Lab 03 Maldocs

In this lab, you will analyze a malicious Office document to extract relevant/important information using static analysis tools.

Lab Files

The malware samples are contained within password-protected zip archives with the password: **infected**

Please ensure you are handling the malicious files appropriately. File hashes of the lab samples are:

- MD5 (sample01.doc) = 4a88e83b325aa23da1e4bfa90b4f7c34

Lab Instructions

Complete the following tasks in your analysis environment or using the IA lab. All work should be original, discovered, and reported on by you. Do not rely upon a single source of information (i.e., an online sandbox) to answer all questions. Use the tools and utilities available on your VM.

sample01

1. Determine what type of macros we are working with in this document. The file extension is .doc, but do some due diligence to make sure .doc is indeed the correct extension and not .docx or something else.

2. Find the streams of the file, and determine which of those contain macro code.

3. Once you've found the stream(s), dump them out and decode them using oldeump.py.

4. One of the functions typically will run when the document opens (Document_Open) or closes (Document_Close) to infect the user; find that one to begin.

5. The **Document_Open** function does some nifty string obfuscation with character encoding. You will see function calls to **Chr()** and **Chrw()**, find out what the difference is.

6. There is a variable called **process** that seems important. Decode it and describe what it does. Here's a screenshot of the variable:

```
    Set process = GetObject(ChrW(119) & ChrW(105) & ChrW(110) & ChrW(109) & ChrW(103) & ChrW(109) & ChrW(116) & ChrW(115) & _
  ChrW(58) & ChrW(87) & ChrW(105) & ChrW(110) & ChrW(51) & ChrW(50) & ChrW(95) & ChrW(80) & ChrW(114) & _
  ChrW(111) & ChrW(99) & ChrW(101) & ChrW(115) & ChrW(115))
process.create d, Null, Null, processid
```

7. We can assume that a file is getting dropped on disk by some of the context clues in the macro. There are various names, such as **CurFolder** and **GenerateFileName,** that raise suspicion. Find this block of code, and copy it to Visual Studio Code. Once there, add comments to lines shown as 26-42 in the following screenshot (your line numbers may be different). This will help you learn some VBA!

```
23  CurFolder = "C:\1\Whole"
24
25  d = CurFolder & "\" & "
26  PFSDNSKDF.E " + Chr$(88) + Chr$(69)
27  hextostr = ToggleButton1.Caption
28
29  If Right$(CurFolder, 1) < > "\" Then CurFolder = CurFolder & "\
30  "
31  MakeSureDirectoryPathExists CurFolder
32
33  For i = 1 To Len(hextostr) Step 2
34  strTemp = Chr(Val("&H" + Mid(hextostr, i, 2)))
35  strReturn = strReturn + strTemp
36  Next i
37
38  hextostr = Right(strReturn, Len(strReturn) - 1)
39
40  Open d For Binary As #1
41      Put # 1, , Chr$(77) + hextostr
42  Close #1
```

8. The file dropped by the macro is stored within the Office document. Look at the streams again and see if any of those are quite a bit larger than the others. When you find it, use **oledump.py** to display the contents in ASCII.

9. Write the contents of the stream out to a file in ACII format; you can use oledump.py with the **-S** or **-d** arguments.

10. Determine the type of encoding that was used for the string data that you just extracted. You may want to use CyberChef or base64dump.py to explore different encodings.

11. Once you have identified how the file is encoded, try to extract the data to get a valid binary file that is dropped to disk. You may need to sanitize some of the data.

12. Run **file <filename>; md5sum $_** to show what the file is identified as and its hash.

13. Do a brief bit of sandbox analysis on the dropped file. Determine what it is and whether or not it appears to be malicious.

14. Summarize your findings in a report. Write up a paragraph or so that you would use to brief your boss or a stakeholder on what the binary is, it's capabilities, potential impacts, etc.