网络地址转换实验

范子墨
2021K8009929006

一、 实验内容

1、SNAT 实验

a) 运行给定网络拓扑(nat_topo.py)
b) 在 n1, h1, h2, h3 上运行相应脚本
   i. n1: disable_arp.sh, disable_icmp.sh, disable_ip_forward.sh, disable_ipv6.sh
   ii. h1-h3: disable_offloading.sh, disable_ipv6.sh
c) 在 n1 上运行 nat 程序： n1# ./nat exp1.conf
d) 在 h3 上运行 HTTP 服务：h3# python3 ./http_server.py
e) 在 h1, h2 上分别访问 h3 的 HTTP 服务
   i. h1# wget http://159.226.39.123:8000
   ii. h2# wget http://159.226.39.123:8000

2、DNAT 实验

f) 运行给定网络拓扑(nat_topo.py)
g) 在 n1, h1, h2, h3 上运行相应脚本
   i. n1: disable_arp.sh, disable_icmp.sh, disable_ip_forward.sh, disable_ipv6.sh
   ii. h1-h3: disable_offloading.sh, disable_ipv6.sh
h) 在 n1 上运行 nat 程序： n1# ./nat exp2.conf
i) 在 h1, h2 上分别运行 HTTP Server： h1/h2# python3 ./http_server.py
j) 在 h3 上分别请求 h1, h2 页面
   i. h3# wget http://159.226.39.43:8000
   ii. h3# wget http://159.226.39.43:8001

3、手动构造一个包含两个 nat 的拓扑

k) h1 <-> n1 <-> n2 <-> h2
l) 节点 n1 作为 SNAT， n2 作为 DNAT，主机 h2 提供 HTTP 服务，主机 h1 穿过两个 nat 连接到 h2 并获取相应页面

二、 设计思路

1、 总体设计思路

首先依据原有代码，可以看到在 main 函数中引出两个本次实验相关的函数，一个是 nat_init，一个是 ustack_run。

在 ustack_run 中会持续接收数据包，如果是 ip 数据包，则会跳转到 handle_ip_packet 函数中处理。在 handle_ip_packet 函数中，如果不是目的地址为本接口的 ICMP 包的话，均会跳转至 nat_translate_packet 函数处理。在 nat_translate_packet 函数中首先判断数据包的方向，而后如果是无效方向则发送 icmp，如果协议不是 TCP 则记录错误日志，最后进行网络地址转换，即跳转到 do_translation 函数中。在该函数中，如果该数据包在 NAT 中有对应连接映射，则直接处理即可。若该数据包的方向为 DIR_OUT，且为该 TCP 连接的第一个数据包（请求连接数据包），NAT 中没有对应连接映射 (SNAT)，则分配端口，创建新的映射项。若该数据包的方向为 DIR_IN，为该 TCP 连接的第一个数据包，NAT 中没有对应连接映射，但有对应处理规则 (DNAT)，则新建映射项。最终将处理好的数据包发送。

在 nat_init 的函数中，会在 parse_config 函数中对配置文件进行解析，同时新创一个线

程，在 nat_timeout 中定时清理已完成的 NAT 映射。

2、 get_packet_direction

由于 NAT 只处理两个方向的数据包，即当源地址为内部地址，且目的地址为外部地址时，方向为 DIR_OUT 和当源地址为外部地址，且目的地址为 external_iface 地址时，方向为 DIR_IN。由于只有两个方向，因此可以简化判断条件位源地址为外部地址或内部地址。

图表 1：数据包方向

首先获取 ip 头部，而后对源地址进行最长前缀查找，即在路由表中查询，如果目的地址相应转发条目对应的 iface 与内部接口相同的话，则方向为 DIR_OUT，与外部接口相同则为 DIR_IN。

3、 do_translation

首先提取 ip 头部和 tcp 头部，而后根据数据包方向获取端口号和 ip 地址。

```
u32 addr = (dir == DIR_IN)? ntohl(iphdr->saddr) : ntohl(iphdr->daddr);
 u16 port = (dir == DIR_IN)? ntohs(tcphdr->sport) : ntohs(tcphdr->dport);
```

