Linux Firewall Exploration Lab

57117213 张曙

Task 1: Using Firewall

机器A:

```
[09/16/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:f8:72:b5
          inet addr:192.168.1.106  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::41b:e85b:5e5:7984/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1480 errors:0 dropped:0 overruns:0 frame:0
          TX packets:709 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1788295 (1.7 MB)  TX bytes:73149 (73.1 KB)
```

IP地址为`192.168.1.106`。

机器B:

```
Wireless LAN adapter WLAN:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::dc52:c508:87cc:fab6%18
   IPv4 Address. . . . . . . . . . . : 192.168.1.103
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

IP地址为`192.168.1.103`。

阻止A对B发起`telnet`

```
[09/16/20]seed@VM:~$ sudo ufw reject out telnet
Rule added
Rule added (v6)
[09/16/20]seed@VM:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 23/tcp                     REJECT OUT  Anywhere                   (out)
[ 2] 23/tcp (v6)                REJECT OUT  Anywhere (v6)              (out)
```

检验:

```
[09/16/20]seed@VM:~$ telnet 192.168.1.103
Trying 192.168.1.103...
telnet: Unable to connect to remote host: Connection refused
```

阻止B对A发起`telnet`

```
[09/16/20]seed@VM:~$ sudo ufw reject in telnet
Rule added
Rule added (v6)
[09/16/20]seed@VM:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 23/tcp                     REJECT IN   Anywhere
[ 2] 23/tcp (v6)                REJECT IN   Anywhere (v6)
```

检验：

```
 evian@EVIAN张的XPS  ~  telnet 192.168.1.106
Connecting To 192.168.1.106 ... Could not open connection to the host, on port 23: Connect failed
```

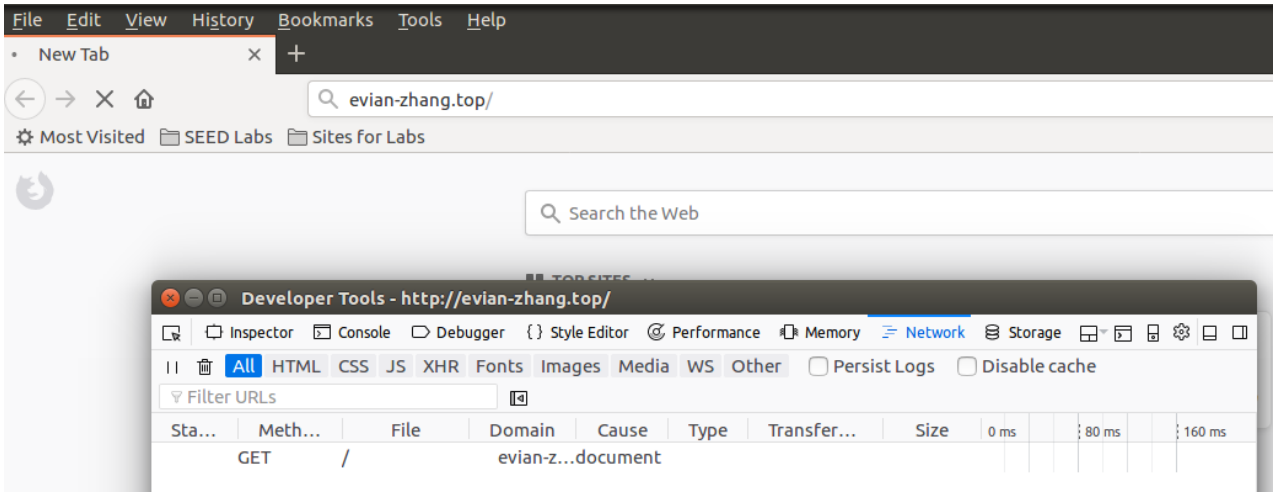
阻止A访问特定的外部网页

这里选择为我的个人网站`evian-zhang.top`，IP地址为`47.100.175.77`。

```
[09/16/20]seed@VM:~$ sudo ufw deny out proto tcp to 47.100.175.77 port 80
Rule added
[09/16/20]seed@VM:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 47.100.175.77 80/tcp       DENY OUT    Anywhere                   (out)
```

检验：



一直无法加载，说明拦截成功。

Task 2: Implementing a Single Firewall

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

