Tabla de contenidos

El autor - 7 -
Preparando el espacio de trabajo - 11 -
Para quién es este libro - 13 -
¿Cómo leer este libro? - 15 -
Introducción - 19 -
Saca el máximo partido de leer este libro - 20 -
Convenciones en el libro - 20 -
Detalles sobre las fuentes - 21 -
Repositorio en Github - 22 -
Alcance de este libro - 22 -
Contactar - 22 -
¡Deja una revisión! - 23 -
Capítulo 0. Mi entorno de trabajo - 25 -
Linux - 25 -
Makefile vs build.sh vs .py - 25 -
nasm — Netwide Assembler - 26 -
gdb - 28 -
GEF, pwndbg y PEDA - 33 -
GEF - 33 -
pwndbg - 35 -
PEDA - 36 -
radare2 - 37 -
QEMU - 39 -
Capítulo 1. Lo más básico - 41 -
El inicio - 43 -
El `triple fault` - 44 -

Primer fallo (`first fault`) ..... - 44 -
Segundo fallo (`double fault`) ..... - 44 -
El proceso con `BIOS Legacy` ..... - 45 -
El MBR y el bootloader ..... - 49 -
La memoria en arranque ..... - 52 -
El Modo Real (Real Mode) ..... - 56 -
Bootloader mínimo ..... - 57 -
Invocación simple de QEMU ..... - 67 -
Crear un disco y cargarlo con QEMU ..... - 68 -
Registros de 32 bits desde Modo Real ..... - 70 -
Capítulo 2. Usando interrupciones ..... - 73 -
La interrupción int 10h ..... - 84 -
La interrupción int 11h ..... - 92 -
La interrupción int 12h ..... - 93 -
La interrupción int 13h ..... - 94 -
Lectura con CHS ..... - 97 -
Puertos serie para hacer debug ..... - 111 -
Lectura en modo Extendido ..... - 114 -
La interrupción int 15h ..... - 125 -
Capítulo 3. Bootloaders más complejos ..... - 129 -
Bootloader con dos etapas ..... - 130 -
Unos dibujos básicos con VGA ..... - 140 -
Capítulo 4. El Modo Irreal (Unreal mode) ..... - 147 -
Línea A20 ..... - 148 -
Global Descriptor Table (GDT) ..... - 150 -
Hidden bases ..... - 152 -
Uso de interrupciones ..... - 152 -

A por el Modo Irreal: versión simplificada ..... - 153 -
Para qué sirve esto ..... - 163 -
Previo a cargar kernel de Linux de más de 1 MB ..... - 163 -
Breve introducción al diagnóstico en arranque ..... - 164 -
Paso 1: QEMU con las opciones de gbd ..... - 165 -
Paso 2: abrimos pwndbg, gdb o r2 ..... - 167 -
Paso 3: diagnosticar cosas relevantes ..... - 190 -
Capítulo 5. Cargando el kernel de Linux ..... - 193 -
Primer paso: `stage1` ..... - 194 -
Segundo paso: stage2 leyendo kernel de Linux ..... - 201 -
Segundo paso bis: debug de la escritura en memoria alta ..... - 209 -
Tercer paso: cómo el kernel de Linux toma el control ..... - 214 -
1. Nuestro propio `initrd` ..... - 224 -
2. Pasamos un initrd real ..... - 236 -
Tercer paso bis: saltar a algo ejecutable en el kernel ..... - 238 -
Cuarto paso: Linux Boot Protocol ..... - 240 -
La cabecera del kernel de Modo Real ..... - 241 -
Estructura de memoria en Modo Real ..... - 251 -
Capítulo 6: Kernel de Linux funcionando ..... - 255 -
A) Continuamos con las decisiones tomadas: parchear ..... - 255 -
Paso 1. Buscar firma "`HdrS`" y leer valor de `setup_sects` ..... - 256 -
Paso 2. Leer el valor de `setup_sects` ..... - 257 -
Paso 3. Calcular el valor de `setup_size` ..... - 258 -
Paso 4. Copiar el `setup` a la dirección correcta ..... - 258 -
Paso 5. Copiar el `payload` a la dirección correcta ..... - 259 -
Paso 6. Ubicar la línea de comando (`kernel_cmdline`) ..... - 261 -
Paso 7. Ubicar `initrd` ..... - 262 -

Paso 8. Tipo de bootloader ..... - 262 -
Paso 9. Verificar los campos del `header` ..... - 263 -
Paso 10. Saltar al punto de entrada del kernel ..... - 263 -
Crear el disco de arranque de nuevo ..... - 264 -
Problemas con el diagnóstico ..... - 271 -
Problemas con qemu+gdb y radare2 ..... - 271 -
Monitor con QEMU ..... - 274 -
¿Qué estamos haciendo mal? ..... - 279 -
B) Corregimos y seguimos el estándar ..... - 350 -
Estructura de memoria predecible ..... - 351 -
Cambios en el código ..... - 353 -
Paso 1: verificación de salud ..... - 356 -
Paso 2: cargar initrd en memoria ..... - 357 -
Paso 3: iniciamos boot protocol con `setup_sects` ..... - 358 -
Paso 4: copiamos el `kernel setup` ..... - 359 -
Paso 5: copiamos el `payload` del kernel ..... - 361 -
Paso 6: kernel `cmdline` ..... - 362 -
Paso 7: informamos a `setup` sobre el `initrd` ..... - 364 -
Paso 8: tipo de bootloader ..... - 364 -
Paso 9: kernel `loadflags` y final del `heap` ..... - 365 -
Paso 10: transferimos el control al kernel ..... - 366 -
Capítulo 7. Modo Protegido (Protected Mode) ..... - 371 -
Cambios en la estructura de las etapas ..... - 372 -
El nuevo `stage1` ..... - 372 -
El nuevo `stage2` ..... - 375 -
Pruebas de verificación de Modo Protegido ..... - 377 -
Lectura de disco en 32 bits ..... - 379 -

Cargando el kernel en Modo Protegido ..... - 382 -
Capítulo 8: `stage2` mejorado ..... - 397 -
Nuevo `build.py` ..... - 400 -
Mejoras en el `initrd` ..... - 401 -
Mejoras en el `init.sh` ..... - 404 -
Cambios en `stage2.asm` ..... - 408 -
Capítulo 9: el Modo Largo ..... - 413 -
Qué es el Modo Largo ..... - 413 -
Entrando en Modo Largo ..... - 414 -
Código en C ..... - 421 -
Mini-kernel en C ..... - 422 -
Mini-kernel en Rust ..... - 426 -
Capítulo 10. Un kernel con `userspace` ..... - 433 -
Niveles de privilegio ..... - 433 -
Syscalls ..... - 434 -
Mejorando nuestro kernel ..... - 436 -
Tabla IDT: `idt64.asm` ..... - 437 -
`GDT64` mejorada ..... - 440 -
Kernel con soporte para syscall ..... - 441 -
Capítulo <EOT>: nuevos libros y trabajo futuro ..... - 447 -
Anexo I. Códigos de resultado de la int 13h ..... - 451 -
Anexo II. Identificadores de bootloaders ..... - 453 -
Anexo III. herramientas estáticas ..... - 455 -
Python 3 ..... - 455 -
Frida server ..... - 458 -
Referencias ..... - 459 -