网络地址转换实验

孙佳钰 2015K8009929051

2018 年 11 月 22 日

1 实验内容

NAT(Network Address Translation) 主要工作为维护私网地址/端口与公网地址/端口的映射关系，和对数据包进行重写。本次实验需要实现：

- 确认数据包的发送方向，然后根据方向对 ip 等进行重写。
- NAT 表的老化功能，将满足一定条件的连接释放，回收端口号。

2 实验流程

由于代码太多，故报告中没有加入完整代码，完整代码可见附件。

2.1 数据包方向确认与重写

1. 根据源 ip 地址查询路由表，根据查询到的 iface 确定方向：

```
// determine the direction of the packet, DIR_IN / DIR_OUT / DIR_INVALID
static int get_packet_direction(char *packet)
{
    struct iphdr *ip = packet_to_ip_hdr(packet);
    u32 saddr = ntohl(ip->saddr);
    rt_entry_t *rt = longest_prefix_match(saddr);
    iface_info_t *iface = rt->iface;
    if (iface->index == nat.internal_iface->index) {
        return DIR_OUT;
    } else if (iface->index == nat.external_iface->index) {
        return DIR_IN;
    }
    return DIR_INVALID;
}
```

2. 在重写过程中，由于两个方向的数据包中服务器地址与端口都是不变的，所以可以根据这一特性，将 < 内部地址，端口 > 到 < 外部地址，端口 > 的映射进行哈希储存。

```
struct iphdr *ip = packet_to_ip_hdr(packet);
u32 addr = (dir == DIR_IN)? ntohl(ip->saddr) : ntohl(ip->daddr);
u8 hash_i = hash8((char*)&addr, 4);
struct list_head *head = &(nat.nat_mapping_list[hash_i]);
struct nat_mapping *mapping_entry = NULL;
struct tcphdr *tcp = packet_to_tcp_hdr(packet);
```

3. 然后根据数据包方向分别处理，如果是从公网到私网，则不存在找不到映射的问题；如果是从私网到公网，可能会找不到映射，此时则需要从端口号池中分配一个端口号，并将映射加入相应链表中。同时要更新连接状态的结构和更新时间，方便老化操作。

```
if (dir == DIR_IN) {
    list_for_each_entry(mapping_entry, head, list) {
        if (mapping_entry->external_ip == ntohl(ip->daddr) && \
        mapping_entry->external_port == ntohs(tcp->dport)) break;
    }
    tcp->dport = htons(mapping_entry->internal_port);
    ip->daddr = htonl(mapping_entry->internal_ip);
    mapping_entry->conn.external_fin = (tcp->flags == TCP_FIN);
    mapping_entry->conn.external_seq_end = tcp->seq;
    if (tcp->flags == TCP_ACK) mapping_entry->conn.external_ack = tcp->ack;
} else {
    int found = 0;
    list_for_each_entry(mapping_entry, head, list) {
        if (mapping_entry->internal_ip == ntohl(ip->saddr) && \
            mapping_entry->internal_port == ntohs(tcp->sport)) {
            found = 1;
            break;
        }
    }
    if (!found) {
        struct nat_mapping *new_entry = (struct nat_mapping*) \
            malloc(sizeof(struct nat_mapping));
        memset(new_entry, 0, sizeof(struct nat_mapping));
        new_entry->internal_ip = ntohl(ip->saddr);
        new_entry->external_ip = nat.external_iface->ip;
        new_entry->internal_port = ntohs(tcp->sport);
```

```
27          new_entry->external_port = assign_external_port();
28          list_insert(&(new_entry->list), &(mapping_entry->list), \
29              mapping_entry->list.next);
30          mapping_entry = new_entry;
31      }
32      tcp->sport = htons(mapping_entry->external_port);
33      ip->saddr = htonl(mapping_entry->external_ip);
34      mapping_entry->conn.internal_fin = (tcp->flags == TCP_FIN);
35      mapping_entry->conn.internal_seq_end = tcp->seq;
36      if (tcp->flags == TCP_ACK) mapping_entry->conn.internal_ack = tcp->ack;
37  }
38  mapping_entry->update_time = time(NULL);
```