由于 NAT 可能需要同时维护数万条映射关系，链表查找方式效率很低，因此考虑使用 hash 方式。在原代码中也提供了一种 hash 函数，使用该 hash 函数对 ip 和 port 进行映射。

```
u8 hash = hash8((char*)&rs, sizeof(rmt_set_t));
```

而后，如果方向是 DIR_IN 的话，对 NAT 映射表进行遍历，如果存在外部 ip 地址和端口符合该数据包目的地址 ip 地址和端口的，则设置 found=1。

如果 found=1，则表示可以直接利用该映射，修改 TCP 头部的目的端口及 IP 头部的目 IP 地址，并更新映射表中的序列号，和连接信息的 ACK 和 FIN 标志，最后更新时间。

如果 found=0，则遍历 DNAT 规则表，检查 rule 对应的外部端口是否被分配，rule 的外部 IP 地址是否与 ip 头部的目标地址匹配且 rule 的外部端口是否与 tcp 头部的目的端口匹配。若满足条件，则按照规则创造新的映射项，并将映射项最终添加到映射表的链表中。最后与 found=1 类似。

```
if (nat.assigned_ports[rule->external_port] == 0 && rule->external_ip == ntohl(iphdr->daddr) \
        && rule->external_port == ntohs(tcphdr->dport))
```

若数据包方向为 DIR_OUT，则同样先寻找到是否有匹配的映射项。若 found=1，则同样更新映射项的信息，包括外部序列号、ACK、FIN 标志等。

如果未找到匹配的映射项，说明是新的连接，需要分配一个外部端口，并创建新的映射项。遍历 NAT_PORT_MIN 到 NAT_PORT_MAX 的端口号，找到第一个未被分配的端口。如果找到可用端口，将其标记为已分配，并将新的映射项的信息设置为新分配的端口和输出方向数据包的源 IP 和源端口。而后也更新映射表的信息。

最终更新校验和后发送数据包。

4、 parse_config

首先查看要解析的配置文件都大概是什么样子的，可以看到首先规定了 internal-iface 和 external-iface，在实验二的配置文件中还包含目标网络地址转换的规则。首先获取内部接口和外部接口，只需读取而后赋值即可。而对于 DNAT 的规则，其形如“dnat-rules: 159.226.39.43:8000 -> 10.21.0.1:8000”，因此按照空格分为四段，这里需要将字符串形式的 ip 地址转换为 32 位整数。在最开始尝试使用函数 inet_addr 或者 inet_aton 但是都出现了问题

（比如说 core_dump），最后决定还是使用最原始的方法将字符串转换为整数。

```
    sscanf(name, "%u.%u.%u.%u", &ip4, &ip3, &ip2, &ip1);
    ip = (ip4 << 24) | (ip3 << 16) | (ip2 << 8) | (ip1);
```

