CSCI 5573: Advanced Operating Systems
Fall 2024

Programming Assignment One

Due Date: 10/13/2024

1. Write a program to report the behavior of your Linux kernel. Your program should run in two different versions. The default version prints the following values on stdout:

   Processor type
   Kernel version
   The amount of memory configured into this computer
   Virtual memory address range mapped to System RAM, Kernel (code, data, bss)
   Total size of RAM and size of memory that can be used for new processes
   Amount of time since the system was last booted

   A second version of the program runs continuously and prints lists of the following dynamic values (each value in the lists is the average over a specified interval):

   The percentage of time the processor(s) spends in user mode, system mode, and the percentage of time the processor(s) are idle
   The amount and percentage of available (or free) memory
   The rate (number of sectors per second) of disk read/write in the system
   The rate (number per second) of context switches in the kernel
   The rate (number per second) of process creations in the system

   If your program (compiled executable) is called *proc_parse*, running it without any parameter prints out information required for the default version. Running it with two parameters "*proc_parse* <*read_rate*> <*print_rate*>" prints out information required for the second version. *read_rate* represents the time interval between two consecutive reads of the /proc file system. *print_rate* indicates the time interval over which the average values is calculated. Both *read_rate* and *print_rate* are in seconds. For instance, *proc_parse 2 60* reads kernel data structures once every two seconds. It then prints out averaged kernel statistics once a minute (average of 30 samples). The second version of your program doesn't terminate.

2. Write a device driver as an LKM for a character device called *chardev* that allows processes to read from it, write into it and seek into it. A process must open the device to use it and close it once it is done. Simulate the device as a fixed buffer of size 1 KB initialized with all characters being '0'. Multiple processes may use this device concurrently, so any data written by a process can be read by another process.

   Include some test programs to demonstrate that your driver works with multiple processes running concurrently.

CSCI 5573: Advanced Operating Systems
Fall 2024

3. Intercepting a system call (For this assignment, use an earlier version of Linux kernel - preferably 4.19.x or earlier).

   A rootkit is a collection of computer software, typically malicious, designed to enable access to a computer or areas of its software that is not otherwise allowed (for example, to an unauthorized user) and often masks its existence or the existence of other software. A common way rootkits work is by manipulating systems calls.

   Your task is to write a very simple rootkit module that affects *ls* such that files/folders whose name begin with prefix *abc* are not listed when *ls* is run. These files/folders should still be available and you should be able to access them using other utilities, e.g. by using an editor.

   To write this rootkit, you will first identify the system call that *ls* uses to find the file listings. Next, you will write an LKM to intercept that system call. Use the methodology discussed in class to do this.

4. Write a kernel probe in your rootkit kernel module that dumps the current state of the module before and after a specific instruction of the module is executed. You can choose any instruction in the module to trap. The dumped state should include information such as current values of registers, process ids, etc.

5. (Extra Credit) The method for modifying a system call discussed in class does not work on the later Linux version. This is because sensitive bits in CR0 and CR4 are pinned since version 5.3, when updated via kernel functions such as *write_cr0* and *write_cr4*. However, if you're in supervisor mode, you can always write CR0 directly, which should avoid the pinning. Try this and see if it works.