计算机网络实验报告-10

阮星程
2015K8009929047

一、 实验题目

网络地址转换实验。

二、 实验内容

理解网络地址转换的内容和原理，构建地址转换，并在简单实验环境中验证网络地址转换的效果。

三、 实验过程

首先构建方向查找程序。

```
// determine the direction of the packet, DIR_IN / DIR_OUT / DIR_INVALID
static int get_packet_direction(char *packet)
{
    //log(DEBUG, "deciding dir");
    struct iphdr *ip = packet_to_ip_hdr(packet);
    //struct tcphdr * tcp = packet_to_tcp_hdr(packet);

    //if it's out packet
    if(longest_prefix_match(ntohl(ip->saddr))->iface->index == nat.internal_iface->index \
        && longest_prefix_match(ntohl(ip->daddr))->iface->index == nat.external_iface->index){
        return DIR_OUT;
    }

    //if it's in packet
    if(longest_prefix_match(ntohl(ip->saddr))->iface->index == nat.external_iface->index \
        && ntohl(ip->daddr) == nat.external_iface->ip){
        return DIR_IN;
    }

    return DIR_INVALID;
}
```

对 ip 头部进行检查，若源地址属于私网地址，目的地址属于外网地址，则为 OUT；若源地址为外网地址，目的地址为路由的外部端口 ip，则为 IN。其余皆为 INVALID。

开始构建转换程序。

```
if(dir == DIR_OUT){
    //if map not exist
    if(!nat.nat_mapping_list[(ntohl(ip->daddr) + ntohs(tcp->dport)) % 256  ] || ((struct nat_mapping *)nat.nat_
        struct nat_mapping * nm = (struct nat_mapping *)malloc(sizeof(struct nat_mapping));
        nat.nat_mapping_list[(ntohl(ip->daddr) + ntohs(tcp->dport)) % 256  ] = (struct list_head *) nm;

        nm->internal_ip = ntohl(ip->saddr);
        nm->internal_port = ntohs(tcp->sport);
        nm->external_ip = nat.external_iface->ip;

        //assign port
        do{
            nm->external_port = ++assigned_port;
            log(DEBUG, "current assigned_port %d state: %d ",assigned_port,nat.assigned_ports[assigned_port]);
        }while(nat.assigned_ports[assigned_port] == 1);
        nat.assigned_ports[assigned_port] = 1;

        nm->update_time = time(NULL);
        nm->conn.internal_ack = 0;
        nm->conn.internal_fin = 0;
        nm->conn.internal_seq_end = 0;
        nm->conn.external_ack = 0;
        nm->conn.external_fin = 0;
        nm->conn.external_seq_end = 0;
    }
```

如果映射还没有建立的话，那么建立对应映射，并进行初始化。这里我们使用的 hash 表的映射方式是将目的 ip 在加上目的 port 然后模去 256。在实验中实际上不需要考虑多个终端访问同一个目的地址访问同一端口的情况，为了方便实现，我稍稍修改了下 nat_table 的数据结构，将里面的 list 换成了指向 list_head 的指针数组。如果要解决之前提到的情况，只需要改回到 list_head 数组，然后用对应的 list 操作即可。

```
//update ip&tcp head
ip->saddr = htonl(nm->external_ip);
tcp->sport = htons(nm->external_port);
ip->checksum = ip_checksum(ip);
tcp->checksum = tcp_checksum(ip, tcp);


if(tcp->flags & TCP_RST){
    nat.assigned_ports[nm->external_port] = 0;
    free(nm);
    ip_send_packet(packet, len);
    return;
}

if(tcp->flags & TCP_ACK){
    nm->conn.internal_ack = ntohl(tcp->ack);
}

if(tcp->flags & TCP_FIN){
    nm->conn.internal_fin = 1;
}

nm->update_time = time(NULL);
nm->conn.internal_seq_end = ntohl(tcp->seq);
```

之后开始进行 ip 地址和端口转换，然后再更新下对应的连接状态即可。

在这里 fin 时没有检查并决定是否释放映射，我把它放到 time_out 里面统一处理了。

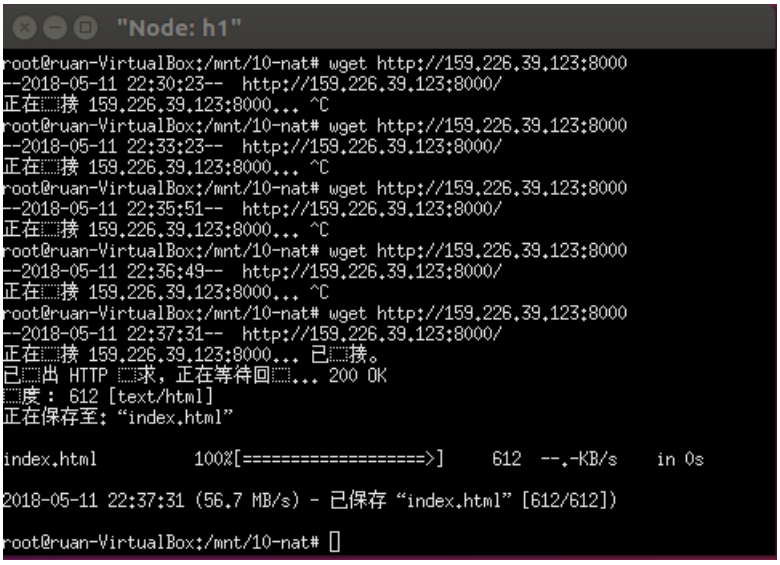
IN 的情况与之类似，不再展示，最后 ip 发包即可。

构建 time_out 函数。

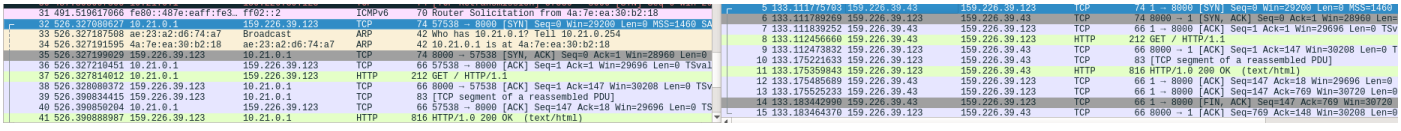
```
for(i = 0; i < 256; i++){
    //if nat map exist
    if(nat.nat_mapping_list[i]){
        //if nat map out of time
        if(time(NULL) - ((struct nat_mapping *)nat.nat_mapping_list[i])->update_time > 60){
            nat.assigned_ports[ ((struct nat_mapping *)nat.nat_mapping_list[i])->external_port] = 0;
            free((struct nat_mapping *)nat.nat_mapping_list[i]);
            nat.nat_mapping_list[i] = NULL;
        }
        //if nat tcp fin
        else if(((struct nat_mapping *)nat.nat_mapping_list[i])->conn.internal_fin && ((struct nat_mapping *)
            nat.assigned_ports[ ((struct nat_mapping *)nat.nat_mapping_list[i])->external_port] = 0;
            free((struct nat_mapping *)nat.nat_mapping_list[i]);
            nat.nat_mapping_list[i] = NULL;
        }
    }
}
```

先检查超时的 map entry，移除，检查都结束了的连接，移除，便结束了。

四、 实验结果



Wget 结果见上图，成功。



抓包结果见上图，符合预期。

五、 实验总结

本次实验实现较易，代码量较小，很快便完成了，出于预先对地址转换的了解，以及之前实验的积累，解决了一个实验无关的 BUG 过后便得到了正确的结果。还是很轻松愉快的。