|  |  |  |
| --- | --- | --- |
| Course # | James Riley Dorough | Scripted Covert Initial Reconnaissance, Scanning, and Enumeration |
| CSIS 462 | **22FEB2022** | **Semester Week 7, Lab 4** |

Title

* Generate a script to execute on target systems to enumerate usable information to assist in gaining and maintain persistence

Introduction

The lab should include the creation of custom malware to enumerate system information, store the data, and covertly exfiltrate the data to local team systems. This custom code should benefit persistence though awareness of blue team systems and structure. I focused on understanding the blue team local user organization and vulnerability. This should help me know what accounts to use in future exploitation.

Method

* Determine the location for your output file and an inconspicuous file name
* Use the “Get-CimInstance” cmdlet to recover specified user accounts
* Format the output to remain readable
  + Currently working on: Encoding output to obfuscate from blue team (Output formatting does not yet output new line characters. I intend to work on data format checking to allow for manual return insertion or figure out how to encode the return characters themselves.
* Output data to specified file
* Execute the code on a target machine
* Recover & Exfil the output file
* Remove the output file from the blue team system (I intend to leave it for the blue team to discover if they look hard enough)
* Clean traces of activity (I intend to leave all logs as is and leave my binary on their systems for their edification)

Results

My focus was to return full lists of local users and pull out any users that didn’t require a password. Referring to my previous payload lab, I have scripts that can make every user on the target windows box an admin. This should allow me to have access to password-less administrator accounts. Here is my main PowerShell code that outputs all of the enumerated users into a retrievable file.

Text

Description automatically generated

The output file can be left without a .txt, or any, extension to minutely help hide it. Here is the result of running the PowerShell script. I return user information along with password requirement, ID, class info, and properties. At the very end of the file are the no password accounts reprinted separately.

Application, table

Description automatically generated

Application, table

Description automatically generated

Here is where things get interesting. I attempted to format the information into Base64 encoding. It worked incredibly well for obfuscating the intent of my code. Especially when I converted my script into a binary. The next step up would be encoding my commands within the PowerShell so they can’t be reverse engineered.

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

However, sadly I haven’t yet figured out how to encode the new line characters so that the output can be read as is without needing another script to find the end of each line and manually adding the returns.

Text

Description automatically generated

Conclusion

Through this lab I gained a much better understanding of Windows cmdlets and the formatting of Windows PowerShell objects. Additionally, I am continuing to improve my covert and obfuscation TTPs. Strangely, file transferring has been one of the hardest obstacles to surmount for the semester so far. I will be building on these exploits to make them much quieter in the future. For now I hope that by leaving them slightly simple on the obfuscation, the blue team can learn from them.