



中南大学

CENTRAL SOUTH

网络安全课后实验

Seed project sniffing and spoofing

学 院： 信息科学与工程学院

班 级： 信息安全 1401

指导老师： 王伟平

姓 名： 苏伟

完成时间： 2016 年 11 月

目录

一、实验原理.....	3
二、实验器材.....	3
三、实验步骤.....	3
四、实验结果.....	4

一、实验原理

Sniffing 就是一种能将本地网卡状态设成‘混杂’状态的模式，当网卡处于这种“混杂”方式时，该网卡具备“广播地址”，它对遇到的每一个帧都产生一个硬件中断以便提醒操作系统处理流经该物理媒体上的每一个报文包。（绝大多数的网卡具备置成混杂模式的能力）

一般来说，sniffing 和 poofing 会联合起来使用。当攻击者嗅探到关键信息时，通常会使用 poofing 技术来构造数据包来劫持会话或者去获取更多信息，通常会造成很大的危害。Poofing 技术就是攻击者自己构造数据包的 ip/tcp 数据包帧头部数据来达到自己的目的。

本次实验就是基于以上原理，在 linux 下模拟整个过程。

二、实验器材

1. Ubuntu12.04。
2. Wireshark 等常用捕包工具。

三、实验步骤

Task1. 编写嗅探程序

嗅探程序可以很容易地使用 pcap 库。利用 PCAP，嗅探器的任务变得在 pcap 库调用一系列简单的程序。在序列结束时，数据包将被放置在缓冲区中，以进一步处理，只要它们被捕获。所有的数据包捕获的细节由 pcap 库处理。Tim Carstens 写了一个教程如何使用 pcap 库写的嗅探程序。

- 1: 深入理解并可以编写嗅探程序。
- 2: 编写过滤器。请为您的嗅探程序捕捉每个写过滤表达式如下。在你的实验报告，你需要包括 screendumps 显示应用这些过滤器的结果。
 - 捕获 ICMP 数据包。
 - 捕获 TCP 数据包有一个目的端口范围从端口 10 - 100。

Task2. 包欺骗

在正常的情况下，当一个用户发送一个数据包时，操作系统通常不允许用户设置所有的在协议头字段（如 TCP，UDP，和 IP 报头）。操作系统将大部分的领域，而

只允许用户设置几个字段，如目标 IP 地址、目标端口号等。但是当用户有有 root 权限，他们可以在数据包标头设置为任意字段。这就是所谓的包欺骗，它可以通

过原始套接字完成。

原始套接字给程序员的数据包结构的绝对控制，允许程序员构建任何任意的数据包，包括设置头字段和有效载荷。使用原始套接字是相当简单的，它包括四个步骤：（1）创建一个原始套接字，（2）设置套接字选项，（3）构建数据包，和（4）通过原始套接字发送数据包。有许多在线教程，可以教你如何使用原始套接字在 C 编程。我们已经把一些教程与实验室的网页联系起来了。请阅读它们，并学习如何写一个 **spoonfing** 程序包。我们展示了一个简单的程序。

Task3: 综合使用

在这个任务中，你将嗅探和欺骗技术实现连接，并实现程序。你需要在同一局域网两虚拟机。从 **VMA ping** 另一个 VM 的 IP，这将产生一个 **ICMP** 回送请求报文。如果 **X** 是活着的，**ping** 程序将收到一个回音答复，并打印出响应。你嗅探到数据包然后伪造程序运行在虚拟机 **B**、监控网络数据包嗅探。每当它看到 **ICMP** 回送请求，不管目标 IP 地址是什么，你的程序应该立即发出回声应答数据包欺骗技术的使用。因此，考虑到机器 **X** 是否是活的，这个程序将总是收到一个回复，这表明 **X** 是活的。你要写这样一个程序，包括在你显示你的程序的工作报告 **screendumps**。请在你的报告中附上代码。