5、 nat_timeout

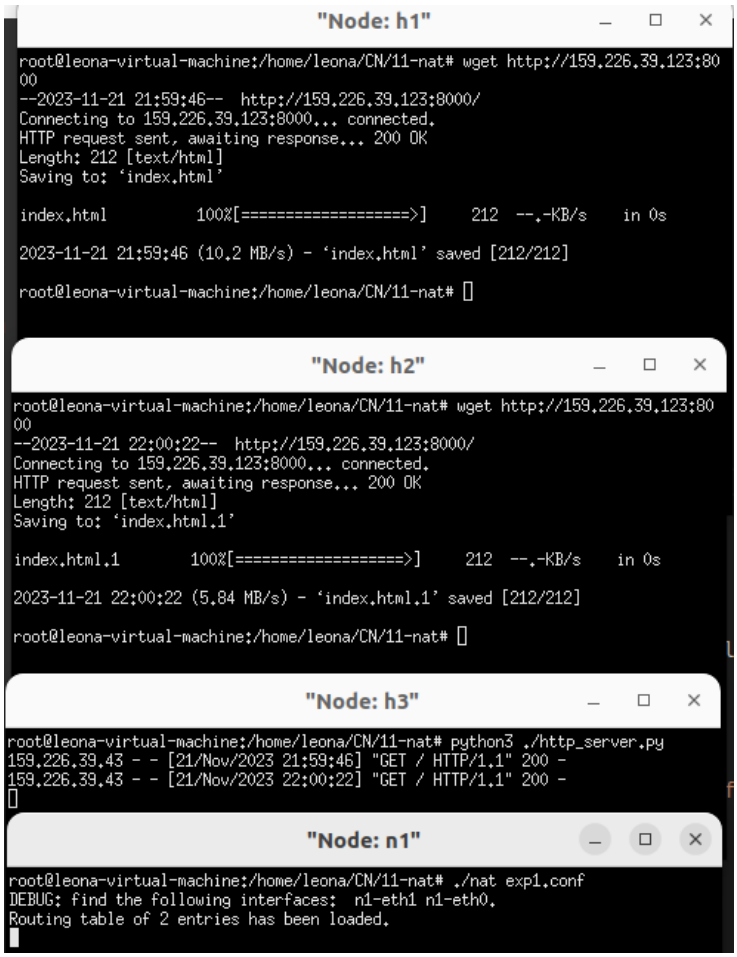
相比于之前实验中的老化操作还更为简单一些，只包含有两种情况，首先是双方都已发送 FIN 且回复相应 ACK 的连接，一方发送 RST 包的连接，可以回收。另一种是双方已经超过 60 秒未传输数据的连接，认为其已经传输结束，可以回收。因此遍历 NAT 映射表的每个槽中的每个映射项，若老化或传输已完成，则收回端口。

```
if (now - cur->update_time > TCP_ESTABLISHED_TIMEOUT || is_flow_finished(&(cur->conn))) {
        nat.assigned_ports[cur->external_port] = 0;
        list_delete_entry(&(cur->list));
        free(cur);}
```

