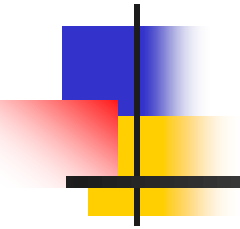
Firewall



Lecture 12

Instructor: Dr. Cong Pu, Ph.D.

cong.pu@okstate.edu

Acknowledgment: Adapted partially from course materials from Dr. Wenliang Du at Syracuse University, Dr. Fengwei Zhang at Southern University of Science and Technology, and Dr. Steven M. Bellovin at Columbia University.

Netfilter

- ***Netfilter*** in Linux: packet processing and filtering framework
- In Linux,
  - each protocol stack defines *hooks* along the packet's traversal path in that stack
    - *hook* is a ***location*** in the kernel that calls out of the kernel to a kernel module routine
  - developers use kernel modules to register callback functions to hooks

Netfilter

- ***Netfilter*** in Linux: packet processing and filtering framework
- In Linux,
  - when packet arrives at a hook, the protocol stack calls netfilter framework with the packet and hook number
  - Netfiler checks if any kernel module has registered a callback function at this hook
  - each registered module will be called to analyze or manipulate packet, and return their *verdict* on packet

Netfilter (cont.)

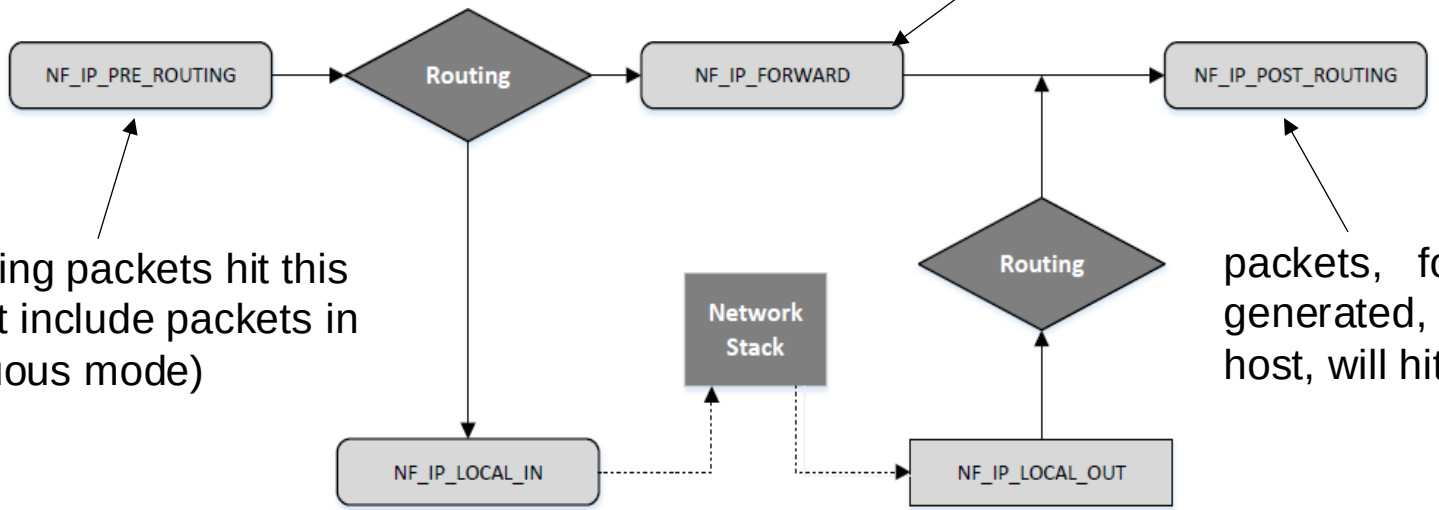
- *Verdict*: return values of kernel module routines:
  - *NF_ACCEPT*: let the packet flow through the stack
  - *NF_DROP*: discard the packet
  - *NF_QUEUE*: pass the packet to the user space via nf_queue facility
    - perform packet handling in user space via asynchronous operations
  - *NF_STOLEN*: inform the netfilter to forget about this packet
    - the packet is further processed by the module
    - the packet is still present and valid in the kernel's internal table
  - *NF_REPEAT*: request the netfilter to call this module again

