BAJANDO EL NIVEL

Metiendo mano en el kernel Linux.

samuel (4nt1) paschuan

Secure Jump

¿quien soy?

cofounder @ securejump
pentester @ pentest spa, sombreroblanco
director @ CUC 2025
“director” @ RESACA Chile
profesor/ayudante @ umayor

hoy hablaremos de...

compilación.

depuración.

tracing.

una intro antes

intro.

syscalls
interacción user-kernel

kernelspace
donde vive el kernel

userspace
donde vive el usuario

locking
tipo de mutex

mutex
objeto de sincronización

rcu
read, copy, update

irq
interrupt request

sched
planificador de ejecución

hardware rings
seguridad física

VSS
virtual subsystem

MMU
memory management unit

compilación.

¿cómo se compila el kernel?

así.

```
make defconfig
make -j$(nproc)
```

no es suficiente.

(para nuestro use-case)

flags de depuración, tracing y tooling.

lo mínimo sería...

```
make defconfig
# o
make kvm_guest.config
# .config
CONFIG_DEBUG_INFO_DWARF_TOOLCHAIN_DEFAULT=y
CONFIG_KGDB=y
CONFIG_FTRACE=y
CONFIG_RANDOMIZE_BASE=n
make olddefconfig
make -j$(nproc)
```

funcional... si.

pero subóptimo.

entonces...

¡¿cómo?!

```
make defconfig
# o
make kvm_guest.config
# vim .config
CONFIG_DEBUG_INFO_DWARF_TOOLCHAIN_DEFAULT=y
CONFIG_KGDB=y
CONFIG_FTRACE=y
CONFIG_BOOTTIME_TRACING=y
CONFIG_FUNCTION_TRACER=y
CONFIG_STACK_TRACER=y
CONFIG_FTRACE_SYSCALLS=y
CONFIG_SECURITYFS=y
CONFIG_RANDOMIZE_BASE=n
CONFIG_CONFIGFS_FS=y
CONFIG_STRICT_KERNEL_RWX=n
CONFIG_KGDB_SERIAL_CONSOLE=y
CONFIG_GDB_SCRIPTS=y
make olddefconfig
```

este setup es mejor...

sin embargo...

solo tenemos depuración...

y tracing...

esto es por diseño.

el tooling dependerá...

del caso a caso.

¿el repertorio?

el siguiente.

KASAN.
Kernel Address Sanitizer

KFENCE.
Kernel Electronic Fence

KPROBE.
Kernel Probes

KCOV.
Kernel Coverage

KMSAN.
Kernel Memory Sanitizer

UBSAN.
Undefined Behaviour Sanitizer

KMLD.
Kernel Memory Leak Detector

KCSAN.
Kernel Concurrency Sanitizer

KUNIT.
Kernel Unit Testing

KTAP.
Kernel Test Anything Protocol

UAPIC.
UAPI Checker

KGSLA.
Kernel GPIO Sloppy Logic Analyzer

dado el tiempo...

solo veremos una.

KASAN.
Kernel Address Sanitizer

habilitemosla.

```
# vim .config
CONFIG_KASAN=y
CONFIG_KASAN_GENERIC=y
make olddefconfig
make -j$(nproc)
# recommended cmdline: kasan.fault=panic
```

KASAN sirve para dos cosas

detección de UAFs...

y OOBs

lo veremos más adelante.

depuración.

tenemos un kernel.

pero necesitamos usarlo.

pero... ¿cómo?

qemu-system-x86_64

la forma básica:

```
qemu-system-x86_64 \
  -kernel arch/x86_64/boot/bzImage \
  -m 2G \
  -s \
  -S
```

ahora, abramos gdb.

```
gdb -ex 'target remote :1234' vmlinux
continue
```

lo que logramos no es mucho.

```
[    3.155688]  ? kmem_cache_alloc_noprof+0x130/0x300
[    3.155737]  ? __asan_memcpy+0x3c/0x60
[    3.155776]  ? getname_kernel+0x148/0x350
[    3.155822]  mount_root+0x29d/0x2f0
[    3.155862]  ? __kasan_slab_alloc+0x63/0x80
[    3.155904]  ? __pfx_mount_root+0x10/0x10
[    3.155948]  ? __pfx_init_mknod+0x10/0x10
[    3.155994]  ? __asan_memcpy+0x3c/0x60
[    3.156042]  ? getname_kernel+0x148/0x350
[    3.156085]  prepare_namespace+0x216/0x280
[    3.156133]  ? __pfx_prepare_namespace+0x10/0x10
[    3.156180]  kernel_init_freeable+0x4b5/0x4d0
[    3.156227]  ? __pfx_kernel_init+0x10/0x10
[    3.156275]  kernel_init+0x19/0x150
[    3.156313]  ? __pfx_kernel_init+0x10/0x10
[    3.156353]  ? __pfx_kernel_init+0x10/0x10
[    3.156395]  ret_from_fork+0x10d/0x1b0
[    3.156436]  ? __pfx_kernel_init+0x10/0x10
[    3.156478]  ret_from_fork_asm+0x1a/0x30
[    3.156526]  </TASK>
[    3.156968] Kernel Offset: 0x33800000 from 0xffffffff81000000 (relocation ran
ge: 0xffffffff80000000-0xffffffffbfffffff)
[    3.157414] ---[ end Kernel panic - not syncing: VFS: Unable to mount root fs
 on unknown-block(0,0) ]---
```

tenemos varias opciones.

generar nuestra propia CPIO...

o una de syzkaller.

veamos cómo generar una CPIO.

```
# NECESITAMOS!!! un binario ELF o script llamado "init".
# en mi caso, compilé Bash estáticamente para esta tarea.
wget https://ftp.gnu.org/gnu/bash/bash-5.3.tar.gz
tar xfv bash-5.3.tar.gz
cd bash5.3
./configure --enable-static-link
# para la build estática, se requieren librerías GCC estáticas.
# busca en Google cuales son las que te sirven en tu distro.
make clean
make -j$(nproc)
mv bash init
mkdir cpio
mv init cpio
cd cpio
ls | cpio -H newc -o > initram.cpio
cd linux/source/code/path
qemu-system-x86_64 \
  -m 2G \
  -kernel arch/x86_64/boot/bzImage \
  -initrd /path/to/initram.cpio \
  -s \
  -S
```

```
[    3.539696] Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[    3.541237] Loaded X.509 cert 'wens: 61c038651aabdcf94bd0ac7ff06c7248db18c600
'
[    3.543285] ALSA device list:
[    3.543393]   No soundcards found.
[    3.547553] faux_driver regulatory: Direct firmware load for regulatory.db fa
iled with error -2
[    3.548103] cfg80211: failed to load regulatory.db
[    3.612468] Freeing unused kernel image (initmem) memory: 5904K
[    3.615687] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i8042
/serio1/input/input3
[    3.618351] Write protecting the kernel read-only data: 67584k
[    3.620377] Freeing unused kernel image (text/rodata gap) memory: 644K
[    3.621154] Freeing unused kernel image (rodata/data gap) memory: 600K
[    3.740652] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[    3.741131] Run /init as init process
init: cannot set terminal process group (-1): Inappropriate ioctl for device
init: no job control in this shell
init-5.3# echo hola desde el kerne[    8.870364] random: crng init done

hola desde el kerne
init-5.3# echo hola desde el kernel
init-5.3# echo hola desde el kernel
hola desde el kernel
init-5.3#
```

mientras esté compilado estáticamente...

todo puede ser un init.

flags útiles de qemu.

-nographic
todo a través de la terminal

-append
añade parámetros a la cmdline

`-enable-kvm`
usa KVM en vez de TCG.

