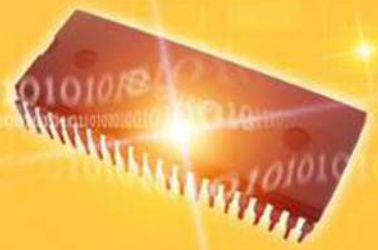
嵌入式系统工程师



网络开发工具包

大纲

- socket原始套接字回顾
- libpcap介绍、安装
- libpcap开发实例
- libpcap练习——网络数据分析器
- Libnet介绍、安装
- Libnet开发实例
- libnet练习——ARP欺骗

大纲

- socket原始套接字回顾
- libpcap介绍、安装
- libpcap开发实例
- libpcap练习——网络数据分析器
- Libnet介绍、安装
- Libnet开发实例
- libnet练习——ARP欺骗

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

socket原始套接字回顾

- 原始套接字
  - 开发者可发送任意的数据包到网上
  - 开发网络攻击等特殊软件
  - 需要开发者手动组织数据、各种协议包头、校验和计算等等
- 创建方法

```
//创建数据链路层的原始套接字
int sock_raw_fd = 0;
sock_raw_fd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
```

大纲

- socket原始套接字回顾
- libpcap介绍、安装
- libpcap开发实例
- libpcap练习——网络数据分析器
- Libnet介绍、安装
- Libnet开发实例
- libnet练习——ARP欺骗

Libpcap介绍

Libpcap概念

- 是一个网络数据捕获开发包
- 平台独立具有强大功能
- 是一套高层的编程接口的集合；其隐藏了操作系统的细节，可以捕获网上的所有,包括到达其他主机的数据包
- 使用非常广泛，几乎只要涉及到网络数据包的捕获的功能，都可以用它开发,如wireshark
- 开发语言为C语言,但是也有基于其它语言的开发包,如Python语言的开发包pycap

Libpcap介绍

- Libpcap主要的作用
  - 捕获各种数据包

    列如：网络流量统计
  - 过滤网络数据包

    列如：过滤掉本地上的一些数据，类似防火墙
  - 分析网络数据包

    列如：分析网络协议，数据的采集
  - 存储网络数据包

    列如：保存捕获的数据以为将来进行分析

Libpcap安装

➢ Libpcap的安装

- apt-get方式

```
edu@edu-T: ~
edu@edu-T:~$ sudo apt-get install libpcap        出错
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
E: 未发现软件包 libpcap
edu@edu-T:~$ sudo apt-get install libpcap-dev    正确
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会安装下列额外的软件包:
  libpcap0.8-dev
下列【新】软件包将被安装:
  libpcap-dev libpcap0.8-dev
升级了 0 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，有 424 个软件包未被升
级。
需要下载 219 kB 的软件包。
解压缩后会消耗掉 548 kB 的额外空间。
您希望继续执行吗？[Y/n]y
获取: 1 http://172.20.220.71/ precise/main libpcap0.8-dev i386 1.1.1-10 [215 kB]
获取: 2 http://172.20.220.71/ precise/main libpcap-dev all 1.1.1-10 [3,396 B]
下载 219 kB，耗时 0秒 (2,452 kB/s)
```

大纲

- socket原始套接字回顾
- libpcap介绍、安装
- libpcap开发实例
- libpcap练习——网络数据分析器
- Libnet介绍、安装
- Libnet开发实例
- libnet练习——ARP欺骗

Libpcap开发实例

- 利用libpcap函数库开发应用程序的基本步骤：
  1. 打开网络设备
  2. 设置过滤规则
  3. 捕获数据
  4. 关闭网络设备
- 捕获网络数据包常用函数
  - pcap_lookupdev( )
  - pcap_open_live( )
  - pcap_lookupnet( )
  - pcap_compile( )、 pcap_setfilter( )
  - pcap_next( )、pcap_loop( )
  - pcap_close( )

Libpcap开发实例

- char *pcap_lookupdev(char *errbuf)
  - 功能：得到可用的网络设备名指针
  - 参数：errbuf：存放相关的错误信息
  - 返回值：成功返回设备名指针；失败返回NULL
  - 例如：

```
53      char *dev = NULL;
54      char err_buf[100] = "";
55      dev = pcap_lookupdev(err_buf);
56      if(NULL == dev)
57      {
58          perror("pcap_lookupdev");
59          exit(-1);
60      }
```

