数据包嗅探与伪造

注：本章实际章节号对于不同的卷来说是不同的，所以它被表示为在本文档中用字母 C 表示。

N4.1. 在书中介绍的基于 pcap 的嗅探器程序中，我们添加了一个检查来查看 `handle` 是否为 NULL。运行该程序时收到错误消息 `"NULL: No such device"`。问题出在哪里？

```
  handle = pcap_open_live("eth1", BUFSIZ, 1, 1000, errbuf);
  if (handle == NULL) {
      perror("NULL");
  }

Error message:
  NULL: No such device
```

N4.2. 在基于 pcap 的嗅探器程序中，我们使用以下代码打开 pcap 会话。运行程序时，只能捕获进出本机的数据包，无法捕获同一网络中其他计算机之间的数据包。原因可能是什么？

```
  handle = pcap_open_live("eth3", BUFSIZ, 0, 1000, errbuf);
```

N4.3. 本书代码清单 C.4 中哪一行需要 root 权限（C 是“数据包嗅探与伪造”一章的章号，具体数值取决于本书版本）？如果程序不以 root 权限运行，会发生什么？

N4.4. 清单 C.4 中的 `pcap_setfilter()` 会在内核中设置过滤器，因此看起来该调用也需要 root 权限。请设计实验验证或推翻这一假设（C 是本章章号，具体数值取决于本书版本）。

N4.5. 嗅探器通常有两种过滤方式。第一种先从系统取得所有数据包，再在用户态丢弃不需要的数据包，然后显示结果或写入文件；第二种使用 `pcap_setfilter` 设置过滤器。请说明两种方式的区别。

N4.6. ★ ★

程序要启用混杂模式，甚至只是在本机嗅探数据包，都需要普通用户不具备的特殊权限。据此，我们检查了 `wireshark` 进程的权限：

```
$ pgrep wireshark
7598
$ ps  -fp 7598
UID   PID   PPID  C STIME TTY  TIME     CMD
seed  7598     1  0 10:01 ?    00:00:01 /usr/bin/wireshark
```

结果表明，wireshark 进程的 UID（有效用户 ID）是 seed，但它却能捕获本地网络中的所有数据包，包括并非发往或来自本机的数据包。按照嗅探器的工作原理，这似乎不可能。请调查并解释其中的原因。

提示：在 wireshark 中开始抓包，然后查看它启动了哪些子进程。可以使用 "pstree -p 7598" 获取进程 7598 的子进程 ID，并重点检查子进程实际执行的程序。

N4.7. tcpdump 也是嗅探器，出现时间早于 wireshark。它只输出文本，或把捕获的数据包保存到文件。程序起初需要 root 权限，但提供了 -Z 选项，可在以 root 启动后放弃权限，并把用户 ID 改成指定账户。例如，"-Z seed" 会把有效用户 ID 降为 seed。为什么要提供该选项？放弃权限后程序为何仍能工作？

N4.8. 根据问题 N4.6. 的调查结果，请说明如何对清单 C.4 中的嗅探器应用最小权限原则，以缩小攻击面（C 是本章章号，具体数值取决于本书版本）。

N4.9. 整数 0xAABBCCDD 从内存地址 0x1000 开始存放。在大端机器上，地址 0x1000、0x1001、0x1002 和 0x1003 中分别存放什么值？在小端机器上又如何存放？

N4.10. 某网络协议用一个四字节整数表示数据包的有效载荷长度。实现程序收到数据包后，先把报头中的长度字段复制到变量 X，却忘记将其从网络字节序转换为主机字节序；随后分配 X 字节内存以复制有效载荷。在小端机器上，如果收到的数据包有效载荷为 255 字节，程序会分配多少内存？这一错误可能造成什么后果？

N4.11. 请分别说明以下程序在小端机器和大端机器上的输出。

```
void main()
{
  int a = 255;
  printf("%u\n", htonl(a));
  printf("%u\n", ntohl(a));
}
```

N4.12. 假设机器 A 和 B 位于同一网络 10.3.2.0/24。A 发送伪造数据包，B 在网络中嗅探。当伪造数据包的目的地址为 1.2.3.4 时，B 总能看到它；但目的地址为

10.3.2.30 时，B 看不到它。伪造和嗅探程序本身都没有问题，显然数据包根本没有发出。可能是什么原因？