ieee802.11数据radiotap介绍

原创 李迟 于 2016-11-16 22:34:20 发布 10266 收藏 26 版权

分类专栏： Linux网络

之前写有文章介绍了在Linux系统用 wireshark 或tcpdump抓无线网卡数据包。分析包时发现每一数据帧前面都有一个叫radiotap的东西。它包含了如信号强度、频率等信息。当时没有研究，直接跳过。本文就对此介绍补充。首先介绍radiotap，然后利用radiotap解析库对一段radiotap数据进行解析，获取其中的信息。

介绍

radiotap比传统的Prism或AVS头部更有灵活性，成为是ieee802.11事实上的标准。支持radiotap的系统较多，如Linux、FreeBSD、NetBSD、OpenBSD，还有Windows(需使用AirPcap)。它的头部定义如下：

```
struct ieee80211_radiotap_header {
        u_int8_t        it_version;     /* set to 0 */
        u_int8_t        it_pad;
        u_int16_t       it_len;         /* entire length */
        u_int32_t       it_present;     /* fields present */
} __attribute__((__packed__));
```

其中第一个字段it_version表示版本号，当前为0。

第二个字段it_pad没有使用，仅仅是为了结构体对齐。

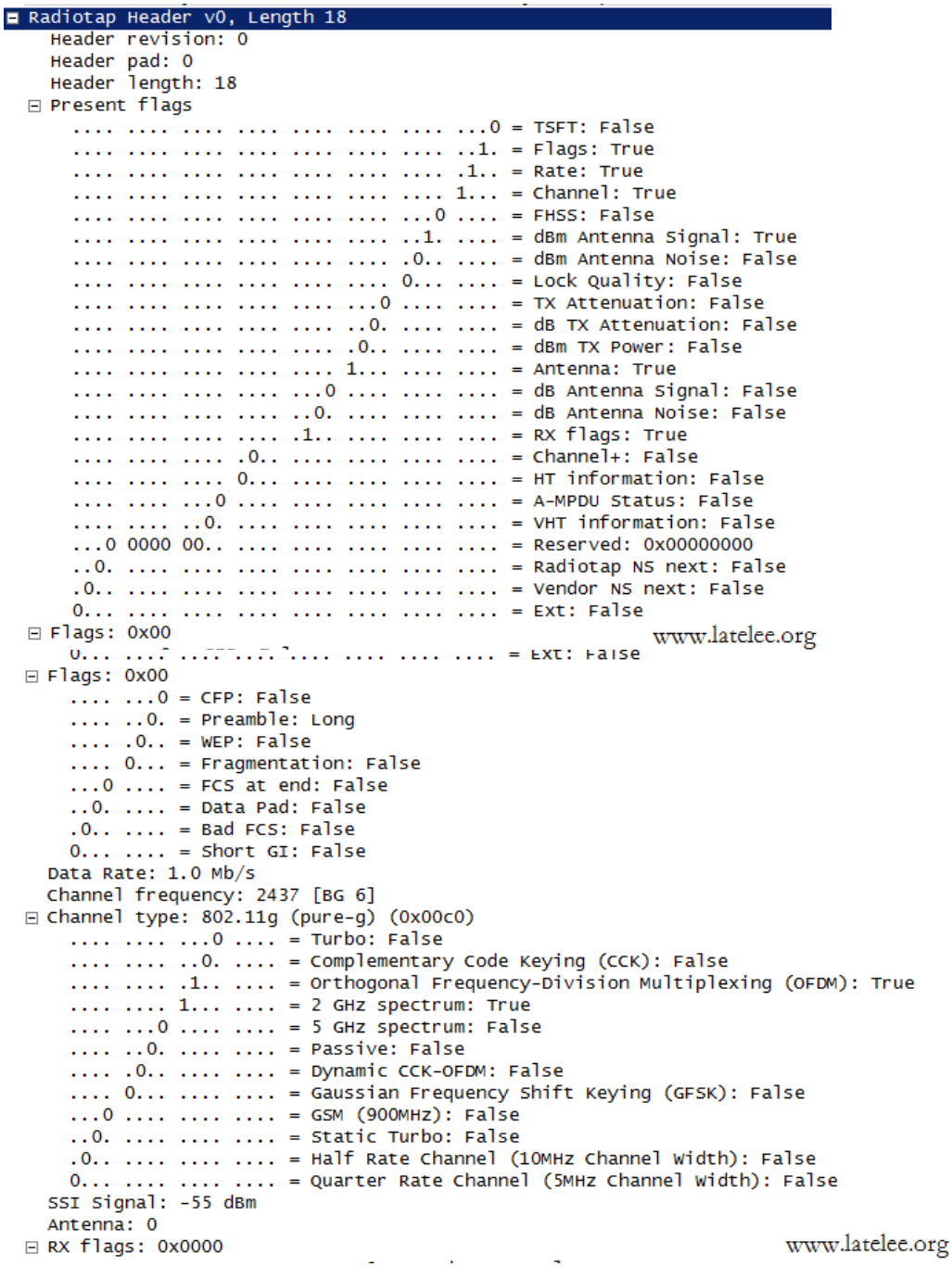
第三个字段it_present表示长度，包括了radiotap头部和数据两部分。此设计的好处在于，如果不需要了解radiotap，则可以跳过直接到ieee802.11头部。——半个月前写抓包程序时，还不了解radio，就是直接跳过的。