三、 结果

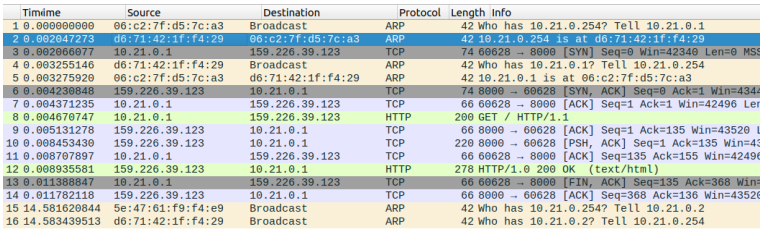
1、实验内容 1

如图在 n1 上运行 nat 程序，h3 上运行 HTTP 服务，在 h1 和 h3 上分别访问 h3 的 HTTP 服务



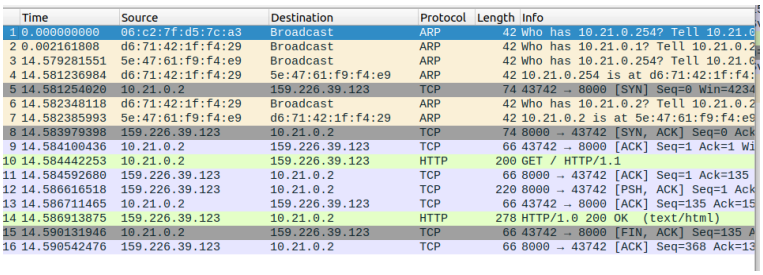
图表 2：实验一终端

抓包结果如下，可以看到 TCP 连接正常建立到关闭的过程。



	Timime	Source	Destination	Protocol	Length	Info
1	0.000000000	06:c2:7f:d5:7c:a3	Broadcast	ARP	42	Who has 10.21.0.254? Tell 10.21.0.1
2	0.002047273	d6:71:42:1f:f4:29	06:c2:7f:d5:7c:a3	ARP	42	10.21.0.254 is at d6:71:42:1f:f4:29
3	0.002066077	10.21.0.1	159.226.39.123	TCP	74	60628 → 8000 [SYN] Seq=0 Win=42340 Len=0 MSS
4	0.003255146	d6:71:42:1f:f4:29	Broadcast	ARP	42	Who has 10.21.0.1? Tell 10.21.0.254
5	0.003275920	06:c2:7f:d5:7c:a3	d6:71:42:1f:f4:29	ARP	42	10.21.0.1 is at 06:c2:7f:d5:7c:a3
6	0.004230848	159.226.39.123	10.21.0.1	TCP	74	8000 → 60628 [SYN, ACK] Seq=0 Ack=1 Win=4344
7	0.004371235	10.21.0.1	159.226.39.123	TCP	66	60628 → 8000 [ACK] Seq=1 Ack=1 Win=42496 Len
8	0.004670747	10.21.0.1	159.226.39.123	HTTP	200	GET / HTTP/1.1
9	0.005131278	159.226.39.123	10.21.0.1	TCP	66	8000 → 60628 [ACK] Seq=1 Ack=135 Win=43520 L
10	0.008453430	159.226.39.123	10.21.0.1	TCP	220	8000 → 60628 [PSH, ACK] Seq=1 Ack=135 Win=43
11	0.008707897	10.21.0.1	159.226.39.123	TCP	66	60628 → 8000 [ACK] Seq=135 Ack=155 Win=42496
12	0.008935581	159.226.39.123	10.21.0.1	HTTP	278	HTTP/1.0 200 OK (text/html)
13	0.011388847	10.21.0.1	159.226.39.123	TCP	66	60628 → 8000 [FIN, ACK] Seq=135 Ack=368 Win=
14	0.011782118	159.226.39.123	10.21.0.1	TCP	66	8000 → 60628 [ACK] Seq=368 Ack=136 Win=43520
15	14.581620844	5e:47:61:f9:f4:e9	Broadcast	ARP	42	Who has 10.21.0.254? Tell 10.21.0.2
16	14.583439513	d6:71:42:1f:f4:29	Broadcast	ARP	42	Who has 10.21.0.2? Tell 10.21.0.254