-pidfile file.pid
almacena el proceso en un PIDfile

-monitor
envio del monitor hacia un dev

-net
configura un dispositivo de red

(s)breaks have no(t much)
power here

muchos usamos el querido break

pero...

sin MMU, vmem...

no tiene poder alguno.

```
[    3.651404] cfg80211: Loading compiled-in X.509 certificates for regulatory d
atabase
[    3.662009] kworker/u4:1 (50) used greatest stack depth: 29360 bytes left
[    3.665892] kworker/u4:0 (51) used greatest stack depth: 29024 bytes left
[    3.676660] Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[    3.678398] Loaded X.509 cert 'wens: 61c038651aabdcf94bd0ac7ff06c7248db18c600
'
[    3.681049] ALSA device list:
[    3.681273]   No soundcards found.
[    3.683837] faux_driver regulatory: Direct firmware load for regulatory.db fa
iled with error -2
[    3.684167] cfg80211: failed to load regulatory.db
[    3.749524] Freeing unused kernel image (initmem) memory: 5904K
[    3.752565] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i8042
/serio1/input/input3
[    3.754902] Write protecting the kernel read-only data: 67584k
[    3.756899] Freeing unused kernel image (text/rodata gap) memory: 644K
[    3.757633] Freeing unused kernel image (rodata/data gap) memory: 600K
[    3.894227] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[    3.894677] Run /init as init process
init: cannot set terminal process group (-1): Inappropriate ioctl for device
init: no job control in this shell
init-5.3# echo y mi break?
y mi break?
init-5.3# _
```

```
(gdb) break kernel_init
Breakpoint 1 at 0xffffffff8453a000: file init/main.c, lin
e 1473.
(gdb) c
Continuing.
```

su funcionamiento es irregular

a veces funciona...

```
SeaBIOS (version 1.17.0-5.fc42)


iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+7EFCAE60+7EF0AE60 CA00



Booting from ROM...
No EFI environment detected.
early console in extract_kernel
input_data: 0x0000000004e6b2c4
input_len: 0x0000000001c25a9f
output: 0x0000000001000000
output_len: 0x0000000005a15c90
kernel_total_size: 0x0000000005a28000
needed_size: 0x0000000005c00000
trampoline_32bit: 0x0000000000000000

Decompressing Linux... Parsing ELF... done.
Booting the kernel (entry_offset: 0x000000000355e2f0).
```

```
(gdb) b start_kernel
Breakpoint 1 at 0xffffffff86244430: file init/main.c, line 910.
(gdb) c
Continuing.

Breakpoint 1, start_kernel () at init/main.c:910
910     {
(gdb)
```

a veces no.

```
[ anti@mothership ]
    01:33 PM
~/Documents/Research/linux-debugging/linux-6.17.7 $ cat ../qemu.sh | bash
```

```
(gdb) break start_kernel
Breakpoint 1 at 0xffffffff86244430: file init/main.c, line 910.
(gdb) c
Continuing.
Warning:
Cannot insert breakpoint 1.
Cannot access memory at address 0xffffffff86244430

Command aborted.
(gdb)
```

y cuando no...

usamos (h)breaks.

hardware assisted breaks.

```
No EFI environment detected.
early console in extract_kernel
input_data: 0x0000000004e6b2c4
input_len: 0x0000000001c25a9f
output: 0x0000000001000000
output_len: 0x0000000005a15c90
kernel_total_size: 0x0000000005a28000
needed_size: 0x0000000005c00000
trampoline_32bit: 0x0000000000000000

Decompressing Linux... Parsing ELF... done.
Booting the kernel (entry_offset: 0x000000000355e2f0).
```

```
(gdb) hb start_kernel
Note: breakpoint 1 (disabled) also set at pc 0xffffffff86244430.
Hardware assisted breakpoint 2 at 0xffffffff86244430: file init/main.c, line 910.
(gdb) c
Continuing.

Breakpoint 2, start_kernel () at init/main.c:910
910     {
(gdb)
```

anatomía de un driver.

los drivers de Linux...

a alto nivel...

son sencillos.

veaamos el esqueleto de uno.

```c
#include<linux/module.h>
#include<linux/init.h>

int kmod_init(void)
{
        printk("kmod-hello - module loaded!\n");
        return 0;
}

void kmod_exit(void)
{
        printk("kmod-hello - module unloaded!\n");
}

module_init(kmod_init);
module_exit(kmod_exit);

MODULE_LICENSE("GPL");
```

si bien solo el esqueleto...

esta estructura es global

función module_init con un valor int de retorno

función module_exit sin valor de retorno

lo que si es necesario...

es el MODULE_LICENSE.

esto permite impedir la carga de modulos propietarios.

si... impedir.

major-minors.

los major-minor values...

son importantes.

cada driver tiene un major.

cada dispositivo un minor.

generalmente asignados dinámicamente.

algunos estáticamente.

```
~ $ ls -l /dev/tty*
crw-rw-rw-. 1 root tty     5,  0 Nov 17 16:31 /dev/tty
crw--w----. 1 root tty     4,  0 Nov 14 20:25 /dev/tty0
crw--w----. 1 root tty     4,  1 Nov 14 20:25 /dev/tty1
crw--w----. 1 root tty     4, 10 Nov 14 20:25 /dev/tty10
crw--w----. 1 root tty     4, 11 Nov 14 20:25 /dev/tty11
[SNIP]
```

estructuras driverescas.

file_operations
from: include/linux/fs.h

```
2156  struct file_operations {
2157          struct module *owner;
2158          fop_flags_t fop_flags;
2159          loff_t (*llseek) (struct file *, loff_t, int);
2160          ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
2161          ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
2162          ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
2163          ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
2164          int (*iopoll)(struct kiocb *kiocb, struct io_comp_batch *,
2165                        unsigned int flags);
2166          int (*iterate_shared) (struct file *, struct dir_context *);
2167          __poll_t (*poll) (struct file *, struct poll_table_struct *);
2168          long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
2169          long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
2170          int (*mmap) (struct file *, struct vm_area_struct *);
2171          int (*open) (struct inode *, struct file *);
2172          int (*flush) (struct file *, fl_owner_t id);
2173          int (*release) (struct inode *, struct file *);
2174          int (*fsync) (struct file *, loff_t, loff_t, int datasync);
2175          int (*fasync) (int, struct file *, int);
2176          int (*lock) (struct file *, int, struct file_lock *);
2177          unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned long);
2178          int (*check_flags)(int);
2179          int (*flock) (struct file *, int, struct file_lock *);
2180          ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t *, size_t, unsigned int);
2181          ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned int);
2182          void (*splice_eof)(struct file *file);
2183          int (*setlease)(struct file *, int, struct file_lease **, void **);
2184          long (*fallocate)(struct file *file, int mode, loff_t offset,
2185                            loff_t len);
2186          void (*show_fdinfo)(struct seq_file *m, struct file *f);
2187  #ifndef CONFIG_MMU
2188          unsigned (*mmap_capabilities)(struct file *);
2189  #endif
2190          ssize_t (*copy_file_range)(struct file *, loff_t, struct file *,
2191                                     loff_t, size_t, unsigned int);
2192          loff_t (*remap_file_range)(struct file *file_in, loff_t pos_in,
2193                                     struct file *file_out, loff_t pos_out,
2194                                     loff_t len, unsigned int remap_flags);
2195          int (*fadvise)(struct file *, loff_t, loff_t, int);
2196          int (*uring_cmd)(struct io_uring_cmd *ioucmd, unsigned int issue_flags);
2197          int (*uring_cmd_iopoll)(struct io_uring_cmd *, struct io_comp_batch *,
2198                                  unsigned int poll_flags);
2199          int (*mmap_prepare)(struct vm_area_desc *);
2200  } __randomize_layout;
```

```
static struct file_operations my_null_fops = {
        .owner = THIS_MODULE,
        .read = my_null_read,
        .write = my_null_write,
        .open = my_null_open,
        .release = my_null_release,
};
```