Ref. (Writing New Netfilter Modules):
https://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO-4.html

Netfiler Hooks for IPv4

- Netfiler defines five hooks for IPv4:

move packets to other hosts (***useful for firewall***)



all incoming packets hit this hook (not include packets in promiscuous mode)

packets, forwarded or generated, going out of host, will hit this hook

incoming packets will go through routing, which decides whether the packet is for

- other machines or
  - will go through forwarding path
- host itself
  - will go through this hook

packets generated by local host reach this hook (the first hook on their way out of host)

Implementing Simple Packet Filter Firewall

- Implementing a packet filter using *netfilter* framework and *loadable kernel module*
  - ***goals***: *blocking* all packets that are going out to port number 23
    - *preventing* users from using *telnet* to connect to other machines

Implementing Simple Packet Filter Firewall (cont.)

- Implementing a ***callback function***, *telnetFilter*, for actual filtering
  - inspect packet (TCP header, port #)
    - if port # is 23, drop packet,
    - otherwise, pass

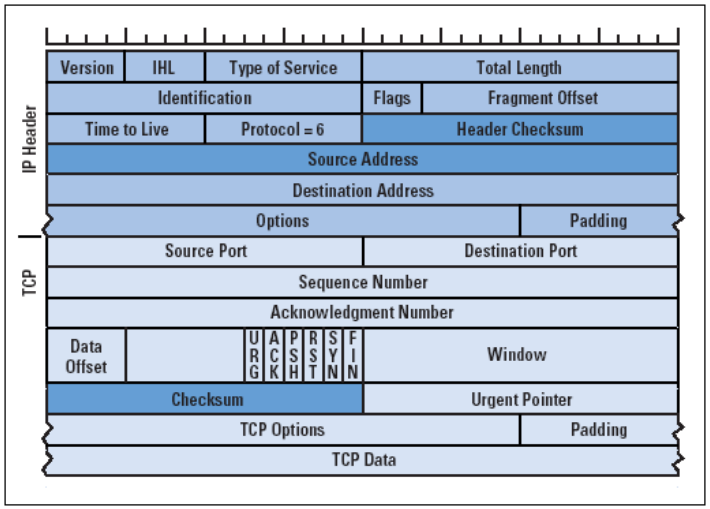
```
unsigned int telnetFilter(void *priv, struct sk_buff *skb,
                const struct nf_hook_state *state)
{
  struct iphdr *iph;
  struct tcphdr *tcph;

  iph = ip_hdr(skb);
  tcph = (void *)iph+iph->ihl*4;

  if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23)) {
    printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
   ((unsigned char *)&iph->daddr)[0],
   ((unsigned char *)&iph->daddr)[1],
   ((unsigned char *)&iph->daddr)[2],
   ((unsigned char *)&iph->daddr)[3]);
    return NF_DROP;
  } else {
    return NF_ACCEPT;
  }
}
```

the reference of entire packet

The filtering logic is hardcoded here.

- drop packet if destination TCP port is 23 (telnet)

decisions



callback function telnetFilter()

- provided with the entire packet, including the headers
- hook telnetFilter() to one of netfilter hooks
  - NF_IP_LOCAL_OUT
  - NF_IP_POST_ROUTING

