

Sistemas de Computación

Trabajo Práctico N°3

Grupo: SampleText

Integrantes:

- Ulla, Juan Ignacio
- Di Giannantonio, Marco Valentino
- Villa, Leandro Augusto

Desafío 1

Desafío: UEFI y coreboot (40 min)

¿Qué es UEFI? ¿cómo puedo usarlo? Mencionar además una función a la que podría llamar usando esa dinámica.

¿Menciona casos de bugs de UEFI que puedan ser explotados?

¿Qué es Converged Security and Management Engine (CSME), the Intel Management Engine BIOS Extension (Intel MEBx).?

¿Qué es coreboot ? ¿Qué productos lo incorporan ? ¿Cuáles son las ventajas de su utilización?

- 1) UEFI (interfaz unificada de firmware extensible) es un software de bajo nivel que corre cuando se prende la PC antes de bootear el sistema operativo. Este, a diferencia de BIOS (sistema básico de entrada y salida), es algo más moderno ya que soporta discos duros de mayor capacidad, tiempos de booteo más rápidos, mayores características de seguridad, interfaz gráfica de usuario y cursores de mouse. Además, UEFI funciona directamente con modos de 32 y 64 bits (mientras que BIOS generalmente con 16). Cabe destacar que UEFI está programado principalmente en lenguaje C mientras que BIOS en una mezcla de C y Assembler. UEFI puede ser accedido mediante una tecla especial durante el booteo del SO (depende de la PC) o mediante algunos pasos a través de la sección "Configuración de Windows" ([link](#)) una vez que el SO se ha iniciado.
Algunas funcionalidades a las que podemos llamar son, por ejemplo, visualizar y configurar el proceso de arranque, los coolers de la pc (temperaturas, tensión, velocidad), funciones de seguridad, etcétera
- 2) Un bug de UEFI que puede ser explotado es aquel que permite que código malicioso pueda desactivar las medidas de seguridad de Windows en el inicio. Esto es posible ya que el fallo se encuentra dentro del **System Management Mode**, el cual está en los paquetes de firmware UEFI. Con esta vulnerabilidad, se pueden desactivar:
 - o UEFI Secure Boot.

- Virtual Secure Mode.
- Protección de credenciales de Windows 10 Enterprise.

El fallo fue descubierto en portátiles de Lenovo y se ha confirmado que afecta también a las de HP y a la gran mayoría de los demás fabricantes de portátiles.

El fallo se cataloga como “de día cero” ya que es parte del código IBV que entrega Intel, el cual va dentro de BIOS y UEFI. El término “de día cero” significa que hasta que la vulnerabilidad sea mitigada, se puede utilizar para afectar programas, datos, computadoras adicionales o redes.

Fuente: [link](#)

- 3) **CSME** es una característica de seguridad que se ejecuta al inicio y es responsable de verificar y autenticar todo el firmware posterior. Por ejemplo, verifica el firmware del UEFI o el controlador de energía para gestionar la fuente de alimentación del chip. En definitiva CSME es la base de seguridad para verificar todos los procesos sensibles. ([link](#))

Intel MEBx brinda la capacidad de cambiar y/o recopilar la configuración del hardware del sistema, es decir que se puede utilizar para ver y cambiar la configuración del motor de administración Intel para la computadora. ([link1](#))([link2](#))

- 4) Coreboot, conocido anteriormente como LinuxBIOS, es un proyecto de software de código libre apuntado a reemplazar BIOS o UEFI (los cuales se encuentran en la mayoría de las computadoras) con un firmware más liviano diseñado para realizar solo la cantidad mínima de tareas necesarias para cargar y ejecutar una PC moderna de 32/64 bits. Terceros como Chrome OS Devices, PC Engines, Star Labs, System76 y Purism producen y entregan su hardware con coreboot.

Las ventajas de coreboot son:

- Código abierto.
- Libre de derechos de autor.
- Footprint pequeño y liviano. Footprint es la cantidad de espacio que una unidad particular de hardware o software ocupa.
- Al hacer una separación de las tareas de inicio permite implementar cosas como la ejecución de programas especializados directamente de la BIOS e incluso sistemas operativos completos sin necesidad de acceder a un medio de almacenamiento masivo. ([link](#))

Desafío 2

Desafío: Linker

Crear un documento donde respondan a las siguientes preguntas

¿Que es un linker? ¿que hace ?

