

**Assignment Report:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Malware Analysis – PDF’s**

Authored By:

**Jon Murray**

8-31-23

Table of Contents

Overview \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg 1

Sources \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg 1

Anti-Cheating Notice \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg 1

Machine Setup \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg 2-4

Example2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg 5-7

Example4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg 7-9

Conclusion \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg 9

**Overview:**

In lab, we are going to be looking at Malware Analysis again; this time we are going to be dealing with a pdf document that has been embedded with Javascript code. We’ll take a look at 2 different examples. In the second example, there is a base64 encoded string, the flag. We’ll extract that flag and decode it.

The first example is a basic walkthrough of the process of analyzing a pdf. We’ll use the ‘peepdf’ tool, and a tool called pdf-parser. Once we have the understanding of the tools down, we’ll apply that knowledge to the second example, where we will extract the flag.

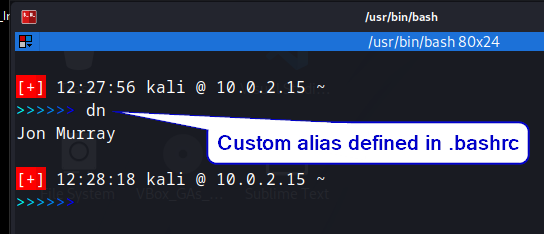
**Sources:**

Intezer - [ https://intezer.com/blog/incident-response/analyze-malicious-pdf-files/ ]

Google.com

**Anti-Cheating Notice:**

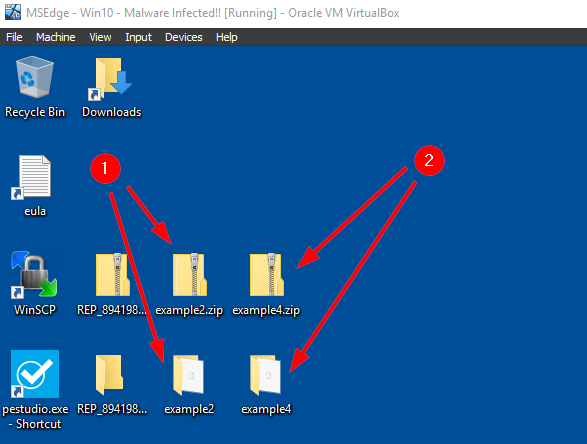
I am running a custom shell on my Kali Linux machine with a custom command prompt as shown. I have also written an alias into my bashrc file to display my name:

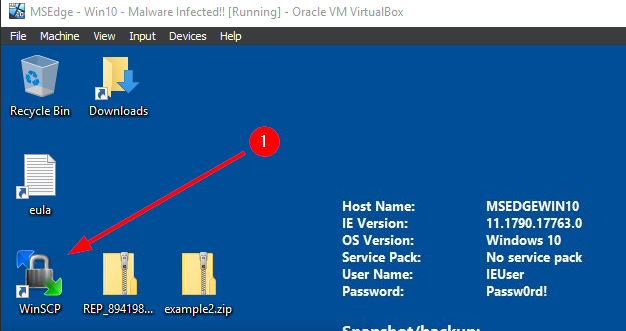


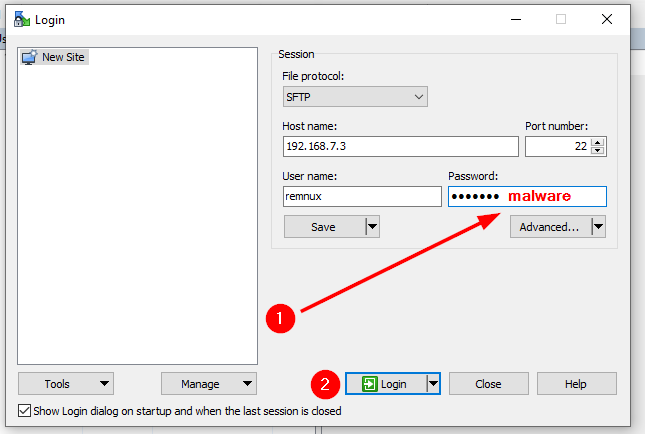
**Machine Setup:**

For this lab we are going to be using 2 example pdf files. I have already demonstrated the setup of the Windows VM we will be using; see my writeup on Malware Analysis for documentation on that process.