Libpcap开发实例

- pcap_t *pcap_open_live(const char *device,
  int snaplen,
  int promisc,
  int to_ms,
  char *ebuf)

- 功能：打开一个用于捕获数据的网络接口
- 返回值：返回一个Libpcap句柄

Libpcap开发实例

- 参数

  device：网络接口的名字

  snaplen：捕获数据包的长度

  promise：1代表混杂模式,其它非混杂模式

  to_ms：等待时间

  ebuf：存储错误信息

- 例如

```
char error_content[PCAP_ERRBUF_SIZE] = "";
pcap_t * pcap_handle = NULL;
pcap_handle = pcap_open_live("eth0",1024,1,0,error_content);
```

Libpcap开发实例

- int pcap_lookupnet(char *device,
  bpf_u_int32 *netp, bpf_u_int32 *maskp,
  char *errbuf)

- 功能：获得指定网络设备的网络号和掩码
- 参数：device：网络设备名
  netp：存放网络号的指针
  maskp：存放掩码的指针
  errbuf：存放出错信息

Libpcap开发实例

- 返回值：成功返回0，失败返回-1
- 例如：

```
80      unsigned int net_ip;                    //网络地址
81      unsigned int net_mask;                  //子网掩码
82      res = pcap_lookupnet(dev,&net_ip,&net_mask,error_content);
83      if(res == -1)
84      {
85          perror("pcap_loopupnet");
86          exit(-1);
87      }
```

Libpcap开发实例

- int pcap_compile(pcap_t *p,
  struct bpf_program *program,
  char *buf,int optimize,
  bpf_u_int32 mask)

- 功能：编译BPF过滤规则
- 返回值：成功返回0，失败返回-1

Libpcap开发实例

- 参数

  p：Libpcap句柄

  program：bpf过滤规则

  buf：过滤规则字符串

  optimize：优化

  mask：掩码

- 例如

```
52      struct bpf_program bpf_filter;
53      char *bpf_filter_string = "arp or ip";
54      if(pcap_compile(pcap_handle,&bpf_filter,bpf_filter_string,0,0xffffff00) < 0)//编译BPF过滤规则
55      {
56          perror("pcap_compile");
57      }
```

Libpcap开发实例

- int pcap_setfilter(pcap *p,
  struct bpf_program*fp)
  - 功能：设置BPF过滤规则
  - 返回值：成功返回0，失败返回-1
  - 参数

    p：Libpcap句柄

    fp：BPF过滤规则

```
57    if(pcap_setfilter(pcap_handle,&bpf_filter) < 0)//设置过滤规则
58    {
59        perror("pcap_setfilter");
60    }
```

Libpcap开发实例

- const u_char *pcap_next(pcap_t *p,
  struct pcap_pkthdr *h)
  - 功能：捕获一个网络数据包
  - 参数
    p：Libpcap句柄
    h：数据包头
  - 返回值：捕获的数据包的地址
  - 例如：

```
64      struct pcap_pkthdr protocol_header;
65      unsigned char *p_packet_content = NULL;
66      p_packet_content = pcap_next(pcap_handle,&protocol_header);
```

Libpcap开发实例

- int pcap_loop(pcap_t *p,int cnt,
  pcap_handler callback,
  u_char *user)
  - 功能:循环捕获网络数据包，直到遇到错误或者满足退出条件；每次捕获一个数据包就会调用callback指示的回调函数，所以，可以在回调函数中进行数据包的处理操作
  - 返回值：成功返回0，失败返回负数

Libpcap开发实例

- 参数

  p：Libpcap句柄

  cnt：指定捕获数据包的个数，如果是-1，就会永无休止的捕获

  callback：回调函数

  user：向回调函数中传递的参数

- 例如：

```
87      if(pcap_loop(pcap_handle,-1,ethernet_protocol_callback,NULL) < 0)
88      {
89          perror("pcap_loop");
90      }
```

