

计网 LAB7 实验报告

姓名：徐佳美

学号：181860117

专业：计算机科学与技术系

任课老师：李文中

邮箱：181860117@smail.nju.edu.cn

一实验名称

防火墙机制实现

二实验目的

建立防火墙，检查网络中的数据包。

三实验内容和核心代码

Task2: Implement firewall rules

Ruleparse 和 switchForParse 函数，按关键字分别处理对应的规则；

（1）文件解析：

按条目存储相关内容在 ruleTable 中。

①无速率限制和 impair:

按格式存储操作，包名，源 IP 和目的 IP，如果是 TCP 或者 UDP 流，加上源端口和目的端口。

②速率限制:

检测到 ratelimit 关键字，在存储操作，包名，源 IP...等基础上，设置该规则的令牌桶，初始化为 0，其后存储 R/2，和上一次更新令牌桶的时间。

③Impair:

检测到 impair 关键字，在表项中最后一项存入 impair 标记。

例如：

```
elif ('tcp'in line or 'udp' in line) and ('ratelimit' in line):
    (operation, headerName, src, srcnet,
port1,srcport ,dst ,dstnet,port2,dstport,rate,byterate) = line.split(' ')
    for i in range(10): #these are all strings
        if i==0 or i%2==1 :#0,1,2:srcnet 3:srcport 4:dstnet 5:dstport
            entry.append(line.split(' ')[i])
        byterate=int(byterate) # attention,it's int
        entry.append(0) #6: ratelimit
```

```

entry.append(byterate/2) # 7: every 0.5s add rings into bucket
entry.append(time.time()) #8:the add time
return entry

```

(2) 规则匹配:

从表中按顺序进行处理，根据包头类型判断要进行哪些匹配，主要是匹配 IP 和端口，分别用两个函数 IPProc 和 PortMatch 来处理，判断包 IP 是否在子网中，端口名是否一致。注意比较时类型的转换。

```

def ipProc(ipsrc, ipdst, srcnet, dstnet):
    match = True #whether matches the rule
    if srcnet!='any':
        srcnetw = IPv4Network(srcnet, strict=False) #
        match = ipsrc in srcnetw
    if match == False:
        return match
    if dstnet!='any':
        dstnetw = IPv4Network(dstnet, strict=False) #
        match = ipdst in dstnetw
    return match

def portMatch(srcport, dstport, port1, port2):
    if port1!='any':
        if srcport!=int(port1):
            return False
    if port2!='any':
        if dstport!=int(port2):
            return False
    return True

```

(3) 找到匹配表项后:

如果操作为 deny，直接 return;

否则。判断是否有其他标志，如果有速率限制，比较令牌数目和包大小，令牌不够则丢包。如果有 impair 标志，进行 impair 操作 (task4)

例如:

```

elif entry[1]=='tcp' and pkt.has_header(TCP) == True:
    srcport=pkt[TCP].src
    dstport=pkt[TCP].dst
    if tcp_udp_Proc(ipsrc,srcport,ipdst,dstport,
entry[2],entry[3],entry[4],entry[5]) == True:
        if entry[0]=='permit':
            if len(entry)==9: #has ratelimit
                size=len(pkt)

```

```

        if pkt.has_header(Ethernet):
            size=(len(pkt[Ethernet]))
            if size <= entry[6]:
                entry[6]-=size
            else:
                return
            elif len(entry)==7 and entry[6]=='impair':
                payload=pkt[RawPacketContents]
                log_info("before impair,payload={}".format(payload))
                payload=str(len(pkt)) #change payload to the len of
pkt
                payload=RawPacketContents(payload)
                pkt[RawPacketContents]=payload
                log_info("after impair,payload={}".format(payload))
                net.send_packet(portpair[input_port], pkt)
            return