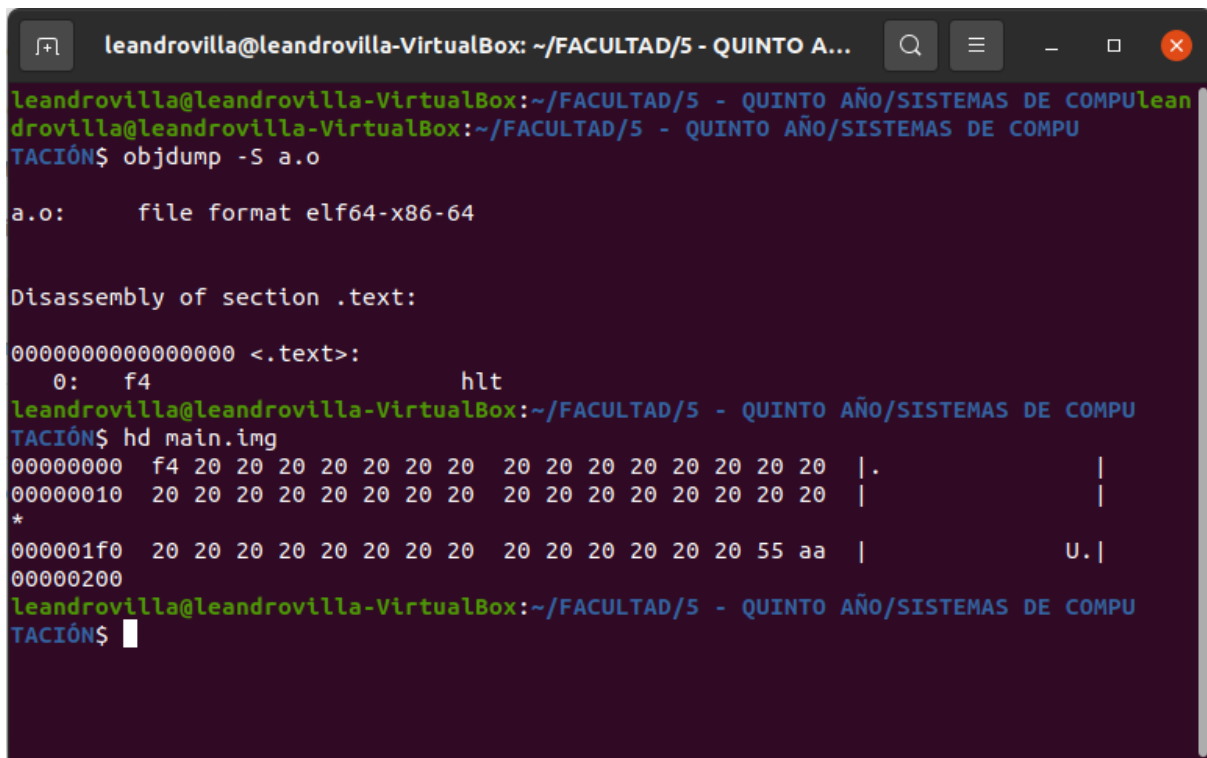
¿Que es la dirección que aparece en el script del linker? ¿Porqué es necesaria ?

Compare la salida de objdump con hd, verifique donde fue colocado el programa dentro de la imagen.

Grabar la imagen en un pendrive y probarla en una pc y subir una foto

¿Para que se utiliza la opción --oformat binary en el linker?

- 1) Un linker o enlazador es un programa encargado de agarrar archivos objeto (.o) que son generados por el compilador o que ya están presentes en una biblioteca, combinarlos y crear un ejecutable.
- 2) La dirección que aparece en el script del linker le dice que todas las instrucciones que va a estar calculando para el ejecutable (código fuente compilado) van a arrancar desde la dirección indicada (en el caso de la BIOS, el código fuente del ejecutable arranca en la dirección 0x7c00).
Esta dirección es necesaria ya que si no se especifica el código fuente va a arrancar en una posición continuada desconocida.
- 3)



```
leandrovilla@leandrovilla-VirtualBox: ~/FACULTAD/5 - QUINTO AÑO/SISTEMAS DE COMPUTACIÓN$ objdump -S a.o

a.o:          file format elf64-x86-64

Disassembly of section .text:

0000000000000000 <.text>:
   0:   f4                hlt
leandrovilla@leandrovilla-VirtualBox:~/FACULTAD/5 - QUINTO AÑO/SISTEMAS DE COMPUTACIÓN$ hd main.img
00000000  f4 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |.          |
00000010  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |          |
*
000001f0  20 20 20 20 20 20 20 20 20 20 20 20 55 aa |          U.|
00000200
leandrovilla@leandrovilla-VirtualBox:~/FACULTAD/5 - QUINTO AÑO/SISTEMAS DE COMPUTACIÓN$
```

4)



5) Cuando compilamos el linker, la opción `-oformat binary` se utiliza para indicarle al kernel que debe linkear un archivo tipo objeto con uno de tipo binario.

Desafío 3

Desafío final: Modo protegido

Crear un código assembler que pueda pasar a modo protegido (sin macros).

¿Cómo sería un programa que tenga dos descriptores de memoria diferentes, uno para cada segmento (código y datos) en espacios de memoria diferenciados?

Cambiar los bits de acceso del segmento de datos para que sea de solo lectura, intentar escribir, ¿Que sucede? ¿Que debería suceder a continuación? (revisar el teórico) Verificarlo con gdb.

En modo protegido, ¿Con qué valor se cargan los registros de segmento ?
¿Porque?

2) Para tener dos descriptores de memoria diferentes deberíamos cambiar las direcciones de base y de límite en el descriptor de cada uno para que no se solapen.

3) Al cambiar el bit de acceso (poner el bit W en 0, bit 41 del descriptor de segmento de datos) para que sea de solo lectura sucede que no puede editar las variables dentro del segmento y no imprime el mensaje que habíamos escrito.



4) Los registros de segmento se cargan con las direcciones de los descriptores de los segmentos que queremos acceder, estos descriptores pueden estar en la tabla global (GDT) o en la tabla local (LDT), dependiendo del bit 2 del registro de segmento (si es 0, se busca el descriptor en la tabla global y si es 1 se lo busca en la local).