|  |  |  |
| --- | --- | --- |
| Week | Malware Analysis Tools and Techniques | Duration |
| 12 | In-Depth Malware Analysis -Part 1 | 120 mins |

**Lesson Objectives**

1. Recognising Malware Packers
2. Automated unpacking of Malware
3. Fixing ASLR Problem
4. Manually unpacking malware

Note:

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

1. Brbbot.exe

Exercise 1: Use appropriate tools to identify packer

Step 1: Use the following tools to check if brbbot.exe is packed and the packer used:

1. Strings – BinText,Strings2
2. Dependency Walker
3. IDA Pro : You receive warning that some imported functions will not be visible
4. Bytehist : Byte-Usage histogram . Highly uniform distributions suggest that file is packed.

Q1. List the packer used to pack brbbot.exe

UPX1, .rsrc

Exercise 2: Unpack brbbot.exe

Step 1: Use appropriate tool(s) to unpack brbbot.exe (Using “upx -d brbbot.exe)

Step 2: Compare the strings and DLL imports of the packed and unpacked brbbot.exe

Q1. List down the important strings and DLLs of the unpacked brbbot.exe

DLL: Wininet.dll, w2\_32.dll

String: POST, HTTP/1.1, CreateFile, DeleteFile, InternetConnect, brbconfig.tmp, Software\Microsoft\Windows\CurrentVersion\Run

Exercise 3: Fixing ASLR Problem

Load brbbot-unpacked.exe into OllyDbg. An easy way to do this is to drag the brbbot unpacked.exe icon onto the OllyDbg icon. After the debugger finishes loading the specimen and its dependencies, run brbbot-unpacked.exe within OllyDbg by selecting Debug > Run (F9) OllyDbg will pause the specimen's execution, unable to continue. You'll see a brief explanation of the problem in the debugger's status bar at the bottom left of OllyDbg's window. The message will explain that an access violation has occurred when writing to memory. This error is due to the ASLR security mechanism built into Windows.

Step 1: Exit OllyDbg, directing it to terminate the brbbot-unpacked.exe process. Confirm that no Brbbot instances are running using Windows Hacker.

Step 2: Load brbbot-packed.exe into CFF Explorer on Windows Workstation by dragging the icon brbbot-packed.exe into the CFF Explorer window or dropping it on top of the tool's icon. In the tree-like hierarchy on the left side of the tool, click on Optional Header, then scroll down in the tool's right pane until you see the DllCharacteristics field.

Step 3: Click the Click here area to the right side of the DLLCharacteristics field. In the window that pops up, uncheck the “DLL can move” checkbox to clear it.

Step 4: Save the modified file by selecting File > Save. Click Yes when the tool asks you whether to overwrite the original file then press OK.  
Once you've tweaked the brbbot-unpack.exe, exit CFF Explorer (File > Exit)

Step 5: Load brbbot-unpacked.exe into OllyDbg and run brbbot-unpacked.exe within OllyDbg by selecting Debug > Run (F9). The Access Violation message should have gone.

Exercise 4: Unpacking Brbbot with Scylla

Step 1: Exit OllyDbg and CFF Explorer if they are still running. Use Process Hacker to terminate all Brbbot instances if they are still running.

Step 2: Disable the "dynamic base" flag in the packed brbbot.exe file. You can use CFF Explorer, as shown in the previous exercise. Next, infect your Windows Workstation with the packed Brbbot specimen, running the program with Administrator privileges.

Step 3: Examine strings within the running brbbot.exe process using Process Hacker. Using Process Hacker, right-click on the running brbbot.exe process and select Properties then switch to the Memory tab.

Click the strings button, then click Ok to extract strings from the process using default settings. Scroll through the extracted strings; you will see more strings than you would find in a packed file, because the specimen unpacked itself into memory when it ran. Close the Results window and the Properties window of Process Hacker. Make a note of the ID of the brbbot.exe process, as shown in Process Hacker.

Step 4: Launch Scylla on your Windows Workstation. Click the drop-down box below the Attach to an active process area of Scylla, then select the brbot.exe process.

Once Scylla attached to the malicious process, click the Dump button to extract it from memory. The tool will ask you where you'd like to save the file. Enter the location (such as the Desktop) and the file name (such as brbbot-dumped.exe), then click OK.

Step 5: Click the IAT Autosearch button in Scylla. Scylla will present the IAT found pop-up; click on it. Then click Get Imports. These steps will direct the tool to automatically locate and extract Import Address Table (IAT) data inside the brbbot.exe process, to which Scylla is currently attached. It's not uncommon for Scylla to fail to resolve some IAT entries, in which case you'll see an "x"  
instead of a checkmark preceding them in the output of the Get Imports action. You probably won't be able to resolve all the entries at all times. However, if you see many such "x" signs, try repeating IAT Autosearch and Get Imports again several times. If this doesn't improve the situation, reboot your Windows REM Workstation, re-infect the VM with brbbot.exe and use Scylla again.

Step 6: Click on each entry that Scylla failed to resolve, then right-click on it and select Delete tree node. Do this for each entry that was preceded with the "x" icon instead of the checkmark.  
Lastly, click the Fix Dump button in Scylla and point the tool at the dumped file you created earlier (e.g., brbbot-dumped.exe). Scylla will embed IAT information in the executable, creating a new version of the file by including" \_SCY" at the end of its name. In our example, fixed file will be called brbbot-dumped\_SCY.exe.  
Terminate the brbbot.exe process and exit Scylla.

You should now be able to load the dumped and fixed file (e.g., brbbot-dumped\_SCY.exe) into tools such as IDA Pro and OllyDbg, which will be able to perform code analysis that was unavailable to them when the specimen was packed. The tools will be able to disassemble the malicious code, show IAT contents and display API call functions by name.