COSC 647 - Fall 2015 - Logan Bair, Harold McGinnis, & Mary Snyder
Lab 7 - Firewall Lab

1. **What were the environment and attack set up? What is/are the ultimate goal(s) for this lab?**

   The ultimate goal of this lab is to teach the basics of how to implement a packet filter in linux written in C. Additional goals were to learn about loadable/unloadable kernel modules, how to compile them, and what their restrictions are. This lab requires a solid understanding of IP packet constructions as well as the C libraries netfilter.h, ip.h, tcp.h, and udp.h. This lab goes one step further than a lab on iptables would have because it forces us to look through the ip header information and determine how the information is stored in the packet.

   Useful overall steps: (<filename> indicates the name of the file without an extension)
   1. To build the kernal object (.ko) file:
      a. make ARGS="<filename>"
   2. To install the compiled .ko file:
      a. sudo insmod <filename>.ko
   3. To remove the installed .ko file:
      a. sudo rmmod <filename>.ko
   4. To view the result:
      a. tail /var/log/syslog
   5. Optional - Install missing packages if needed:
      a. su -- seedubuntu
      b. apt-get install manpages-dev manpages-posix manpages-posix-dev

2. **What were the steps that you take in order to launch the attack? (Note: Make sure your include the shell commands, GDB debugger commands and screenshots of your computer to demonstrate it.)**

   Task 1 (Modify the dropAllPackets.c to drop only WEB traffic):
   a. Copy dropAllPackest.c to dropAllWebPackets.c
   b. Edit the hook_func of dropAllWebPackets.c to:
      i. Confirm/Verify the socket buffer (skb) object is not null
      ii. Populate the ip header object (iph) using the socket buffer object (skb)
      iii. Confirm/Verify the ip header object (iph) is not null
      iv. If the ip header object protocol (iph->protocol) is not 0 and is TCP:
         1. populate the tcp header object (tcph)
         2. If the tcp header object source (tcph->source) is the http port for web traffic (80), then drop the packet
      v. If the ip header object protocol (iph->protocol) is not 0 and is UDP:
         1. populate the udp header object (udph)
         2. If the upd header object source (udph->source) is the http port for web traffic (80), then return NF_DROP to drop the packet
      vi. Otherwise return NF_ACCEPT to accept the packet
   c. Things to note:
      i. http_port "\x00\x50" evaluates to port 80 which is the port for web traffic
      ii. iph->protocol == IPPROTO_TCP checks if the protocol is tcp
      iii. iph->protocol == 17 checks if the protocol is upd

d. Excerpt of the hook_func code:

```
struct iphdr * iph;
struct tcphdr * tcph;
struct udphdr * udph;
unsigned char * http_port = "\x00\x50";

// Drop all WEB packets
if(skb)
{
        iph = ip_hdr(skb);

        if (iph && iph->protocol && (iph->protocol == IPPROTO_TCP))
        {
                tcph = (struct tcphdr *)((__u32 *)iph + iph->ihl);
                if((tcph->source) == *(unsigned short *)http_port)
                {
                        // Write the log to /var/log/syslog
                        printk(KERN_INFO "dropAllWebPackets.c -- hook_func()
                dropped tcp web packets\n");
                        return NF_DROP;
                }
        }
        else if (iph && iph->protocol && (iph->protocol == 17))
        {
                udph = (struct udphdr *)((__u32 *)iph + iph->ihl);
                if((udph->source) == *(unsigned short *)http_port)
                {
                        // Write the log to /var/log/syslog
                         printk(KERN_INFO "dropAllWebPackets.c -- hook_func()
                dropped udp web packets\n");
                        return NF_DROP;
                }
        }
 }
 return NF_ACCEPT;
```

e. This can be simplified in a few ways:
    i. iph->protocol does not need to be checked if it is 0 because we check if it is a specific value
    ii. Once the type of protocol is determined (tcp or udp), the value of the source can be extracted.This means:
        1. A tcphdr or udphdr object does not need to be populated directly
        2. Checking if the source matches the http_port can be done once outside the protocol type checks instead of inside both if/else if statements

f. Excerpt of updated code:

```
struct iphdr * iph;
unsigned char * http_port = "\x00\x50";
```

```
__u16 hdr_src = 0;

// Drop all WEB packets
if(skb)
{
        iph = ip_hdr(skb);

        if (iph && iph->protocol && (iph->protocol == IPPROTO_TCP))
        {
                hdr_src = ((struct tcphdr *)((__u32 *)iph + iph->ihl))->source;
        }
        else if (iph && iph->protocol && (iph->protocol == UDP_PROTOCOL))
        {
                hdr_src = ((struct udphdr *)((__u32 *)iph + iph->ihl))->source;
        }
        if(hdr_src == *(unsigned short *)http_port)
        {
                // Write the log to /var/log/syslog
                printk(KERN_INFO "dropAllWebPackets.c -- hook_func() dropped
                web packets\n");
                return NF_DROP;
        }
}
return NF_ACCEPT;
```

