

实验六 VPN 设计, 实现与分析

161130118 尹浚宇

实验目的

了解 vpn 的原理
了解 vpn 各种有关协议的细节
学会设计一个简单的 vpn 机制

数据结构说明

本次实验自定义的数据结构如下:

```
struct route_item
{
    char destination[16]; //目的子网
    char gateway[16]; //网关
    char netmask[16]; //子网掩码
    char interface[16]; //网卡名称
} route_table[MAX_ROUTE_SIZE];
int route_item_nr = 0;
```

route_table 对应于主机中的路由表, 其中每一项对应于一个路由表项的简化版, 因为这里只需要保证实现静态路由机制即可. route_item_nr 记录了路由表的表项数量. 该表在程序中的作用有, 查询一个目的 ip 与本机是否处于同一子网, 查询下一跳 ip, 确定转发的源端口号.

```
struct arp_item
{
    char ip_addr[16]; //ip 地址
    char mac_addr[18]; //mac 地址
} arp_table[MAX_ARP_SIZE];
int arp_item_nr = 0;
```

arp_table 对应于主机中的 arp 表, 该表在程序中的作用有, 根据确定好的下一跳 ip 地址确定目的 mac 地址.

```
struct device_item
{
    char interface[14]; //网卡名称
    char mac_addr[18]; //mac 地址
    char ip_addr[16]; //ip 地址
    char netmask[16]; //子网掩码
    int is_entrance; //vpn 入口标识
} device_table[MAX_DEVICE_SIZE];
int device_item_nr = 0;
```

device_table 对应于主机中的设备表, 其表项对应于 ifconfig 命令给出的参数, 在程序中的作用有, 根据源端口号确定源 ip 地址和源 mac 地址. 这里新加了一个参数标识 vpn 的入口, 如

果 is_entrance 为 1, 说明该端口为子网的 vpn 接入端口, 如果 is_entrance 为 0, 说明该端口是和外部网络的接入端口.

```
struct vpn_item
{
    char dst_ip[16]; // vpn 隧道的终点 ip 地址
    char src_ip[16]; // vpn 隧道的起点 ip 地址
} vpn_info;
```

在内部网络的数据包到达 vpnserver 要添加新 ip 包头时使用, 用于填充新包头的源 ip 和目的 ip 字段.

本次实验用到的系统库提供的结构体如下:

```
struct ip
{
    u_int ip_v:4; //版本
    u_int ip_hl:4; //报头长度
    u_char ip_tos; //服务类型
    u_short ip_len; //ip 包长度
    u_short ip_id; //标识号
    u_short ip_off; //分片标志
    u_char ip_ttl; //寿命
    u_char ip_p; //上层协议
    u_short ip_sum; //ip 头检验和
    struct in_addr ip_src; //源 ip 地址
    struct in_addr ip_dst; //目的 ip 地址
};
```

该结构体对应于 ip 头结构, 用于解析和填充以太网帧的 ip 部分.

```
struct icmp
{
    u_char icmp_type; //报文类型
    u_char icmp_code; //报文类型子码
    u_short icmp_cksum; //icmp 头检验和
    u_short icmp_id; //标识号
    u_short icmp_seq; //顺序号
    char icmp_data[1]; //icmpdata
};
```

该结构体对应于 icmp 头结构, 用于解析和填充以太网帧的 icmp 部分.

```
struct sockaddr_ll
{
    unsigned short int sll_family; /* 一般为 AF_PACKET */
    unsigned short int sll_protocol; /* 上层协议 */
    int sll_ifindex; /* 接口类型 */
};
```

```
    unsigned short int sll_hatype; /* 报头类型 */
    unsigned char sll_pkttype; /* 包类型 */
    unsigned char sll_halen; /* 地址长度 */
    unsigned char sll_addr[8]; /* MAC 地址 */
};
```

因为本次实验中需要发送和接受的都是以太网帧，所以这里使用该结构体配合 sendto 函数从而实现在数据链路层收发包的功能。

配置文件说明

配置文件存放在 config 文件夹中，不同的 vpn server 有不同的配置文件，配置文件的读取已经硬编码到程序中，只需要将放有配置文件的文件夹放入可执行程序的同目录下，并且将文件夹更名为 config，运行程序后即可读取。

程序设计思路及运行流程

设计思路:

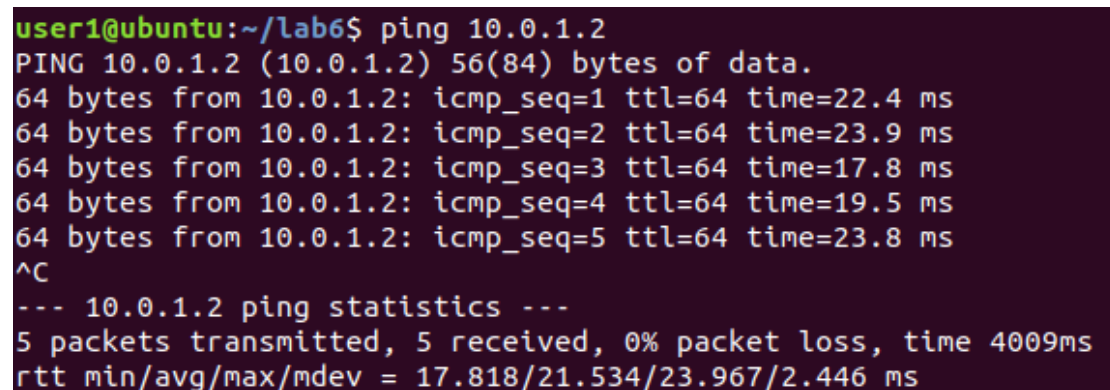
本实验要实现的简易 vpn 的原理大致为，在 vpn server 上将内部网络的数据包作为数据字段封装在一个新的 ip 包里，这里我们需要做的就是添加新的 ip 包头并转发该数据包。对于外部网络的数据包，我们需要解封其数据包，然后再根据解封后的数据包进行转发。这样我们就可以达到在公网上通过虚拟 ip 地址进行通信的目的。