file
from: include/linux/fs.h

```
1100  struct file {
1101          spinlock_t                      f_lock;
1102          fmode_t                         f_mode;
1103          const struct file_operations    *f_op;
1104          struct address_space            *f_mapping;
1105          void                            *private_data;
1106          struct inode                    *f_inode;
1107          unsigned int                    f_flags;
1108          unsigned int                    f_iocb_flags;
1109          const struct cred               *f_cred;
1110          struct fown_struct              *f_owner;
1111          /* --- cacheline 1 boundary (64 bytes) --- */
1112          struct path                     f_path;
1113          union {
1114                  /* regular files (with FMODE_ATOMIC_POS) and directories */
1115                  struct mutex            f_pos_lock;
1116                  /* pipes */
1117                  u64                     f_pipe;
1118          };
1119          loff_t                          f_pos;
1120  #ifdef CONFIG_SECURITY
1121          void                            *f_security;
1122  #endif
1123          /* --- cacheline 2 boundary (128 bytes) --- */
1124          errseq_t                        f_wb_err;
1125          errseq_t                        f_sb_err;
1126  #ifdef CONFIG_EPOLL
1127          struct hlist_head               *f_ep;
1128  #endif
1129          union {
1130                  struct callback_head    f_task_work;
1131                  struct llist_node       f_llist;
1132                  struct file_ra_state    f_ra;
1133                  freeptr_t               f_freeptr;
1134          };
1135          file_ref_t                      f_ref;
1136          /* --- cacheline 3 boundary (192 bytes) --- */
1137  } __randomize_layout
1138    __attribute__((aligned(4)));  /* lest something weird decides that 2 is OK */
```

inode
from: include/linux/fs.h

```
struct inode {
        [SNIP]
        dev_t i_rdev;
        struct cdev *i_cdev;
        [SNIP]
}
```

para entender esto...

debemos entender...

la VFS! Virtual File System

la cual no veremos hoy.

¡pero!

pregunta de investigación

cada fs tiene su propio driver

EXT4, BTRFS, etc.

sus propias funciones.

ext4_file_open,
btrfs_file_open, etc.

pero nosotros no usamos esas funciones.

¿cómo?

¿por qué?

¿y qué tiene que ver la VFS?

depurando un use-after-free.

veamos el siguiente código:

```
#include<linux/module.h>
#include<linux/init.h>
#include<linux/slab.h>

int use_after_free(void)
{
        printk("kmod-uaf - module loading!\n");
        char *leak = kzalloc(100, GFP_KERNEL);
        if (!leak)
        {
            printk(KERN_ERR "Memory allocation failed\n");
            goto out;
        }
        strcpy(leak, "This is a use-after-free example");
        kfree(leak);
        printk(KERN_INFO "Using freed memory: %s\n", leak);
out:
        return 0;
}

void kmod_exit(void)
{
        printk("kmod-uaf - module unloaded!\n");
}

module_init(use_after_free);
module_exit(kmod_exit);

MODULE_LICENSE("GPL");
```

```c
#include<linux/module.h>
#include<linux/init.h>
#include<linux/slab.h>

int use_after_free(void)
{
        printk("kmod-uaf - module loading!\n");
        char *leak = kzalloc(100, GFP_KERNEL);
        if (!leak)
        {
            printk(KERN_ERR "Memory allocation failed\n");
            goto out;
        }
        strcpy(leak, "This is a use-after-free example");
        kfree(leak);
        printk(KERN_INFO "Using freed memory: %s\n", leak);
out:
        return 0;
}

void kmod_exit(void)
{
        printk("kmod-uaf - module unloaded!\n");
}

module_init(use_after_free);
module_exit(kmod_exit);

MODULE_LICENSE("GPL");
```

el uaf, en este caso, es aparente.

pero no siempre es así.

desarrollando...

¿cómo lo encontramos?

