Sistemas de Computación
Trabajo Práctico N°4

Grupos: !(amantes-del-assembler) - SampleText
Integrantes:

- Mercaú Nievas, Nehemías
- Más Casariego, Sebastián
- Ulla, Juan Ignacio
- Di Giannantonio, Marco Valentino
- Villa, Leandro Augusto

Desafío 1

Revisar la bibliografía para impulsar acciones que permitan mejorar la seguridad del kernel, concretamente: evitando cargar módulos que no estén firmados.

Algunas acciones que permiten mejorar la seguridad del kernel es no permitir que se desactive el arranque seguro de UEFI ya que de esta manera sólo se pueden cargar aquellos módulos del kernel firmados cuyas firmas fueron autenticadas contra claves del llavero del sistema y del llavero de la plataforma, otra acción es que se tenga que especificar el parámetro del kernel “module.sig_enforce”. En otras palabras, si el arranque seguro está habilitado, los módulos tienen que ser firmados con una clave privada y autenticados con la clave pública correspondiente para poder finalizar el proceso de arranque.

Desafío 2

Debe tener respuestas precisas a las siguientes preguntas y sentencias:

- ¿ Qué funciones tiene disponible un programa y un módulo ?

  Un programa puede tener múltiples funciones que son definidas por el programador (o exportadas de librerías) y que se pueden implementar a nivel de usuario. El programa comienza con una función main() que, al finalizar la ejecución, retorna el control al sistema operativo.
  Un módulo del kernel, en cambio, es un archivo de tipo objeto que se pueden sumar (habilitar / deshabilitar) en tiempo de ejecución. Para implementar módulos, se deben utilizar 2 funciones (module_init y module_exit) que se les pasará por parámetro 2 funciones que implementarán acciones al cargar (init) o eliminar el módulo (exit). Cuando compile mi código del módulo, se generará un archivo de tipo “.ko” que, al

insertarlo, pasará a estar en el espacio del kernel (quien interpreta los símbolos) al que el usuario no podrá acceder. Cabe aclarar que las funciones externas que se pueden implementar están limitadas por el kernel y se pueden ver en /proc/kallsyms.

- Espacio de usuario o espacio del kernel.

  Espacio del kernel: El sistema operativo Linux y en especial su kernel se ocupan de gestionar los recursos de hardware de la máquina de una forma eficiente y sencilla, ofreciendo al usuario una interfaz de programación simple y uniforme. El kernel, y en especial sus drivers, constituyen así un puente o interfaz entre el programador de aplicaciones para el usuario final y el hardware. Toda subrutina que forma parte del kernel tales como los módulos o drivers se consideran que están en el espacio del kernel.

  Espacio de usuario: Los programas que utiliza el usuario final, tales como las "shell" u otras aplicaciones con ventanas como por ejemplo "kpresenter", residen en el espacio de usuario . Como es lógico estas aplicaciones necesitan interaccionar con el hardware del sistema, pero no lo hacen directamente, sino a través de las funciones que soporta el kernel.

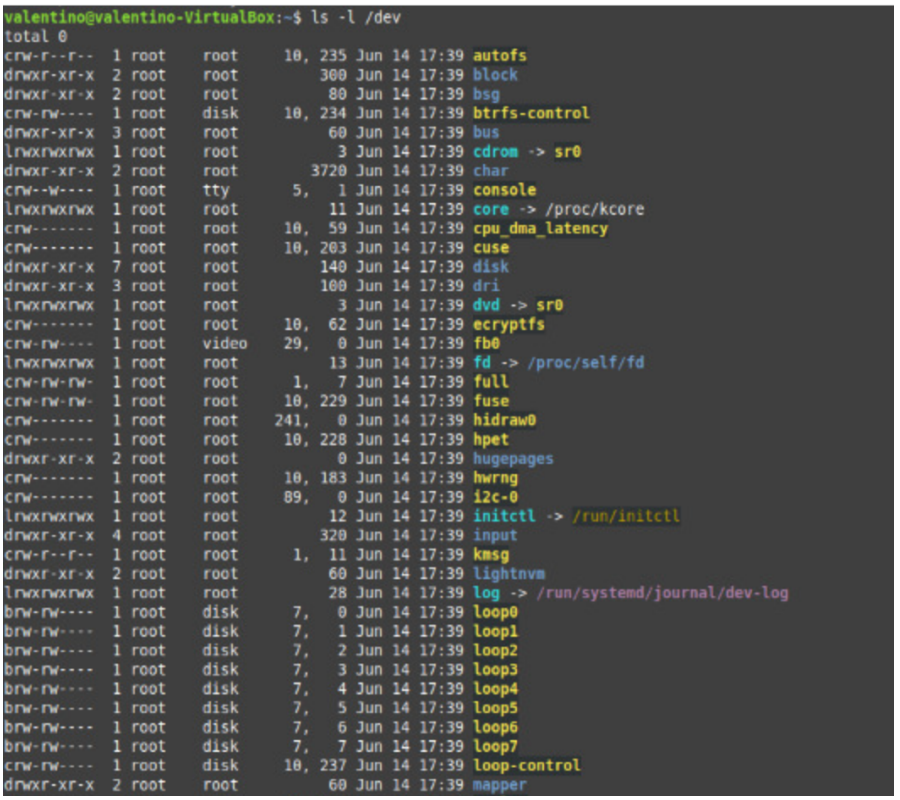
  > Para acceder al espacio del kernel se necesitan privilegios especiales, que una aplicación en el espacio de usuarios no dispone. Existen mecanismos para poder acceder a este espacio de mayor privilegios desde el espacio de usuario (system calls).