四、实验结果以及思考

Task1: 运行结果如下：

```
Device: eth0
Number of packets: 10
Filter expression: ip

Packet number 1:
    From: 192.168.129.132
    To: 128.230.208.76
    Protocol: TCP
    Src port: 40021
    Dst port: 80

Packet number 2:
    From: 192.168.129.132
    To: 128.230.208.76
    Protocol: TCP
```

```

Protocol: TCP
Src port: 40021
Dst port: 80
Payload (369 bytes):
00000  47 45 54 20 2f 7e 77 65 64 75 2f 73 65 65 64 2f  GET /~wedu/seed/
00016  6c 61 62 5f 65 6e 76 2e 68 74 6d 6c 20 48 54 54  lab_env.html HTT
00032  50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 77 77 77  P/1.1..Host: www
00048  2e 63 69 73 2e 73 79 72 2e 65 64 75 0d 0a 55 73  .cis.syr.edu..Us
00064  65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c  er-Agent: Mozill
00080  61 2f 35 2e 30 20 28 58 31 31 3b 20 55 62 75 6e  a/5.0 (X11; Ubun
00096  74 75 3b 20 4c 69 6e 75 78 20 69 36 38 36 3b 20  tu; Linux i686;
00112  72 76 3a 32 33 2e 30 29 20 47 65 63 6b 6f 2f 32  rv:23.0) Gecko/2
00128  30 31 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f  0100101 Firefox/
00144  32 33 2e 30 0d 0a 41 63 63 65 70 74 3a 20 74 65  23.0..Accept: te
00160  78 74 2f 68 74 6d 6c 2c 61 70 70 6c 69 63 61 74  xt/html,applicat
00176  69 6f 6e 2f 78 68 74 6d 6c 2b 78 6d 6c 2c 61 70  ion/xhtml+xml,ap
00192  70 6c 69 63 61 74 69 6f 6e 2f 78 6d 6c 3b 71 3d  plication/xml;q=

```

部分代码如下：

```

int main(int argc, char **argv)
{
    char *dev = NULL;          /* capture device name */
    char errbuf[PCAP_ERRBUF_SIZE]; /* error buffer */
    pcap_t *handle;            /* packet capture handle */
    char filter_exp[] = "ip";   /* filter expression [3] */
    struct bpf_program fp;      /* compiled filter program (expression) */
    bpf_u_int32 mask;          /* 子网掩码 */
    bpf_u_int32 net;           /* IP 地址 */
    int num_packets = 10;      /* number of packets to capture */
    print_app_banner();
    /* check for capture device name on command-line */
    if (argc == 2) {
        dev = argv[1];
    }
    else if (argc > 2) {
        fprintf(stderr, "error: unrecognized command-line options\n\n");
        print_app_usage();
        exit(EXIT_FAILURE);
    }
    else {
        /* find a capture device if not specified on command-line */
        dev = pcap_lookupdev(errbuf);
        if (dev == NULL) {
            fprintf(stderr, "Couldn't find default device: %s\n",
                    errbuf);
            exit(EXIT_FAILURE);
        }
    }
}