```

Task3: token bucket

在 main 函数的 while 循环中，每次循环都执行一次 addring 函数。
该函数检查 ruleTable 中的表项，是否有速率限制，如有，判断当前时间与上次更新令牌桶时间是否过 0.5s，若是，加入 $R/2$ ，并保证不超过 $2R$ ，同时更新表项中存储的时间。否则，时间不满足 0.5s，则不操作。

```

def addRing(ruleTable):
    size=len(ruleTable)
    for i in range (size):
        if len(ruleTable[i]) == 9: #tcp,udp: has ring bucket,attention the max is
2r
            if time.time()-ruleTable[i][8] >=0.5:
                ruleTable[i][6]+=ruleTable[i][7]
                ruleTable[i][6] = min (ruleTable[i][6], 4*ruleTable[i][7])
                ruleTable[i][8]=time.time()
            elif ruleTable[i][1]=='icmp' and len(ruleTable[i])==7: #icmp limit
                if time.time()-ruleTable[i][6] >=0.5:
                    ruleTable[i][4]+=ruleTable[i][5]
                    ruleTable[i][4] = min (ruleTable[i][4], 4*ruleTable[i][5])
                    ruleTable[i][6]=time.time() #update time

```

Task4: impair

如果在匹配表项中有 impair 标记，将包的负载内容换成包的长度。注意用 rawpacketcontents 时先检查有没有该包头。

```

elif len(entry)==7 and entry[6]=='impair':
    payload=""
    newpayload=str(len(pkt)) #change payload to the len of pkt
    newpayload=RawPacketContents(newpayload)

```

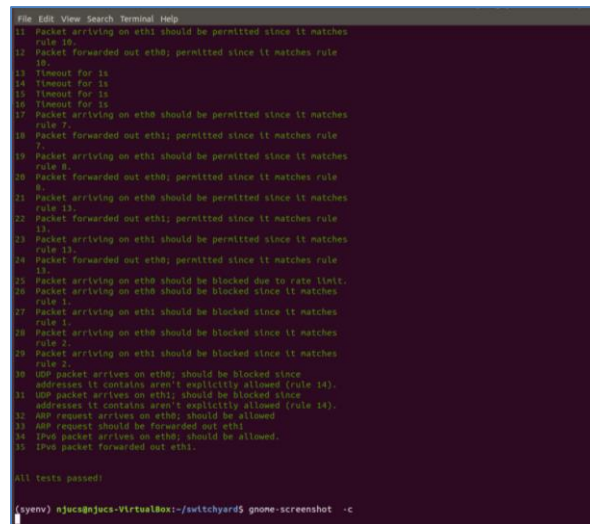
```

if pkt.has_header(RawPacketContents)==False:
    log_info("before impair,payload=" ")
    pkt+=newpayload
else:
    payload=pkt[RawPacketContents]
    log_info("before impair,payload={}".format(payload))
    indexi = pkt.get_header_index(RawPacketContents)
    pkt[indexi]=newpayload
    log_info("after impair,payload={}".format(newpayload))
    net.send_packet(portpair[input_port], pkt)
return

```

Task5: TEST

(1) 整体文件测试结果



```

File Edit View Search Terminal Help
11 Packet arriving on eth0 should be permitted since it matches
    rule 10.
12 Packet forwarded out eth0; permitted since it matches rule
    10.
13 Timeout for 1s
14 Timeout for 1s
15 Timeout for 1s
16 Timeout for 1s
17 Packet arriving on eth0 should be permitted since it matches
    rule 7.
18 Packet forwarded out eth0; permitted since it matches rule
    7.
19 Packet arriving on eth0 should be permitted since it matches
    rule 8.
20 Packet forwarded out eth0; permitted since it matches rule
    8.
21 Packet arriving on eth0 should be permitted since it matches
    rule 11.
22 Packet forwarded out eth0; permitted since it matches rule
    11.
23 Packet arriving on eth0 should be permitted since it matches
    rule 12.
24 Packet forwarded out eth0; permitted since it matches rule
    12.
25 Packet arriving on eth0 should be blocked due to rate limit.
26 Packet arriving on eth0 should be blocked since it matches
    rule 1.
27 Packet arriving on eth0 should be blocked since it matches
    rule 1.
28 Packet arriving on eth0 should be blocked since it matches
    rule 2.
29 Packet arriving on eth0 should be blocked since it matches
    rule 2.
30 ICMP packet arrives on eth0; should be blocked since
    addresses it contains aren't explicitly allowed (rule 14).
31 ICMP packet arrives on eth0; should be blocked since
    addresses it contains aren't explicitly allowed (rule 14).
32 ADD request arrives on eth0; should be allowed
33 ADD request should be forwarded out eth0
34 IPvs packet arrives on eth0; should be allowed.
35 IPvs packet forwarded out eth0.

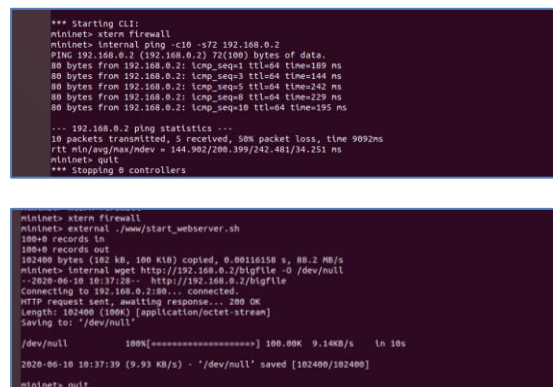
All tests passed!
(syenv) njucs@njucs-VirtualBox:~/switchyard$ gnome-screenshot -c