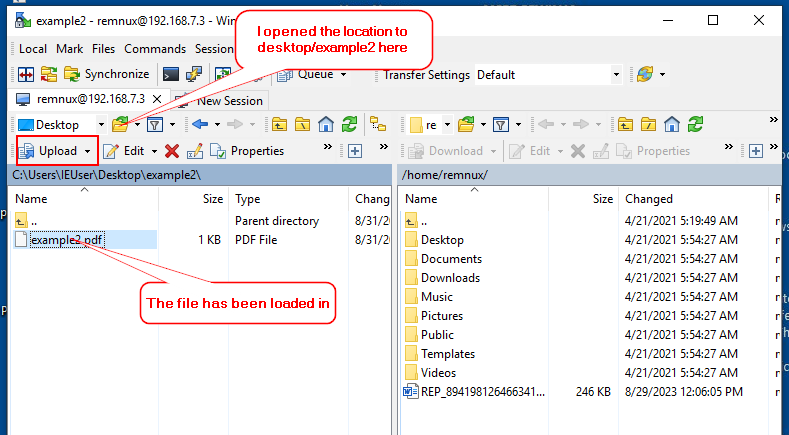
I started by reconnecting the machine to the internet so I could download 2 zipped folders called Example2.zip and Example4.zip. I then extracted each file to the desktop:

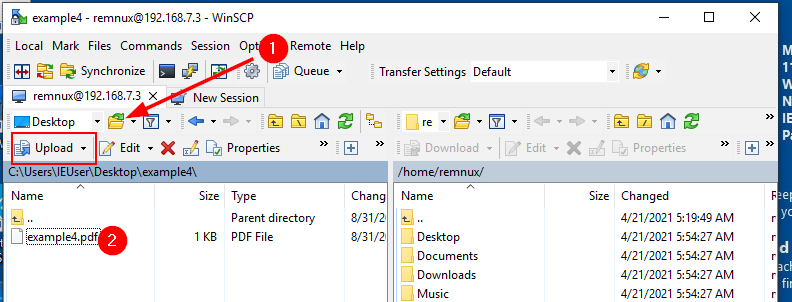


We’re going to start with the example2.pdf which is located inside the example2 folder. This file needs to be transferred to the Remnux machine where we can analyze it. We will do this with WinSCP. At the login screen, I set the hostname to the IP address of the Remnux machine – 192.168.7.3, the username as remnux, and the password is: malware, then clicked the login button:



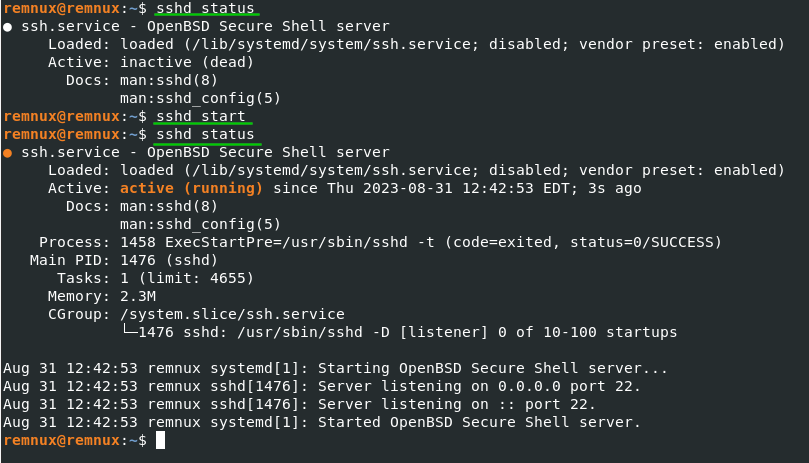
Once logged in, I opened the location to the example2 folder on the desktop. The transferred the file to the Remnux machine by clicking the upload button. I repeated this process for the example4 folder as well:





On the Remnux machine, I started the sshd service so that FTP would work, using commands:

<**sshd status**> <**sshd start**> <**sshd status**>



And as we can see, both files are now on the Remnux machine:



This completes the setup, now let’s move on to the analysis.

**Example2:**

OK, let’s start with how to analyze the pdf we transferred onto the Remnux machine. First, we are going to use a tool called ‘peepdf’ to extract the information or data, related to the file we are working with. The command is: <**peepdf example2.pdf**>

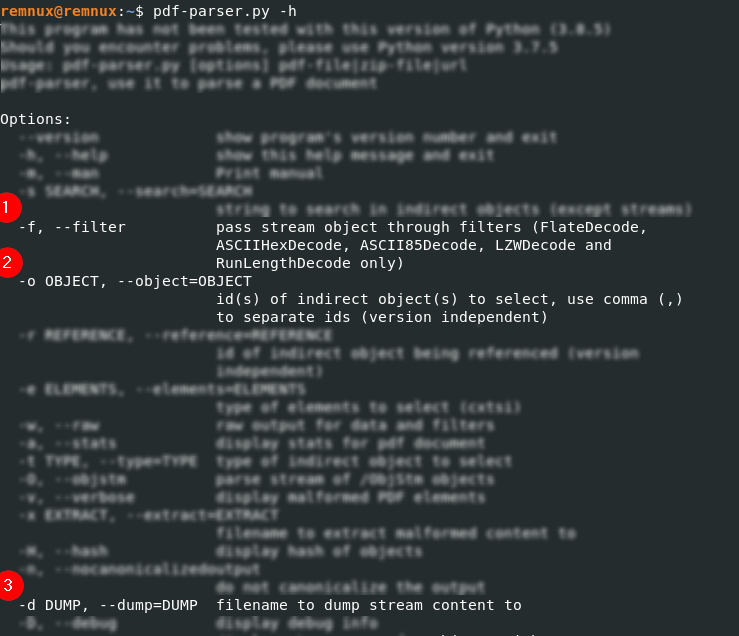


We learn that there are 6 objects in this file. I went through and extracted all 6 objects using a tool called pdf-parser. The command is:

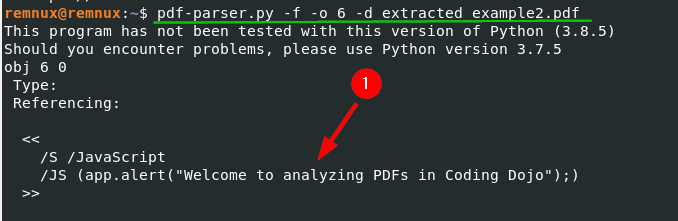
<**pdf-parser.py -f -o (object #) -d (output file name) (file to be extracted)**>



Let’s have a look at the help section for the tool to get a better idea of what this command is doing:



Here we learn that the -f option is passing the data through filters. The -o option is referencing a specific object in the pdf. The -d option is specifying the output file. We know from the ‘peepdf’ command that object 6 is the object that has the javascript code in it so lets take a look at the output from that object:



We get a plain text string that says “Welcome to analyzing PDF’s in Coding Dojo”. Nice. The next example we are going to deal with has a base64 encoded flag in the PDF somewhere. Using what we learned about the two tools we just used, let’s see if we can extract that flag.

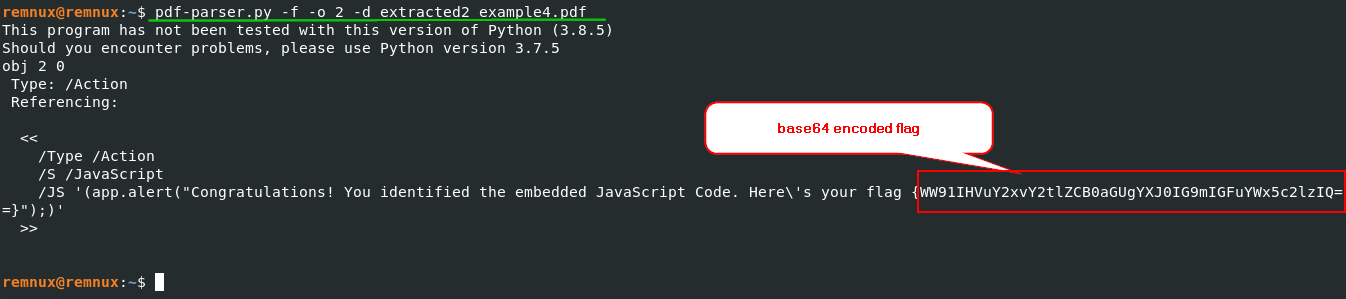
**Example4:**

We’re going to start with the <**peepdf example4.pdf**> command. And here’s output:

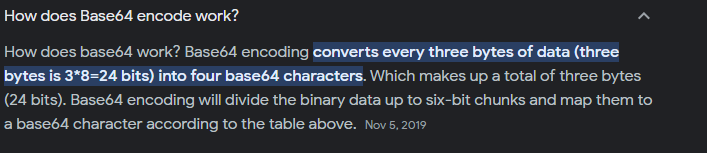


We have 5 objects in this pdf, and object 2 is the object that the javascript in it. Let’s take a look at that object using command:

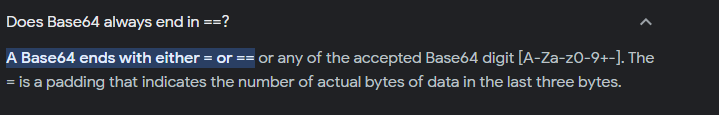
<**pdf-parser.py -f -o 2 -d extracted2 example4.pdf**>



Here we have located our base64 encoded flag. Let’s see what Google has to say about base64 encoding:



[ Google.com ]



[ Google.com ]

So, this information tells us that base64 encoded strings will typically end in either 1 or 2 “=”. This is how you can tell that the flag is base64 encoded; it has a trailing “=”. To decode this, we need to send the string to a tool called base64, which is pre-installed on most Linux platforms, including Remnux.

To decode the flag, our command will be:

<**echo “WW91IHVuY2xvY2tlZCB0aGUgYXJ0IG9mIGFuYWx5c2lzIQ=” | base64 -d**>

This command is using echo to repeat the base64 encoded string to the stdout data stream. Then it is being piped “ | “ to the base64 tool which is using the -d flag to decode the flag. Here’s the output of the command:



We got the flag! “You unclocked the art of analysis!”

**Conclusion:**

In this lab we got to look at and use two different tools, pdf-parser and peepdf. These tools allowed us to examine the PDF files and see the javascript located inside them. If these were malicious files, the javascript code inside would have been activated when the user opened the documents. We found a bae64 encoded string and decoded it to reveal the flag!