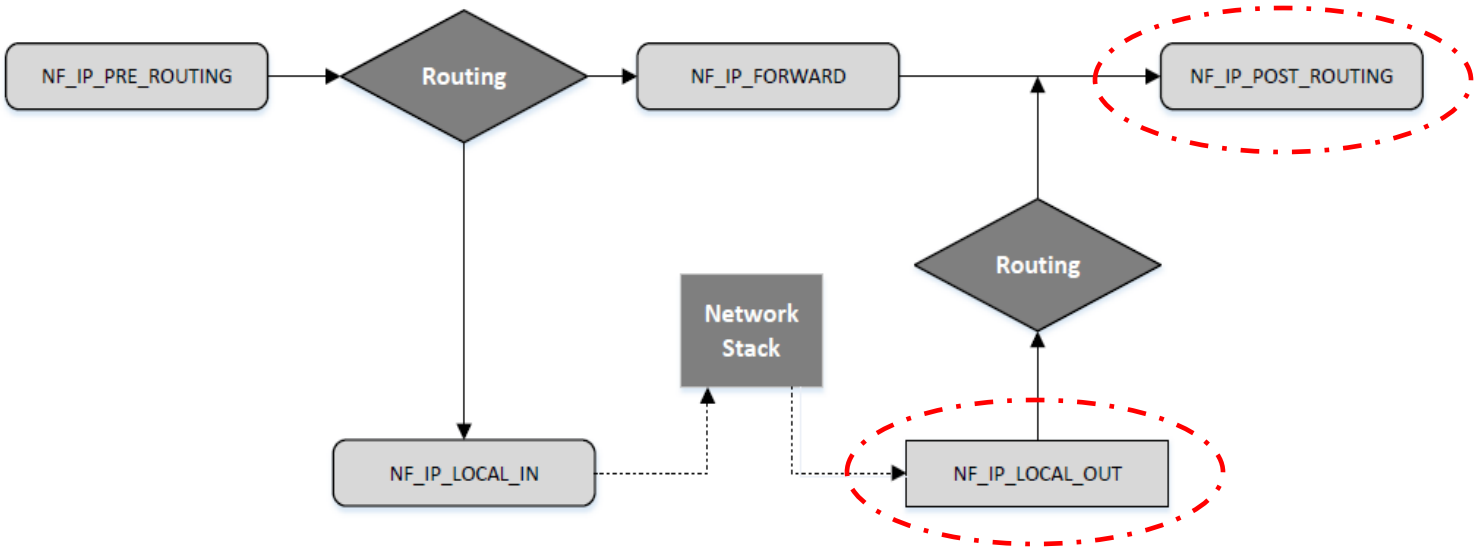
Implementing Simple Packet Filter Firewall (cont.)

- *struct sk_buff* (means socket buffers)
  - core structure in Linux networking
  - socket buffers are the buffers where the Linux kernel handles network packets
    - packets are received by network card
    - put into a socket buffer
    - passed to network stack for processing

ref.: https://www.kernel.org/doc/htmldocs/networking/API-struct-sk-buff.html

Implementing Simple Packet Filter Firewall (cont.)

- Hook telnetFilter() to one netfilter hook
  - NF_IP_LOCAL_OUT or NF_IP_POST_ROUTING



Implementing Simple Packet Filter Firewall (cont.)

- Hook telnetFilter() to one netfilter hook
  - NF_IP_LOCAL_OUT or NF_IP_POST_ROUTING

```
int setUpFilter(void) {
   printk(KERN_INFO "Registering a Telnet filter.\n");
   telnetFilterHook.hook = telnetFilter;
   telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
   telnetFilterHook.pf = PF_INET;
   telnetFilterHook.priority = NF_IP_PRI_FIRST;

   // Register the hook.
   nf_register_hook(&telnetFilterHook);
   return 0;
}

void removeFilter(void) {
   printk(KERN_INFO "Telnet filter is being removed.\n");
   nf_unregister_hook(&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);
```

hook this callback function

use this netfilter hook

IPv4 packet family

register the hook

Testing Firewall Implementation

```
$ sudo insmod telnetFilter.ko
$ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: ...    ← Blocked!
$ dmesg
......
[1166456.149046] Registering a Telnet filter.
[1166535.962316] Dropping telnet packet to 10.0.2.5
[1166536.958065] Dropping telnet packet to 10.0.2.5

// Now, let's remove the kernel module

$ sudo rmmod telnetFilter
$ telnet 10.0.2.5
telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 12.04.2 LTS
ubuntu login:                    ← Succeeded!
```