- Espacio de datos.

  “Un espacio de datos es un rango de hasta dos gigabytes de direcciones de almacenamiento virtual contiguas que un programa puede manipular directamente mediante instrucciones del ensamblador. A diferencia de un espacio de direcciones, un espacio de datos contiene sólo datos; no contiene áreas comunes ni datos o programas del sistema. El código del programa no se ejecuta en un espacio de datos, aunque un programa puede residir en un espacio de datos como código no ejecutable.” (Fuente)

- Drivers. Investigar contenido de /dev.

  Un driver es un tipo de módulo que está en un controlador de dispositivo y nos permite comunicarnos con el hardware.
  El directorio /dev contiene los archivos de dispositivos especiales para todos los dispositivos de hardware. Los archivos de dispositivos se nombran utilizando convenciones especiales. En cuanto al tema de los privilegios, podemos encontrar que cada uno de los dispositivos generalmente comienzan con la letra “b” si corresponde a un bloque, con la letra “c” si es un archivo de tipo carácter (estos poseen un número de mayor y menor) o con la letra “d” si se hace referencia a un directorio.


valentino@valentino-VirtualBox:~$ ls -l /dev
total 0
crw-r--r-- 1 root   root   10, 235 Jun 14 17:39 autofs
drwxr-xr-x 2 root   root   300 Jun 14 17:39 block
drwxr-xr-x 2 root   root   80 Jun 14 17:39 bsg
crw-rw---- 1 root   disk   10, 234 Jun 14 17:39 btrfs-control
drwxr-xr-x 3 root   root   60 Jun 14 17:39 bus
lrwxrwxrwx 1 root   root   3 Jun 14 17:39 cdrom -> sr0
drwxr-xr-x 2 root   root   3720 Jun 14 17:39 char
crw--w---- 1 root   tty   5,   1 Jun 14 17:39 console
lrwxrwxrwx 1 root   root   11 Jun 14 17:39 core -> /proc/kcore
crw------- 1 root   root   10, 59 Jun 14 17:39 cpu_dma_latency
crw------- 1 root   root   10, 203 Jun 14 17:39 cuse
drwxr-xr-x 7 root   root   140 Jun 14 17:39 disk
drwxr-xr-x 3 root   root   100 Jun 14 17:39 dri
lrwxrwxrwx 1 root   root   3 Jun 14 17:39 dvd -> sr0
crw------- 1 root   root   10, 62 Jun 14 17:39 ecryptfs
crw-rw---- 1 root   video   29,   0 Jun 14 17:39 fb0
lrwxrwxrwx 1 root   root   13 Jun 14 17:39 fd -> /proc/self/fd
crw-rw-rw- 1 root   root   1,   7 Jun 14 17:39 full
crw-rw-rw- 1 root   root   10, 229 Jun 14 17:39 fuse
crw------- 1 root   root   241,   0 Jun 14 17:39 hidraw0
crw------- 1 root   root   10, 228 Jun 14 17:39 hpet
drwxr-xr-x 2 root   root   0 Jun 14 17:39 hugepages
crw------- 1 root   root   10, 183 Jun 14 17:39 hwrng
crw------- 1 root   root   89,   0 Jun 14 17:39 i2c-0
lrwxrwxrwx 1 root   root   12 Jun 14 17:39 initctl -> /run/initctl
drwxr-xr-x 4 root   root   320 Jun 14 17:39 input
crw-r--r-- 1 root   root   1, 11 Jun 14 17:39 kmsg
drwxr-xr-x 2 root   root   60 Jun 14 17:39 lightnvm
lrwxrwxrwx 1 root   root   28 Jun 14 17:39 log -> /run/systemd/journal/dev-log
brw-rw---- 1 root   disk   7,   0 Jun 14 17:39 loop0
brw-rw---- 1 root   disk   7,   1 Jun 14 17:39 loop1
brw-rw---- 1 root   disk   7,   2 Jun 14 17:39 loop2
brw-rw---- 1 root   disk   7,   3 Jun 14 17:39 loop3
brw-rw---- 1 root   disk   7,   4 Jun 14 17:39 loop4
brw-rw---- 1 root   disk   7,   5 Jun 14 17:39 loop5
brw-rw---- 1 root   disk   7,   6 Jun 14 17:39 loop6
brw-rw---- 1 root   disk   7,   7 Jun 14 17:39 loop7
crw-rw---- 1 root   disk   10, 237 Jun 14 17:39 loop-control
drwxr-xr-x 2 root   root   60 Jun 14 17:39 mapper