```

```

    }
}

/* get network number and mask associated with capture device */
if (pcap_lookupnet(dev, &net, &mask, errbuf) == -1) {
    fprintf(stderr, "Couldn't get netmask for device %s: %s\n",
            dev, errbuf);
    net = 0;
    mask = 0;
}

/* print capture info */
printf("Device: %s\n", dev);
printf("Number of packets: %d\n", num_packets);
printf("Filter expression: %s\n", filter_exp);

/* open capture device */
handle = pcap_open_live(dev, SNAP_LEN, 1, 1000, errbuf);
if (handle == NULL) {
    fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
    exit(EXIT_FAILURE);
}

/* make sure we're capturing on an Ethernet device [2] */
if (pcap_datalink(handle) != DLT_EN10MB) {
    fprintf(stderr, "%s is not an Ethernet\n", dev);
    exit(EXIT_FAILURE);
}

if (pcap_compile(handle, &fp, filter_exp, 0, net) == -1) /*过滤表达式*/
    fprintf(stderr, "Couldn't parse filter %s: %s\n",
            filter_exp, pcap_geterr(handle));
    exit(EXIT_FAILURE);
}

if (pcap_setfilter(handle, &fp) == -1) {
    fprintf(stderr, "Couldn't install filter %s: %s\n",
            filter_exp, pcap_geterr(handle));
    exit(EXIT_FAILURE);
}

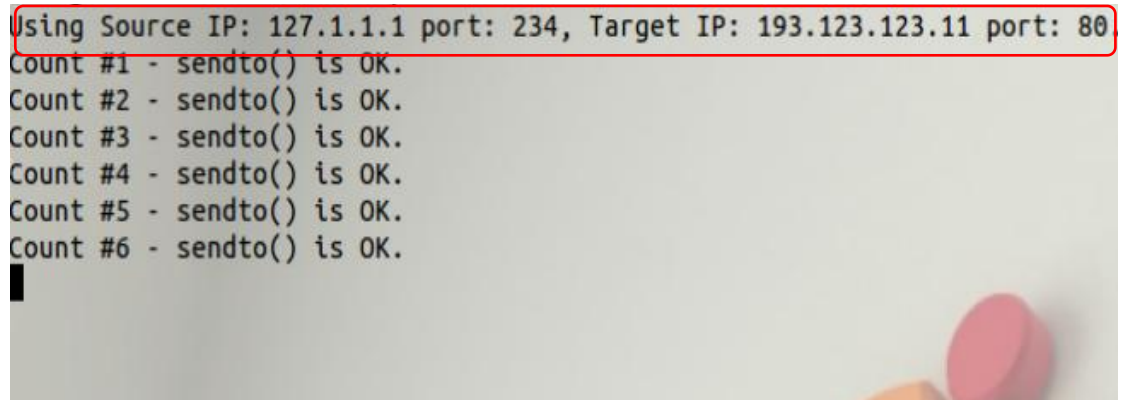
pcap_loop(handle, num_packets, got_packet, NULL);
pcap_freecode(&fp);

```

```
pcap_close(handle);  
printf("\nCapture complete.\n");  
return 0;  
}
```

对于任务一的 2，主要是修改 filter 中的过滤条件，要实现只捕获 ICMP 类型的数据包，只需要将 char filter_exp[] = "ip" 中的 ip 改为 ICMP，然后要捕获端口在 10-100 之间的 tcp 数据包，同理，将这条语句中的条件改为 'tcp and dst portrange 10-100' 即可。

运行结果如下：

A terminal window showing the configuration of a network capture. The first line is highlighted with a red box: "Using Source IP: 127.1.1.1 port: 234, Target IP: 193.123.123.11 port: 80". Below this, there are six lines indicating the status of sendto() calls: "Count #1 - sendto() is OK.", "Count #2 - sendto() is OK.", "Count #3 - sendto() is OK.", "Count #4 - sendto() is OK.", "Count #5 - sendto() is OK.", and "Count #6 - sendto() is OK.". The terminal background is dark, and the text is light-colored. There are some colorful objects (pink and orange) visible in the bottom right corner of the terminal window.

```
Using Source IP: 127.1.1.1 port: 234, Target IP: 193.123.123.11 port: 80  
Count #1 - sendto() is OK.  
Count #2 - sendto() is OK.  
Count #3 - sendto() is OK.  
Count #4 - sendto() is OK.  
Count #5 - sendto() is OK.  
Count #6 - sendto() is OK.  
█
```

这里便实施了包欺骗

对于 Task3，实则是对前两个任务的总结，即将嗅探和欺骗混合在一起，也就可以通过将前两个实验的代码进行混合，但由于能力有限，没有完成，代码运行不出想要的结果，感到十分得遗憾。但是，通过实质得编程，还是增进不少。特别是网络编程。