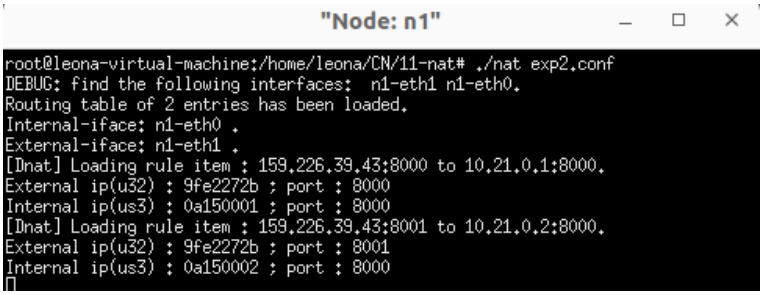
图表 3：h1 访问 h3 抓包



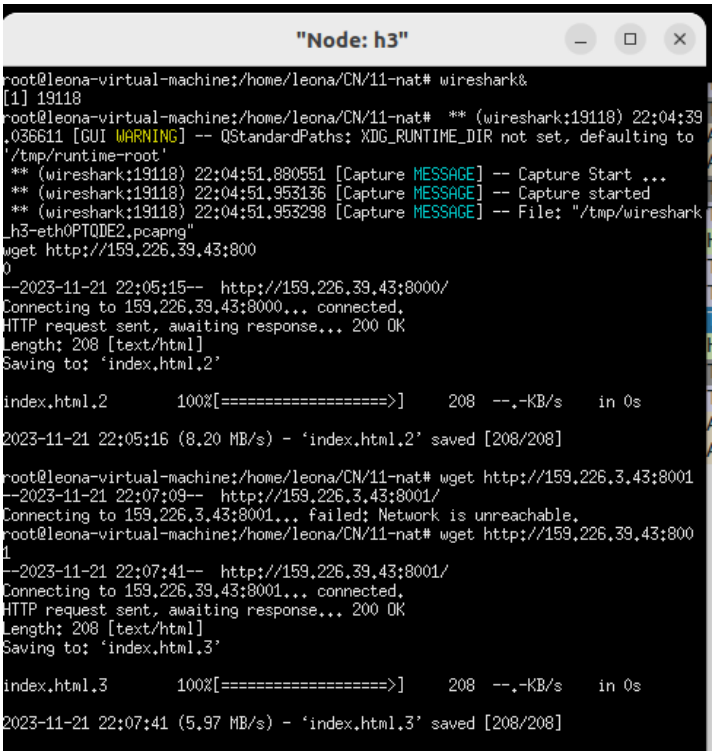
	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	06:c2:7f:d5:7c:a3	Broadcast	ARP	42	Who has 10.21.0.254? Tell 10.21.0
2	0.002161808	d6:71:42:1f:f4:29	Broadcast	ARP	42	Who has 10.21.0.1? Tell 10.21.0.2
3	14.579281551	5e:47:61:f9:f4:e9	Broadcast	ARP	42	Who has 10.21.0.254? Tell 10.21.0
4	14.581236984	d6:71:42:1f:f4:29	5e:47:61:f9:f4:e9	ARP	42	10.21.0.254 is at d6:71:42:1f:f4:
5	14.581254020	10.21.0.2	159.226.39.123	TCP	74	43742 → 8000 [SYN] Seq=0 Win=4234
6	14.582348118	d6:71:42:1f:f4:29	Broadcast	ARP	42	Who has 10.21.0.2? Tell 10.21.0.2
7	14.582385993	5e:47:61:f9:f4:e9	d6:71:42:1f:f4:29	ARP	42	10.21.0.2 is at 5e:47:61:f9:f4:e9
8	14.583979398	159.226.39.123	10.21.0.2	TCP	74	8000 → 43742 [SYN, ACK] Seq=0 Ack
9	14.584100436	10.21.0.2	159.226.39.123	TCP	66	43742 → 8000 [ACK] Seq=1 Ack=1 Wi
10	14.584442253	10.21.0.2	159.226.39.123	HTTP	200	GET / HTTP/1.1
11	14.584592680	159.226.39.123	10.21.0.2	TCP	66	8000 → 43742 [ACK] Seq=1 Ack=135
12	14.586616518	159.226.39.123	10.21.0.2	TCP	220	8000 → 43742 [PSH, ACK] Seq=1 Ack
13	14.586711465	10.21.0.2	159.226.39.123	TCP	66	43742 → 8000 [ACK] Seq=135 Ack=15
14	14.586913875	159.226.39.123	10.21.0.2	HTTP	278	HTTP/1.0 200 OK (text/html)
15	14.590131946	10.21.0.2	159.226.39.123	TCP	66	43742 → 8000 [FIN, ACK] Seq=135 A
16	14.590542476	159.226.39.123	10.21.0.2	TCP	66	8000 → 43742 [ACK] Seq=368 Ack=13

