ESTR 3102

Adding system call (Linux kernel v4.0.5)

Helen Chan

SHB 118

hwchan@cse.cuhk.edu.hk

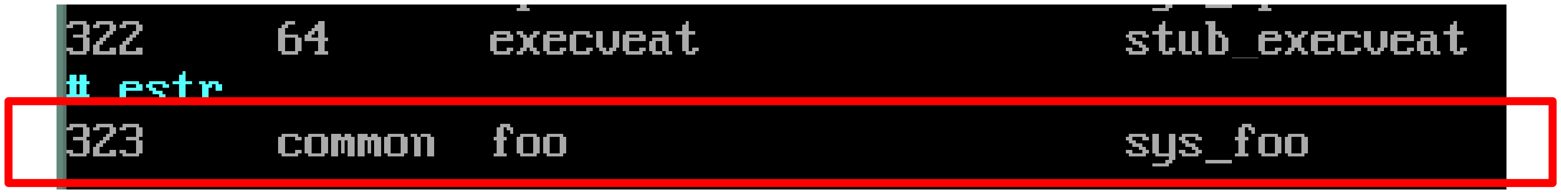
(the hacking technique also works on kernel v3.14, prepared by Dr. Q. Huang)

Step 1 – Add a System Call Entry

- **`# cd /usr/src/linux`**
- Edit file:

  `arch/x86/syscalls/syscall_64.tbl`

- Add the new entry in the red box **after the system call with no. 322**



```
322     64      execveat                stub_execveat
# estr
323     common  foo                     sys_foo
```

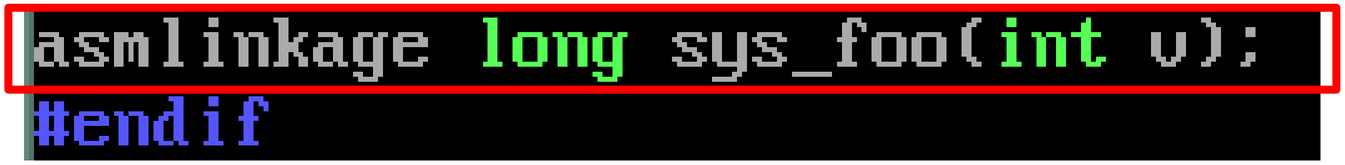
- The entry consists of 4 fields (separated by tabs)
  - syscall number
  - abi type
  - name
  - Entry function name

Step 2 – Define the Func. Prototype

- Edit file:

  `include/linux/syscalls.h`

- Add the prototype to nearly the end of the file **just before #endif**



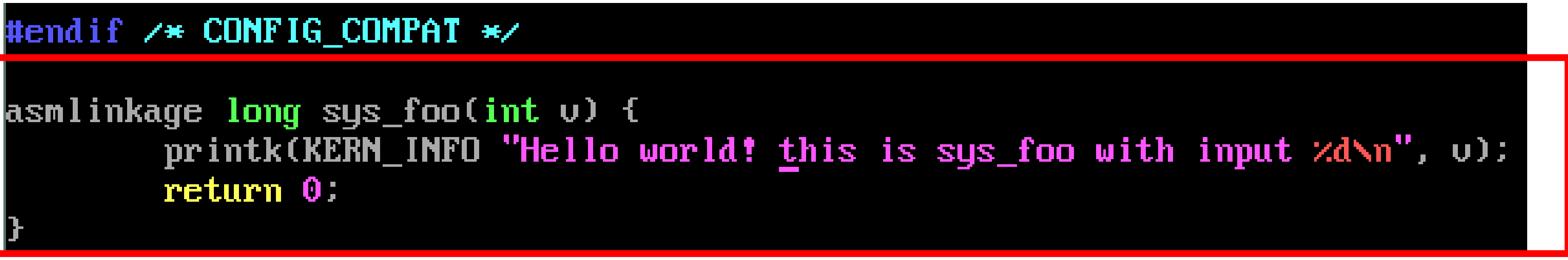
```
asmlinkage long sys_foo(int v);
#endif
```

