West Chester University

CSC 472

Dr. Si Chen

Fall 2023 Lab 5

Submitted by

Sean Berlin

11/30/23

1. **Introduction:** The purpose of lab 5 laboratory was to gain a practical understanding of kernel exploitation concepts, specifically focusing on exploiting a Use-After-Free (UAF) vulnerability in kernel space. The experiment aims to provide hands-on experience in launching kernel exploitation by employing the QEMU emulator. The objectives include tweaking the default file system, compiling and executing kernel exploitation shellcode, and gaining insights into the concept of UAF

2. **Analysis and Results:**

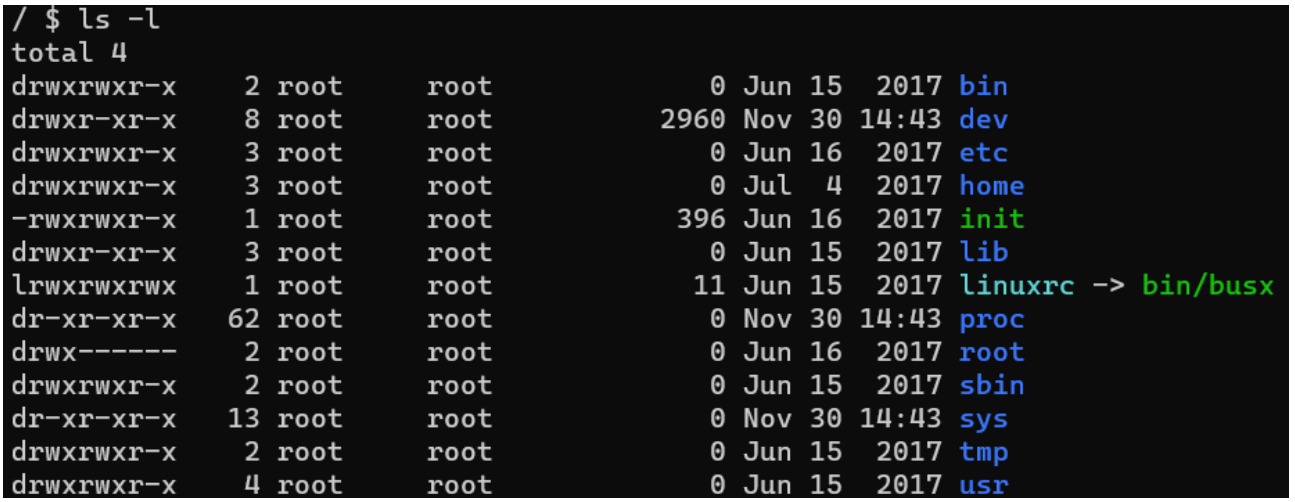
Questions:

Question 1: How many folders are there inside the root (/) folder?

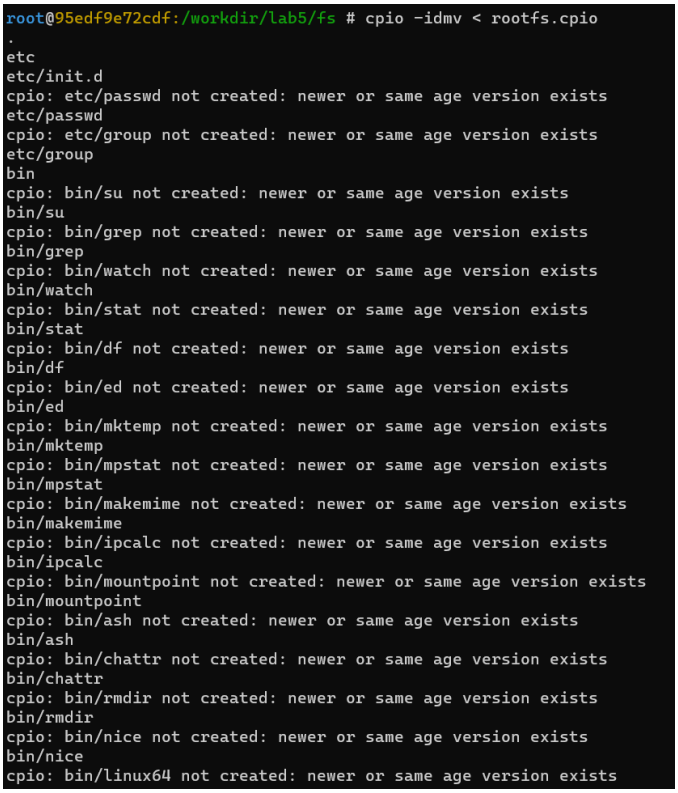
Answer: There are 11 folders inside the root (/) folder.

