

In the doc file, use the text editor to see its content. The VBA script part shows the function "AutoOpen" which means the script will run automatically. Inside the AutoOpen, I found a function MsgBox which contains "EICAR test ..". It means when the file is open it will automatically show the message box with "EICAR test" as a message.

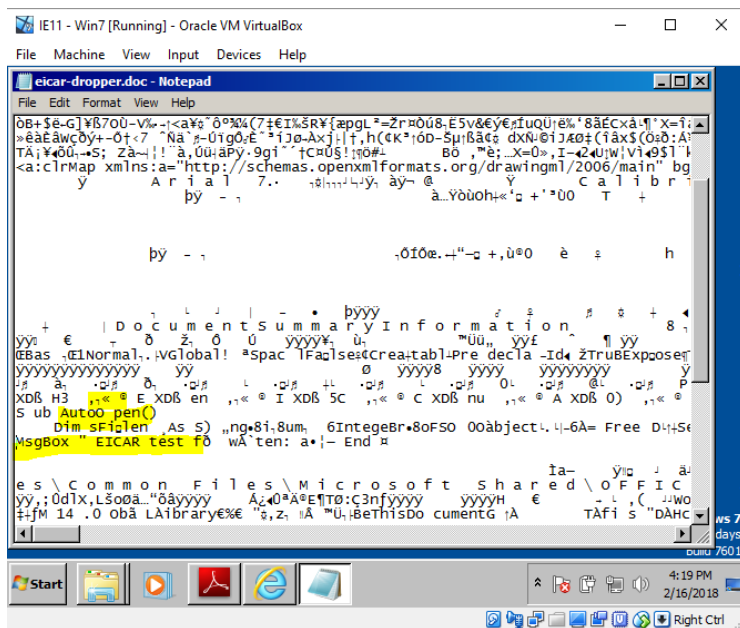


Fig 1.2 Investigate doc file with text editor

Part 3

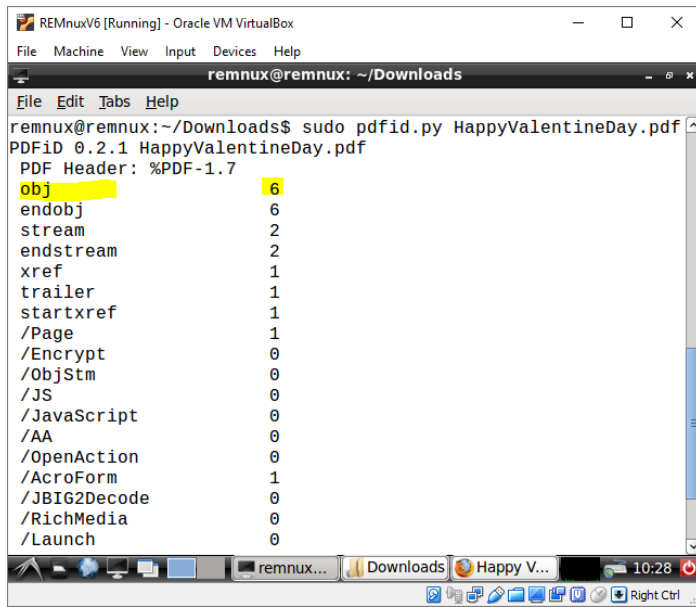
Analyze malicious file from blackboard system

(Malicious Sample Files\Assignment 1 Malicious PDF File\Happy Valentine's Day PDF File)

This analysis ran on REMnux virtual machine.

Q1 Report the number of objects in the file.

By using pdfid.py, I found 6 objects are in the pdf file.