Libpcap开发实例

- void pcap_close(pcap_t *p)
  - 功能：关闭Libpcap操作，并销毁相应的资源
  - 参数

    p:需要关闭的Libpcap句柄
  - 返回值：无
  - 例如

```
pcap_close(pcap_handle);
```

Libpcap开发实例-demo

➢ 捕获第一个网络数据包（01-Libpcap-demo）

- gcc编译时需要加上-lpcap
- 运行时需要使用超级权限

```
edu@edu-T: ~/share/libpcap
edu@edu-T:~/share/libpcap$ sudo ./a.out
------------------------------------------------------------------------
capture a Packet from p_net_interface_name :eth0
Capture Time is :Sat Jun 15 06:50:13 2013
Packet Lenght is :142
Mac Source Address is 10:78:d2:93:ed:42
Mac Destination Address is 00:0c:29:c4:c0:0a
Ethernet type is :0800  The network layer is IP protocol
edu@edu-T:~/share/libpcap$ sudo ./a.out
------------------------------------------------------------------------
capture a Packet from p_net_interface_name :eth0
Capture Time is :Sat Jun 15 06:50:17 2013
Packet Lenght is :60
Mac Source Address is c8:9c:dc:ba:79:30
Mac Destination Address is ff:ff:ff:ff:ff:ff
Ethernet type is :0806  The network layer is ARP protocol
edu@edu-T:~/share/libpcap$
```

Libpcap开发实例-demo

- 捕获指定类型数据包，如ARP（02-Libpcap-demo）

```
edu@edu-T: ~/share/libpcap
edu@edu-T:~/share/libpcap$ sudo ./a.out
------------------------------------------------------------
capture a Packet from p_net_interface_name :eth0
Capture Time is :Sat Jun 15 07:02:50 2013
Packet Lenght is :60
Mac Source Address is 00:24:54:b7:f1:36
Mac Destination Address is ff:ff:ff:ff:ff:ff
Ethernet type is :0806  The network layer is ARP protocol
edu@edu-T:~/share/libpcap$
```

- 注意：为了更具有说明请多次执行此程序

Libpcap开发实例-demo

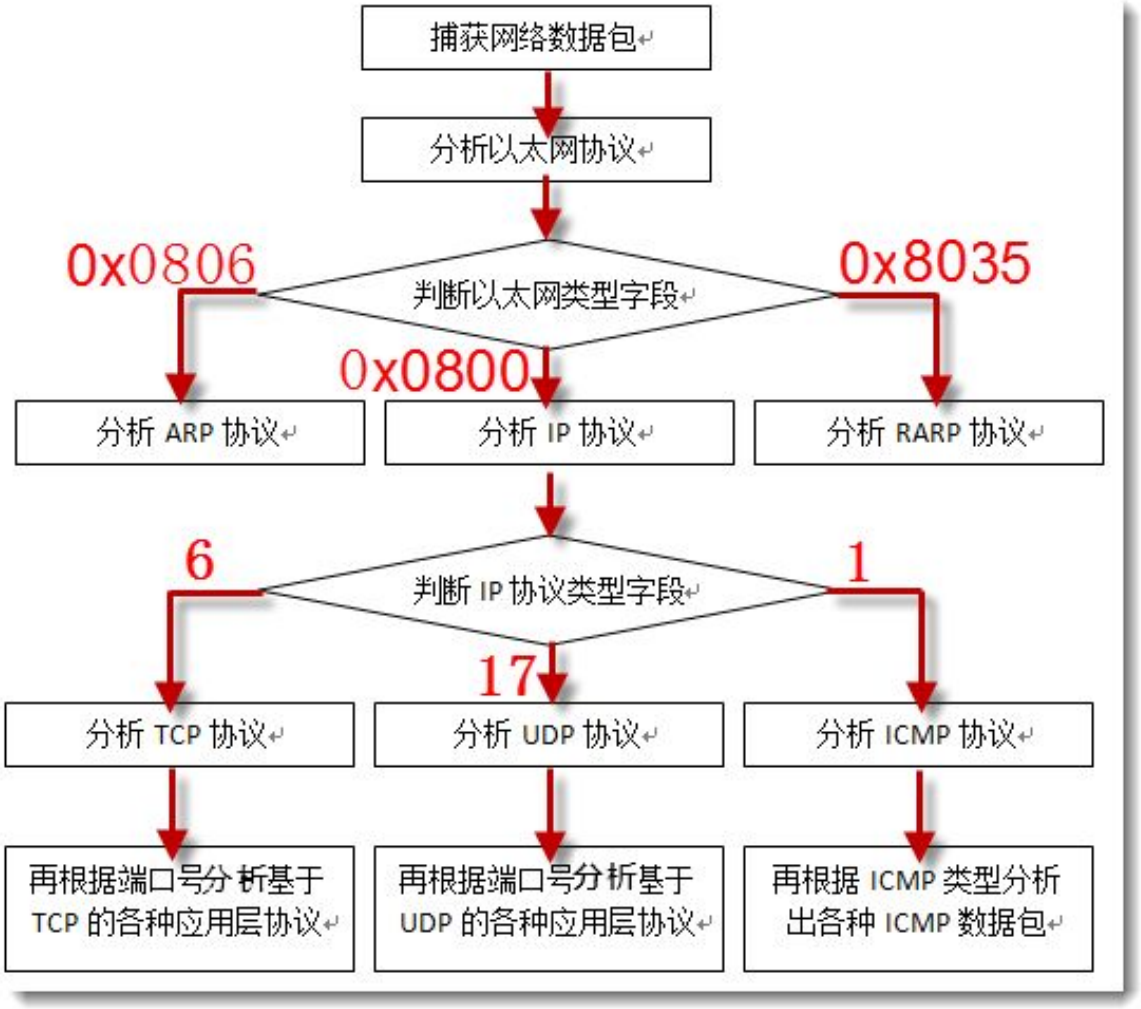
- 捕获多个网络数据包（03-Libpcap-demo）

```
edu@edu-T: ~/share/libpcap
edu@edu-T:~/share/libpcap$ sudo ./a.out
---------------------------------------------------
Mac Source Address is 10:78:d2:93:ed:42
Mac Destination Address is 00:0c:29:c4:c0:0a
Ethernet type is :0800
The network layer is IP protocol
---------------------------------------------------
Mac Source Address is 10:78:d2:93:ed:42
Mac Destination Address is 00:0c:29:c4:c0:0a
Ethernet type is :0800
The network layer is IP protocol
---------------------------------------------------
Mac Source Address is 00:0c:29:c4:c0:0a
Mac Destination Address is 10:78:d2:93:ed:42
Ethernet type is :0800
The network layer is IP protocol
```