Step 3 – Fill in the Func. Body

- Edit file:

  `kernel/sys.c`

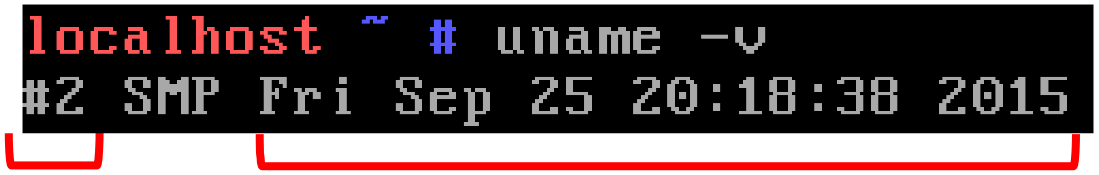
- Add the function implementation at the end of file **after**
  `#endif /* CONFIG_COMPACT */`



```
#endif /* CONFIG_COMPAT */

asmlinkage long sys_foo(int v) {
        printk(KERN_INFO "Hello world! this is sys_foo with input %d\n", v);
        return 0;
}
```

Step 4 – Recompile Kernel & Reboot

- Compile and install new kernel image
  - `# make && make install`
- Reboot into the new kernel image
  - `# reboot`
- Check the "version" of the kernel image
  - `# uname -v`
  - (Output)

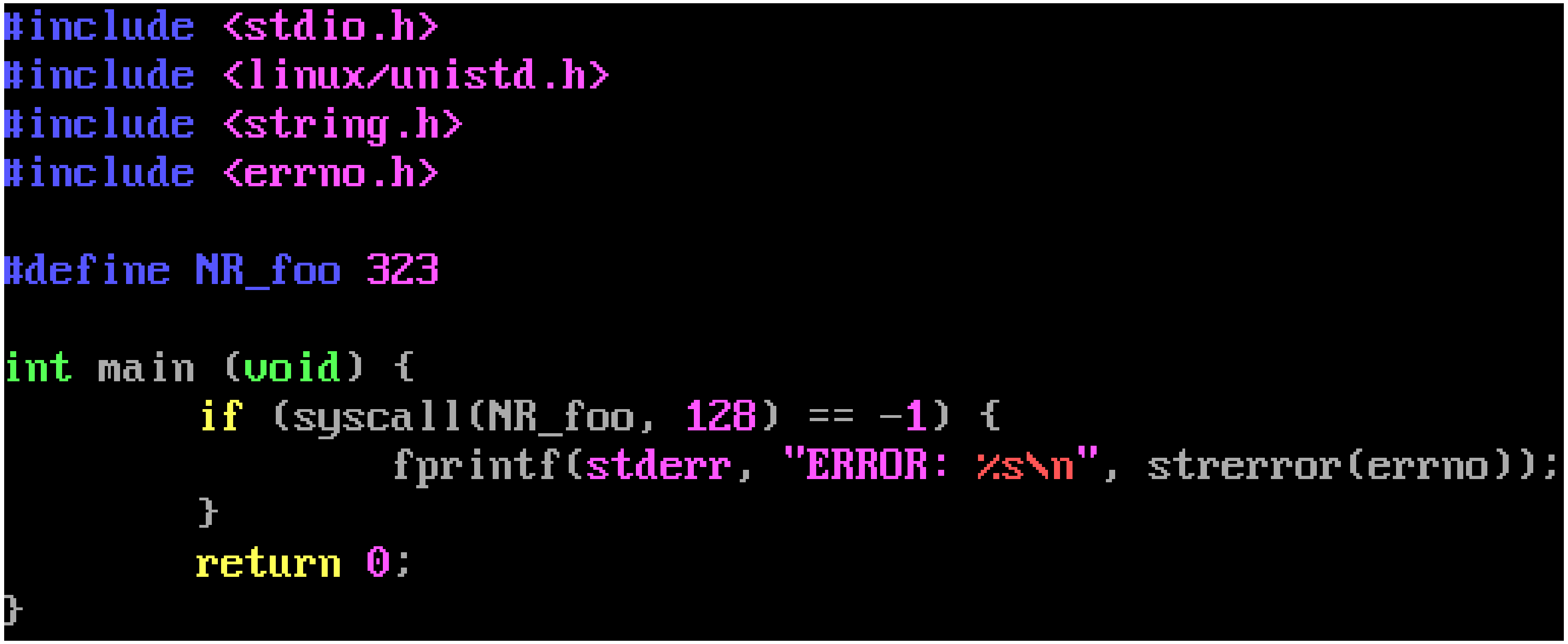


```
localhost ~ # uname -v
#2 SMP Fri Sep 25 20:18:38 2015
```