图表 4：h2 访问 h3 抓包

2、实验内容 2

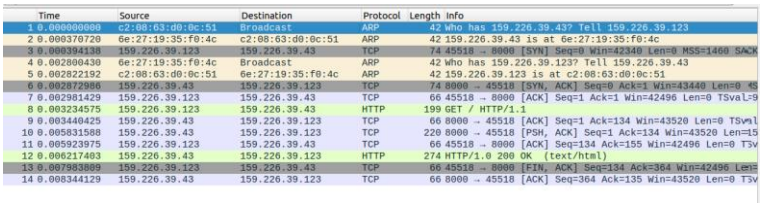


图表 5：n1 读取规则成功



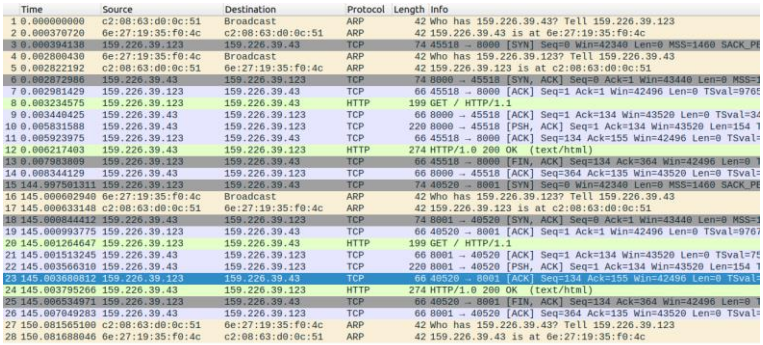
图表 6：h3 请求 h1，h2 页面

以下可以看到 TCP 连接正常建立到关闭的过程。



	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	c2:08:63:d0:0c:51	Broadcast	ARP	42	Who has 159.226.39.43? Tell 159.226.39.123
2	0.000370720	6e:27:19:35:f0:4c	c2:08:63:d0:0c:51	ARP	42	159.226.39.43 is at 6e:27:19:35:f0:4c
3	0.000394138	159.226.39.123	159.226.39.43	TCP	74	45518 → 8000 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK
4	0.002800430	6e:27:19:35:f0:4c	Broadcast	ARP	42	Who has 159.226.39.123? Tell 159.226.39.43
5	0.002822192	c2:08:63:d0:0c:51	6e:27:19:35:f0:4c	ARP	42	159.226.39.123 is at c2:08:63:d0:0c:51
6	0.002872986	159.226.39.43	159.226.39.123	TCP	74	8000 → 45518 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=0 45
7	0.002981429	159.226.39.123	159.226.39.43	TCP	66	45518 → 8000 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSval=9
8	0.003234575	159.226.39.123	159.226.39.43	HTTP	199	GET / HTTP/1.1
9	0.003440425	159.226.39.43	159.226.39.123	TCP	66	8000 → 45518 [ACK] Seq=1 Ack=134 Win=43520 Len=0 TSval
10	0.005831588	159.226.39.43	159.226.39.123	TCP	220	8000 → 45518 [PSH, ACK] Seq=1 Ack=134 Win=43520 Len=15
11	0.005923975	159.226.39.123	159.226.39.43	TCP	66	45518 → 8000 [ACK] Seq=134 Ack=155 Win=42496 Len=0 TSv
12	0.006217403	159.226.39.43	159.226.39.123	HTTP	274	HTTP/1.0 200 OK (text/html)
13	0.007983809	159.226.39.123	159.226.39.43	TCP	66	45518 → 8000 [FIN, ACK] Seq=134 Ack=364 Win=42496 Len=
14	0.008344129	159.226.39.43	159.226.39.123	TCP	66	8000 → 45518 [ACK] Seq=364 Ack=135 Win=43520 Len=0 TSv

图表 7：h1 抓包



	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	c2:08:63:d0:0c:51	Broadcast	ARP	42	Who has 159.226.39.43? Tell 159.226.39.123
2	0.000370720	6e:27:19:35:f0:4c	c2:08:63:d0:0c:51	ARP	42	159.226.39.43 is at 6e:27:19:35:f0:4c
3	0.000394138	159.226.39.123	159.226.39.43	TCP	74	45518 → 8000 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PE
4	0.002800430	6e:27:19:35:f0:4c	Broadcast	ARP	42	Who has 159.226.39.123? Tell 159.226.39.43
5	0.002822192	c2:08:63:d0:0c:51	6e:27:19:35:f0:4c	ARP	42	159.226.39.123 is at c2:08:63:d0:0c:51
6	0.002872986	159.226.39.43	159.226.39.123	TCP	74	8000 → 45518 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=0 MSS=
7	0.002981429	159.226.39.123	159.226.39.43	TCP	66	45518 → 8000 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSval=9765
8	0.003234575	159.226.39.123	159.226.39.43	HTTP	199	GET / HTTP/1.1
9	0.003440425	159.226.39.43	159.226.39.123	TCP	66	8000 → 45518 [ACK] Seq=1 Ack=134 Win=43520 Len=0 TSval=34
10	0.005831588	159.226.39.43	159.226.39.123	TCP	220	8000 → 45518 [PSH, ACK] Seq=1 Ack=134 Win=43520 Len=154 T
11	0.005923975	159.226.39.123	159.226.39.43	TCP	66	45518 → 8000 [ACK] Seq=134 Ack=155 Win=42496 Len=0 TSval=
12	0.006217403	159.226.39.43	159.226.39.123	HTTP	274	HTTP/1.0 200 OK (text/html)
13	0.007983809	159.226.39.123	159.226.39.43	TCP	66	45518 → 8000 [FIN, ACK] Seq=134 Ack=364 Win=42496 Len=0
14	0.008344129	159.226.39.43	159.226.39.123	TCP	66	8000 → 45518 [ACK] Seq=364 Ack=135 Win=43520 Len=0 TSval=
15	144.997501311	159.226.39.123	159.226.39.43	TCP	74	40520 → 8001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PE
16	145.000602940	6e:27:19:35:f0:4c	Broadcast	ARP	42	Who has 159.226.39.123? Tell 159.226.39.43
17	145.000633148	c2:08:63:d0:0c:51	6e:27:19:35:f0:4c	ARP	42	159.226.39.123 is at c2:08:63:d0:0c:51
18	145.000844412	159.226.39.43	159.226.39.123	TCP	74	8001 → 40520 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=0 MSS=
19	145.000993775	159.226.39.123	159.226.39.43	TCP	66	40520 → 8001 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSval=9767
20	145.001264647	159.226.39.123	159.226.39.43	HTTP	199	GET / HTTP/1.1
21	145.001513245	159.226.39.43	159.226.39.123	TCP	66	8001 → 40520 [ACK] Seq=1 Ack=134 Win=43520 Len=0 TSval=75
22	145.003566310	159.226.39.43	159.226.39.123	TCP	220	8001 → 40520 [PSH, ACK] Seq=1 Ack=134 Win=43520 Len=154 T
23	145.003698812	159.226.39.123	159.226.39.43	TCP	66	40520 → 8001 [ACK] Seq=134 Ack=155 Win=42496 Len=0 TSval=
24	145.003795266	159.226.39.43	159.226.39.123	HTTP	274	HTTP/1.0 200 OK (text/html)
25	145.006534971	159.226.39.123	159.226.39.43	TCP	66	40520 → 8001 [FIN, ACK] Seq=134 Ack=364 Win=42496 Len=0
26	145.007049283	159.226.39.43	159.226.39.123	TCP	66	8001 → 40520 [ACK] Seq=364 Ack=135 Win=43520 Len=0 TSval=
27	150.081565100	c2:08:63:d0:0c:51	6e:27:19:35:f0:4c	ARP	42	Who has 159.226.39.43? Tell 159.226.39.123
28	150.081688046	6e:27:19:35:f0:4c	c2:08:63:d0:0c:51	ARP	42	159.226.39.43 is at 6e:27:19:35:f0:4c