```c
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/socket.h>

unsigned int outTelnetFilter(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state) {
    struct iphdr *iph = ip_hdr(skb);
    struct tcphdr *tcph = (void *)iph + iph->ihl * 4;

    char ip_src[16];
    snprintf(ip_src, 16, "%pI4", &iph->saddr);

    if (strcmp(ip_src, "192.168.1.106") == 0 && iph->protocol == IPPROTO_TCP &&
tcph->dest == htons(23)) {
        printk("DROP out telnet.\n");
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}

unsigned int inTelnetFilter(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state) {
    struct iphdr *iph = ip_hdr(skb);
    struct tcphdr *tcph = (void *)iph + iph->ihl * 4;

    char ip_dst[16];
    snprintf(ip_dst, 16, "%pI4", &iph->daddr);

    if (strcmp(ip_dst, "192.168.1.106") == 0 && iph->protocol == IPPROTO_TCP &&
tcph->dest == htons(23)) {
        printk("DROP in telnet.\n");
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}

unsigned int inSshFilter(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state) {
    struct iphdr *iph = ip_hdr(skb);
    struct tcphdr *tcph = (void *)iph + iph->ihl * 4;

    char ip_dst[16];
    snprintf(ip_dst, 16, "%pI4", &iph->daddr);
```

```
    if (strcmp(ip_dst, "192.168.1.106") == 0 && iph->protocol == IPPROTO_TCP &&
tcph->dest == htons(22)) {
        printk("DROP in ssh.\n");
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}

unsigned int evianzhangFilter(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state) {
    struct iphdr *iph = ip_hdr(skb);
    struct tcphdr *tcph = (void *)iph + iph->ihl * 4;

    char ip_src[16];
    snprintf(ip_src, 16, "%pI4", &iph->daddr);;
    if (strcmp(ip_src, "47.100.175.77") == 0 && iph->protocol == IPPROTO_TCP &&
tcph->dest == htons(80)) {
        printk("DROP connection to 47.100.175.77:80.\n");
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}

unsigned int evianzhangHttpsFilter(void *priv, struct sk_buff *skb, const
struct nf_hook_state *state) {
    struct iphdr *iph = ip_hdr(skb);
    struct tcphdr *tcph = (void *)iph + iph->ihl * 4;

    char ip_src[16];
    snprintf(ip_src, 16, "%pI4", &iph->daddr);;
    if (strcmp(ip_src, "47.100.175.77") == 0 && iph->protocol == IPPROTO_TCP &&
tcph->dest == htons(443)) {
        printk("DROP connection to 47.100.175.77:443.\n");
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}


struct nf_hook_ops inTelnetHook;
struct nf_hook_ops outTelnetHook;
struct nf_hook_ops inSshHook;
struct nf_hook_ops evianzhangHook;
struct nf_hook_ops evianzhangHttpsHook;

static int kmodule_init(void) {
```

```
    inTelnetHook.hook = inTelnetFilter;
    inTelnetHook.hooknum = NF_INET_POST_ROUTING;
    inTelnetHook.pf = PF_INET;
    inTelnetHook.priority = NF_IP_PRI_FIRST;

    outTelnetHook.hook = outTelnetFilter;
    outTelnetHook.hooknum = NF_INET_POST_ROUTING;
    outTelnetHook.pf = PF_INET;
    outTelnetHook.priority = NF_IP_PRI_FIRST;

    inSshHook.hook = inSshFilter;
    inSshHook.hooknum = NF_INET_POST_ROUTING;
    inSshHook.pf = PF_INET;
    inSshHook.priority = NF_IP_PRI_FIRST;

    evianzhangHook.hook = evianzhangFilter;
    evianzhangHook.hooknum = NF_INET_POST_ROUTING;
    evianzhangHook.pf = PF_INET;
    evianzhangHook.priority = NF_IP_PRI_FIRST;

    evianzhangHttpsHook.hook = evianzhangHttpsFilter;
    evianzhangHttpsHook.hooknum = NF_INET_POST_ROUTING;
    evianzhangHttpsHook.pf = PF_INET;
    evianzhangHttpsHook.priority = NF_IP_PRI_FIRST;

    nf_register_hook(&inTelnetHook);
    nf_register_hook(&outTelnetHook);
    nf_register_hook(&inSshHook);
    nf_register_hook(&evianzhangHook);
    nf_register_hook(&evianzhangHttpsHook);
    return 0;
}

static void kmodule_exit(void) {
    nf_unregister_hook(&inTelnetHook);
    nf_unregister_hook(&outTelnetHook);
    nf_unregister_hook(&inSshHook);
    nf_unregister_hook(&evianzhangHook);
    nf_unregister_hook(&evianzhangHttpsHook);
}

module_init(kmodule_init);
module_exit(kmodule_exit);