con **KASAN.**

y con la cmdline
kasan.fault=panic.

```
[    0.768208] scsi 1:0:0:0: CD-ROM            QEMU     QEMU DVD-ROM     2.5+ PQ: 0 ANSI: 5
[    0.769487] sd 0:0:0:0: [sda] Preferred minimum I/O size 512 bytes
[    0.781207] sd 0:0:0:0: [sda] Attached SCSI disk
[    0.795243] sr 1:0:0:0: [sr0] scsi3-mmc drive: 4x/4x cd/rw xa/form2 tray
[    0.796341] cdrom: Uniform CD-ROM driver Revision: 3.20
[    0.814677] sr 1:0:0:0: Attached scsi generic sg1 type 5
[    0.946200] e1000 0000:00:03.0 eth0: (PCI:33MHz:32-bit) 52:54:00:12:34:56
[    0.947302] e1000 0000:00:03.0 eth0: Intel(R) PRO/1000 Network Connection
[    0.948390] e1000e: Intel(R) PRO/1000 Network Driver
[    0.949128] e1000e: Copyright(c) 1999 - 2015 Intel Corporation.
[    0.950056] sky2: driver version 1.30
[    0.950981] usbcore: registered new interface driver usblp
[    0.951846] usbcore: registered new interface driver usb-storage
[    0.952800] i8042: PNP: PS/2 Controller [PNP0303:KBD,PNP0f13:MOU] at 0x60,0x64 irq 1,12
[    0.954892] serio: i8042 KBD port at 0x60,0x64 irq 1
[    0.955658] serio: i8042 AUX port at 0x60,0x64 irq 12
[    0.957136] input: AT Translated Set 2 keyboard as /devices/platform/i8042/serio0/input/input1
[    0.958568] rtc_cmos 00:05: RTC can wake from S4
[    0.960040] rtc_cmos 00:05: registered as rtc0
[    0.961605] rtc_cmos 00:05: alarms up to one day, y3k, 242 bytes nvram, hpet irqs
[    0.964845] device-mapper: ioctl: 4.50.0-ioctl (2025-04-28) initialised: dm-devel@lists.linux.dev
[    0.966552] amd_pstate: the _CPC object is not present in SBIOS or ACPI disabled
[    0.967789] hid: raw HID events driver (C) Jiri Kosina
[    0.968993] usbcore: registered new interface driver usbhid
[    0.969824] usbhid: USB HID core driver
```

```
Breakpoint 1, use_after_free () at drivers/uaf/uaf.c:7
7       {
(gdb) l
2       #include <linux/init.h>
3       #include <linux/slab.h>
4       #include <linux/string.h>
5
6       static int __init use_after_free(void)
7       {
8               char *leak = kzalloc(100, GFP_KERNEL);
9               if (!leak)
10              {
11                  printk(KERN_ERR "Memory allocation failed\n");
(gdb)
12                  goto out;
13              }
14              strcpy(leak, "This is a use-after-free example");
15              kfree(leak);
16              // Using the pointer after freeing it, which is undefined behavior
17              printk(KERN_INFO "Using freed memory: %s\n", leak); // This is dangerous!
18      out:
19              return 0;
20      }
21
(gdb) a
```

```
[    0.768208] scsi 1:0:0:0: CD-ROM            QEMU     QEMU DVD-ROM     2.5+ PQ: 0 ANSI: 5
[    0.769487] sd 0:0:0:0: [sda] Preferred minimum I/O size 512 bytes
[    0.781207] sd 0:0:0:0: [sda] Attached SCSI disk
[    0.795243] sr 1:0:0:0: [sr0] scsi3-mmc drive: 4x/4x cd/rw xa/form2 tray
[    0.796341] cdrom: Uniform CD-ROM driver Revision: 3.20
[    0.814677] sr 1:0:0:0: Attached scsi generic sg1 type 5
[    0.946200] e1000 0000:00:03.0 eth0: (PCI:33MHz:32-bit) 52:54:00:12:34:56
[    0.947302] e1000 0000:00:03.0 eth0: Intel(R) PRO/1000 Network Connection
[    0.948390] e1000e: Intel(R) PRO/1000 Network Driver
[    0.949128] e1000e: Copyright(c) 1999 - 2015 Intel Corporation.
[    0.950056] sky2: driver version 1.30
[    0.950981] usbcore: registered new interface driver usblp
[    0.951846] usbcore: registered new interface driver usb-storage
[    0.952800] i8042: PNP: PS/2 Controller [PNP0303:KBD,PNP0f13:MOU] at 0x60,0x64 irq 1,12
[    0.954892] serio: i8042 KBD port at 0x60,0x64 irq 1
[    0.955658] serio: i8042 AUX port at 0x60,0x64 irq 12
[    0.957136] input: AT Translated Set 2 keyboard as /devices/platform/i8042/serio0/input/input1
[    0.958568] rtc_cmos 00:05: RTC can wake from S4
[    0.960040] rtc_cmos 00:05: registered as rtc0
[    0.961605] rtc_cmos 00:05: alarms up to one day, y3k, 242 bytes nvram, hpet irqs
[    0.964845] device-mapper: ioctl: 4.50.0-ioctl (2025-04-28) initialised: dm-devel@lists.linux.dev
[    0.966552] amd_pstate: the _CPC object is not present in SBIOS or ACPI disabled
[    0.967789] hid: raw HID events driver (C) Jiri Kosina
[    0.968993] usbcore: registered new interface driver usbhid
[    0.969824] usbhid: USB HID core driver
```

```
8               char *leak = kzalloc(100, GFP_KERNEL);
9               if (!leak)
10              {
11                  printk(KERN_ERR "Memory allocation failed\n");
(gdb)
12                  goto out;
13              }
14              strcpy(leak, "This is a use-after-free example");
15              kfree(leak);
16              // Using the pointer after freeing it, which is undefined behavior
17              printk(KERN_INFO "Using freed memory: %s\n", leak); // This is dangerous!
18      out:
19              return 0;
20      }
21
(gdb) n
8               char *leak = kzalloc(100, GFP_KERNEL);
(gdb)
9               if (!leak)
(gdb)
14              strcpy(leak, "This is a use-after-free example");
(gdb)
15              kfree(leak);
(gdb)
17              printk(KERN_INFO "Using freed memory: %s\n", leak); // This is dangerous!
(gdb)
```

```
[    0.969824] usbhid: USB HID core driver
[    0.972451] ================================================================
[    0.972909] BUG: KASAN: slab-use-after-free in string+0x476/0x570
[    0.972909] Read of size 1 at addr ffff8880080b1300 by task swapper/0/1
[    0.972909]
[    0.972909] CPU: 0 UID: 0 PID: 1 Comm: swapper/0 Not tainted 6.17.7 #36 PREEMPT(voluntary)
[    0.972909] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.17.0-5.fc42 04/01/2014
[    0.972909] Call Trace:
[    0.972909]  <TASK>
[    0.972909]  dump_stack_lvl+0x4d/0x70
[    0.972909]  print_report+0x170/0x4f3
[    0.972909]  ? __pfx__raw_spin_lock_irqsave+0x10/0x10
[    0.972909]  kasan_report+0xda/0x110
[    0.972909]  ? string+0x476/0x570
[    0.972909]  ? string+0x476/0x570
[    0.972909]  string+0x476/0x570
[    0.972909]  ? unwind_next_frame+0x53a/0x23f0
[    0.972909]  ? arch_stack_walk+0x74/0x100
[    0.972909]  ? __pfx_string+0x10/0x10
[    0.972909]  ? __pfx_format_decode+0x10/0x10
[    0.972909]  ? unwind_next_frame+0x53a/0x23f0
[    0.972909]  vsnprintf+0x2f0/0x13f0
[    0.972909]  ? write_profile+0x90/0xf0
[    0.972909]  ? __pfx_vsnprintf+0x10/0x10
[    0.972909]  ? _raw_spin_trylock+0xaf/0x120
[    0.972909]  ? __pfx__raw_spin_trylock+0x10/0x10
```

```
10              {
11                  printk(KERN_ERR "Memory allocation failed\n");
(gdb)
12                  goto out;
13              }
14              strcpy(leak, "This is a use-after-free example");
15              kfree(leak);
16              // Using the pointer after freeing it, which is undefined behavior
17              printk(KERN_INFO "Using freed memory: %s\n", leak); // This is dangerous!
18      out:
19              return 0;
20      }
21
(gdb) n
8               char *leak = kzalloc(100, GFP_KERNEL);
(gdb)
9               if (!leak)
(gdb)
14              strcpy(leak, "This is a use-after-free example");
(gdb)
15              kfree(leak);
(gdb)
17              printk(KERN_INFO "Using freed memory: %s\n", leak); // This is dangerous!
(gdb) n
19              return 0;
(gdb)
```

```
[    0.972909]  ? __pfx__raw_spin_trylock+0x10/0x10
[    0.972909]  vprintk_store+0x2c8/0x880
[    0.972909]  ? stack_depot_save_flags+0x4b5/0x8f0
[    0.972909]  ? __pfx_vprintk_store+0x10/0x10
[    0.972909]  ? kasan_save_stack+0x3f/0x50
[    0.972909]  ? __kasan_slab_free+0x63/0x70
[    0.972909]  ? kfree+0xf2/0x3b0
[    0.972909]  ? use_after_free+0x73/0x90
[    0.972909]  ? do_one_initcall+0x9d/0x310
[    0.972909]  ? kernel_init_freeable+0x458/0x4d0
[    0.972909]  vprintk_emit+0x10b/0x440
[    0.972909]  ? __pfx_vprintk_emit+0x10/0x10
[    0.972909]  ? __pfx_use_after_free+0x10/0x10
[    0.972909]  _printk+0xc2/0x100
[    0.972909]  ? __pfx__printk+0x10/0x10
[    0.972909]  ? ret_from_fork_asm+0x1a/0x30
[    0.972909]  ? ret_from_fork+0x10d/0x1b0
[    0.972909]  use_after_free+0x82/0x90
[    0.972909]  do_one_initcall+0x9d/0x310
[    0.972909]  ? __pfx_do_one_initcall+0x10/0x10
[    0.972909]  ? __pfx_parse_args+0x10/0x10
[    0.972909]  ? __pfx_kasan_poison+0x10/0x10
[    0.972909]  kernel_init_freeable+0x458/0x4d0
[    0.972909]  ? __pfx_kernel_init+0x10/0x10
[    0.972909]  kernel_init+0x19/0x150
[    0.972909]  ? __pfx_kernel_init+0x10/0x10
```

```
[    0.979210]  ? kfree+0xf2/0x3b0
[    0.979210]  ? use_after_free+0x73/0x90
[    0.979210]  ? do_one_initcall+0x9d/0x310
[    0.979210]  ? kernel_init_freeable+0x458/0x4d0
[    0.979210]  vprintk_emit+0x10b/0x440
[    0.979210]  ? __pfx_vprintk_emit+0x10/0x10
[    0.979210]  ? __pfx_use_after_free+0x10/0x10
[    0.979210]  _printk+0xc2/0x100
[    0.979210]  ? __pfx__printk+0x10/0x10
[    0.979210]  ? ret_from_fork_asm+0x1a/0x30
[    0.979210]  ? ret_from_fork+0x10d/0x1b0
[    0.979210]  use_after_free+0x82/0x90
[    0.979210]  do_one_initcall+0x9d/0x310
[    0.979210]  ? __pfx_do_one_initcall+0x10/0x10
[    0.979210]  kernel_init_freeable+0x458/0x4d0
[    0.979210]  ? __pfx_kernel_init+0x10/0x10
[    0.979210]  kernel_init+0x19/0x150
[    0.979210]  ? __pfx_kernel_init+0x10/0x10
[    0.979210]  ? __pfx_kernel_init+0x10/0x10
[    0.979210]  ret_from_fork+0x10d/0x1b0
[    0.979210]  ? __pfx_kernel_init+0x10/0x10
[    0.979210]  ret_from_fork_asm+0x1a/0x30
[    0.979210]  </TASK>
[    0.979210] Kernel Offset: disabled
[    0.979210] ---[ end Kernel panic - not syncing: kasan.fault=panic set ... ]---
```

```
#0  delay_tsc (cycles=3892716) at arch/x86/lib/delay.c:79
#1  0xffffffff8120ce7b in udelay (usec=1000) at ./include/asm-generic/delay.h:62
#2  vpanic (fmt=fmt@entry=0xffffffff84fd1b67 "kasan.fault=panic set ...\n", args=args@entry=0xffff888006e2f798) at kernel/panic.c:555
#3  0xffffffff8120cf69 in panic (fmt=fmt@entry=0xffffffff84fd1b67 "kasan.fault=panic set ...\n") at kernel/panic.c:566
#4  0xffffffff8122a348 in end_report (addr=<optimized out>, flags=0xffff888006e2f868, is_write=false) at mm/kasan/report.c:233
#5  end_report (flags=0xffff888006e2f868, addr=<optimized out>, is_write=<optimized out>) at mm/kasan/report.c:219
#6  0xffffffff81abbd9b in kasan_report (addr=addr@entry=0xffff8880080b0300, size=size@entry=1, is_write=is_write@entry=false, ip=18446744071634551814) at mm/kasan/report.c:597
#7  0xffffffff81abd824 in __asan_report_load1_noabort (addr=addr@entry=0xffff8880080b0300) at mm/kasan/report_generic.c:378
#8  0xffffffff84520406 in string_nocheck (buf=0xffff888006e2fc0e "", end=<optimized out>, s=0xffff8880080b0301 "", spec=...) at lib/vsprintf.c:653
#9  string (buf=<optimized out>, buf@entry=0xffff888006e2fc0e "", end=end@entry=0xffff888006e2fc00 "", s=<optimized out>, spec=...) at lib/vsprintf.c:735
#10 0xffffffff8452a6e0 in vsnprintf (buf=buf@entry=0xffff888006e2fbf8 "\0016Using ", size=<optimized out>, size@entry=8, fmt_str=fmt_str@entry=0xffffffff84b058a0 "\0016Using freed memory: %s\n",
    args=args@entry=0xffff888006e2fbb8) at lib/vsprintf.c:2926
#11 0xffffffff815f2188 in vprintk_store (facility=facility@entry=0, level=level@entry=-1, dev_info=dev_info@entry=0x0, fmt=fmt@entry=0xffffffff84b058a0 "\0016Using freed memory: %s\n",
    args=args@entry=0xffff888006e2fd38) at kernel/printk/printk.c:2279
#12 0xffffffff815f502b in vprintk_emit (facility=facility@entry=0, level=level@entry=-1, dev_info=dev_info@entry=0x0, fmt=fmt@entry=0xffffffff84b058a0 "\0016Using freed memory: %s\n",
    args=args@entry=0xffff888006e2fd38) at kernel/printk/printk.c:2426
#13 0xffffffff815f5388 in vprintk_default (fmt=fmt@entry=0xffffffff84b058a0 "\0016Using freed memory: %s\n", args=args@entry=0xffff888006e2fd38) at kernel/printk/printk.c:2465
#14 0xffffffff815f6c49 in vprintk (fmt=fmt@entry=0xffffffff84b058a0 "\0016Using freed memory: %s\n", args=args@entry=0xffff888006e2fd38) at kernel/printk/printk_safe.c:82
#15 0xffffffff812174b2 in _printk (fmt=fmt@entry=0xffffffff84b058a0 "\0016Using freed memory: %s\n") at kernel/printk/printk.c:2475
#16 0xffffffff8630be12 in use_after_free () at drivers/uaf/uaf.c:17
#17 0xffffffff8131887d in do_one_initcall (fn=0xffffffff8630bd90 <use_after_free>) at init/main.c:1281
#18 0xffffffff8624dc98 in do_initcall_level (level=6, command_line=0xffff888006da8200 "console") at init/main.c:1343
#19 do_initcalls () at init/main.c:1359
#20 do_basic_setup () at init/main.c:1378
#21 kernel_init_freeable () at init/main.c:1591
--Type <RET> for more, q to quit, c to continue without paging--
```

