MP5

Checkpoint 2

1. Reverse engineering a kernel module

What is a kernel module?

A loadable kernel module (LKM) is an object file that contains code to extend the running kernel. LKMs are typically used to support new hardware, filesystems, or system calls.

Why kernel modules? Access (read/write) to privileged kernel objects that user mode processes cannot

- **Rootkits**
  - hide a process, a file, or a network connection by hooking system calls
  - write directly to a hard drive MBR
- **Keyloggers:** intercept keypresses by hooking input devices
- **Firewall rules:** drop or accept a network packet before the packet reaches user mode
- **Trace and measure** application performances.

You can be a kernel hacker!

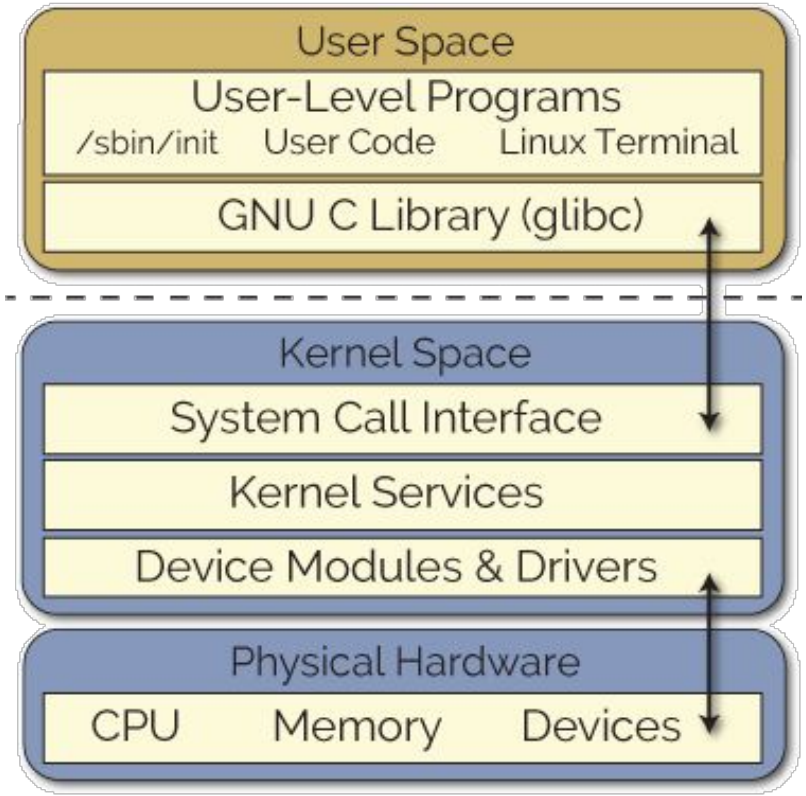
Basic commands and architecture

Given a kernel module, e.g., *knockd.ko*

`lsmod`: list running modules

`insmod ./knockd.ko` : load a module

`rmmod knockd`: unload a module



A Hello World kernel module

There is no *main* function, instead

module_init is called when `insmod`

module_exit is called when `rmmod`

printk outputs to */var/log/kern.log*

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

static int __init my_init(void)
{
    printk(KERN_INFO "hello, my module\n");
    return 0;
}

static void __exit my_exit(void)
{
    printk(KERN_INFO "good bye, my module\n" );
}

