P '(/Q /C %HOMEDRIVE% & cd %HOMEPATH%&(#546865205365637265742063 6f6465206973
3a523131373936343136 echo @echo "The secret code is hiding"> text.bat)&(start text.ba
t)\n\n\n\n\n\n\n\n\n\nThis is the Launch Message)'
      >>
  >>
```

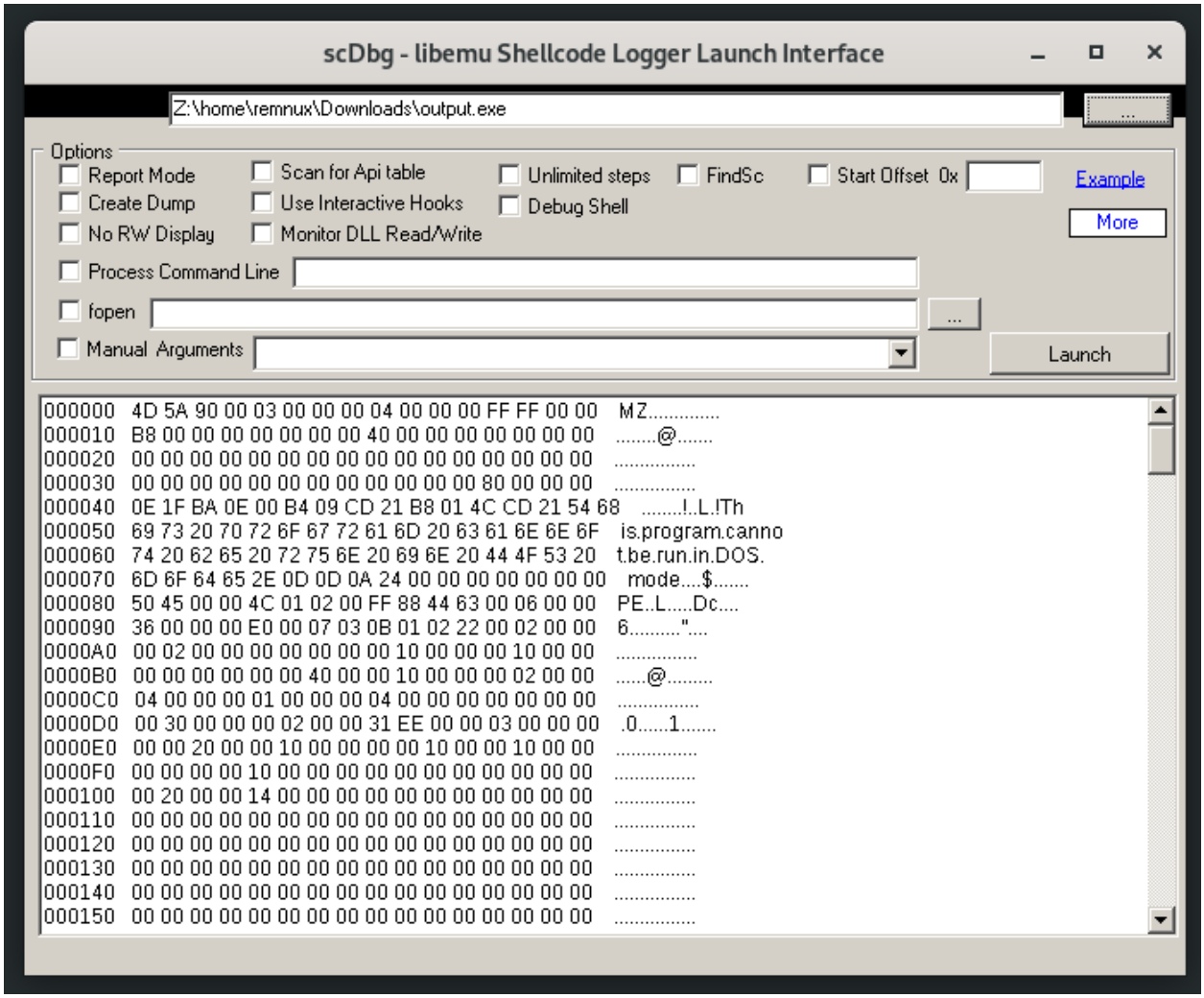
Shell code

We used a command *shcode2exe -s object10_shellcode.txt* to make .exe executable from object 10. It gives output.exe.

