

A tutorial approach

Introduction

Alex Newman

Oct 11th 2005

With Acroread, **CTRL-L** switch
between full screen and window mode

1 – Introduction	3
2 – Setup you enviroment	4
3 – Your first kernel hack	5
4 – Kernel versus Userland	10
5 – The kernel's network drivers	11
6 – Userland alternatives	12
7 – Kernel level network programming	14
8 – Style and politics	16

1 – Introduction

I am here to ramp you up to proper linux kernel coding

FreeBSD's netgraph is not covered but you should check it out.

Background material raise your hand or yell when you are missing vocab

2 – Setup you enviroment

Does hello world works (i.e. write a program which prints hello world to the screen)

cd into your kernel tree.

make bzImage && make modules

While it is running we will move on

3 – Your first kernel hack

```
} Bring up init/main.c
```

```
add in to calibrate_delay(void)
```

```
printk("*** Hello kernelland! ***\n");
```

```
make bzimage
```

```
reboot
```

```
dmesg | less
```

```
Now try
```

```
printk("The address of loops_per_jiffy is  
%p\n", &loops_per_jiffy);
```

The function `start_kernel()` (in `init/main.c`)

The function `init()` (in `init/main.c`)

The user level "init" program

nto drivers/misc/

```
include <linux/module.h>
include <linux/config.h>
include <linux/init.h>

static int __init mymodule_init(void)

printk ("My module worked!\n");
return 0;

static void __exit mymodule_exit(void)

printk ("Unloading my module.\n");
return;

module_init(mymodule_init);
module_exit(mymodule_exit);
MODULE_LICENSE("GPL");
```

In the makefile add `obj-m += mymodule.o` Right after `O_TARGET`

```
make -C /home/net/linux-2.4.18-3/ SUBDIRS=$PWD modules
```

```
sudo insmod mymodule.o
```

4 – Kernel versus Userland

What is virtual memory or fork in kernel mode?

Try adding in floating point

What happens when we block/sleep?

In kernel/printk.c add int

```
my_variable = 0;
```

In your module try to access the extern variable i.e.

```
extern int my_variable;  
printk ("my_variable is %d\n", my_variable);  
my_variable++;
```

what errors do you get

Now in the var declaration try

```
EXPORT_SYMBOL(my_variable);
```

5 – The kernel's network drivers

ow picture and give them the journey

6 – Userland alternatives

Divert sockets

another option is netfilter it goes into the hooks before in our journey.

```
t main(int argc, char** argv) {
fprintf(stderr, "%s:Creating a socket\n", argv[0]);
fd=socket(AF_INET, SOCK_RAW, IPPROTO_DIVERT);
fprintf(stderr, "%s:Binding a socket\n", argv[0]);
ret=bind(fd, &bindPort, sizeof(struct sockaddr_in));
while(1) {
    n=recvfrom(fd, packet, BUFSIZE, 0, &sin, &sinlen);
    hdr=(struct iphdr*)packet;
    printf("%s: The packet looks like this:\n", argv[0]);
        for( i=0; i<40; i++) {
            printf("%02x ", (int)*(packet+i));
            if (!(i+1)%16) printf("\n");
        };
    addr.s_addr=hdr->saddr;
    printf("\n%s: Source address: %s\n", argv[0], inet_ntoa(addr));
}
```

7 – Kernel level network programming

How about a modification which messes with an IDS

The client module catches every TCP packet with the SYN flag on and swaps it with a FIN flag.

The server module does exactly the opposite, swaps the FIN for a SYN.

Check your `7otp.c` file

normal connection

```
telnet hardbitten
```

A regular connection (without the modules loaded) looks like this

```
03:29:56.766445 foo.1025 > bar.23: tcp (SYN)
```

```
03:29:56.766580 bar.23 > foo.1025: tcp (SYN ACK)
```

```
03:29:56.766637 foo.1025 > bar.23: tcp (ACK)
```

A cracked out connection

```
03:35:30.576331 foo.1025 > bar.80: tcp (FIN)
```

```
03:35:30.576440 bar.80 > foo.1025: tcp (FIN ACK)
```

8 – Style and politics