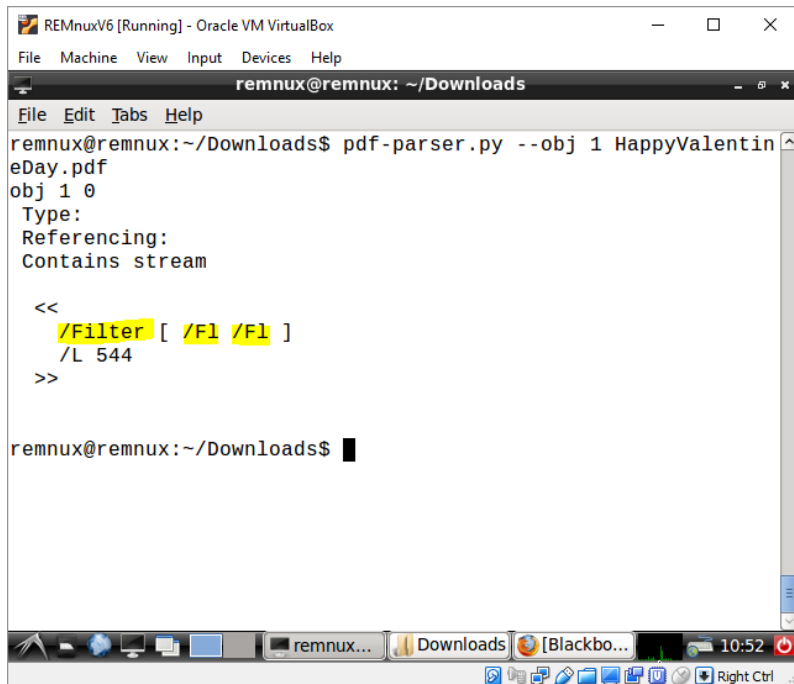


```
REMnuxV6 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
remnux@remnux: ~/Downloads
File Edit Tabs Help
remnux@remnux:~/Downloads$ sudo pdfid.py HappyValentineDay.pdf
PDFiD 0.2.1 HappyValentineDay.pdf
PDF Header: %PDF-1.7
obj 6
endobj 6
stream 2
endstream 2
xref 1
trailer 1
startxref 1
/Page 1
/Encrypt 0
/ObjStm 0
/JS 0
/JavaScript 0
/AA 0
/OpenAction 0
/AcroForm 1
/JBIG2Decode 0
/RichMedia 0
/Launch 0
```

Fig.3.1 pdfid.py shows number of objects

Q2 Determine whether the file is compressed or not.

It has 2 fl decode in the object one. So it is compressed.



```
REMnuxV6 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
remnux@remnux: ~/Downloads
File Edit Tabs Help
remnux@remnux:~/Downloads$ pdf-parser.py --obj 1 HappyValentin
eDay.pdf
obj 1 0
Type:
Referencing:
Contains stream

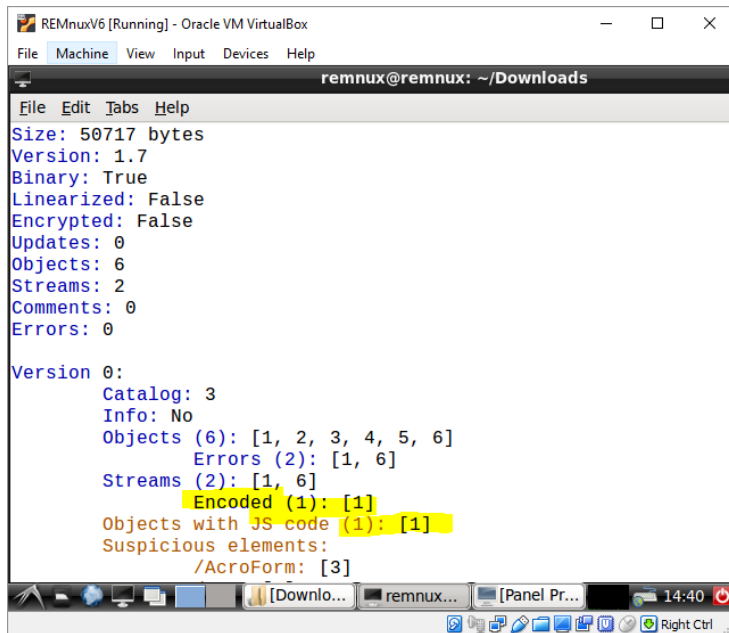
<<
  /Filter [ /F1 /F1 ]
  /L 544
>>

remnux@remnux:~/Downloads$
```

Fig.3.2 pdf-parser.py shows that it used Filter

Q3 Determine whether the file is obfuscated or not.

By using peepdf the result shown in Fig 3.3. The object [1] is Javascript and had been encoded (obfuscated).



```
remnux@remnux: ~/Downloads
File Edit Tabs Help
Size: 50717 bytes
Version: 1.7
Binary: True
Linearized: False
Encrypted: False
Updates: 0
Objects: 6
Streams: 2
Comments: 0
Errors: 0

Version 0:
  Catalog: 3
  Info: No
  Objects (6): [1, 2, 3, 4, 5, 6]
  Errors (2): [1, 6]
  Streams (2): [1, 6]
  Encoded (1): [1]
  Objects with JS code (1): [1]
  Suspicious elements:
    /AcroForm: [3]
```

Fig. 3.3 The file is encoded (obfuscated)

Q4 Find and Extract Javascript

As shown in Fig. 3.4, I used pdf-parser to extract object [1] which is suspected and contain Javascript.