运行流程:

1. 读取配置文件
2. 抓包
3. 抓到包，判断包的方向
 - 3.1 包来自内部，加包头封装
 - 3.2 包来自外部，解封数据包
 - 3.3 其他包，不处理
4. 调用实验 4 实现的转发程序进行转发
5. 回到步骤 2

运行结果截图

ping 结果:



```
user1@ubuntu:~/lab6$ ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=22.4 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=23.9 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=17.8 ms
64 bytes from 10.0.1.2: icmp_seq=4 ttl=64 time=19.5 ms
64 bytes from 10.0.1.2: icmp_seq=5 ttl=64 time=23.8 ms
^C
--- 10.0.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 17.818/21.534/23.967/2.446 ms
```

vpn1 上运行截图:

```
type: IP (0x0800)
receive a packet from 10.0.0.2 to 10.0.1.2
vpn server repack the packet
ip_next_hop is 192.168.0.1
interface is ens38
dst mac is 00:0c:29:7e:a4:6e
src mac is 00:0c:29:7a:d9:de

type: IP (0x0800)
receive a packet from 192.168.0.2 to 172.0.0.2
no need for processing, drop the packet

type: IP (0x0800)
receive a packet from 172.0.0.2 to 192.168.0.2
vpn server unpack the packet
ip_next_hop is 10.0.0.2
interface is ens33
dst mac is 00:0c:29:cb:88:0d
src mac is 00:0c:29:7a:d9:d4

type: IP (0x0800)
receive a packet from 10.0.1.2 to 10.0.0.2
no need for processing, drop the packet
```

vpn2 上运行截图:

```
type: IP (0x0800)
receive a packet from 192.168.0.2 to 172.0.0.2
vpn server unpack the packet
ip_next_hop is 10.0.1.2
interface is ens38
dst mac is 00:0c:29:c0:cd:38
src mac is 00:0c:29:b5:4f:88

type: IP (0x0800)
receive a packet from 10.0.0.2 to 10.0.1.2
no need for processing, drop the packet

type: IP (0x0800)
receive a packet from 10.0.1.2 to 10.0.0.2
vpn server repack the packet
ip_next_hop is 172.0.0.1
interface is ens33
dst mac is 00:0c:29:7e:a4:78
src mac is 00:0c:29:b5:4f:7e

type: IP (0x0800)
receive a packet from 172.0.0.2 to 192.168.0.2
no need for processing, drop the packet
```

vpn2 上抓包截图:

Time	Source	Destination	Protocol	Length	Info
8	10.0.0.2	10.0.1.2	ICMP	100	Echo (ping) request id=0x0a28, seq=96/24576, ttl=64 (reply in 9)
9	10.0.1.2	10.0.0.2	ICMP	100	Echo (ping) reply id=0x0a28, seq=96/24576, ttl=64 (request in 8)
10	10.0.0.2	192.168.0.2	IPv4	120	Fragmented IP protocol (proto=IPIP 4, off=512, ID=b10a)
11	192.168.0.2	172.0.0.2	IPv4	120	Fragmented IP protocol (proto=IPIP 4, off=512, ID=e20a)
12	10.0.0.2	10.0.1.2	ICMP	100	Echo (ping) request id=0x0a28, seq=97/24832, ttl=64 (reply in 13)
13	10.0.1.2	10.0.0.2	ICMP	100	Echo (ping) reply id=0x0a28, seq=97/24832, ttl=64 (request in 12)
14	10.0.0.2	192.168.0.2	IPv4	120	Fragmented IP protocol (proto=IPIP 4, off=512, ID=b10a)
15	192.168.0.2	172.0.0.2	IPv4	120	Fragmented IP protocol (proto=IPIP 4, off=512, ID=e20a)

关闭 vpn 程序后 ping:

```
user1@ubuntu:~/lab6$ ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
^C
--- 10.0.1.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1016ms
```

通过上述截图可以观察到, 开启 vpn 程序后 pc1 到 pc2 能够通信, 关闭后就无法通信. 在 vpn server 上抓到的在隧道中的包是通过虚拟 ip 进行传输的. 这符合实验的原理.

相关参考资料

网络上关于 ip 头协议字段的详细介绍的一些文章

对比样例程序

无

代码个人创新以及思考

根据数据包里的目的 mac 地址和设备表里的 is_entrance 字段判断包来自内部还是外部, 比较合理. 同时很大程度上复用了实验 4 的代码, 没有对代码进行整体重构.

该程序的应用场景

可用于实现加密的点到点通信