

Applying DLL and Code Injection

Offensive and Defensive Tool Construction

Table of Contents

[Objectives 4](#_Toc477850605)

[Background Reading 4](#_Toc477850606)

[Important Information 4](#_Toc477850607)

[Introduction 5](#_Toc477850608)

[Problem 1 5](#_Toc477850609)

[Questions 5](#_Toc477850610)

[Problem 2 5](#_Toc477850611)

[Problem 3 5](#_Toc477850612)

Offensive and Defensive Tool Construction

Applying DLL and Code Injection

Objectives

This lab focuses on the following objectives:

* Create a remote thread.
* Inject code into a DLL.
* Evaluate file hiding techniques.
* Write backdoor code.
* Compile Python script into a Windows executable file.

Background Reading

Read Chapter 7 in the *Gray Hat Python* textbook. The following links are also useful:

* <https://www.sans.org/reading-room/whitepapers/malicious/basic-reverse-engineering-immunity-debugger-36982>
* <https://sgros-students.blogspot.ca/2014/05/immunity-debugger-basics-part-1.html>

# Important Information

* For *every* lab and home assignment, store all your work in your personal repository in a subdirectory named **mXX**, where XX is the module number. Carefully name the program as described in each problem.
* Your programs are extracted from your repository by a Python script. If there are any errors in the program name, then your instructor will never see your program, and you will receive a mark of zero.
* Push your work to the server often, and ensure that you push the final version of a program by the deadline specified, because the script extracting them can be run at any time after the deadline.

# Introduction

In this lab, first create a remote thread by calling Windows function **CreateRemoteThread()** in kernel32.dll. We will use the **ghp\_inject.dll** by downloading the code from the Grey Hat Python website. The PyCommands are found in **/c/Program Files/Immunity Inc/Immunity Debugger/PyCommands**.

To edit files at that location, you must start your code editor (e.g., Emacs) with admin privileges.

# Problem 1

1. Using the **dll\_injector.py** from the *Grey Hat Python* textbook, start the process **calc.exe** and find its process ID. You need to inject **ghp\_inject.dll** from the Grey Hat Python website into the process. This is a simple DLL that opens a message box when loaded.
2. Use the **LoadLibrary()** function call from kernel32.dll.
3. Record the address kernel allocated for the DLL path.
4. Record the location of the LoadLibraryA() function.
5. Answer the following questions:

* What is the meaning of the values the following parameters are set to in dll\_injector.py?
* PAGE\_READWRITE
* PROCESS\_ALL\_ACCESS
* VIRTUAL\_MEM
* What is the purpose of the h\_loadlib variable?

# Problem 2

1. Practise injecting code using the **code\_injector.py** from the textbook by creating one **cmd.exe** and one **calc.exe** process. Use cmd.exe as the process to inject, and calc.exe as the process to kill.
2. Run the script with appropriate parameters and demonstrate the victim process being killed.

# Problem 3

Exploit alternative data streams on the NTFS file system. Follow Section 7.2.1 in the *Grey Hat Python* textbook and write the **file\_hider.py** code.

Improve the code so that it first asks for the name of the file to attach to, and then for the name of the DLL to be hidden interactively.

Submit your result as a file named **m09p02.py**.

**Note:** The rest of the problems for this module are available in the homework assignment. See your course schedule for details.