MODULE_LICENSE("GPL");
```

代码如上，实现的功能有：

- 禁止本机向外部发起`telnet`

- 禁止外部向本机发起`telnet`
- 禁止外部向本机发起`ssh`
- 禁止本机访问`http://evain-zhang.top`
- 禁止本机访问`https://evian-zhang.top`

将模块编译好后载入内核，执行上述几个操作，结果与Task 1类似，均被阻断。

此时用`dmesg`查看内核输出：

```
[ 6978.892592] DROP out telnet.
[ 6987.084550] DROP out telnet.
[ 7099.965414] DROP connection to 47.100.175.77:80.
[ 7100.290985] DROP connection to 47.100.175.77:80.
[ 7100.426051] DROP connection to 47.100.175.77:80.
[ 7100.976653] DROP connection to 47.100.175.77:80.
[ 7101.146663] DROP connection to 47.100.175.77:80.
[ 7101.300429] DROP connection to 47.100.175.77:80.
```

（输出特别多，因为TCP包特别多，这里只截取一小部分）

确实被NetFilter阻断了。

Task 3: Evading Egress Filtering

这里由于需要三台虚拟机，所以我就不再用Task 1和2中VM+本机的方式来模拟了，而是采用三个Docker容器。（并且恰好做到这里的时候SEED虚拟机的桥接网卡功能又崩了。。🌚）

容器A：

```
root@99cb3c611dc1:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 10275  bytes 15258522 (15.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4516  bytes 248300 (248.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

容器名为`99cb3c611dc1`，IP地址为`172.17.0.2`。模拟VM A，也就是在内部网络中的机器。

容器B：

```
root@7268104a1106:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
        RX packets 10298  bytes 15259580 (15.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4179  bytes 230082 (230.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

容器名为`7268104a1106`，IP地址为`172.17.0.3`。模拟VM B，也就是在防火墙之外的机器。

容器C：

```
root@03902d86edbb:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.4  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:04  txqueuelen 0  (Ethernet)
        RX packets 10443  bytes 15268103 (15.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4404  bytes 243909 (243.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

容器名为`03902d86edbb`，IP地址为`172.17.0.4`。模拟VM C，也就是`telnet`服务器。

在容器A中按题目要求，阻断了所有向外部`telnet`服务器的请求，也阻断了对`47.100.175.77`（我的个人服务器）的web访问请求：

```
root@99cb3c611dc1:/# ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 23/tcp                     REJECT OUT  Anywhere                   (out)
[ 2] 47.100.175.77 80/tcp       REJECT OUT  Anywhere                   (out)
[ 3] 23/tcp (v6)                REJECT OUT  Anywhere (v6)              (out)
```

Task 3.1: Telnet to Machine B through the firewall

在容器A中向容器B发起SSH请求，以B为跳板访问容器C：

```
root@99cb3c611dc1:/# ssh -4 -L 8000:172.17.0.4:23 root@172.17.0.3
root@172.17.0.3's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 4.19.128-microsoft-standard x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Thu Sep 17 11:26:17 2020 from 172.17.0.2
root@7268104a1106:~#
```

此时另外再在容器A中开一个shell（也可以在上一步SSH连接时直接选择静默模式），对自身的`8000`端口发起`telnet`连接：

```
root@99cb3c611dc1:/# telnet 0.0.0.0 8000
Trying 0.0.0.0...
Connected to 0.0.0.0.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
03902d86edbb login: root
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 4.19.128-microsoft-standard x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@03902d86edbb:~#
```

成功连接到了容器C。

在连接之前，在容器B中架起`tcpdump`可以观察到连接的情况：

```
11:38:16.145961 IP 172.17.0.2.38076 > 7268104a1106.ssh: Flags [P.], seq 2349278130:2349278166, ack 901609179, win 501, options [nop,nop,TS val 2322610070 ecr 2402205927], length 36
11:38:16.146583 IP 7268104a1106.58718 > 172.17.0.4.telnet: Flags [P.], seq 3554564641:3554564642, ack 3746443947, win 501, options [nop,nop,TS val 1361577597 ecr 3424326367], length 1
11:38:16.147723 IP 172.17.0.4.telnet > 7268104a1106.58718: Flags [P.], seq 1:2, ack 1, win 510, options [nop,nop,TS val 3424381308 ecr 1361577597], length 1
11:38:16.147846 IP 7268104a1106.58718 > 172.17.0.4.telnet: Flags [.], ack 2, win 501, options [nop,nop,TS val 1361577598 ecr 3424381308], length 0
11:38:16.148062 IP 7268104a1106.50878 > 192.168.65.1.domain: 48627+ PTR? 2.0.17.172.in-addr.arpa. (41)
11:38:16.148096 IP 7268104a1106.ssh > 172.17.0.2.38076: Flags [P.], seq 1:37, ack 36, win 501, options [nop,nop,TS val 2402260869 ecr 2322610070], length 36
```