第四个字段it_present表示radiotap数据的位掩码。radiotap的数据紧跟其头部。当其中的位掩码为true时，表示有对应的数据，可以认为每一比特表示一种类型。比如bit5为1表示有通道数据，则可以获取到信号强度。反之就是没有对应的数据。因此radiotap的长度其实是不固定的。bit31为1表示还有多个it_present。不过目前笔者还没有碰到此情况，没有深入了解。

radiotap的每个类型都是有严格的顺序的。另外，radiotap数据的字序是小端格式(little endian byte-order)——包括头部的it_len和it_present。

目前应用比较广的解析库是radiotap-library——在horst软件和Linux内核中都使用到。关于每个类型的解释，可以参考radiotap.h文件的ieee80211_radiotap_type注释。

下面看一下wireshark抓到的包的radiotap头部数据：



由图可见，radiotap包括的东西挺多的。

解析

radiotap解析库使用十分简单，先对此有个基本面的认知概念：

1、首先

完整代码如下：

```
/**
radiotap头部解析
使用radiotap库，源码地址：
https://github.com/radiotap/radiotap-library

*/

#include <unistd.h>
#include <stdio.h>
```

```
#include <stdint.h>
                    #include <endian.h>
#include <errno.h>
#include <string.h>

#include "radiotap_iter.h"

// 根据wireshark抓包抽取的radiotap头部数据
char radiotap_buf[][18] = {
                {0x00, 0x00, 0x12, 0x00, 0x2e, 0x48,
                 0x00, 0x00, 0x00, 0x02, 0x85, 0x09,
                 0xc0, 0x00, 0xc9, 0x00, 0x00, 0x00},
                {0x00, 0x00, 0x12, 0x00, 0x2e, 0x48,
                 0x00, 0x00, 0x00, 0x02, 0x85, 0x09,
                 0xa0, 0x00, 0xa8, 0x00, 0x00, 0x00},
                };

static void print_radiotap_namespace(struct ieee80211_radiotap_iterator *iter)
{
    char signal = 0;
    uint32_t phy_freq = 0;

    switch (iter->this_arg_index)
    {
    case IEEE80211_RADIOTAP_TSFT:
        printf("\tTSFT: %llu\n", le64toh(*(unsigned long long *)iter->this_arg));
        break;
    case IEEE80211_RADIOTAP_FLAGS:
        printf("\tflags: %02x\n", *iter->this_arg);
        break;
    // 速率?
    case IEEE80211_RADIOTAP_RATE:
        printf("\trate: %.2f Mbit/s\n", (double)*iter->this_arg/2);
        break;

#define IEEE80211_CHAN_A \
    (IEEE80211_CHAN_5GHZ | IEEE80211_CHAN_OFDM)
#define IEEE80211_CHAN_G \
    (IEEE80211_CHAN_2GHZ | IEEE80211_CHAN_OFDM)
    // 通信信息
    case IEEE80211_RADIOTAP_CHANNEL:
        phy_freq = le16toh(*(uint16_t*)iter->this_arg); // 信道
        iter->this_arg = iter->this_arg + 2; // 通道信息如2G、5G，等
        int x = le16toh(*(uint16_t*)iter->this_arg);
        printf("\tfreq: %d type: ", phy_freq);
        if ((x & IEEE80211_CHAN_A) == IEEE80211_CHAN_A)
        {
            printf("A\n");
        }
        else if ((x & IEEE80211_CHAN_G) == IEEE80211_CHAN_G)
        {
            printf("G\n");
        }
        else if ((x & IEEE80211_CHAN_2GHZ) == IEEE80211_CHAN_2GHZ)
        {
            printf("B\n");
        }
        break;
    // 信号强度
    case IEEE80211_RADIOTAP_DBM_ANTSIGNAL:
        signal = *(signed char*)iter->this_arg;
        printf("\tsignal: %d dBm\n", signal);
        break;
        break;
```

```