```
remnux@remnux:~/Downloads$ shcode2exe -s object10_shellcode.txt
remnux@remnux:~/Downloads$
```

8) **Analyze shell code and determine what is does or even execute it using sctest or spider monkey.**

Then we used *scdbg* command to execute and analyze the shellcode.



While launching the .exe file, it shows error showing it cannot be disassembled as show in screenshot below.

```
remnux@remnux: ~/Downloads
remnux@remnux:~/Downloads$ scdbg
remnux@remnux:~/Downloads$ shcode2exe -s object10_shellcode.txt
remnux@remnux:~/Downloads$ scdbg
Loaded d3d bytes from file Z:\home\remnux\DOWN~NTG\output.exe
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

401003   error accessing 0x00000000 not mapped

401003   0003                          add [ebx],al            step: 3  foffset: 3
eax=0         ecx=0         edx=0         ebx=0
esp=12fe04    ebp=12ffef    esi=0         edi=0          EFL 0

401005   0000                          add [eax],al
401007   000400                        add [eax+eax],al
40100a   0000                          add [eax],al
40100c   ???? Can Not Disassemble ff ff 0 0 b8

Stepcount 3



Z:\opt\scdbg>
```

Since the pdf is not encrypted we can easily see what it does and analyze it.

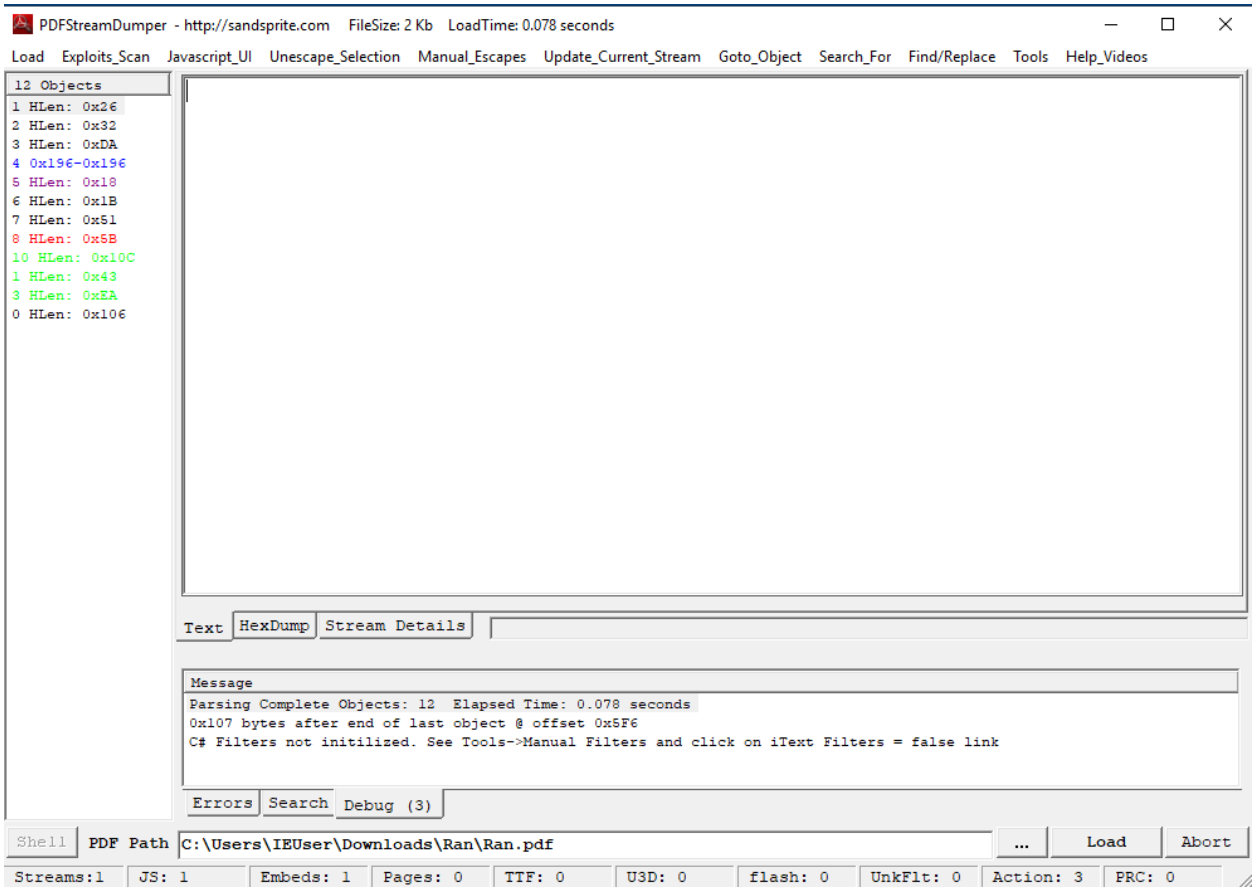
```
    /Win
      <<
        /F (cmd.exe)
        /D '(c:\\\\windows\\\\system32)'
        /P '(/Q /C %HOMEDRIVE% & cd %HOMEPATH%&(#546865205365637265742063 6f6465206973
3a523131373936343136 echo @echo "The secret code is hiding"> text.bat)&(start text.ba
t)\n\n\n\n\n\n\n\n\n\nThis is the Launch Message)'
      >>
  >>
```

Shell code

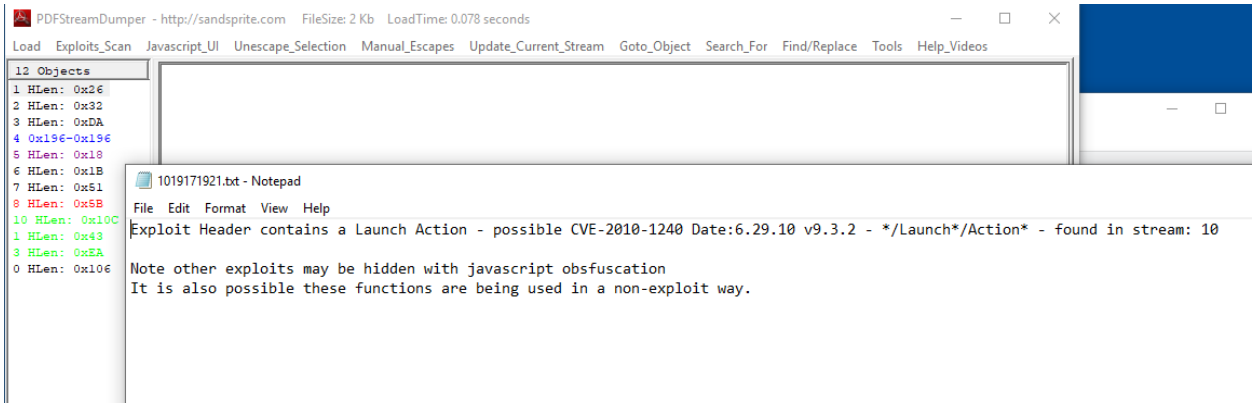
It simply writes some texts in text.bat file and executes it. That's all.