-rwxrwxr-x 1 root root 396 Jun 16 2017 init : Regular file indicated by '-'

lrwxrwxrwx 1 root root 11 Jun 15 2017 linuxrc -> bin/busx :l: Symbolic link indicated by 'l' in "lrwxrwxrwx"

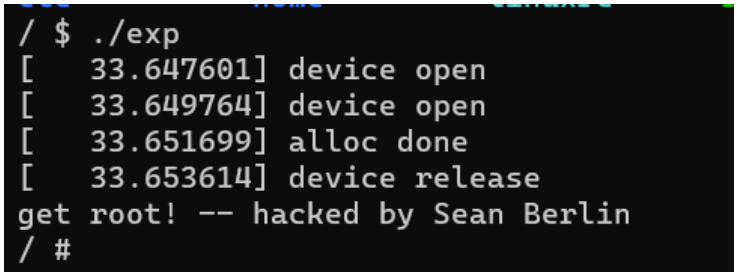


Question 2: Please take a screenshot and show me the output after typing the command: ‘cpio -idmv ¡ rootfs.cpio‘.
‘cpio -idmv ¡ rootfs.cpio‘ is invalid but ‘cpio -idmv < rootfs.cpio‘ produces an output
Screenshot:



Explanation: the ‘cpio -idmv < rootfs.cpio’ command extracts the contents of the CPIO archive rootfs.cpio with output showing the directory structure and files within the just extracted filesystem.

Question 3: (Inside the QEMU Linux virtual machine) Please take a screenshot of the output after typing the command: ‘./exp‘.
Screenshot:



Question 4: In the shellcode (exp.c), why do we want to open the device (/dev/babydev) twice?
Answer:
The opening of the device twice looks to trigger a race condition. A race condition occurs when multiple processes or threads are trying to perform operations at the same time with the behavior depending on the timing of their execution. To be specific, by opening both devices at the same time, the second time will overwrite the first allocated space. Similarly, if the first one is released, then the second one is released causing a UAF. This race condition is then used by the following code exploits.

Question 5: In the shellcode (exp.c), what's the purpose of ioctl(fd1, 0x1001, 0xa8)? Why use 0xa8?
Answer:
To start on the most basic level, the line of code performs an ioctl operation on 'fd1' with the command '0x1001' and the parameter '0xa8'. In other words, The ioctl() call manipulates the underlying device parameters of special files. The value '0xa8' is a parameter passed to change the size of the device to the size of the cred structure via the iotcl command.

Question 6: In the shellcode (exp.c), what's the meaning of write(fd2, zeros, 28)?
Answer:
To start on the most basic level, the line of code writes 28 bytes from the buffer "zeros" to the file descriptor fd2. Specifically, it is writing to the space that was recently overlapped as a result of "forking" a new process.

Analysis:
The initial phase of the experiment involved booting up QEMU which is a generic machine emulator and is essential for creating a controlled environment for kernel exploitation. The default file system was then modified to fit the experiment's specifications. The compilation and execution of kernel exploitation shellcode were then performed, emphasizing the importance of understanding the underlying UAF concepts.

One critical aspect of the analysis focused on answering specific questions related to the root folder's structure and the output of key commands inside the QEMU Linux virtual machine. The examination of the 'exp.c' shellcode revealed the utilization of a race condition by opening the device file /dev/babydev twice. The ioctl operation with the command '0x1001' and the parameter '0xa8' was identified as a crucial step, manipulating the device parameters to the size of the cred structure. Additionally, the 'write(fd2, zeros, 28)' command was found to exploit the recently overlapped space resulting from forking a new process.

3. Discussion and Conclusion:

Generated by ChatGPT- In conclusion, the laboratory experiment successfully met its objectives, providing a practical exploration of kernel exploitation techniques. The utilization of QEMU, customization of the file system, and the execution of kernel exploitation shellcode contributed to a comprehensive understanding of UAF vulnerabilities. The observed behaviors, such as the race condition triggered by opening the device file twice and the specific use of ioctl and write commands, demonstrated the intricacies of kernel-level exploits. The alignment between theoretical concepts and practical outcomes reinforces the importance of security awareness and ethical considerations when delving into kernel exploitation.