pero, usemos KASAN...

el reporte de KASAN tiene:

call trace (bt),

alocaciones de memoria,

liberaciones de memoria.

en nuestro caso...

alocamos en
use_after_free+0x41/0x90
(slab.h)

liberamos en
use_after_free+0x73/0x90

y la bt es:
use_after_free+0x82/0x90

usemos gdb para encontrar...

el código ofensor.

la bt:

```
[    0.979210]  ? __pfx_string+0x10/0x10
[    0.979210]  ? __pfx_format_decode+0x10/0x10
[    0.979210]  ? unwind_next_frame+0x53a/0x23f0
[    0.979210]  vsnprintf+0x2f0/0x13f0
[    0.979210]  ? write_profile+0x90/0xf0
[    0.979210]  ? __pfx_vsnprintf+0x10/0x10
[    0.979210]  ? _raw_spin_trylock+0xaf/0x120
[    0.979210]  ? __pfx__raw_spin_trylock+0x10/0x10
[    0.979210]  vprintk_store+0x2c8/0x880
[    0.979210]  ? stack_depot_save_flags+0x4b5/0x8f0
[    0.979210]  ? __pfx_vprintk_store+0x10/0x10
[    0.979210]  ? kasan_save_stack+0x3f/0x50
[    0.979210]  ? __kasan_slab_free+0x63/0x70
[    0.979210]  ? kfree+0xf2/0x3b0
[    0.979210]  ? use_after_free+0x73/0x90
[    0.979210]  ? do_one_initcall+0x9d/0x310
[    0.979210]  ? kernel_init_freeable+0x458/0x4d0
[    0.979210]  vprintk_emit+0x10b/0x440
[    0.979210]  ? __pfx_vprintk_emit+0x10/0x10
[    0.979210]  ? __pfx_use_after_free+0x10/0x10
[    0.979210]  _printk+0xc2/0x100
[    0.979210]  ? __pfx__printk+0x10/0x10
[    0.979210]  ? ret_from_fork_asm+0x1a/0x30
[    0.979210]  ? ret_from_fork+0x10d/0x1b0
[    0.979210]  use_after_free+0x82/0x90
[    0.979210]  do_one_initcall+0x9d/0x310
(gdb) list *(use_after_free+0x82/0x90)
0xffffffff8630bd90 is in use_after_free (drivers/uaf/uaf.c:7).
2       #include <linux/init.h>
3       #include <linux/slab.h>
4       #include <linux/string.h>
5
6       static int __init use_after_free(void)
7       {
8               char *leak = kzalloc(100, GFP_KERNEL);
9               if (!leak)
10              {
11                  printk(KERN_ERR "Memory allocation failed\n");
(gdb)
```

alocación:

```
[    0.979210]  ? __pfx_kernel_init+0x10/0x10
[    0.979210]  ? __pfx_kernel_init+0x10/0x10
[    0.979210]  ret_from_fork+0x10d/0x1b0
[    0.979210]  ? __pfx_kernel_init+0x10/0x10
[    0.979210]  ret_from_fork_asm+0x1a/0x30
[    0.979210]  </TASK>
[    0.979210]
[    0.979210] Allocated by task 1:
[    0.979210]  kasan_save_stack+0x30/0x50
[    0.979210]  kasan_save_track+0x14/0x30
[    0.979210]  __kasan_kmalloc+0x9a/0xb0
[    0.979210]  use_after_free+0x41/0x90
[    0.979210]  do_one_initcall+0x9d/0x310
[    0.979210]  kernel_init_freeable+0x458/0x4d0
[    0.979210]  kernel_init+0x19/0x150
[    0.979210]  ret_from_fork+0x10d/0x1b0
[    0.979210]  ret_from_fork_asm+0x1a/0x30
[    0.979210]
[    0.979210] Freed by task 1:
[    0.979210]  kasan_save_stack+0x30/0x50
[    0.979210]  kasan_save_track+0x14/0x30
[    0.979210]  kasan_save_free_info+0x3b/0x70
[    0.979210]  __kasan_slab_free+0x63/0x70
[    0.979210]  kfree+0xf2/0x3b0
[    0.979210]  use_after_free+0x73/0x90
[    0.979210]  do_one_initcall+0x9d/0x310
(gdb) list *(use_after_free+0x90)
(gdb) list *(use_after_free+0x41/0x90)
0xffffffff8630bd90 is in use_after_free (drivers/uaf/uaf.c:7).
2       #include <linux/init.h>
3       #include <linux/slab.h>
4       #include <linux/string.h>
5
6       static int __init use_after_free(void)
7       {
8               char *leak = kzalloc(100, GFP_KERNEL);
9               if (!leak)
10              {
11                  printk(KERN_ERR "Memory allocation failed\n");
(gdb)
```