To further analyze and make sure we were analyzing the pdf correctly, we even used stream dumper on windows. Following were the foundings:-

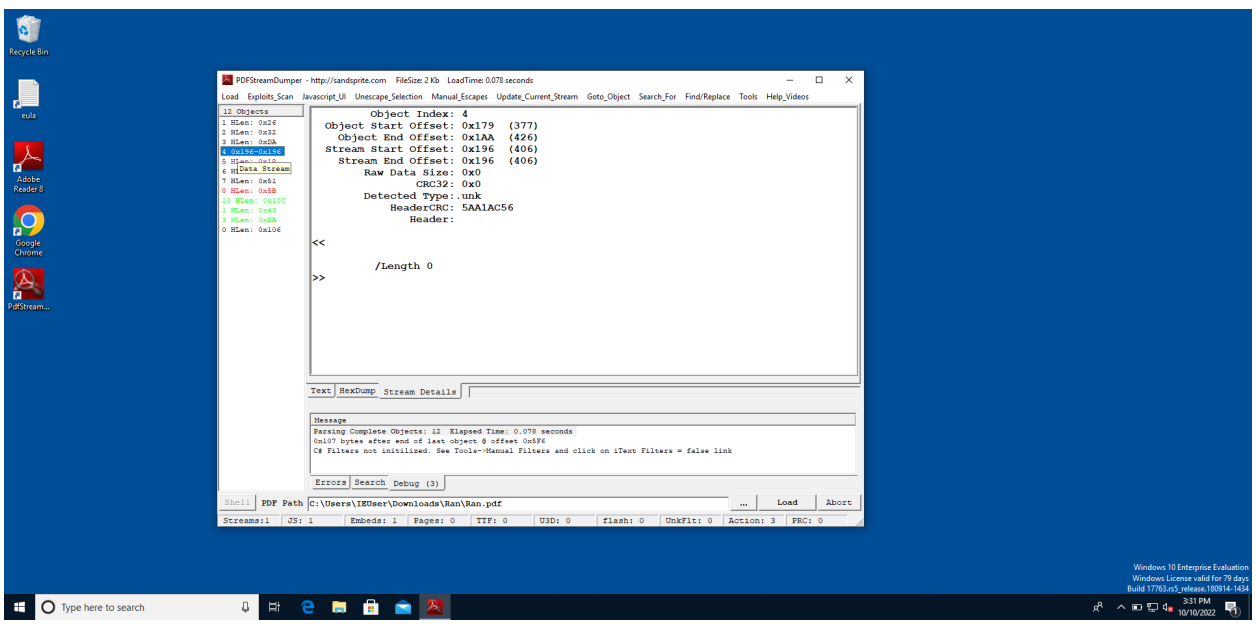
There were 11 objects in the pdf file.



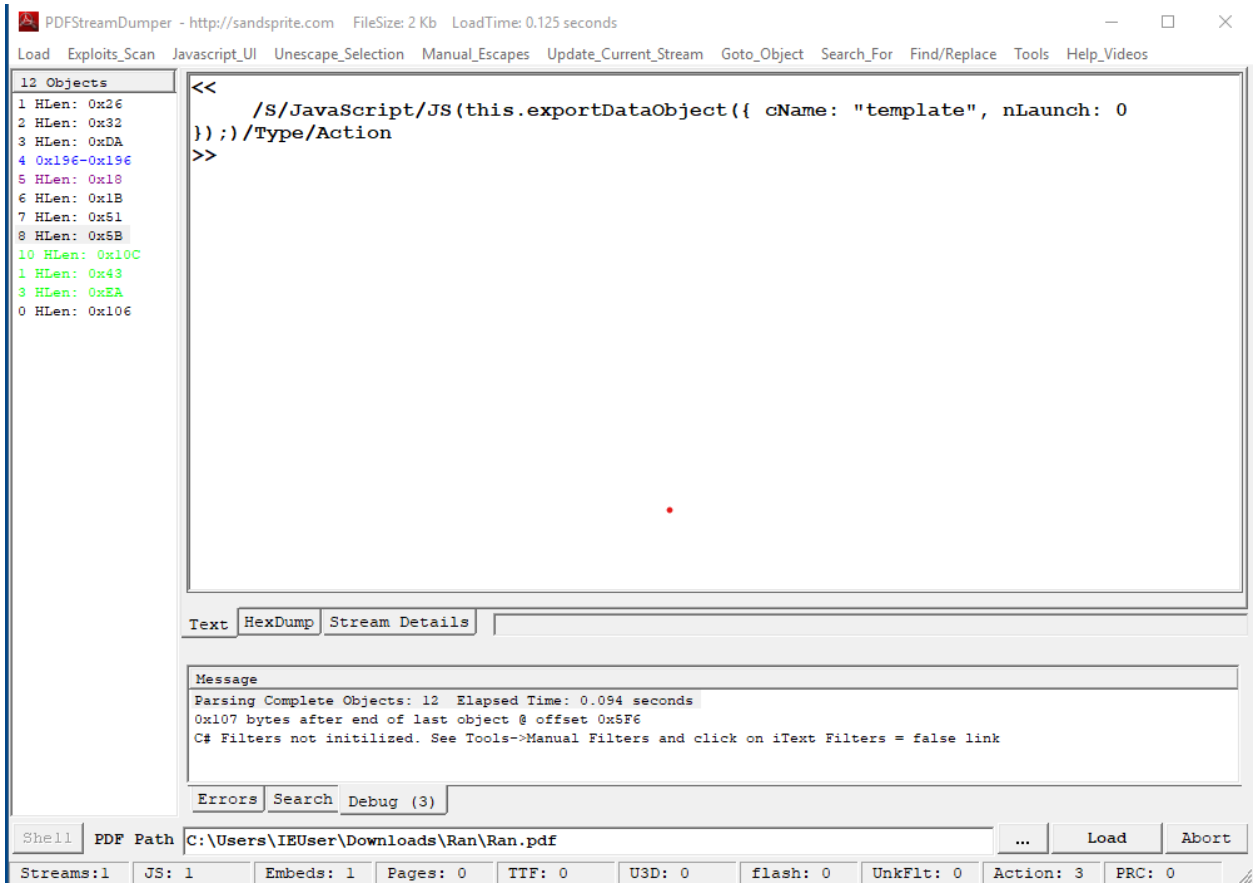
Following screenshot shows the vulnerability exploited while making the pdf:-



The data stream as shown below in screenshot is empty.



Now for the extracting the javascript embedded in the pdf file, object 8 contains few JS code as shown in screenshot below:-



Now trying to analyze what that JS code does we can go to Javascript_UI and test using shellcode analysis (scdbg):-

PDFStreamDumper - http://sandsprite.com FileSize: 2 Kb LoadTime: 0.125 seconds

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

12 Objects
1 HLen: 0x26
2 HLen: 0x32
3 HLen: 0xDA
4 0x196-0x196
5 HLen: 0x18
6 HLen: 0x1B
7 HLen: 0x51
8 HLen: 0x5B
10 HLen: 0x10C
1 HLen: 0x43
3 HLen: 0xEA
0 HLen: 0x106

