CS 499/ISA 564: Lab 6 – Advanced Malware

Lab Submission Guidelines

Submissions must be in Microsoft Word or PDF format. Be sure to clearly label what question you're answering. Where possible, screenshots should be embedded directly in the document. Screenshots should be cropped to only include what is necessary to answer the question. If you need to save them as a separate file, save them in compressed format (gif, jpg, png) and name them after the question they pertain to. If your submission contains multiple files, archive them (zip, 7z, tar.gz) and submit the file via Blackboard.

Lab Requirements

- 32-bit Windows 7 VM
- Files in the Lab Resources folder

Windows VM Setup

- Install the Microsoft .NET Framework from Microsoft .NET Framework 4.7.1.exe
- Install PowerShell 5.1 from PowerShell 5.1.msu (this will require a reboot)

Lab 6: Task 1 – PowerShell Obfuscation and Debugging

For this exercise we will be using Lab6.1.ps1. The purpose of this exercise is to learn how malware authors can use various obfuscation techniques to make PowerShell code harder to read and analyze, as well as how to use PowerShell's debugger and/or output capability to manually deobfuscate malicious code.

Right-click on Lab6.1.ps1 and click Edit, this will launch Windows PowerShell Integrated Scripting Environment (ISE), an IDE for PowerShell code. You'll note that the script code is obfuscated; this makes it impossible to learn what the code does. Your objective is to use PowerShell's ISE debugger to deobfuscate the code.

Some hints on how to use PowerShell's ISE debugger:

- F1 will do a help lookup for a term, F5 will run the script, F9 will set a breakpoint, F10 will step over
- Typing *echo $var* in the console (lower pane) will print the contents of *$var* while the script is running
- While at a breakpoint, mouseover a variable to reveal its contents

Question 1.1 – Provide a screenshot of the fully deobfuscated script. You must provide the fully deobfuscated contents of every variable, expand every alias into its full command, provide the final command that is executed, and finally provide the Base64 decoded string.

Extra Credit (up to 10 points) – Explain how the various obfuscation techniques in the script work. Several different techniques are employed, the more of them you cover and the more thorough your explanation, the more points you will receive.

Lab 6: Task 2 – PowerShell Obfuscation and Logging

For this exercise we will be using Lab6.2.ps1. The purpose of this exercise is to continue delving into PowerShell obfuscation techniques, as well as how Windows logging can be used to defeat these obfuscation techniques and reveal the actual executed code.

Examine the contents of Lab6.2.ps1. You will see that just like with Lab6.1.ps1, it is obfuscated making it difficult to tell what the code will do. While this code can be deobfuscated manually like in Task 1, the obfuscation techniques used in this script are trivially bypassed by PowerShell's logging capability. PowerShell's logs can be viewed using the Windows Event Viewer. Events of interest will have Event IDs 4103, and 4104. First run win.reg to enable logging, click yes when prompted, and then reboot your VM.

To access PowerShell logs in the Windows Event Viewer, do the following: Go to Start→Run→eventvwr.msc (enter)→Navigate to Application and Services Logs→Microsoft→Windows→PowerShell→Operational.

Question 2.1 – Execute Lab6.2.ps1. It will generate multiple interesting events. Provide screenshots of the 4104 event that contains the complete script block code and the 4103 event that contains the parameters for the command invocation of Out-File.

Question 2.2 – Lab6.2.ps1 ultimately writes a Base64 encoded string to a file. Although the filepath is revealed in the 4103 event asked for in Question 1.1, it does not reveal how that path is constructed. Dig through the other 4103 and 4104 events and provide an explanation of how the path was constructed. There are two specific references that your explanation must contain (Hint: they're where the heart is). Finally, provide the Base64 decoded string.

Grading

- **Task 1 – 50 points (50 points per question)**
- **Task 2 – 50 points (25 points per question)**
- **Extra Credit – Up to 10 points**

Lab Survey - https://cryptomancer.typeform.com/to/GmDK2A