module_init(my_init);
module_exit(my_exit);
```

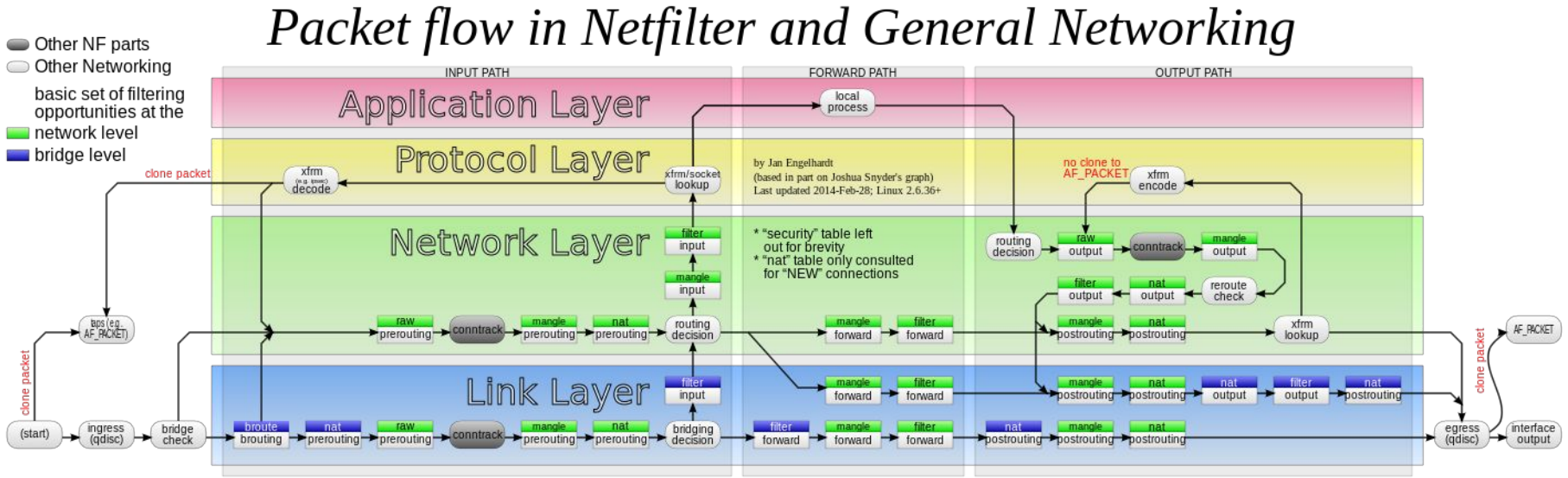
```
# make clean build
```

gives the kernel module obj (.ko)

(see Makefile in _shared/mp5/portknocking-helloworld)

Netfilter: a network hooking framework

Netfilter enables packet filtering, network address translation, and port translation. Network hooks are implemented in the form of customized handlers.



A Hello World netfilter hook (_shared/mp5/)

The hook must be registered in the module init function

Extract port from pkt header

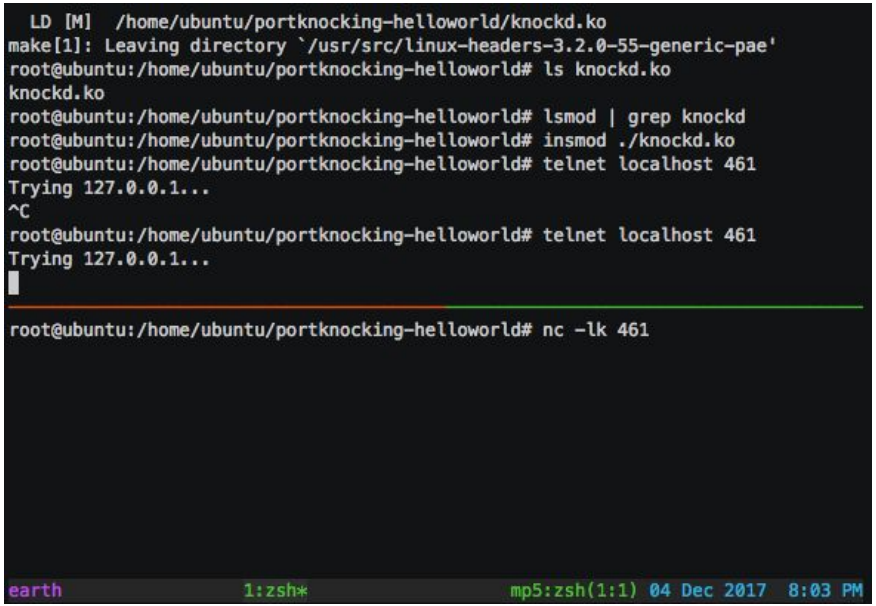
Drop any packets going to the PROTECTED_PORT

Allow other packets

```
22  static unsigned int knockd_filter_function(void *priv, struct sk_buff *skb,
23                                              const struct nf_hook_state *state){
24
25    struct iphdr *ip_header;
26    struct tcphdr *tcp_header;
27    ip_header = ip_hdr(skb);
28
29    tcp_header= (struct tcphdr *)((__u32 *)ip_header+ ip_header->ihl);
30
31    unsigned int dst_port;
32    dst_port = htons((unsigned short int)tcp_header->dest);
33
34    // guard the protected port
35    if (dst_port == PROTECTED_PORT) {
36        return NF_DROP;
37    }
38    return NF_ACCEPT;
39  }
```

Actual code for knockd is different!!!

Demo



https://asciinema.org/a/LlVNyuFNWuXb86tkJah969qaG

Jiffy: kernel timer

A jiffy is the time between two ticks of the system timer interrupt.

`Jiffies` is a global variable in kernel, measuring the ticks so far.

The timer interrupt rate (and jiffy increment rate) is defined by a compile-time constant called HZ.

Important: In our kernel module, HZ = 250. Use this HZ to calculate how long will the filtered port be opened after receiving the knocking sequence?

For example, at time t1, jiffies = 100 and at time t2, jiffies = 350

Physical time difference between t1 and t2 is (350 - 100) / 250 = 1s

2. Side channels in object deserialization attack

How to extract information out of the target?

1. Setup your own server that is reachable from the target
2. Establish a communication channel

In this MP, it does not have to be a covert channel, so you can use telnet or http

In the real-world, using a public and trusted service, such as github / twitter for receiving side channels information is a good idea.

- Twitter/Github bot
- DNS tunneling
- ICMP tunnel
- ...

3. Exploiting a TOCTTOU race condition

In our MP, the digital wallet first:

1. Checks for your balance
2. Processes the withdraw
3. Updates the new balance

You have to trigger the race condition at the at Step 2.

Report the race condition token returned by the server.

```
{

    race_condition_token: “xxxxx”

}
```

4. Passwords cracking

Start from 1-length password

Increase password length until you crack all passwords

Hints:

- Expand the password vocab, including non-ascii and potential unicode characters
- Think of novel ways to generate passwords beyond Markov, e.g., substitution, shifting, foreign languages, etc.
- Run multiple instances of hashcat using GPU VMs on public clouds
- Use the hint, i.e., passwords from CP1, to generate a password candidates.

http://www.jbonneau.com/doc/DBCBW14-NDSS-tangled_web.pdf