Increment after each compilation
time when this image was compiled

Step 5 – Write a Test Program

- Create and edit **`mytest.c`**

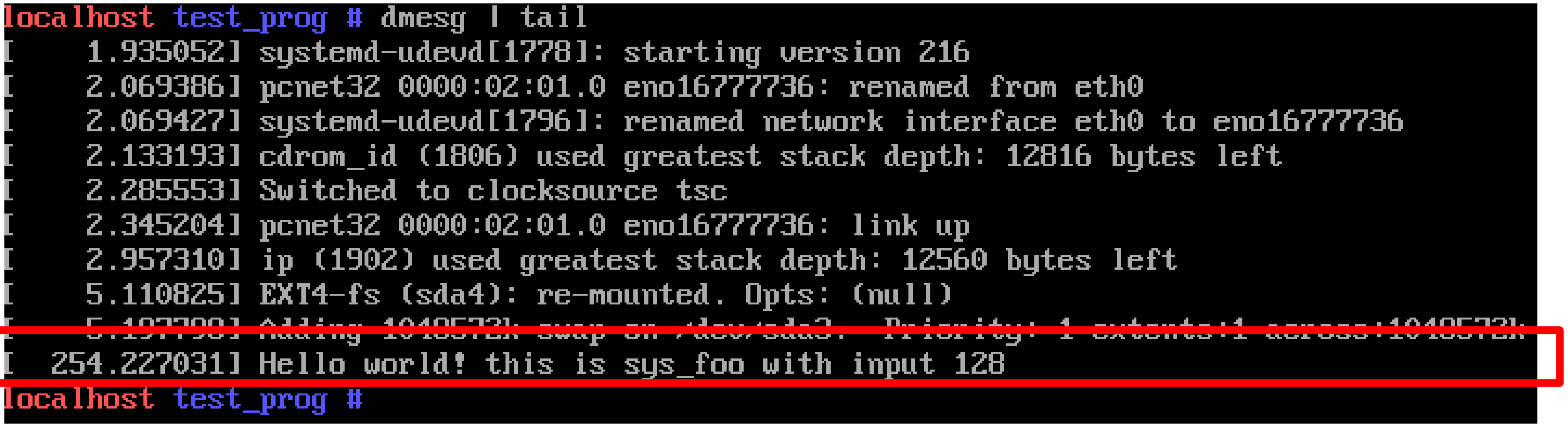


```
#include <stdio.h>
#include <linux/unistd.h>
#include <string.h>
#include <errno.h>

#define NR_foo 323

int main (void) {
        if (syscall(NR_foo, 128) == -1) {
                fprintf(stderr, "ERROR: %s\n", strerror(errno));
        }
        return 0;
}
```