    // 接收标志
            case IEEE80211_RADIOTAP_RX_FLAGS:
        printf("\tRX flags: %#.4x\n", le16toh(*(uint16_t *)iter->this_arg));
        break;
    case IEEE80211_RADIOTAP_ANTENNA:
        printf("\tantenna: %x\n", *iter->this_arg);
        break;
    // 忽略下面的
    case IEEE80211_RADIOTAP_RTS_RETRIES:
    case IEEE80211_RADIOTAP_DATA_RETRIES:
    case IEEE80211_RADIOTAP_FHSS:
    case IEEE80211_RADIOTAP_DBM_ANTNOISE:
    case IEEE80211_RADIOTAP_LOCK_QUALITY:
    case IEEE80211_RADIOTAP_TX_ATTENUATION:
    case IEEE80211_RADIOTAP_DB_TX_ATTENUATION:
    case IEEE80211_RADIOTAP_DBM_TX_POWER:
    case IEEE80211_RADIOTAP_DB_ANTSIGNAL:
    case IEEE80211_RADIOTAP_DB_ANTNOISE:
    case IEEE80211_RADIOTAP_TX_FLAGS:
        break;
    default:
        printf("\tBOGUS DATA\n");
        break;
    }
}

int main(void)
{
    struct ieee80211_radiotap_iterator iter;
    int err;
    int i, j;

    for (i = 0; i < sizeof(radiotap_buf)/sizeof(radiotap_buf[0]); i++)
    {
        printf("parsing [%d]\n", i);
        // 初始化
        err = ieee80211_radiotap_iterator_init(&iter,
        (struct ieee80211_radiotap_header *)radiotap_buf[i], sizeof(radiotap_buf[i]), NULL);
        if (err)
        {
            printf("not valid radiotap...\n");
            return -1;
        }
        j = 0;
        /**
        遍历时，this_arg_index表示当前索引(如IEEE80211_RADIOTAP_TSFT等)，
        this_arg表示当前索引的值，this_arg_size表示值的大小。
        只有flag为true时才会进一步解析。
        */
        while (!(err = ieee80211_radiotap_iterator_next(&iter)))
        {
            printf("next[%d]: index: %d size: %d\n",
                    j, iter.this_arg_index, iter.this_arg_size);
            if (iter.is_radiotap_ns) // 表示是radiotap的命名空间
            {
                print_radiotap_namespace(&iter);
            }
            j++;
        }
        printf("=================================\n");
    }

    return 0;
}
```

执行结果：

```
parsing [0]
next[0]: index: 1 size: 1
        flags: 00
next[1]: index: 2 size: 1
        rate: 1.00 Mbit/s
next[2]: index: 3 size: 4
        freq: 2437 type: G
next[3]: index: 5 size: 1
        signal: -55 dBm
next[4]: index: 11 size: 1
        antenna: 0
next[5]: index: 14 size: 2
        RX flags: 0000
==================================
parsing [1]
next[0]: index: 1 size: 1
        flags: 00
next[1]: index: 2 size: 1
        rate: 1.00 Mbit/s
next[2]: index: 3 size: 4
        freq: 2437 type: B
next[3]: index: 5 size: 1
        signal: -88 dBm
next[4]: index: 11 size: 1
        antenna: 0
next[5]: index: 14 size: 2
        RX flags: 0000
==================================
```

参考资料：

http://www.radiotap.org/

https://github.com/radiotap/radiotap-library