liberación:

```
[    0.979210]  </TASK>
[    0.979210]
[    0.979210] Allocated by task 1:
[    0.979210]  kasan_save_stack+0x30/0x50
[    0.979210]  kasan_save_track+0x14/0x30
[    0.979210]  __kasan_kmalloc+0x9a/0xb0
[    0.979210]  use_after_free+0x41/0x90
[    0.979210]  do_one_initcall+0x9d/0x310
[    0.979210]  kernel_init_freeable+0x458/0x4d0
[    0.979210]  kernel_init+0x19/0x150
[    0.979210]  ret_from_fork+0x10d/0x1b0
[    0.979210]  ret_from_fork_asm+0x1a/0x30
[    0.979210]
[    0.979210] Freed by task 1:
[    0.979210]  kasan_save_stack+0x30/0x50
[    0.979210]  kasan_save_track+0x14/0x30
[    0.979210]  kasan_save_free_info+0x3b/0x70
[    0.979210]  __kasan_slab_free+0x63/0x70
[    0.979210]  kfree+0xf2/0x3b0
[    0.979210]  use_after_free+0x73/0x90
[    0.979210]  do_one_initcall+0x9d/0x310
[    0.979210]  kernel_init_freeable+0x458/0x4d0
[    0.979210]  kernel_init+0x19/0x150
[    0.979210]  ret_from_fork+0x10d/0x1b0
[    0.979210]  ret_from_fork_asm+0x1a/0x30
[    0.979210]
```

```
(gdb) list *(use_after_free+0x73)
0xffffffff8630be03 is in use_after_free (drivers/uaf/uaf.c:17).
12                  goto out;
13              }
14              strcpy(leak, "This is a use-after-free example");
15              kfree(leak);
16              // Using the pointer after freeing it, which is undefined behavior
17              printk(KERN_INFO "Using freed memory: %s\n", leak); // This is dangerous!
18      out:
19              return 0;
20      }
21
(gdb)
```

es necesario jugar con los valores...

para encontrar los bugs.

tracing.

mucha depuración, ahora pasemos a la exploración.

ftrace: el papi de strace.

muchos conocemos strace.

pero ftrace está un paso más abajo.

the function tracer.

pasamos de solo ver
syscalls...

a ver el código completo.

ftrace es parte de la arquitectura del kernel.

basado en callbacks.

trace-cmd

podríamos usar ftrace a través de la interfaz...

sysfs /sys/kernel/tracing.

para eso existe trace-cmd.

veamos el clásico printf.

```
hola amigos


[ anti@mothership ]
[    11:41 AM     ]
~/Documents/Research/linux-debugging/ftrace/code_example $
```

strace cuenta una historia...

```
~/Documents/Research/linux-debugging/ftrace/code_example $ strace ./main
execve("./main", ["./main"], 0x7ffe150f1400 /* 61 vars */) = 0
brk(NULL)                               = 0x3eed1000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f2496893000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=145907, ...}) = 0
mmap(NULL, 145907, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f249686f000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\00007\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2447520, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2038832, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f249667d000
mmap(0x7f24967ec000, 479232, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16f000) = 0x7f24967ec000
mmap(0x7f2496861000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e3000) = 0x7f2496861000
mmap(0x7f2496867000, 31792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f2496867000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f249667a000
arch_prctl(ARCH_SET_FS, 0x7f249667a740) = 0
set_tid_address(0x7f249667aa10)         = 2331067
set_robust_list(0x7f249667aa20, 24)     = 0
rseq(0x7f249667a680, 0x20, 0, 0x53053053) = 0
mprotect(0x7f2496861000, 16384, PROT_READ) = 0
mprotect(0x402000, 4096, PROT_READ)     = 0
mprotect(0x7f24968d1000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=16384*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f249686f000, 145907)          = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...}) = 0
getrandom("\xaa\x3d\x8a\xc3\xff\x17\xe3\x62", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x3eed1000
brk(0x3eef2000)                         = 0x3eef2000
write(1, "hola amigos\n", 12hola amigos
)           = 12
exit_group(0)                           = ?
+++ exited with 0 +++
```

interesante...

pero limitado.

pasemos a ftrace.

```
~/Documents/Research/linux-debugging/ftrace/code_example $ sudo trace-cmd record -p function_graph -F ./main
  plugin 'function_graph'
hola amigos
CPU0 data recorded at offset=0x3ef000
    0 bytes in size (0 uncompressed)
CPU1 data recorded at offset=0x3ef000
    0 bytes in size (0 uncompressed)
CPU2 data recorded at offset=0x3ef000
    0 bytes in size (0 uncompressed)
CPU3 data recorded at offset=0x3ef000
    0 bytes in size (0 uncompressed)
CPU4 data recorded at offset=0x3ef000
    0 bytes in size (0 uncompressed)
CPU5 data recorded at offset=0x3ef000
    0 bytes in size (0 uncompressed)
CPU6 data recorded at offset=0x3ef000
    309893 bytes in size (1802240 uncompressed)
CPU7 data recorded at offset=0x43b000
    0 bytes in size (0 uncompressed)
CPU8 data recorded at offset=0x43b000
    0 bytes in size (0 uncompressed)
CPU9 data recorded at offset=0x43b000
    0 bytes in size (0 uncompressed)
CPU10 data recorded at offset=0x43b000
    0 bytes in size (0 uncompressed)
CPU11 data recorded at offset=0x43b000
    0 bytes in size (0 uncompressed)
```

hmmm...

```
~/Documents/Research/linux-debugging/ftrace/code_example $ sudo trace-cmd report | wc -l
29188
```

¡¿29000 funciones!?