大纲

- socket原始套接字回顾
- libpcap介绍、安装
- libpcap开发实例
- libpcap练习——网络数据分析器
- Libnet介绍、安装
- Libnet开发实例
- libnet练习——ARP欺骗

Libpcap开发流程



libpcap练习

- 项目要求
  - 判断数据包的类型ARP/RARP/IP/TCP/UDP
  - 分析

    MAC(src、dst)

    IP(src、dst)

    PORT(src、dst)
  - 分析出的UDP/TCP应用程序的通信数据

大纲

- socket原始套接字回顾
- libpcap介绍、安装
- libpcap开发实例
- libpcap练习——网络数据分析器
- Libnet介绍、安装
- Libnet开发实例
- libnet练习——ARP欺骗

Libnet介绍

- Libnet概念
  - 专业的构造和发送网络数据包的开发工具包
  - 是个高层次的API函数库，允许开发者自己构造和发送网络数据包
- Libnet特点
  - 隐藏了很多底层细节，省去了很多麻烦；如缓冲区管理、字节流顺序、校验和计算等问题，使开发者把重心放到程序的开发中
  - 可以轻松、快捷的构造任何形式的网络数据包，从而开发各种各样的网络程序

Libnet介绍

- 使用非常广泛，例如著名的软件Ettercap、Firewalk、Snort、Tcpreplay等
- 在1998年就出现了，但那时还有很多缺陷，比如计算校验和非常繁琐等；从2001年开始Libnet的作者Mike Schiffman对其进行了完善，而且功能更加强大。至此，可以说Libnet开发包已经非常完美了，使用Libnet开发包的人越来越多

Libnet安装

➢ Libnet的安装

- apt-get安装

```
edu@edu-T: ~
edu@edu-T:~$ sudo apt-get install libnet-dev
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
注意，选取 libnet1-dev 而非 libnet-dev
将会安装下列额外的软件包：
  libnet1
下列【新】软件包将被安装：
  libnet1 libnet1-dev
升级了 0 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，有 424 个软件包未被升级。
需要下载 166 kB 的软件包。
解压缩后会消耗掉 599 kB 的额外空间。
您希望继续执行吗？[Y/n]y
获取: 1 http://172.20.220.71/ precise/main libnet1 i386 1.1.4-2.1 [51.4 kB]
```