当在容器A中输入命令时，会先通过SSH将命令发送给容器B，然后容器B再将命令发送给容器C的Telnet，容器C的Telnet返回给容器B之后，容器B再把结果返回给容器A。

Task 3.b: Connect to Facebook using SSH Tunnel

这里选择访问我的个人主页`http://evian-zhang.top`。

在容器A中使用

```
ssh -D 9000 -C root@172.17.0.3
```

然后再在容器A的另一个Shell连接中，使用`9000`端口作为Socksv5的代理，请求`http://evian-zhang.top`：

```
root@99cb3c611dc1:/# curl --socks5 "http://localhost:9000" http://evian-zhang.top
<html>
<head><title>302 Found</title></head>
<body>
<center><h1>302 Found</h1></center>
<hr><center>nginx/1.17.10 (Ubuntu)</center>
</body>
</html>
```

访问成功（这里我服务器端会把向80端口的请求重定向到443端口，所以直接用`curl`得到的是302）。

此时容器B的`tcpdump`中显示

```
11:49:07.903425 IP 172.17.0.2.38090 > 7268104a1106.ssh: Flags [P.], seq 84:200, ack 37, win 501, options [nop,nop,TS
 val 2323261828 ecr 2402912624], length 116
11:49:07.903513 IP 7268104a1106.39434 > 47.100.175.77.http: Flags [P.], seq 1:80, ack 1, win 502, length 79: HTTP: G
ET / HTTP/1.1
11:49:07.904662 IP 47.100.175.77.http > 7268104a1106.39434: Flags [.], ack 80, win 65535, length 0
11:49:07.929660 IP 47.100.175.77.http > 7268104a1106.39434: Flags [P.], seq 1:365, ack 80, win 65535, length 364: HT
TP: HTTP/1.1 302 Moved Temporarily
11:49:07.929691 IP 7268104a1106.39434 > 47.100.175.77.http: Flags [.], ack 365, win 501, length 0
11:49:07.929883 IP 7268104a1106.ssh > 172.17.0.2.38090: Flags [P.], seq 37:297, ack 200, win 501, options [nop,nop,T
S val 2402912650 ecr 2323261828], length 260
```

和上一个类似，A容器通过SSH向B容器传输请求，B容器访问`47.100.175.77`的80端口，并把数据返回给A容器。

将SSH关闭后，再次使用`curl`请求数据发生错误：

```
root@99cb3c611dc1:/# curl --socks5 "http://localhost:9000" http://evian-zhang.top
curl: (7) Failed to connect to localhost port 9000: Connection refused
```

根据抓包，其原理为：A容器与B容器通过SSH建立隧道，A容器通过自己的`9000`端口作为Socksv5代理，由B成功访问了`47.100.175.77`的`80`端口。

Task 4: Evading Ingress Filtering

在这个Task中，机器A由容器A模拟：

```
root@99cb3c611dc1:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 10275  bytes 15258522 (15.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4516  bytes 248300 (248.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

机器B由容器B模拟：

```
root@7268104a1106:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
        RX packets 10298  bytes 15259580 (15.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4179  bytes 230082 (230.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

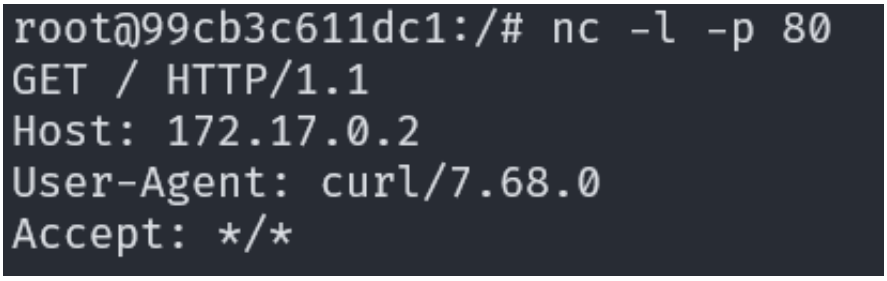
在容器A中阻断外部向内部发起`80`端口和`22`端口的连接，同时用`nc`在80端口监听。

此时，在容器A中使用

```
ssh -fCNR 172.17.0.3:2333:172.17.0.2:2334 root@172.17.0.3
```

对容器B的`2333`端口发起反向SSH隧道。

在容器B中，向自身的`2333`端口发出TCP连接请求，在容器A中可观察到



说明容器A虽然阻断了外部访问80端口的请求，但容器B通过建立的反向SSH隧道，成功访问到了A内部的80端口。