图表 8：h2 抓包

3、更改拓扑文件如下

```
    def build(self):
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        n1 = self.addHost('n1')
        n2 = self.addHost('n2')

        self.addLink(h1, n1)
        self.addLink(h2, n2)
        self.addLink(n1, n2)

if __name__ == '__main__':
    topo = NATTopo()
    net = Mininet(topo = topo, switch = OVSBridge, controller = None)

    h1, h2, n1, n2 = net.get('h1', 'h2', 'n1', 'n2')

    h1.cmd('ifconfig h1-eth0 10.21.0.1/16')
    h1.cmd('route add default gw 10.21.0.254')

    h2.cmd('ifconfig h2-eth0 10.21.0.2/16')
    h2.cmd('route add default gw 10.21.0.254')

    n1.cmd('ifconfig n1-eth0 10.21.0.254/16')
    n1.cmd('ifconfig n1-eth1 159.226.39.11/24')

    n2.cmd('ifconfig n2-eth0 10.21.0.254/16')
    n2.cmd('ifconfig n2-eth1 159.226.39.22/24')
```

设置两个 NAT 节点的配置文件分别为

```
internal-iface: n1-eth0
external-iface: n1-eth1
dnat-rules: 159.226.39.11:8001 -> 10.21.0.1:8000
```

```
internal-iface: n2-eth0
external-iface: n2-eth1
dnat-rules: 159.226.39.22:8001 -> 10.21.0.2:8000
```

而后进行访问，可以看到能够正常连接。


root@leona-virtual-machine:/home/leona/CN/11-nat# wget http://159.226.39.22:800
1
--2023-11-23 15:48:00-- http://159.226.39.22:8001/
Connecting to 159.226.39.22:8001... connected.
HTTP request sent, awaiting response... 200 OK
Length: 207 [text/html]
Saving to: 'index.html.13'
index.html.13   100%[==================>]   207 --.-KB/s   in 0s
2023-11-23 15:48:00 (5.51 MB/s) - 'index.html.13' saved [207/207]

图表 9：h1 访问 h2