Multi-Stage Exploits

This attack will include several other attacks such as Information Leakage, GOT Overwrite and the ROP attack. In the lab4.c we see the main function and the vuln function. We see many problems with this program, the buffer length is 25 but there is a read function that goes up to 100 in the buffer which will trigger a stack overflow which we can use, there is also the puts function which is also a vulnerable function to use and we can use it to hack into this function.

First, we need to gdb into the binary file and then disassemble the write function to the get the starting memory address which is 0x080490b4 as shown in Image 1 then use the memory address into which it is jumping into which is 0x804c018 as shown in Image 1 to then to launch the GOT. We write the command: x/xw 0x804c018 to check as shown in Image 2 and note down the memory address 0x08049070. Then we disassemble the read function and note down its memory address: 0x08049084, then go into the ROP gadget of the binary file lab4 to find the address of the pop\_pop\_pop\_ret which is: 0x080492b1. We also go into the libc database to under the right version to find the offset of these data and calculate the system@libc address. Using the grep “ed” command we find the address for a string as shownin Image 4. We then follow the payload structure in the shell code like so:

|  |
| --- |
| Dummy “A” \* (25+12) |
| write@plt address -> 0x080490b4 |
| Pop\_pop\_pop\_ret -> 0x080492b1 |
| 1 |
| write@got -> 0x804c018 |
| 4 |
| read@plt -> 0x08049084 |
| Pop\_pop\_pop\_ret -> 0x080492b1 |
| 0 |
| write@got -> 0x804c018 |
| 4 |
| system@plt to write@plt |
| 0xdeadbeef |
| “ed” string -> 0x80482c5 |

As shown in Image 5 the python attack file as represented in the payload structure. With this we run it and get inside the shell which we then can find the flag.txt and see what is in there. As shown in Image 6 having successfully gotten inside the flag.txt through the remote server.

All in all we see through this attack having utilized many different attacks learnt previously such as the ROP attack, the GOT attack, and the stack overflow attack. We further see the vulnerabilities of the reads and writes functions as well as the system function again. This lab did a great job of binding all these attack to create a multi-stage exploit as the name explains it.

# Source Images:

Text

Description automatically generated Text

Description automatically generated

Image Image 2

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with low confidence

Image Image 4

Text

Description automatically generated A picture containing diagram

Description automatically generated

Image Image 6