- **`# gcc -o mytest mytest.c`**

Step 6 – Test the New System Call

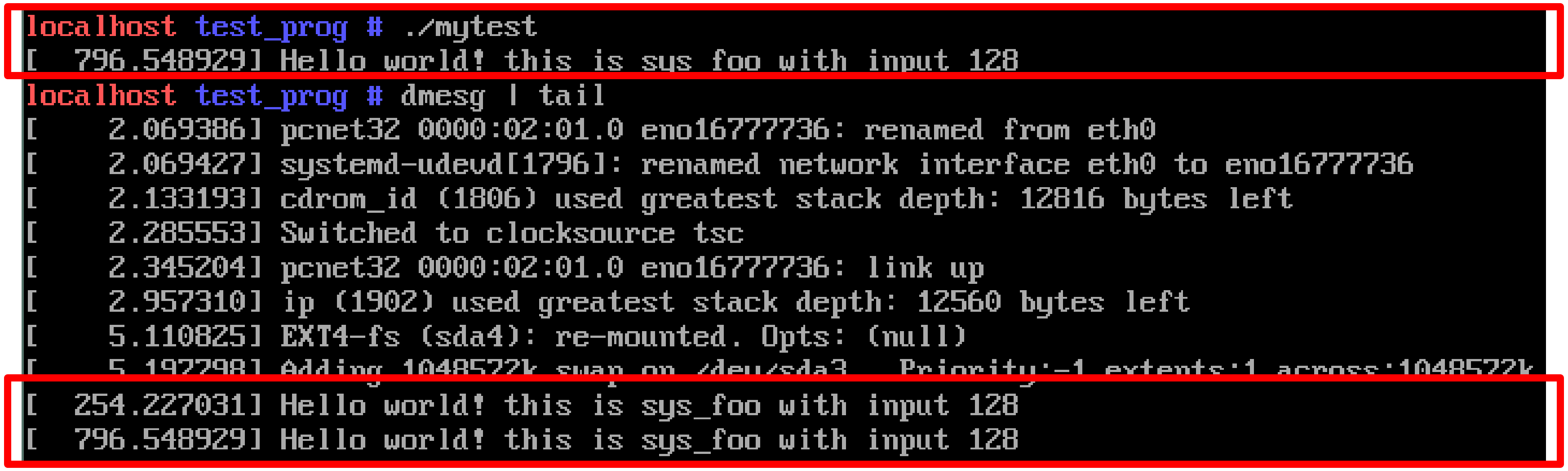
- Run the program
  - **`# ./mytest`**
  - (No output to console)
- Show the end of system log
  - **`# dmesg | tail`**



```
localhost test_prog # dmesg | tail
[    1.935052] systemd-udevd[1778]: starting version 216
[    2.069386] pcnet32 0000:02:01.0 eno16777736: renamed from eth0
[    2.069427] systemd-udevd[1796]: renamed network interface eth0 to eno16777736
[    2.133193] cdrom_id (1806) used greatest stack depth: 12816 bytes left
[    2.285553] Switched to clocksource tsc
[    2.345204] pcnet32 0000:02:01.0 eno16777736: link up
[    2.957310] ip (1902) used greatest stack depth: 12560 bytes left
[    5.110825] EXT4-fs (sda4): re-mounted. Opts: (null)
[    5.197798] Adding 1048572k swap on /dev/sda3.  Priority:-1 extents:1 across:1048572k
[  254.227031] Hello world! this is sys_foo with input 128
localhost test_prog #
```

Step 6 – Test the New System Call

- Adjust the message level of `printk`
  - **`# echo 8 > /proc/sys/kernel/printk`**
- Run the program again
  - (Output) The message appears in both the console and the system log



```
localhost test_prog # ./mytest
[  796.548929] Hello world! this is sys_foo with input 128
localhost test_prog # dmesg | tail
[    2.069386] pcnet32 0000:02:01.0 eno16777736: renamed from eth0
[    2.069427] systemd-udevd[1796]: renamed network interface eth0 to eno16777736
[    2.133193] cdrom_id (1806) used greatest stack depth: 12816 bytes left
[    2.285553] Switched to clocksource tsc
[    2.345204] pcnet32 0000:02:01.0 eno16777736: link up
[    2.957310] ip (1902) used greatest stack depth: 12560 bytes left
[    5.110825] EXT4-fs (sda4): re-mounted. Opts: (null)
[    5.192298] Adding 1048572k swap on /dev/sda3.  Priority:-1 extents:1 across:1048572k
[  254.227031] Hello world! this is sys_foo with input 128
[  796.548929] Hello world! this is sys_foo with input 128
```