g. To test:
   i. Verify the web traffic flows:
      1. Open Firefox and go to www.google.com and/or www.bing.com
      2. Pages should resolve
   ii. Build the kernel object
      1. make ARGS=dropAllWebPackets
   iii. Install the compiled .ko object
      1. sudo insmod dropAllWebPackets.ko
   iv. Verify the web traffic does not flow:
      1. Open Firefox and go to www.google.com and/or www.bing.com
      2. Pages should not resolve
   v. View the log to confirm drop messages are printed
      1. tail /var/log/syslog
   vi. Remove the installed .ko object
      1. sudo rmmod dropAllWebPackets.ko
   vii. Verify the web traffic flows:
      1. Open Firefox and go to www.google.com and/or www.bing.com
      2. Pages should resolve
   viii. View the log to confirm drop messages are no longer printed
      1. tail /var/log/syslog

Task 2 (Modify the dropAllPackets.c to drop only WEB traffic from a specific domain or ip range, e.g., google.com):

a. Copy dropAllPackest.c to dropGooglePackets.c
b. Edit the hook_func of dropGooglePackets.c to:
    i. Confirm/Verify the socket buffer (skb) object is not null
    ii. Populate the ip header object (iph) using the socket buffer object (skb)
    iii. Confirm/Verify the ip header object (iph) is not null
    iv. To determine the source address to be blocked:
        1. Create a printk statement to print the ip header source address (iph->saddr)
        2. In the web browser, go to www.google.com
        3. Check/tail the /var/log/syslog to see the source address
c. Excerpt of hook_func code:

```
struct iphdr * iph;

// Drop all Google packets
if(skb)
{
        iph = ip_hdr(skb);
        if (iph)
        {
                // Write the saddr to /var/log/syslog
                printk(KERN_INFO "dropGooglePackets.c -- hook_func() Got a
                packet at address: %u\n", iph->saddr);
        }
}
return NF_ACCEPT;
```

d. Again edit the hook_func of dropGooglePackets.c to:
    i. Check the source address of the packet against the value of the address obtained in the previous steps.
    ii. If the source address matches the packet source address, drop the packet
    iii. Otherwise return NF_ACCEPT to accept the packet
e. Excerpt of the code:

```
struct iphdr * iph;
__u32 src_addr = 4041506122;

// Drop all Google packets
if(skb)
{
        iph = ip_hdr(skb);

        if(iph)
        {
                if(iph->saddr == src_addr))
                {
                        // Write the saddr to /var/log/syslog
                printk(KERN_INFO "dropGooglePackets.c -- hook_func()
            dropped Google packet");
                return NF_DROP;
            }
        }
    }
    return NF_ACCEPT;
```

f. To test:
   i. Verify the web traffic flows:
      1. Open Firefox and go to www.google.com and www.bing.com
      2. Pages should resolve
   ii. Build the kernel object
      1. make ARGS=dropGooglePackets
   iii. Install the compiled .ko object
      1. sudo insmod dropGooglePackets.ko
   iv. Verify the web traffic does not flow for Google, but does for Bing:
      1. Open Firefox and go to www.google.com
      2. Page should not resolve
      3. Open www.bing.com in Firefox
      4. Page should resolve
   v. View the log to confirm drop messages are printed for Google, but not Bing
      1. tail /var/log/syslog
   vi. Remove the installed .ko object
      1. sudo rmmod dropGooglePackets.ko
   vii. Verify the web traffic flows:
      1. Open Firefox and go to www.google.com
      2. Page should resolve
      3. Open www.bing.com in Firefox
      4. Page should resolve
   viii. View the log to confirm drop messages are no longer printed at all
      1. tail /var/log/syslog

3. **What have you learned from this lab? Make at least 3 bullets.**
   - Packets can be filtered in the kernel using many different pieces of information stored in the ip header including protocol type(TCP/UDP/ICMP/etc), port (src/dest), and address (src/dest).
   - This lab provided us with a much more in depth understanding of what firewalls are doing in the background and how they apply their filtering rules.
   - We learned about loadable/unloadable kernel modules, the commands used to install/remove them, and how to compile them. In addition, we learned that certain packages and libraries are not able to be used in kernel modules.