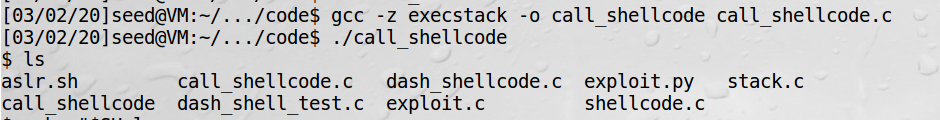
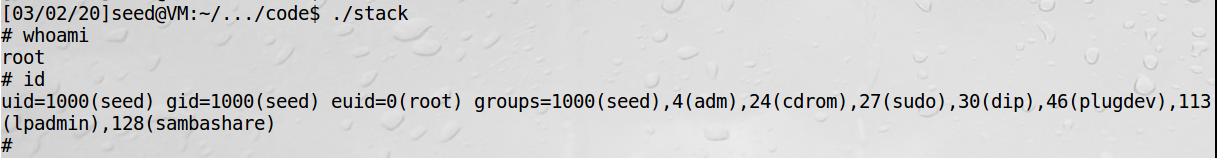
Q1: None

Q2: By running **call\_shellcode**, the shell we previously linked to (i.e., zsh) is invoked.



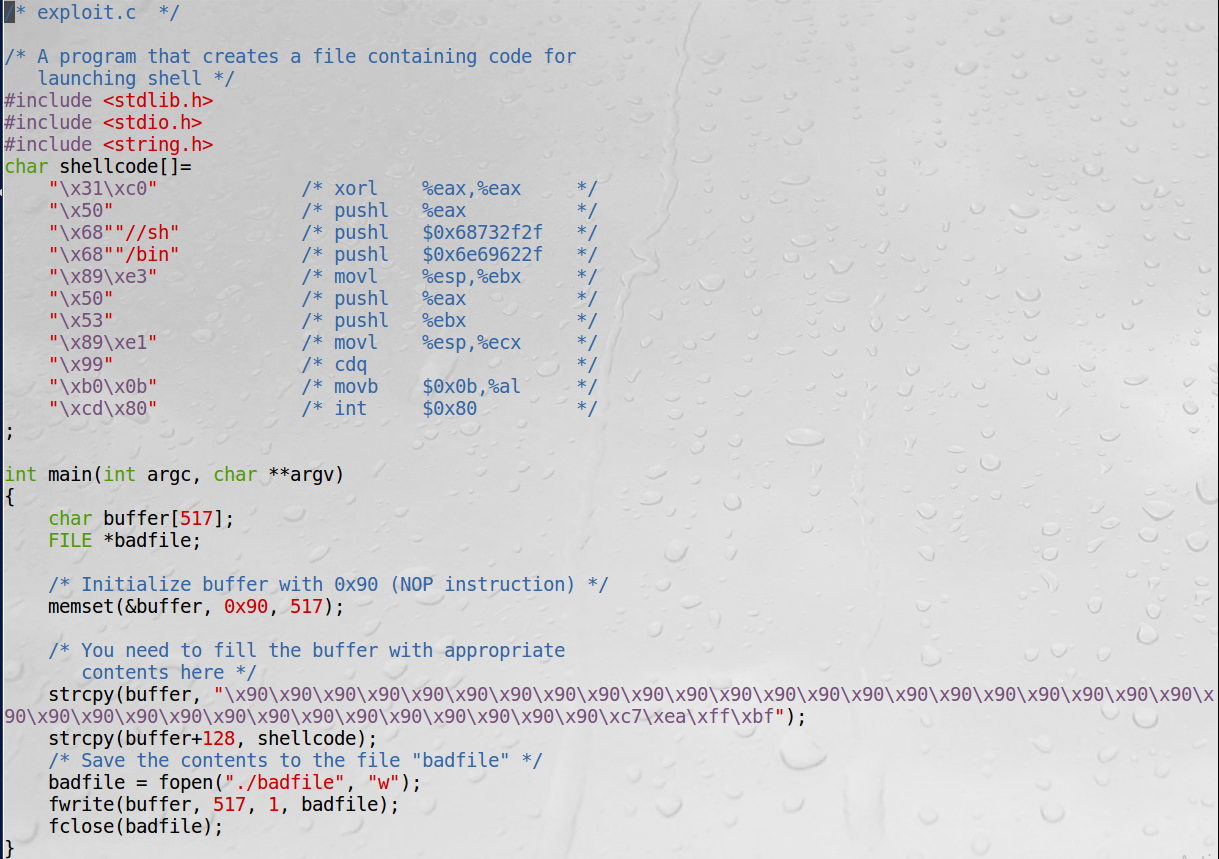
Q3: By running the set-uid program **stack**, we invoked the zsh shell with root privilege.