大纲

- socket原始套接字回顾
- libpcap介绍、安装
- libpcap开发实例
- libpcap练习——网络数据分析器
- Libnet介绍、安装
- Libnet开发实例
- libnet练习——ARP欺骗

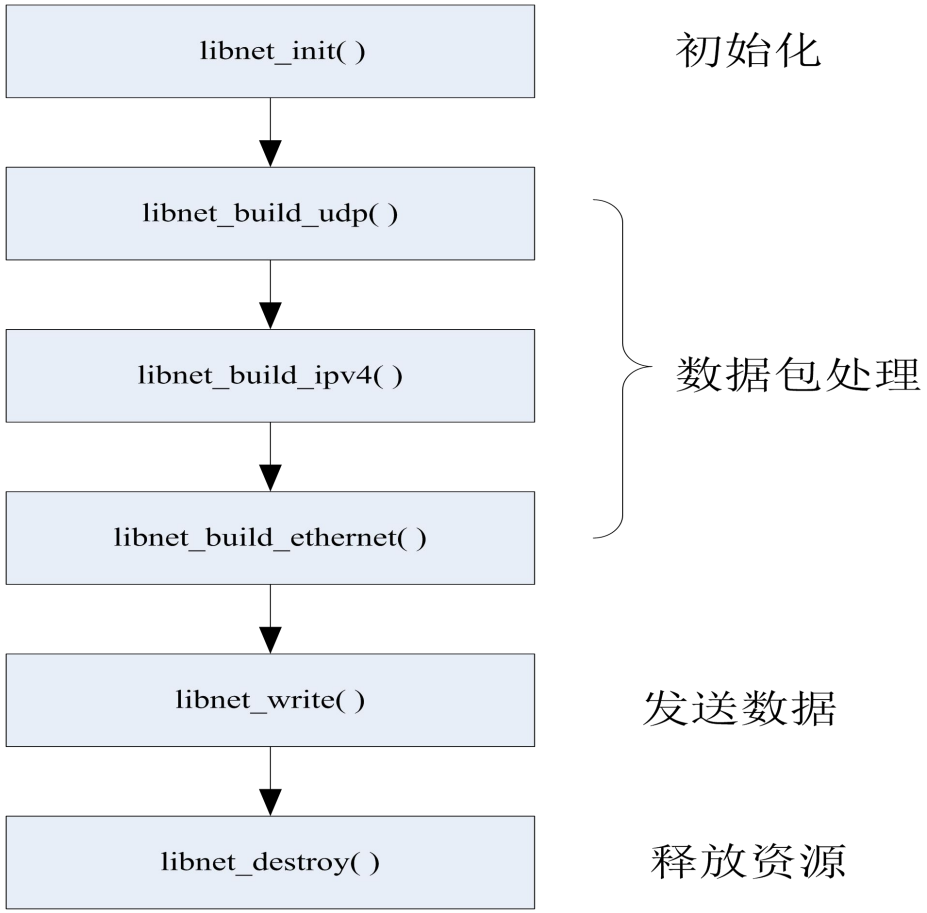
Libnet开发流程

- 利用libnet函数库开发应用程序的基本步骤：
  1. 数据包内存初始化
  2. 构造数据包
  3. 发送数据
  4. 释放资源

- libnet库主要功能：
  - 内存管理
  - 地址解析
  - 包处理

Libnet开发流程

➢ 飞鸽欺骗



Libnet开发流程

➢ 内存管理相关函数：

libnet_t *libnet_init(int injection_type, char *device, char *err_buf)

- 功能：数据包内存初始化及环境建立
- 参数：

  injection_type：构造的类型(LIBNET_LINK,LIBNET_RAW4, LIBNET_LINK_ADV,LIBNET_RAW4_ADV)

  device：网络接口，如"eth0",或IP地址，亦可为NULL(自动查询搜索)

  err_buf：存放出错的信息
- 返回值：成功返回一个libnet句柄；失败返回NULL

```
33      lib_net = libnet_init(LIBNET_LINK_ADV, "eth0", err_buf);
34      if(NULL == lib_net)
35      {
36          perror("libnet_init");
37          exit(-1);
38      }
```

Libnet开发流程

- 内存管理相关函数：

  void libnet_destroy(libnet_t *l);

