|  |  |  |
| --- | --- | --- |
| Course # | James Riley Dorough | Using Shell Code and Shell Code Obfuscation Techniques |
| CSIS 462 | **05FEB22** | **Semester Week 4, Lab 1** |

Title

* Generate Useful code to benefit red team operations
* Attempt to increase persistence in blue team networks
* Obfuscate the code such that it is not easily readable or understandable to the blue team

Abstract

In this lab I have created a simple PowerShell script to get a list of all the local users on a target machine and assign them all to the administrators group. After writing and testing the PowerShell script, I compiled it into a binary executable. Using this executable, I had to decide how I wanted to run it. I have decided on leaving the executable hidden on the target machine in a specific location. With this I will import a task schedule I have created to run the executable at a specific time.

Introduction

Most noteworthy content falls in the code’s demonstration. It is hard to show the code running as it has no visible execution. However, if the code is run manually, you can see a 1.5~ second popup window. I was able to catch it in execution for this report. This shouldn’t pose a problem as if a blue team member finds the executable itself, it won’t matter if they run it.

Method

* Create any shell code you want
* Compile to a binary (I used the ps2exe command)
* Test the executable
* Implant the executable on target systems
* Set up trigger systems (task scheduler, C2 server, social engineering, manual run)

Results

My first goal was to decide what outcome I wanted from my code. I decided to attempt to ensure that all local accounts on the blue team system would have admin access semi consistently. I wrote my script in PowerShell:

Graphical user interface, text

Description automatically generated

At one point my experimented method of executing the code would halt when it reached the Administrator account. I had made a version of the code that skipped the administrator account. However, using time-based task scheduling from the administrator account ignores this hang up. The next issue was keeping the running code as hidden as possible. Currently, running the PowerShell script has a split-second popup. The executable has two options. The first is running the script itself which has a short display. The second is a shortcut that allows to hide the execution window.

A screenshot of a computer

Description automatically generated with medium confidence

This window is not visible if you run the executable from any method other than manually through a logged in user. I have created and exported a scheduled task to run this code at 0300 every morning. It should run as administrator and the executable should have admin rights as well.

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

This runs fine on test systems and the output is a list of users all with Administrator permissions.

Graphical user interface, application

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Conclusion

Through this lab I learned quite a bit about how PowerShell works within a domain connected machine along with lesser-known windows options and system settings. I learned how compiling to a binary executable allows attacks to be harder to recognize. Additionally, many third-party services for data storage or transfer flag executables and zipped files containing these binaries as malicious code. To get your attack onto a target system takes a more direct route. Windows shares, SCP, and PSCP are all viable options for me to use to deploy this payload.