Lab 01 – Getting Started with ROP

There are multiple parts to this lab to help you get warmed-up for the many topics that we’re going to cover this semester. Please complete them in order, as they will build upon one other.

This is intended to be a refresher on ROP, not an introduction to the topic.

The target binary is a hash cracker. The intent of the program is inherently flawed, and similarly this program contains a vulneraiblity.

Total points: 20

**Part 1 - Finding the Exploit (threaded-server.exe)**

* Run the program; understand how it works and how you can interact with it. Think about the flow of user-supplied data; trace those functions to find where the buffer overflow occurs.
* This is a toy binary, so it is not overly complex. However, this one does have a lot of distractions inside it—things that may not be pertinent to a vulnerability or may not serve any useful purpose. What is important and what is not? This lab is just intended to get people thinking again on ROP.
* Once you’ve identified the vulnerable function, determine how large your payload needs to be in order to exploit. That is, understand the stack frame (the size of the buffer) plus any additional stack variables.
* Create a Python script to help you interact with this tool.
  + It is okay to share general tips and hints on the Homework Help forum on how to create a script or where to begin, if unsure.
* Provide details about the vulnerability such as how you found it and how you were able to exploit it. Show a screenshot of what you believe to be the vulnerability. Also provide evidence via a debugger that you had control of EIP at this point in the lab. This could be, for instance, capturing EIP with 4 A’s. Note, 4 A’s is unique; 400 A’s is not! You must be **precise** in showing the amount of overflow needed.
  + This is not a complex task that requires numerous screenshots or extended narrative. This is just getting you warmed up.

**Part 2 – Bypassing DEP with ROP**

* Once you’ve been able to control EIP, your next step is to create a ROP chain that will bypass DEP. It is up to you to decide on the most effective technique; the goal is to bypass DEP with ROP to execute your shellcode.
* Provide details about your ROP chain. At a minimum, discuss:
  + How did begin execution of your ROP chain?
  + What technique does your ROP chain use to bypass DEP. Is the technique something supplied by Mona, or did you go and use an alternative approach? Demonstrate that it works. An exploit that does not work is not useful.
  + Show that your shellcode is ready to be executed after your ROP chain completes. This will depend on the technique you used to bypass DEP. An example of this is showing that you were able to get instructions (any instructions) to be executing in the debugger, rather than being interpreted as virtual memory addresses. Provide screenshot evidence of success.

**Part 3 - Writing Shellcode**

Now it’s time to create your own shellcode! You may use the provided shellcode to help get you started or create your own. In any case, do NOT use a framework such as MSFVENOM to generate your shellcode. You will execute your shellcode after you’ve successfully bypassed DEP. Your shellcode should do something useful, such as establish a REVERSE TCP session or download a payload from a server (for example, host calc somewhere and drop and execute it).

* Provide an analysis of your shellcode. At a minimum, discuss:
  + How does your shellcode make Windows API calls
  + What functionality is it providing and what APIs does it need.
  + Show that your shellcode worked.

**You may use shellcode from the previous lab**, if you created your own shellcode. If you did not create your own shellcode, then you must create your shellcode here. Students may not use shellcode from MSFvenom or online sources. Supply the .ASM or your shellcode.

**Deliverable**

* A Word document or PDF with requested answers to the previous tasks.
* **Page limit is 11 pages. Anything beyond that is not read or used for calculating your grade.**
* **Standards.** At DSU we are proud of the work our students do, and we expect all students to live up to high standards. Your submission should be well written, concise but with specific and relevant details, orderly, with either numbers or headings indicating what you are responding to. Screenshots should be **cropped, readable, and appropriately size.** Messy submissions with some of the aforementioned problems may not be graded. Depending on the issue or frequency, the professor may offer you a chance to resubmit, or may give a zero.