After look though the extracted file, the javascript code was found as shown in Fig. 3.5

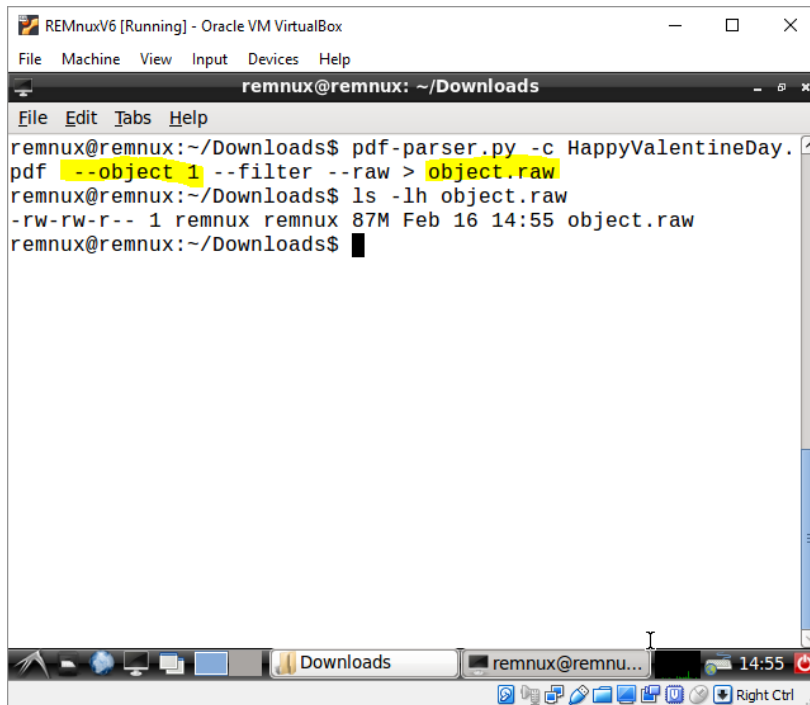


Fig. 3.4 Using pdf-parser to extract object [1]

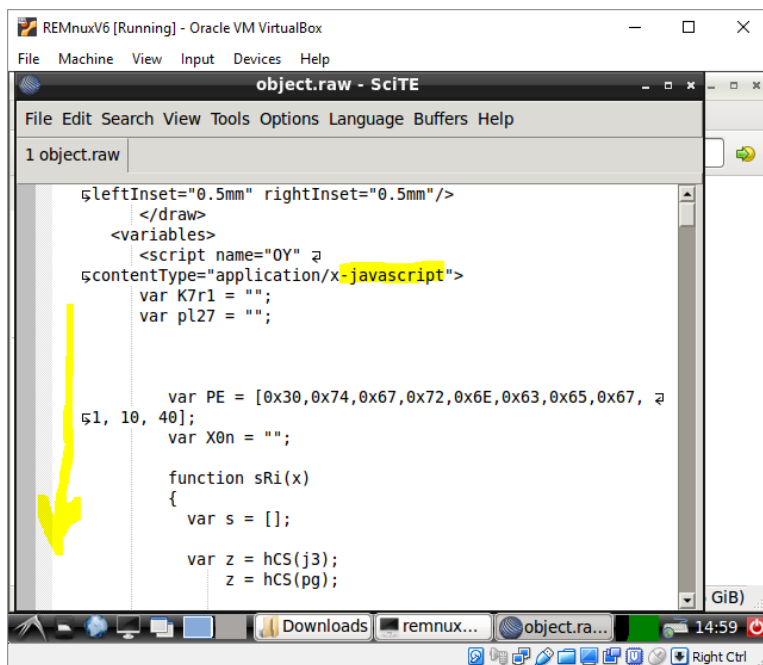
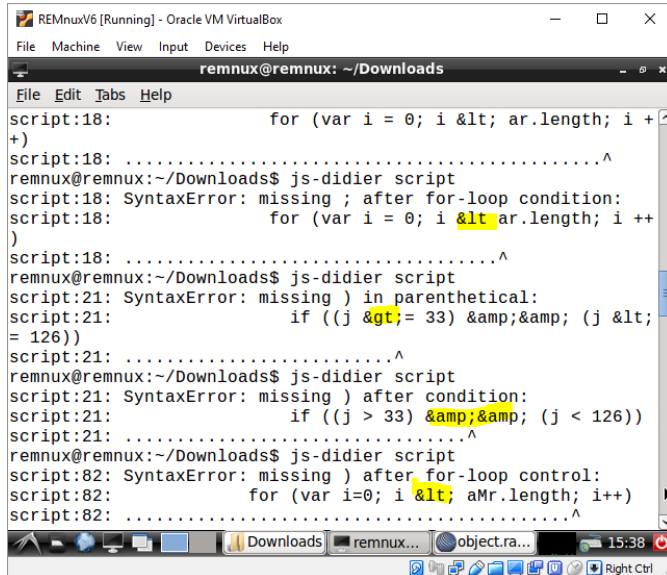