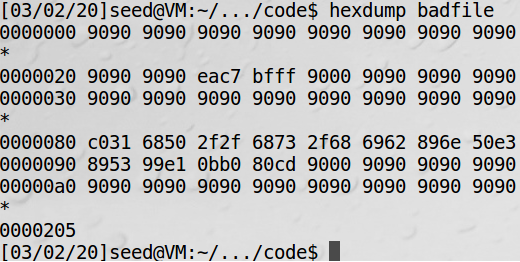
Q4: Memory address of the variable buffer: 0xbfffea08



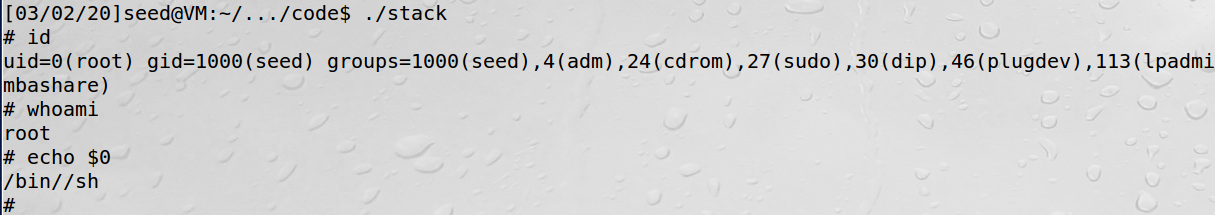
Q5:



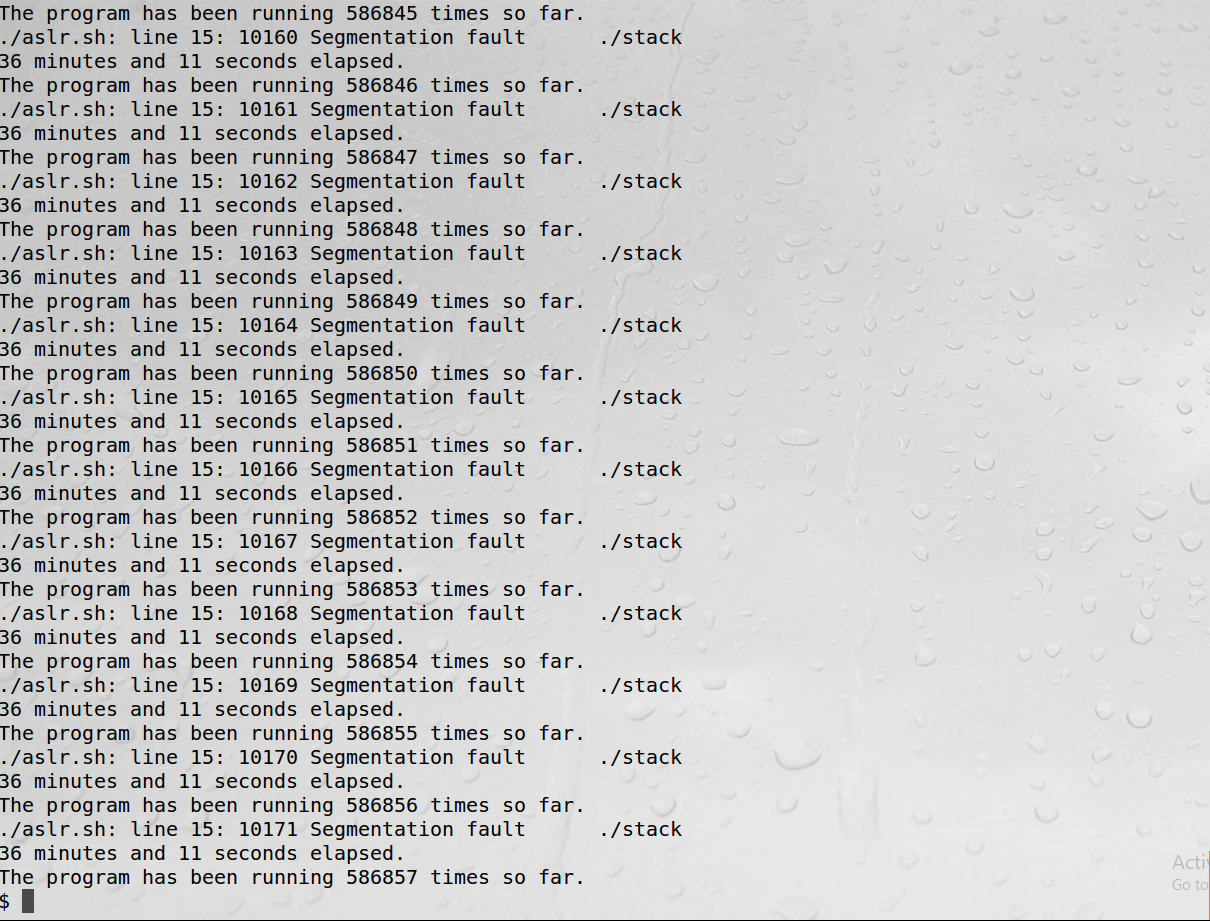
Q6:



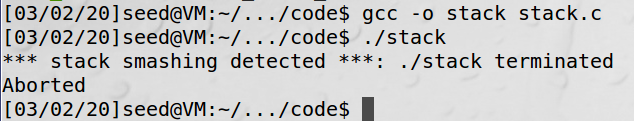
Q7: By running the set-uid program **stack**, we invoked the dash shell with root privilege. Task 3 defeats dash’s countermeasure by setting uid to 0 (which stands for root) before the check in Line 11 happens (which compares real and effective user/group IDs). Since now both real and effective user IDs are root, and both real and effective group IDs are seed, the program passes the check and continues to invoke the dash shell with root privilege.



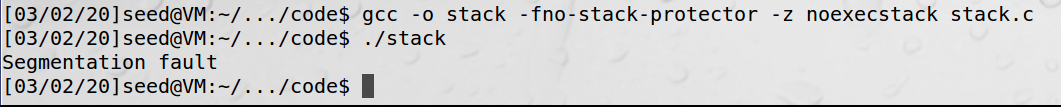
Q8: After 36 minutes and 11 seconds, 586857 iterations, we managed to invoke the shell.

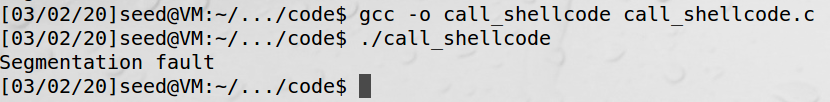


Q9: When StackGuard is present, the **stack** program reports an error: “stack smashing detected” and the program is terminated.



Q10: Segmentation fault is observed. Since non-executable stack protection is enabled, when trying to execute code in a non-executable stack, the system detects the violation and aborts the program.





Q11: Using the hint, we can allocate the buffer on the heap instead of on the stack. And so the function bof is able to copy arbitrary long inputs in the file **badfile** into the heap without corrupting memory.