  - 功能：释放资源
  - 参数：

    l：libnet句柄
  - 返回值：无
  - 例如：

```
91      libnet_destroy(lib_net);
```

Libnet开发实例

- libnet_ptag_t libnet_build_udp(
  u_int16_t sp,u_int16_t dp,
  u_int16_t len,u_int16_t sum,
  u_int8_t *payload,u_int32_t payload_s,
  libnet_t *l,libnet_ptag_t ptag)
  - 功能：构造udp数据包
  - 返回值：成功返回协议标记；失败返回-1

Libnet开发实例

- 参数
  - sp：源端口号
  - dp：目的端口号
  - len：udp包总长度
  - sum：校验和，设为0，libnet自动填充
  - payload：负载，可设置为NULL
  - payload_s：负载长度，或为0
  - l：libnet句柄
  - ptag：协议标记

Libnet开发实例

- libnet_ptag_t libnet_build_ipv4(
  u_int16_t ip_len,u_int8_t tos,
  u_int16_t id,u_int16_t flag,
  u_int8_t ttl,u_int8_t prot,
  u_int16 sum,u_int32_t src,
  u_int32_t dst,u_int8_t *payload,
  u_int32_t payload_s,
  libnet_t *l,libnet_ptag_t ptag)
  - 功能：构造一个IPv4数据包

Libnet开发实例

- 参数
  - ip_len：ip包总长
  - tos：服务类型
  - id：ip标识
  - flag：片偏移
  - ttl：生存时间
  - prot：上层协议
  - sum：校验和，设为0，libnet自动填充
  - src：源ip地址

Libnet开发实例

- dst：目的ip地址
- payload：负载，可设置为NULL
- payload_s：负载长度，或为0
- l：libnet句柄
- ptag：协议标记

➢ 返回值：成功返回协议标记；失败返回-1

Libnet开发实例

- libnet_ptag_t libnet_build_ethernet(
  u_int8_t *dst,u_int8_t *src,
  u_int16_t type,
  u_int8_t *payload,
  u_int32_t payload_s,
  libnet_t *l,libnet_ptag_t ptag)
  - 功能：构造一个以太网数据包

Libnet开发实例

- 参数
  - dst：目的mac
  - src：源mac
  - type：上层协议类型
  - payload：负载，即附带的数据
  - payload_s：负载长度
  - l：libnet句柄
  - ptag：协议标记
- 返回值：成功返回协议标记；失败返回-1

libnet练习

- int libnet_write(libnet_t * l)
  - 功能：发送数据到网络
  - 参数：
    l：libnet句柄
  - 返回值：失败返回-1，成功返回其他

飞鸽欺骗——示例

```c
int main(int argc, char *argv[])
{
    char send_msg[1000] = "";
    char err_buf[100] = "";
    libnet_t *lib_net = NULL;
    int lens = 0;
    libnet_ptag_t lib_t = 0;
    unsigned char src_mac[6] = {0x00,0x0c,0x29,0x1b,0x22,0x0a};//发送者网卡地址
    unsigned char dst_mac[6] = {0xc8,0x9c,0xdc,0xb7,0x0f,0x19};//接收者网卡地址
    char *src_ip_str = "172.20.226.3"; //源主机IP地址
    char *dst_ip_str = "172.20.226.11"; //源主机IP地址
    unsigned long src_ip,dst_ip = 0;

    lens = sprintf(send_msg, "1_lbt4_5#131200#C89CDCB70F19#0#0#0#305b#5:%d:%s:%s:%d:%s", 123,"he.liang",
    "HE-LIANG",32,"lh");

    lib_net = libnet_init(LIBNET_LINK_ADV, "eth0", err_buf);     //初始化
    if(NULL == lib_net)
    {
        perror("libnet_init");
        exit(-1);
    }

    src_ip = libnet_name2addr4(lib_net,src_ip_str,LIBNET_RESOLVE);    //将字符串类型的ip转换为顺序网络字节流
    dst_ip = libnet_name2addr4(lib_net,dst_ip_str,LIBNET_RESOLVE);

    lib_t = libnet_build_udp(2425, 2425, 8+lens, 0, send_msg, lens, lib_net, 0);//构造udp数据包

    lib_t = libnet_build_ipv4(20+8+lens,0,500,0,10,17,0,src_ip,dst_ip,NULL,0,lib_net,0); //构造ip数据包

    lib_t = libnet_build_ethernet((u_int8_t *)dst_mac,(u_int8_t *)src_mac,ETHERTYPE_IP,
                                  NULL,0,lib_net,0);     //构造以太网数据包

    libnet_write(lib_net);       //发送数据包

    libnet_destroy(lib_net);        //销毁资源
    return 0;
}
```