```

(2) 速率测试

测试 rule 13:令牌桶速率为 150, 最多存 300 个令牌。每个包大小为 100. 因此, 基本上应该是每两个 request 发出去收到一个 reply。

测试 rule 7, 8. 限速是 12.5kB/S, 测试得到的平均速率为 9.93KB/S, 符合情况。



```

*** Starting CLI:
mininet> sterm firewall
mininet> Internal ping -c10 -s72 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 72(100) bytes of data:
 80 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=189 ms
 80 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=144 ms
 80 bytes from 192.168.0.2: icmp_seq=5 ttl=64 time=242 ms
 80 bytes from 192.168.0.2: icmp_seq=8 ttl=64 time=223 ms
 80 bytes from 192.168.0.2: icmp_seq=10 ttl=64 time=195 ms

--- 192.168.0.2 ping statistics ---
10 packets transmitted, 5 received, 50% packet loss, time 9092ms
rtt min/avg/max/mdev = 144.902/208.399/242.481/34.251 ms
mininet> quit
*** Stopping 0 controllers

mininet> sterm firewall
mininet> external ./www/start_webserver.sh
100+0 records out
102400 bytes (102 kB, 100 KiB) copied, 0.00111158 s, 88.2 MB/s
mininet> Internal wget http://192.168.0.2/bigfile -O /dev/null
--2020-06-10 10:37:28-- http://192.168.0.2/bigfile
Connecting to 192.168.0.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 102400 (100K) [application/octet-stream]
Saving to: '/dev/null'

/dev/null 100%[=====] 100.00K 9.14KB/s in 10s

2020-06-10 10:37:39 (9.93 KB/s) - '/dev/null' saved [102400/102400]
mininet> quit

```

(3) impair 测试

修改的是有效负载，每次匹配到 impair，打印出修改前的 payload 和修改后的 payload。这里是把负载内容改成了包的长度。

```
"Node: firewall"
19:58:34 2020/06/10 INFO after impair,payload=RawPacketContents (2 bytes) b'
54'
19:58:34 2020/06/10 INFO before impair,payload=''
19:58:34 2020/06/10 INFO after impair,payload=RawPacketContents (2 bytes) b'
54'
19:58:34 2020/06/10 INFO before impair,payload=RawPacketContents (147 bytes)
b' /bigfile '...
19:58:34 2020/06/10 INFO after impair,payload=RawPacketContents (3 bytes) b'
201'
19:58:35 2020/06/10 INFO before impair,payload=RawPacketContents (147 bytes)
b' /bigfile '...
19:58:35 2020/06/10 INFO after impair,payload=RawPacketContents (3 bytes) b'
201'
19:58:36 2020/06/10 INFO before impair,payload=RawPacketContents (147 bytes)
b' /bigfile '...
19:58:36 2020/06/10 INFO after impair,payload=RawPacketContents (3 bytes) b'
201'
19:58:40 2020/06/10 INFO before impair,payload=RawPacketContents (147 bytes)
```