Fig. 3.5 Javascript found in extracted object 1

Q5 De-obfuscate JavaScript

The Javascript are separated obfuscate in to 4 part. I tried to put every part together.

The tool that used to de-obfuscate this JavaScript is SpiderMonkey. However it require manual code fixing, for example changing '\$lt;' to '<' and '\$gt;' to '>'. After manually fixing the code was still cause errors and SpiderMonkey can not compile the JavaScript.



```
REMnuxV6 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
remnux@remnux: ~/Downloads
File Edit Tabs Help
script:18:          for (var i = 0; i &lt;; ar.length; i +
+)
script:18: .....^
remnux@remnux:~/Downloads$ js-didier script
script:18: SyntaxError: missing ; after for-loop condition:
script:18:          for (var i = 0; i &lt; ar.length; i ++
)
script:18: .....^
remnux@remnux:~/Downloads$ js-didier script
script:21: SyntaxError: missing ) in parenthetical:
script:21:          if ((j &gt;= 33) &&; (j &lt;
= 126))
script:21: .....^
remnux@remnux:~/Downloads$ js-didier script
script:21: SyntaxError: missing ) after condition:
script:21:          if ((j > 33) &&; (j < 126))
script:21: .....^
remnux@remnux:~/Downloads$ js-didier script
script:82: SyntaxError: missing ) after for-loop control:
script:82:          for (var i=0; i &lt;; aMr.length; i++)
script:82: .....^
```

Fig. 3.6 Manual edit is required on obfuscate Javascript

Q6 – Q7

The file is the same as provided on

<https://countuponsecurity.com/2014/09/22/malicious-documents-pdf-analysis-in-5-steps/>

The author did some manual work to edit the JavaScript code but didn't provide the detail.

I tried to fix the code but it still can't run on SpiderMonkey and can't generate the SHELL code.

Q8 Analyze what it does

From the result of peepdf, it shows suspicious elements. One of them is BMP/RLE heap corruption (CVE-2013-2719) which allow attackers to execute arbitrary code or cause a denial of service (memory corruption) via unspecified vectors.

```
REMnuxV6 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
remnux@remnux: ~/Downloads
File Edit Tabs Help
Objects: 6
Streams: 2
Comments: 0
Errors: 0

Version 0:
  Catalog: 3
  Info: No
  Objects (6): [1, 2, 3, 4, 5, 6]
    Errors (2): [1, 6]
  Streams (2): [1, 6]
    Encoded (1): [1]
  Objects with JS code (1): [1]
  Suspicious elements:
    /AcroForm: [3]
    /XFA: [2]
    BMP/RLE heap corruption (CVE-2013-2729): [1]

remnux@remnux:~/Downloads$
```

The screenshot shows a terminal window titled 'REMnuxV6 [Running] - Oracle VM VirtualBox'. The terminal is running the 'peepdf' command, which outputs PDF analysis results. The output includes statistics like 'Objects: 6', 'Streams: 2', and 'Errors: 0'. It also shows a 'Version 0' section with details about the PDF's catalog, objects, streams, and suspicious elements. The line 'BMP/RLE heap corruption (CVE-2013-2729): [1]' is highlighted in yellow. The terminal prompt is 'remnux@remnux:~/Downloads\$'.

Fig. 3.7 peepdf shows suspicious elements