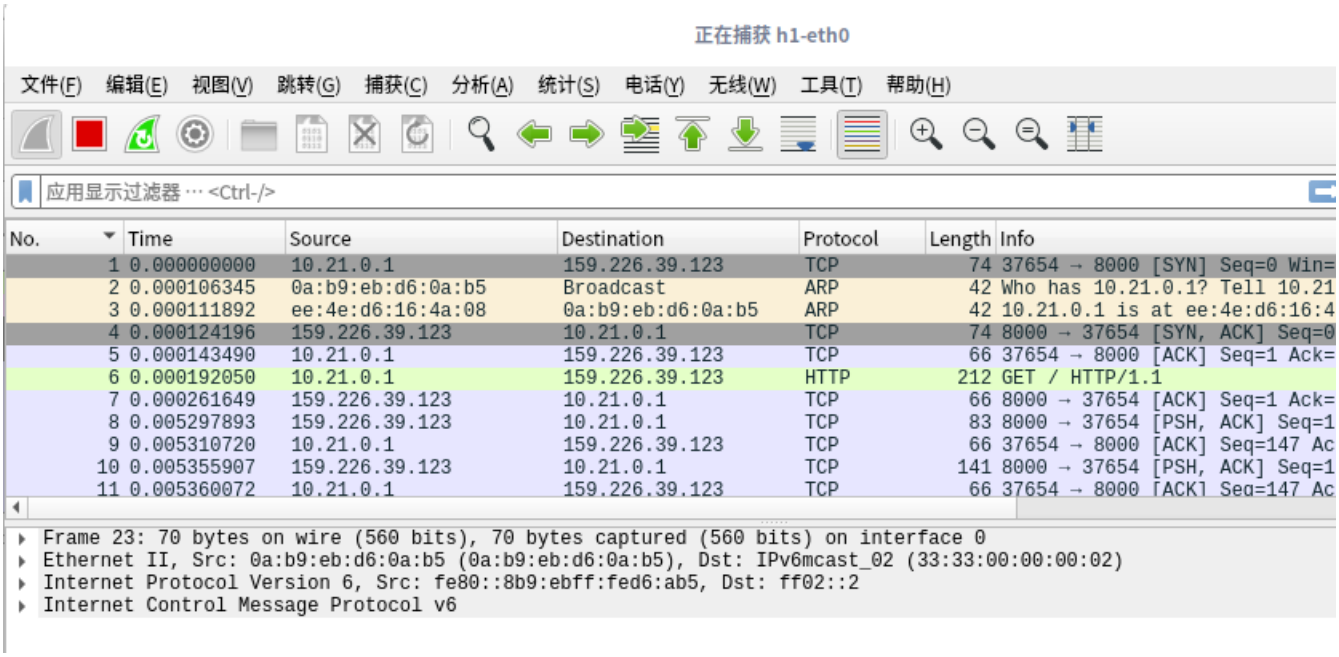
2.2 NAT 老化操作

老化操作中实现了对超过 60 秒没有连接和双方都已发送 FIN 且回复相应 ACK 操作的连接进行删除表项，回收端口号的操作。

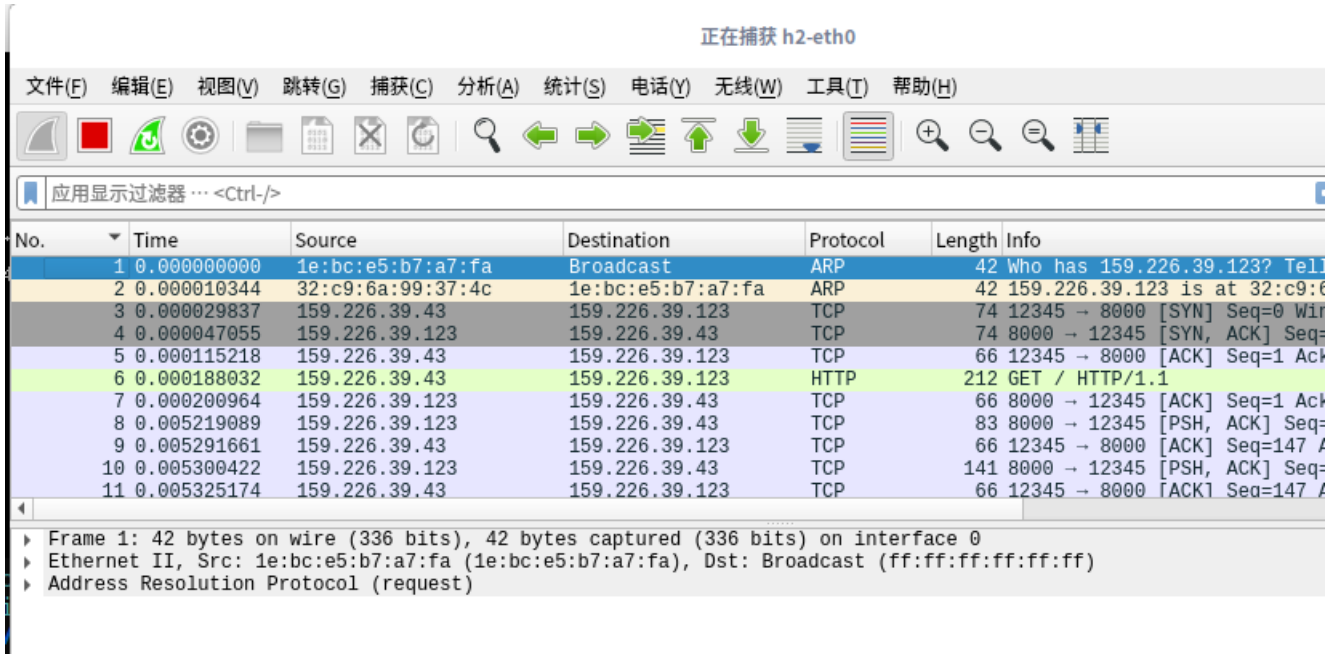
```
1   struct nat_mapping *mapping, *q;
2   list_for_each_entry_safe(mapping, q, head, list) {
3       if (now - mapping->update_time > TCP_ESTABLISHED_TIMEOUT) {
4           nat.assigned_ports[mapping->external_port] = 0;
5           list_delete_entry(&mapping->list);
6           free(mapping);
7           continue;
8       }
9       if (mapping->conn.internal_fin && mapping->conn.external_fin && \
10          mapping->conn.internal_ack == mapping->conn.external_seq_end + 1 && \
11          mapping->conn.external_ack == mapping->conn.internal_seq_end + 1) {
12          nat.assigned_ports[mapping->external_port] = 0;
13          list_delete_entry(&mapping->list);
14          free(mapping);
15      }
16  }
```

3 实验结果及分析

上图是 h1 的 xterm 中显示 wget 成功界面。



上图是对 h1 的端口进行抓包显示从私网地址到服务器地址的请求与从服务器地址到私网地址的返回。



上图是对 h2 的端口进行抓包显示从转换后的公网地址到服务器地址的请求与从服务器地址到公网地址的返回。