|  |  |  |
| --- | --- | --- |
| Week | Malware Analysis Tools and Techniques | Duration |
| 7 | Examining Malicious Documents | 120 mins |

**Lesson Objectives**

Analyse Malicious Microsoft Office documents and PDF files

Note:

Copy the files on your Windows 8 Workstation virtual machine desktop.

1. locked.doc
2. media.docm

Copy the file on your REMnux virtual machine

1. pdf1.pdf

Exercise 1: Extracting VBA macros from locked.doc using OfficeMalScanner

Step 1: Type the following command (in command prompt) to direct OfficeMalScanner to examine locked.doc and extract the VBA Macro code embedded into it:

OfficeMalScanner locked.doc info

OfficeMalScanner will extract the embedded macros into the folder on the desktop called LOCKED.DOC-Macros.

Q1. How many macros does the locked.doc contain? List the filenames.

There are 2 macros, Avira and Mdulo1.

Step 2: Examine the extracted macros using a text editor to understand the nature of the malicious code.

Q2. List the host-based and network-based indictors of the malicious code.

Host-based:

* Módulo1
* Avira
* urlmon
* \conhost.exe
* APPDATA

Network-based:

* http://limitless.hints.me/aboki.scr

Exercise 2: Extracting VBA macros from media.docm using OfficeMalScanner

Step 1: Extract contents of the media.docm file using the OfficeMalScanner’s “inflate” command.

OfficeMalScanner media.docm inflate

OfficeMalScanner will extract the XML-based Microsoft Office document, placing the resulting files into the %Temp%\DecompressedMsOfficeDocument\word folder.

Step 2: Process the vbaProject.bin file, extracted from media.docm, using OfficeMalScanner’s “info” command to extract macros

OfficeMalScanner vbaProject.bin info

Q1. How many macros does the media.docm contain? List the filenames.

There are 2 macros, NewMacros and ThisDocument.

Step 2: Examine the extracted macros using a text editor to understand the nature of the malicious code.

Q2. List the host-based and network-based indictors of the malicious code.

Host-baes:

* NewMacros
* WScript.Shell
* %APPDATA%
* Adodb.Stream
* Microsoft.XMLHTTP
* Scripting.FileSystemObject
* ThisDocument
* 1Normal.ThisDocument

Network-mased:

* <http://softtonic.biz/cr/20014.exe>
* GET
* xHttp.responseBody

Step 3: Extract Unicode and ASCII-encoded strings from the vbaProject.bin file by typing the following command:

Strings2 –nh vbaProject.bin | grep -I http

Q3: List the important strings found.

* <https://dl.dropboxusercontent.com/u/32611948/working.exe>
* http://softtonic.biz/cr/20014.exe

Step 4: Examine the \_\_SRP\_0 stream of vbaProject.bin using SSView to examine remnants of the earlier version of the malicious VBA macro by typing the following command:

start .

Load the vbaProject.bin file into SSView by dragging the vbaProject.bin icon onto the icon the icon of the SSView window within which vbaProject.bin is loaded.

In SSView expand the VBA folder on the top left side of the screen, then click on the \_\_SRP\_0 icon.

Q4: List the important strings found in the stream.

* C:\PROGA~2\COMMON~1\MICROS~1\VBA\VBA7.1\VBE7.DLL
* C:\Program Files (x86)\Microsoft Offices\Office15\MSWORD.OLB
* C:\Windows\SysWOW64\stdole2.tlb
* C:\Program Files (x86)\Common Files\Microsoft Shared\OFFICE15\MSO.DLL
* <https://dl.dropboxusercontent.com/u/32611948/working.exe>
* WScript.Shell
* Adobe.Stream
* VBE7.DLL

Exit SSView by selecting File > Exit.

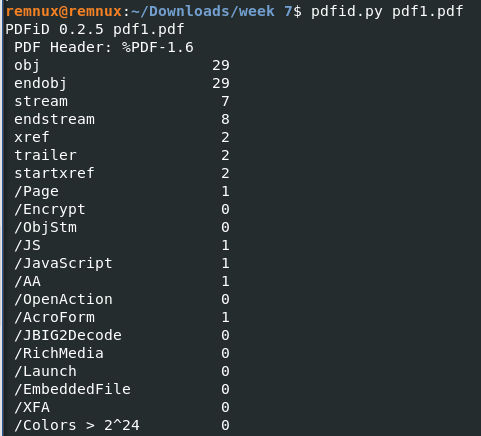
Exercise 3: Analysing pdf1.pdf using pdfid and pdf-parser

Step 1: Scan pdf1.pdf using pdfid.py, looking for indicators that the document file is malicious.

In the terminal window of REMux, while located in the week12 folder, run the following command:

pdfid pdf1.pdf

Q1: List the malicious indicators of pdf1.pdf.



Step 2: Scan pdf1.pdf using pdf-parser to locate object that contain “/JavaScript” by typing the following command in terminal window:

pdf-parser.py – search /JavaScript pdf1.pdf

Q2: In which object is the JavaScript located?



Step 3: Examine the object where the JavaScript is located using pdf-parser, saving the output to file out.js.

Type the following command in terminal window:

pdf-parser.py –object ?? –filter –raw pdf1.pdf > out.js

(Note: Replace ?? with the object number)

Step 4: Examine out.js using SciTE to understand the nature of the malicious JavaScript code.

(Note: Use relevant tools to deobfuscate JavaScript)