```
<<
    /S/JavaScript/JS(this.exportDataObject({ cName: "template", nLaunch: 0
});)/Type/Action
>>
```

PDF Stream Dumper - JS UI

Load Format_Javascript Unescape_Selection (F6) Manual_Escapes Exploit_Scan Simplify_Selection_Quotes Shellcode_Analysis Find/Replace Deobsfuscation Tools

<-- to clipboard Script

Saved Scripts

Functions loc

msg data

```
<<
    /S/JavaScript/JS(this.exportDataObject({ cName: "template", nLaunch: 0
});)/Type/Action
>>
```

THIS RUNS SCRIPTS LIVE -- NO SANDBOX -- (also watch for Adobe specific objects) Double click a word to highlight all instances of it

<-- to clipboard app.viewerVersion : 9.2 this.pageNum 0 ^ to script pane Options Run No Reset

Shell PDF Pat

Streams:1 JS

PDFStreamDumper - http://sandsprite.com FileSize: 2 Kb LoadTime: 0.125 seconds

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

12 Objects
1 HLen: 0x26
2 HLen: 0x32
3 HLen: 0xDA
4 0x196-0x196
5 HLen: 0x18
6 HLen: 0x1B
7 HLen: 0x51
8 HLen: 0x5B
10 HLen: 0x10C
1 HLen: 0x43
3 HLen: 0xEA
0 HLen: 0x106

```
<<
    /S/JavaScript/JS(this.exportDataObject({ cName: "template", nLaunch: 0
});)/Type/Action
>>
```

PDF Stream Dumper - JS UI

Load Format_Javascript Unescape_Selection (F6) Manual_Escapes Exploit_Scan Simplify_Selection_Quotes Shellcode_Analysis Find/Replace Deobsfus

<-- to clipboard Script

Saved Scripts

Functions loc

msg data

```
<<
    /S/JavaScript/JS(this.exportDataObject({ cName: "template", nLaunch
});)/Type/Action
>>
```

scDbg - libemu Shellcode Logger Launch Interface

Options

Report Mode | Scan for Api table | Unlimited steps | FindSc | Start Offset 0x 0

Create Dump | Use Interactive Hooks | Debug Shell | Do not log RW | Monitor DLL Read/Write

fopen C:\Users\IEUser\Downloads\Ran\Ran.pdf

temp C:\Users\IEUser\Downloads\Ran

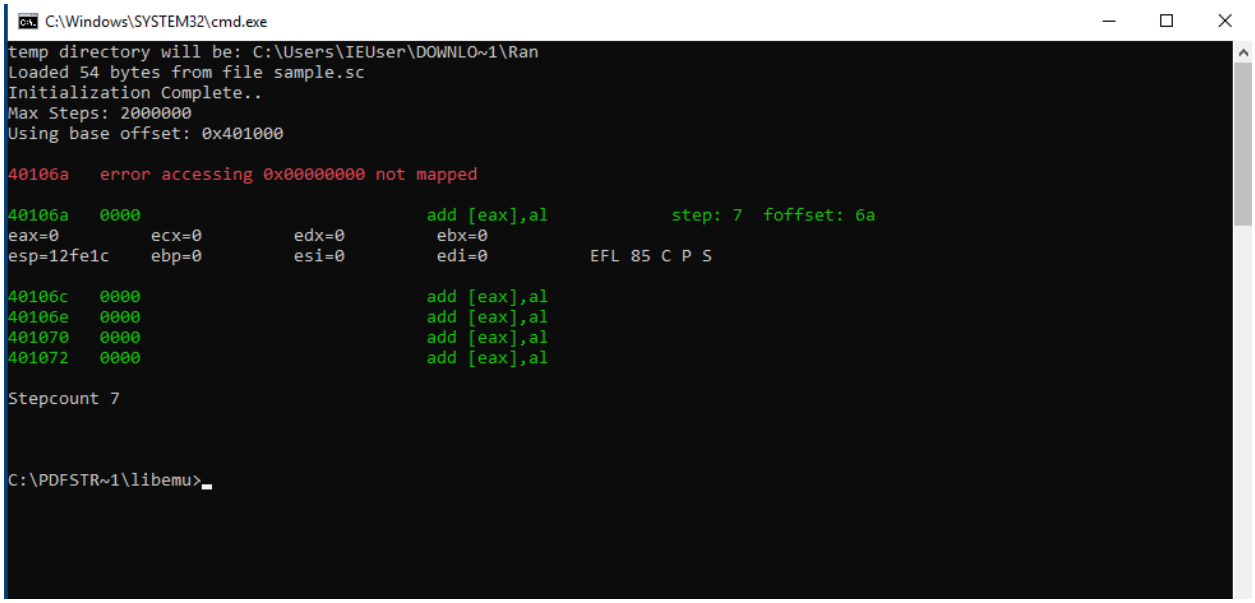
Manual Arguments

Launch

Manually Load File | Libemu HomePage | scdbg homepage | cmdline | Video Demo | Help | Example | Save dump

