Lab 06 – JOP with Vet APP

**Part 1 - Finding the Memory Disclosure**

* **[1 points]** You already found the ASLR bypass that you will need to complete this lab. There is an additional UAF due to poor programming practices. This one can be used to reveal a location on the heap. Can you find this UAF? Explain where it is and the mechanics behind it.Identify the function responsible for this. Feel free to use Gflags or static analysis to identify this.
* **[6 points]** What is the new memory disclosure? (Note: This was present in the previous lab, but not emphasized. Some of you found this whilst searching for the other UAF.) How would you weaponize this and use it as means to fill it with JOP gadgets for your exploit. Yes, this means, you will want to store your gadgets on the heap for this JOP exploit. You will need to figure out a way to use Vet to store the JOP gadgets; there are only so many possibilities! Show your work and explain.

**Part 3 – Bypassing DEP with JOP**

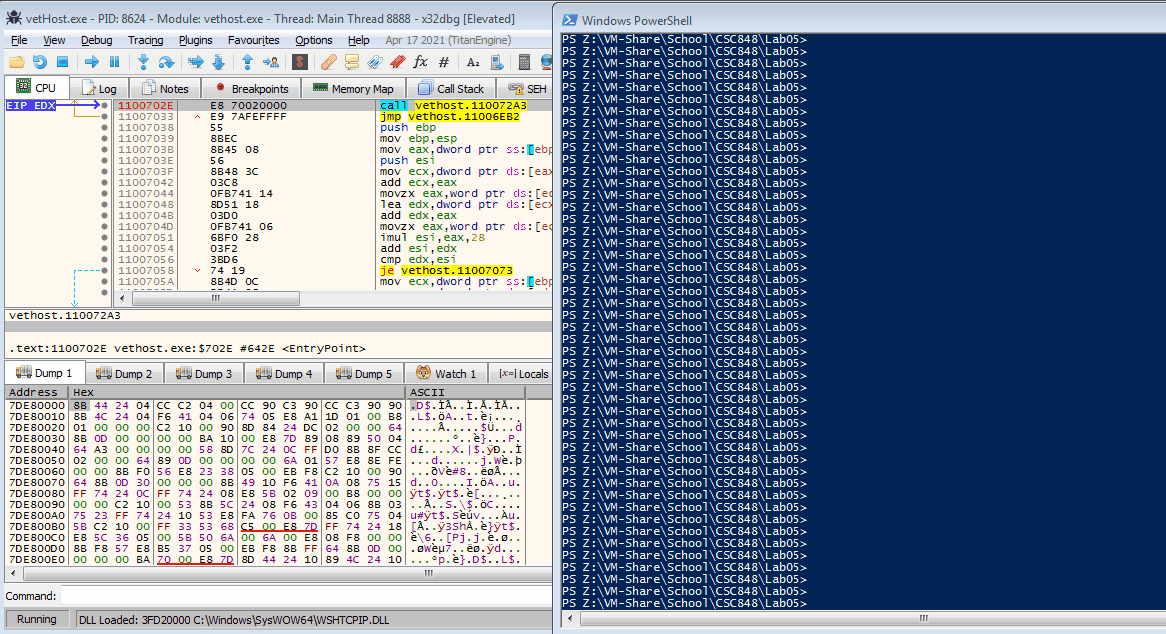
**Note: Demonstrated use of JOP ROCKET is required for any points to be given in this section.**

* This step is not possible without ASLR bypass. Do not submit hypothetical, “what if there were no ASLR”-code. Make sure ASLR is bypassed via memory disclosure before embarking on this part.
* Figure out how to bypass DEP with JOP. You may need to use an arbitrary write, rather than pushing gadgets. It is okay to use one or two ROP gadgets to set up the JOP. Additionally, if you use multiple Windows API’s, you may need to use a tiny bit of ROP to chain them together, to get back into JOP; that is acceptable. Otherwise, you will need to stick to JOP.
* **[12 points]** Provide a detailed narrative that details your JOP chain. At a minimum, discuss:
  + **[1 points]** How did you begin execution of your JOP chain?
  + **[9 points]** What technique does your JOP chain use to bypass DEP. Show that it works. **Include the Python script showing that this works on the Windows 7 VM provided by Dr. Brizendine; use the provided VM, so that your script can be tested in the same environment. Make sure to have detailed comments in the script, explaining functionality.** Include a detailed, well written narrative that walks through and explains what you did and how you did it, any obstacles, anything noteworthy. You do not need to explain every detail. A screenshot is not a narrative.
  + **[2 points]** How did you use the stack in this JOP chain? How did you get your values there and prepare them for execution of payload? Were there any problems, and If so, what workarounds did you utilize?

**Part 2 - Writing Shellcode**

**The shellcode is the same as before. So the overall process goes like:**

1. **Download second stage from Github as “not\_malware.test”**
   1. **It is saved to the same location the executable is ran**
2. **Execute the payload**
   1. **Sends email/sms notif to me that the payload was ran using pyinstaller executable generated from custom python script**
3. **Sleep/Wait for 5000ms**
4. **Delete the payload named “not\_malware.test”**
5. **ExitProcess() gracefully**

****

**You should be able to follow that process in the following GIF.**

**Files Included:**

* **micah (bin)**
* **micah.asm**
* **micah.py**