Taking Screenshots

Most pieces of malware and penetration testing frameworks include the capability to take screenshots

against the remote target. This can help capture images, video frames, or other sensitive data that you

might not see with a packet capture or keylogger. Thankfully, we can use the PyWin32 package (see

Installing the Prerequisites) to make native calls to the Windows API to grab them.

A screenshot grabber will use the Windows Graphics Device Interface (GDI) to determine necessary

properties such as the total screen size, and to grab the image. Some screenshot software will only

grab a picture of the currently active window or application, but in our case we want the entire

screen. Let’s get started. Crack open screenshotter.py and drop in the following code:

import win32gui

import win32ui

import win32con

import win32api

# grab a handle to the main desktop window

➊ hdesktop = win32gui.GetDesktopWindow()

# determine the size of all monitors in pixels

➋ width = win32api.GetSystemMetrics(win32con.SM\_CXVIRTUALSCREEN)

height = win32api.GetSystemMetrics(win32con.SM\_CYVIRTUALSCREEN)

left = win32api.GetSystemMetrics(win32con.SM\_XVIRTUALSCREEN)

top = win32api.GetSystemMetrics(win32con.SM\_YVIRTUALSCREEN)

# create a device context

➌ desktop\_dc = win32gui.GetWindowDC(hdesktop)

img\_dc = win32ui.CreateDCFromHandle(desktop\_dc)

# create a memory based device context

➍ mem\_dc = img\_dc.CreateCompatibleDC()

# create a bitmap object

➎ screenshot = win32ui.CreateBitmap()

screenshot.CreateCompatibleBitmap(img\_dc, width, height)

mem\_dc.SelectObject(screenshot)

# copy the screen into our memory device context

➏ mem\_dc.BitBlt((0, 0), (width, height), img\_dc, (left, top), win32con.SRCCOPY)

➐ # save the bitmap to a file

screenshot.SaveBitmapFile(mem\_dc, 'c:\\WINDOWS\\Temp\\screenshot.bmp')

# free our objects

mem\_dc.DeleteDC()

win32gui.DeleteObject(screenshot.GetHandle())

Let’s review what this little script does. First we acquire a handle to the entire desktop ➊, which

includes the entire viewable area across multiple monitors. We then determine the size of the

screen(s) ➋ so that we know the dimensions required for the screenshot. We create a device

context[18] using the GetWindowDC ➌ function call and pass in a handle to our desktop. Next we need

to create a memory-based device context ➍ where we will store our image capture until we store the

bitmap bytes to a file. We then create a bitmap object ➎ that is set to the device context of our

desktop. The SelectObject call then sets the memory-based device context to point at the bitmap

object that we’re capturing. We use the BitBlt ➏ function to take a bit-for-bit copy of the desktop

image and store it in the memory-based context. Think of this as a memcpy call for GDI objects. The

final step is to dump this image to disk ➐. This script is easy to test: Just run it from the command

line and check the C:\WINDOWS\Temp directory for your screenshot.bmp file. Let’s move on to