podemos decir que el output es un tanto más grande.

```
cpus=12
          main-2331557 [006] ...2. 227930.190666: funcgraph_entry:        1.222 us   |  mutex_unlock(); (ret=0xffff8e4e5461b080)
          main-2331557 [006] ...2. 227930.190668: funcgraph_entry:                   |  __f_unlock_pos() {
          main-2331557 [006] ...2. 227930.190668: funcgraph_entry:        0.080 us   |    mutex_unlock(); (ret=0xffff8e4e5461b080)
          main-2331557 [006] ...1. 227930.190668: funcgraph_exit:         0.250 us   |  } (ret=0xffff8e4e5461b080)
          main-2331557 [006] ...2. 227930.190668: funcgraph_entry:        0.061 us   |  syscall_exit_to_user_mode_prepare(); (ret=0x1)
          main-2331557 [006] ...2. 227930.190669: funcgraph_entry:                   |  __x64_sys_execve() {
          main-2331557 [006] ...2. 227930.190669: funcgraph_entry:                   |    getname_flags() {
          main-2331557 [006] ...2. 227930.190669: funcgraph_entry:                   |      getname_flags.part.0() {
          main-2331557 [006] ...2. 227930.190669: funcgraph_entry:        0.100 us   |        kmem_cache_alloc_noprof(); (ret=0xffff8e4c826a7000)
          main-2331557 [006] ...2. 227930.190669: funcgraph_entry:                   |        __check_object_size() {
          main-2331557 [006] ...2. 227930.190669: funcgraph_entry:                   |          __check_object_size.part.0() {
          main-2331557 [006] ...2. 227930.190670: funcgraph_entry:        0.060 us   |            check_stack_object(); (ret=0x0)
          main-2331557 [006] ...2. 227930.190670: funcgraph_entry:        0.070 us   |            is_vmalloc_addr(); (ret=0x0)
          main-2331557 [006] ...2. 227930.190670: funcgraph_entry:        0.070 us   |            __virt_addr_valid(); (ret=0x1)
          main-2331557 [006] ...2. 227930.190670: funcgraph_entry:        0.050 us   |            __check_heap_object(); (ret=0xfe0)
          main-2331557 [006] ...1. 227930.190670: funcgraph_exit:         0.651 us   |          } (ret=0xffff8e5305200000)
          main-2331557 [006] ...1. 227930.190670: funcgraph_exit:         0.792 us   |        } (ret=0xffff8e5305200000)
          main-2331557 [006] ...1. 227930.190670: funcgraph_exit:         1.172 us   |      } (ret=0xffff8e4c826a7000)
          main-2331557 [006] ...1. 227930.190670: funcgraph_exit:         1.302 us   |    } (ret=0xffff8e4c826a7000)
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |    do_execveat_common.isra.0() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |      alloc_bprm() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |        do_open_execat() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |          do_filp_open() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |            path_openat() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |              alloc_empty_file() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |                kmem_cache_alloc_noprof() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |                  __memcg_slab_post_alloc_hook() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:                   |                    obj_cgroup_charge() {
          main-2331557 [006] ...2. 227930.190671: funcgraph_entry:        0.080 us   |                      consume_obj_stock(); (ret=0x1)
          main-2331557 [006] ...1. 227930.190672: funcgraph_exit:         0.230 us   |                    } (ret=0x0)
          main-2331557 [006] ...2. 227930.190672: funcgraph_entry:        0.070 us   |                    __rcu_read_lock(); (ret=0x1)
          main-2331557 [006] ...2. 227930.190672: funcgraph_entry:        0.060 us   |                    __rcu_read_unlock(); (ret=0x0)
          main-2331557 [006] ...2. 227930.190672: funcgraph_entry:        0.090 us   |                    mod_objcg_state(); (ret=0xffff8e5bb7b70000)
          main-2331557 [006] ...1. 227930.190672: funcgraph_exit:         0.881 us   |                  } (ret=0x1)
          main-2331557 [006] ...1. 227930.190672: funcgraph_exit:         1.032 us   |                } (ret=0xffff8e5373bde6c0)
          main-2331557 [006] ...2. 227930.190672: funcgraph_entry:                   |                init_file() {
          main-2331557 [006] ...2. 227930.190672: funcgraph_entry:                   |                  security_file_alloc() {
          main-2331557 [006] ...2. 227930.190672: funcgraph_entry:        0.080 us   |                    kmem_cache_alloc_noprof(); (ret=0xffff8e5064c32260)
          main-2331557 [006] ...2. 227930.190673: funcgraph_entry:        0.060 us   |                    selinux_file_alloc_security(); (ret=0x0)
          main-2331557 [006] ...2. 227930.190673: funcgraph_entry:        0.060 us   |                    bpf_lsm_file_alloc_security(); (ret=0x0)
          main-2331557 [006] ...2. 227930.190673: funcgraph_entry:        0.070 us   |                    hook_file_alloc_security(); (ret=0x0)
          main-2331557 [006] ...1. 227930.190673: funcgraph_exit:         0.641 us   |                  } (ret=0x0)
          main-2331557 [006] ...2. 227930.190673: funcgraph_entry:        0.070 us   |                  __mutex_init(); (ret=0xffff8e5373bde720)
          main-2331557 [006] ...1. 227930.190673: funcgraph_exit:         0.942 us   |                } (ret=0x0)
          main-2331557 [006] ...1. 227930.190673: funcgraph_exit:         2.184 us   |              } (ret=0xffff8e5373bde6c0)
          main-2331557 [006] ...2. 227930.190673: funcgraph_entry:                   |              path_init() {
          main-2331557 [006] ...2. 227930.190673: funcgraph_entry:        0.070 us   |                __rcu_read_lock(); (ret=0x1)
          main-2331557 [006] ...1. 227930.190674: funcgraph_exit:         0.250 us   |              } (ret=0xffff8e4c826a7020)
          main-2331557 [006] ...2. 227930.190674: funcgraph_entry:                   |              link_path_walk.part.0.constprop.0() {
          main-2331557 [006] ...2. 227930.190674: funcgraph_entry:                   |                inode_permission() {
```

no veremos el output completo...

pero nos sirve para hacernos una idea.

del poder de ftrace.

trace-cmd; profundizando.

los principales comandos que usaremos...

son trace-cmd record...

y trace-cmd report.

por lo general, utilizaremos flags en trace-cmd record.

man trace-cmd-record
nos enseña las flags.

pero veamos algunas.

-F programa
filtra por llamadas hechas por solamente lo que vayamos a ejecutar.

-p plugin
seleccionamos el plugin de visualización (generalmente function_graph).

-n funcion
nos permite "ignorar" ciertas trazas.

-g funcion
nos permite enfocar el tracing a ciertas funciones.

-o archivo
guardar el trace.dat en un archivo 'archivo'.

--max-graph-depth n
profundidad máxima de funciones.

ahora...

¡combinemos!

```
sudo trace-cmd record \
          -p function_graph  \
          -g __x64_sys_write \
          -n rw_verify_area  \
          -F ./main
```

```
~/Documents/Research/linux-debugging/ftrace/code_example $ trace-cmd report | wc -l
273
```

273 es un valor más entendible

la idea no es leer 29k funciones

explorar y minimizar el alcance.

kernelshark

en vez de usar trace-cmd report
para observar la traza...

podemos utilizar kernelshark.

como el nombre indica...

es como wireshark.

pero para trazas del kernel.

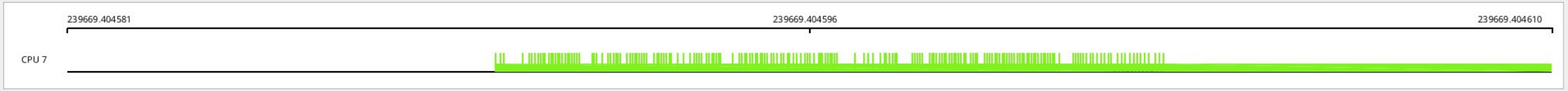
```
cpus=12
           main-2361126 [007] ...2. 239669.404589: funcgraph_entry:                   |  do_output_char() {
           main-2361126 [007] ...2. 239669.404590: funcgraph_entry:                   |    pty_write() {
           main-2361126 [007] ...2. 239669.404590: funcgraph_entry:                   |      tty_insert_flip_string_and_push_buffer() {
           main-2361126 [007] ...2. 239669.404590: funcgraph_entry:       0.070 us    |        _raw_spin_lock_irqsave(); (ret=0x292)
           main-2361126 [007] d..3. 239669.404591: funcgraph_entry:                   |        __tty_insert_flip_string_flags() {
           main-2361126 [007] d..3. 239669.404591: funcgraph_entry:       0.040 us    |          __tty_buffer_request_room(); (ret=0x2)
           main-2361126 [007] d..2. 239669.404591: funcgraph_exit:        0.160 us    |        } (ret=0x2)
           main-2361126 [007] d..3. 239669.404591: funcgraph_entry:       0.050 us    |        _raw_spin_unlock_irqrestore(); (ret=0xffff8e4cdb3f3800)
           main-2361126 [007] ...2. 239669.404591: funcgraph_entry:                   |        queue_work_on() {
           main-2361126 [007] d..2. 239669.404591: funcgraph_entry:       0.060 us    |          clear_pending_if_disabled(); (ret=0x0)
           main-2361126 [007] d..2. 239669.404591: funcgraph_entry:                   |          __queue_work() {
           main-2361126 [007] d..2. 239669.404591: funcgraph_entry:       0.040 us    |            __rcu_read_lock(); (ret=0x1)
           main-2361126 [007] d..2. 239669.404591: funcgraph_entry:       0.050 us    |            wq_select_unbound_cpu(); (ret=0x7)
           main-2361126 [007] d..2. 239669.404591: funcgraph_entry:       0.050 us    |            _raw_spin_lock(); (ret=0x0)
           main-2361126 [007] d..3. 239669.404591: funcgraph_entry:       0.051 us    |            pwq_tryinc_nr_active(); (ret=0x1)
           main-2361126 [007] d..3. 239669.404591: funcgraph_entry:       0.040 us    |            insert_work(); (ret=0x3)
           main-2361126 [007] d..3. 239669.404592: funcgraph_entry:                   |            kick_pool() {
           main-2361126 [007] d..3. 239669.404592: funcgraph_entry:                   |              wake_up_process() {
           main-2361126 [007] d..3. 239669.404592: funcgraph_entry:                   |                try_to_wake_up() {
           main-2361126 [007] d..4. 239669.404592: funcgraph_entry:       0.040 us    |                  _raw_spin_lock_irqsave(); (ret=0x82)
           main-2361126 [007] d..6. 239669.404592: funcgraph_entry:                   |                  __traceiter_sched_waking() {
           main-2361126 [007] d..6. 239669.404592: funcgraph_entry:       0.040 us    |                    _raw_spin_lock_irqsave(); (ret=0x82)
           main-2361126 [007] d..7. 239669.404592: funcgraph_entry:       0.050 us    |                    _raw_spin_unlock_irqrestore(); (ret=0xffff8e4c974b9000)
           main-2361126 [007] d..5. 239669.404592: funcgraph_exit:        0.310 us    |                  } (ret=0x0)
           main-2361126 [007] d..5. 239669.404592: funcgraph_entry:                   |                  select_task_rq_fair() {
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:       0.040 us    |                    __rcu_read_lock(); (ret=0x2)
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:                   |                    wake_affine() {
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:       0.040 us    |                      available_idle_cpu(); (ret=0x0)
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:       0.050 us    |                      available_idle_cpu(); (ret=0x0)
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:       0.140 us    |                      task_h_load(); (ret=0x0)
           main-2361126 [007] d..4. 239669.404593: funcgraph_exit:        0.431 us    |                    } (ret=0xb)
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:                   |                    select_idle_sibling() {
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:       0.040 us    |                      available_idle_cpu(); (ret=0x0)
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:       0.060 us    |                      cpus_share_cache(); (ret=0xffff8e5bb7df0001)
           main-2361126 [007] d..5. 239669.404593: funcgraph_entry:       0.090 us    |                      available_idle_cpu(); (ret=0x0)
           main-2361126 [007] d..5. 239669.404594: funcgraph_entry:                   |                      select_idle_cpu() {
           main-2361126 [007] d..5. 239669.404594: funcgraph_entry:                   |                        select_idle_core.isra.0() {
           main-2361126 [007] d..5. 239669.404594: funcgraph_entry:       0.050 us    |                          available_idle_cpu(); (ret=0x1)
           main-2361126 [007] d..5. 239669.404594: funcgraph_entry:       0.040 us    |                          available_idle_cpu(); (ret=0x1)
           main-2361126 [007] d..4. 239669.404594: funcgraph_exit:        0.310 us    |                        } (ret=0x0)
           main-2361126 [007] d..4. 239669.404594: funcgraph_exit:        0.490 us    |                      } (ret=0x0)
           main-2361126 [007] d..4. 239669.404594: funcgraph_exit:        1.022 us    |                    } (ret=0x0)
           main-2361126 [007] d..5. 239669.404594: funcgraph_entry:       0.040 us    |                    __rcu_read_unlock(); (ret=0x0)
           main-2361126 [007] d..4. 239669.404594: funcgraph_exit:        1.773 us    |                  } (ret=0x0)
           main-2361126 [007] d..5. 239669.404594: funcgraph_entry:       0.040 us    |                  kthread_is_per_cpu(); (ret=0x0)
           main-2361126 [007] d..5. 239669.404595: funcgraph_entry:                   |                  set_task_cpu() {
           main-2361126 [007] d..5. 239669.404595: funcgraph_entry:                   |                    migrate_task_rq_fair() {
```