1. 初始化

2.数据包处理

3.发送数据

4.释放资源

Libnet开发流程

- 更多处理函数请参见附录
- 如：
  - libnet_build_tcp( );
  - libnet_build_tcp_options( );
  - libnet_build_ipv4_options( );
  - libnet_build_arp( );

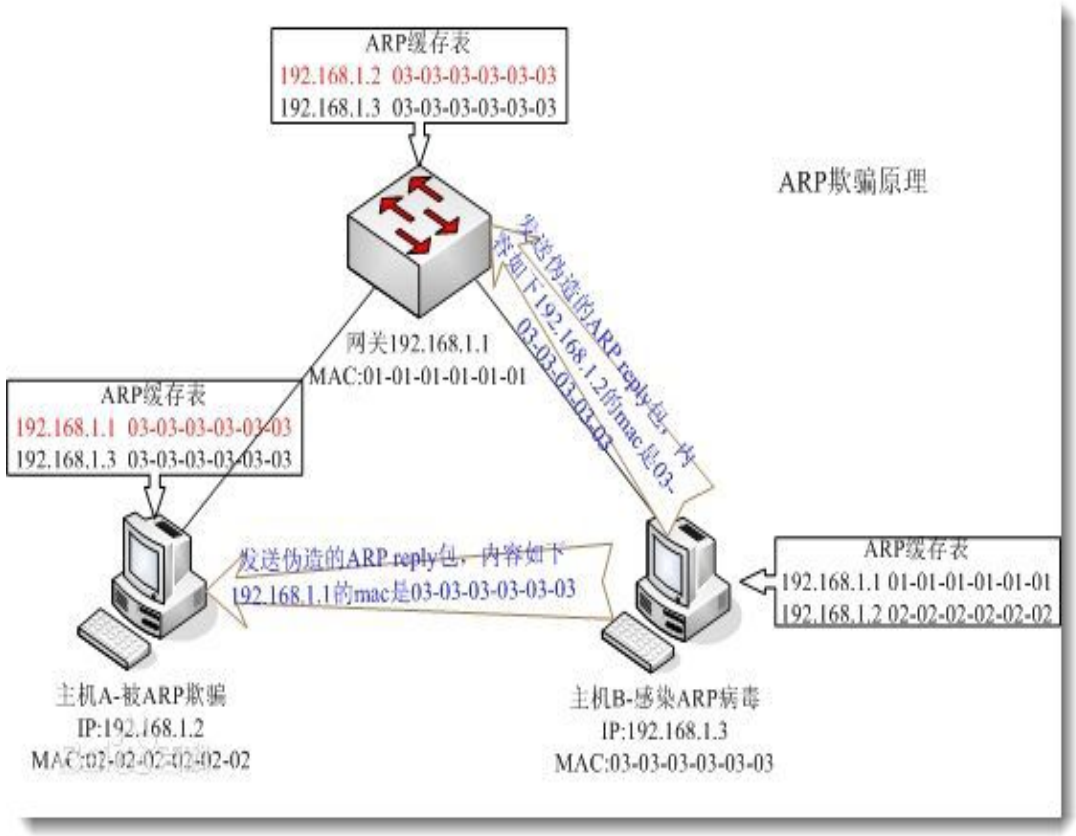
大纲

- socket原始套接字回顾
- libpcap介绍、安装
- libpcap开发实例
- libpcap练习——网络数据分析器
- Libnet介绍、作用
- Libnet安装
- Libnet开发实例
- libnet练习——ARP欺骗

libnet练习

- 项目分析

局域网中的网络流通是按照MAC地址进行传输的，可以通过欺骗通信双方的MAC地址，即可达到干扰通信的目的



libnet练习

- 项目要求
  - 用户可选择要欺骗的ip
  - 使用ARP方式来进行欺骗
  - 被欺骗方不能正常通信

libnet练习

- 项目步骤
  1. 发送ARP请求
  2. 获得ARP表(并存储)
  3. 选择欺骗目标ip
  4. 根据目标ip找到其MAC
  5. 使用库函数构造ARP回应包
  6. 发送

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com