**CAP 4145 Introduction to Malware Analysis**

**Assignment 5 – Manually Generating Shellcode**

**10 points**

### Instructions:

1. **Note**: Blue text points to a web link. Ctrl + Click to follow link.
2. This is a team assignment. However, every student MUST submit the term project report even if all members of a group submit the same report.
3. Answers to all questions must be put into **ONE** document. That is, every time, each student can only submit one report document, answering all questions of this assignment, if not explicitly stated otherwise.
4. Students must put answers following each question in this assignment. The instructor will not grade a report with only answers in it and the student gets zero for such an assignment. An assignment report must include original questions.
5. Students MUST submit the finished assignment in either Microsoft Word or pdf format to Webcourse. The doc must be submitted as ONE standalone file and cannot be tarred or zipped into a container.
6. All required files or docs must be submitted in one submission (last submission). Note: Blackboard allows unlimited number of submission of one assignment by students.
7. Refer to [Print screen](http://en.wikipedia.org/wiki/Print_screen) on how to take a screenshot. Pressing the Alt key in combination with PrtSc will capture the currently selected window.

**Problems:**

***Answer each question following the original question. Do NOT delete the original question.***

Students are provided an example assembly code *shellcode\_cmd\_fixed.asm* and an example shellcode testing code *TestShellcode.c*. Windows API addresses in *shellcode\_cmd\_fixed.asm* must be changed, and the shellcode in *TestShellcode.c* must be changed in the context of the student’s VM so that the shellcode works.

**Notes [1]**:

“Most Windows process (\*.exe) are loaded in (user mode) memory address 0x00400000, that's what we call the "virtual address" (VA) - because they are visible only to each process, and will be converted to different physical addresses by the OS (visible by the kernel / driver layer).”

“Regarding RVA (Relative Virtual Address), it's simply designed to ease relocation. When loading relocable modules (eg, DLL) the system will try to slide it through process memory space. So in file layout it puts a "relative" address to help calculation.”

**Hints:**

* To manually get the address of a function in a dll,
	+ Get the base address of the dll using [*listdlls*](https://docs.microsoft.com/en-us/sysinternals/downloads/listdlls)
	+ Get the RVA of the function in the dll with *peview*
	+ The address of the function = dll base address + function RVA
* Compile with nasm and link with [*GoLink*](http://www.godevtool.com/Golink.zip)
* Get the shellcode with *OllyDbg*
* Compile the shellcode test code with gcc from [Mingw-*w64*](https://mingw-w64.org/doku.php)

**Requirements:**

* To manually get the address of a function in a dll,
	+ Get the base address of the dll using [*listdlls*](https://docs.microsoft.com/en-us/sysinternals/downloads/listdlls). Please provide a screenshot of the obtained base address. (1 point)
	+ Get the RVA of the function in the dll with *peview*. Please provide a screenshot of the base address for each of the two Windows functions (WinExec and ExitProcess) in *peview*. (1 point)
	+ The address of the Windows function = dll base address + function RVA. Write down the addresses of the two functions below. (1 point)
* Update *shellcode\_cmd\_fixed.asm* with correct addresses of the two Windows functions/APIs, compile the assembly with [nasm](http://www.nasm.us/pub/nasm/releasebuilds/2.14.02/) and link the object file with [GoLink](http://www.godevtool.com/Golink.zip). The instructions of compilation and linking are inside the .asm file. Please provide a screenshot of the compilation and linking. (1 point)
* Get the shellcode with OllyDbg. Please provide a screenshot of the shellcode in OllyDbg. (1 point)
* Copy the shellcode into *TestShellcode.c*, and compile it with gcc from [i686-posix-dwarf](https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/8.1.0/threads-posix/dwarf/i686-8.1.0-release-posix-dwarf-rt_v6-rev0.7z) of [Mingw-w64](https://mingw-w64.org/doku.php). Please provide a screenshot of the compilation. (1 point)
* Run the testing shellcode code on the target VM. Please provide a screenshot of the running result. (4 points)

**References**

1. [VA (Virtual Address) & RVA (Relative Virtual Address)](https://stackoverflow.com/questions/2170843/va-virtual-address-rva-relative-virtual-address), Jul 3 '18 at 17:31