```
000000   3C 3C 2F 53 2F 4A 61 76 61 53 63 72 69 70 74 2F   <</S/JavaScript/
000010   4A 53 28 74 68 69 73 2E 65 78 70 6F 72 74 44 61   JS(this.exportDa
000020   74 61 4F 62 6A 65 63 74 28 7B 63 4E 61 6D 65 3A   taObject({cName:
000030   74 65 6D 70 6C 61 74 65 2C 6E 4C 61 75 6E 63 68   template,nLaunch
000040   3A 30 7D 29 3B 29 2F 54 79 70 65 2F 41 63 74 69   :0});)/Type/Acti
000050   6F 6E 3E 3E                                       on>>
```

Launching or running the JS code gives the following error. Seems like the malicious code is poorly configured in the pdf.



9) **What is the secret code?**

As seen in step 2, while executing command *pdf-parser.py --content Ran.pdf*, we can see in object 10 has some launch message which has secret code hidden in it. We can use command *pdf-parser.py -c Ran.pdf --object 10 --filter* to get detail of object 10 as seen in screenshot below.

```
remnux@remnux:~/Downloads$ pdf-parser.py -c Ran.pdf --object 10 --filter
obj 10 0
 Type: /Action
 Referencing:

  <<
    /S /Launch
    /Type /Action
    /Win
      <<
        /F (cmd.exe)
        /D '(c:\\\\windows\\\\system32)'
        /P '(/Q /C %HOMEDRIVE% & cd %HOMEPATH%&(#546865205365637265742063 6f6465206973
3a523131373936343136 echo @echo "The secret code is hiding"> text.bat)&(start text.ba
t)\n\n\n\n\n\n\n\n\n\nThis is the Launch Message)'
      >>
  >>

 [(1, '\r'), (2, '<<'), (2, '/S'), (2, '/Launch'), (2, '/Type'), (2, '/Action'), (2,
/Win'), (2, '<<'), (2, '/F'), (2, '('), (3, 'cmd.exe'), (2, ')'), (2, '/D'), (2, '('
), (3, 'c:\\\\windows\\\\system32'), (2, ')'), (2, '/P'), (2, '('), (2, '/Q'), (1, '
'), (2, '/C'), (1, ' '), (2, '%HOMEDRIVE% & cd %HOMEPATH%&(#546865205365637265742063 6
f64652069733a523131373936343136 echo @echo "The secret code is hiding"> text.bat)&(st
art text.bat)\n\n'), (1, '\n\n\n\n\n\n\n\n'), (3, 'This'), (1, ' '), (3, 'is'), (1, '
'), (3, 'the'), (1, ' '), (3, 'Launch'), (1, ' '), (3, 'Message'), (2, ')'), (2, '>>
'), (2, '>>'), (1, '\r')]
<</S/Launch/Type/Action/Win<</F(cmd.exe)/D(c:\\windows\\system32)/P(/Q /C %HOMEDRIVE%
 & cd %HOMEPATH%&(#546865205365637265742063 6f64652069733a523131373936343136 echo @ech
o "The secret code is hiding"> text.bat)&(start text.bat)
```

Secret code: hiding