trace-cmd report, feo!

File Filter Plots Tools Help

Pointer:

< + - > ++ -- Marker A Marker B A,B Delta:



Search: Column # contains Next Prev Graph follows

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	7	239669.404589	main	2361126	...2.	ftrace/funcgraph_entry	| do_output_char() {
1	7	239669.404590	main	2361126	...2.	ftrace/funcgraph_entry	| pty_write() {
2	7	239669.404590	main	2361126	...2.	ftrace/funcgraph_entry	| tty_insert_flip_string_and_push_buffer() {
3	7	239669.404590	main	2361126	...2.	ftrace/funcgraph_entry	0.070 us | _raw_spin_lock_irqsave(); (ret=0x292)
4	7	239669.404590	main	2361126	d..2.	ftrace/funcgraph_exit	0.070 us | } (ret=0x292)
5	7	239669.404591	main	2361126	d..3.	ftrace/funcgraph_entry	| __tty_insert_flip_string_flags() {
6	7	239669.404591	main	2361126	d..3.	ftrace/funcgraph_entry	0.040 us | __tty_buffer_request_room(); (ret=0x2)
7	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_exit	0.040 us | } (ret=0x2)
8	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_exit	0.160 us | } (ret=0x2)
9	7	239669.404591	main	2361126	d..3.	ftrace/funcgraph_entry	0.050 us | _raw_spin_unlock_irqrestore(); (ret=0xffff8e4cdb3f3800)
10	7	239669.404591	main	2361126	...1.	ftrace/funcgraph_exit	0.050 us | } (ret=0xffff8e4cdb3f3800)
11	7	239669.404591	main	2361126	...2.	ftrace/funcgraph_entry	| queue_work_on() {
12	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_entry	0.060 us | clear_pending_if_disabled(); (ret=0x0)
13	7	239669.404591	main	2361126	d..1.	ftrace/funcgraph_exit	0.060 us | } (ret=0x0)
14	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_entry	| __queue_work() {
15	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_entry	0.040 us | __rcu_read_lock(); (ret=0x1)
16	7	239669.404591	main	2361126	d..1.	ftrace/funcgraph_exit	0.040 us | } (ret=0x1)
17	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_entry	0.050 us | wq_select_unbound_cpu(); (ret=0x7)
18	7	239669.404591	main	2361126	d..1.	ftrace/funcgraph_exit	0.050 us | } (ret=0x7)
19	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_entry	0.050 us | _raw_spin_lock(); (ret=0x0)
20	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_exit	0.050 us | } (ret=0x0)
21	7	239669.404591	main	2361126	d..3.	ftrace/funcgraph_entry	0.051 us | pwq_tryinc_nr_active(); (ret=0x1)
22	7	239669.404591	main	2361126	d..2.	ftrace/funcgraph_exit	0.051 us | } (ret=0x1)
23	7	239669.404591	main	2361126	d..3.	ftrace/funcgraph_entry	0.040 us | insert_work(); (ret=0x3)
24	7	239669.404592	main	2361126	d..2.	ftrace/funcgraph_exit	0.040 us | } (ret=0x3)
25	7	239669.404592	main	2361126	d..3.	ftrace/funcgraph_entry	| kick_pool() {
26	7	239669.404592	main	2361126	d..3.	ftrace/funcgraph_entry	| wake_up_process() {
27	7	239669.404592	main	2361126	d..3.	ftrace/funcgraph_entry	| try_to_wake_up() {
28	7	239669.404592	main	2361126	d..4.	ftrace/funcgraph_entry	0.040 us | _raw_spin_lock_irqsave(); (ret=0x82)
29	7	239669.404592	main	2361126	d..4.	ftrace/funcgraph_exit	0.040 us | } (ret=0x82)
30	7	239669.404592	main	2361126	d..6.	ftrace/funcgraph_entry	| __traceiter_sched_waking() {
31	7	239669.404592	main	2361126	d..6.	ftrace/funcgraph_entry	0.040 us | _raw_spin_lock_irqsave(); (ret=0x82)
32	7	239669.404592	main	2361126	d..6.	ftrace/funcgraph_exit	0.040 us | } (ret=0x82)
33	7	239669.404592	main	2361126	d..7.	ftrace/funcgraph_entry	0.050 us | _raw_spin_unlock_irqrestore(); (ret=0xffff8e4c974b9000)
34	7	239669.404592	main	2361126	d..5.	ftrace/funcgraph_exit	0.050 us | } (ret=0xffff8e4c974b9000)
35	7	239669.404592	main	2361126	d..5.	ftrace/funcgraph_exit	0.310 us | } (ret=0x0)

kernelshark! bonito!

outro

hoy hemos pasado por muchas cosas.

desde compilación...

flags de depuración...

uso de GDB+KGDB...

depuramos un UAF...

vimos ftrace...

trace-cmd...

y kernelshark.

adios.
un besito a todos.

samuel@securejump.cl
4nt1@resacachile.cl
@4nt1@lile.cl
@clp_c tg

un besito a mi esposa tambien