PACP学习笔记三：PCAP方法说明

原创 山鬼谣me 已于 2022-07-01 17:30:03 修改 2755 收藏 14 版权
分类专栏： pcap 文章标签： pcap

创建捕获句柄

pcap_open_live – 创建嗅探会话

创建嗅探会话的任务非常简单。为此，我们使用 `pcap_open_live(3PCAP)` 。这个函数的原型如下：

```
#include <pcap/pcap.h>
char errbuf[PCAP_ERRBUF_SIZE];
pcap_t *pcap_open_live(const char *device, int snaplen, int promisc, int to_ms, char *errbuf);
```

参数字段	说明
第一个参数	第一个参数是我们在上一节中指定的设备。
第二个参数	snaplen 是一个整数，它定义了 pcap 捕获的最大字节数。
第三个参数	promisc，当设置为 true 时，使接口进入混杂模式（然而，即使它设置为 false，在特定情况下，接口也可能处于混杂模式，无论如何）。
第四个参数	to_ms 是以毫秒为单位的读取超时时间（值 0 表示没有超时；至少在某些平台上，这意味着您可能要等到足够数量的数据包到达才能看到任何数据包，因此您应该使用非零暂停）
第五个参数	ebuf 是一个字符串，我们可以在其中存储任何错误消息（就像我们在上面使用 errbuf 所做的那样）。
返回值	该函数返回我们的会话处理程序

描述

`pcap_open_live()` 用于获取数据包捕获句柄以查看网络上的数据包。 device 是一个字符串，指定要打开的网络设备 ；在具有 2.2 或更高版本内核的 Linux 系统上，设备参数“any”或 NULL 可用于捕获来自所有接口的数据包。

`snaplen` 指定要在句柄上设置的快照长度。

`promisc` 指定是否将接口置于混杂模式。如果 promisc 非零，则设置混杂模式，否则不设置。

返回值

`pcap_open_live()` 成功时返回 pcap_t *，失败时返回 NULL。如果返回 NULL，则 errbuf 会填充适当的错误消息。
当 `pcap_open_live()` 成功时，errbuf 也可以设置为警告文本；为了检测这种情况，调用者应在调用 pcap_open_live() 之前将零长度字符串存储在 errbuf 中，如果 errbuf 不再是零长度字符串，则向用户显示警告。假设 errbuf 至少能够保存 **PCAP_ERRBUF_SIZE** 字符。

pcap_create - 创建实时捕获句柄

```
#include <pcap/pcap.h>

char errbuf[PCAP_ERRBUF_SIZE];
```

```
4
5  pcap_t *pcap_create(const char *source, char *errbuf);
```

描述

`pcap_create()` 用于创建数据包捕获句柄(`handle`)以查看网络上的数据包。 **source** 是一个字符串，指定要打开的网络设备；在具有 2.2 或更高版本内核的 Linux 系统上，可以使用“any”或 NULL 的源参数来捕获来自所有接口的数据包。
返回的句柄必须先用 `pcap_activate(3PCAP)` 激活，然后才能用它捕获数据包；捕获的选项，例如混杂模式，可以在激活之前在句柄上设置。

返回值

`pcap_create()` 成功时返回 pcap_t *，失败时返回 NULL。如果返回 NULL，则 errbuf 会填充适当的错误消息。假设 errbuf 至少能够保存 **PCAP_ERRBUF_SIZE** (自己设置)字符。

pcap_activate - 激活捕获句柄

说明

```
1  #include <pcap/pcap.h>
2
3  int pcap_activate(pcap_t *p);
```

描述

`pcap_activate()` 用于激活数据包捕获句柄以查看网络上的数据包，并且在句柄上设置的选项生效。

返回值

pcap_activate() 成功且没有警告返回0，成功有警告返回非0正值，错误时返回负值；非零返回值指示发生了什么警告或错误情况。

具体参考：https://www.tcpdump.org/manpages/pcap_activate.3pcap.html

对句柄进行设置

编译程序 pcap_compile

为了编译程序，我们调用 `pcap_compile()` 。原型将其定义为：

```
1  int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize,
2      bpf_u_int32 netmask)
```

参数字段说明	描述
第一个参数	第一个参数是我们的会话句柄（pcap_t *handle 在我们之前的例子中）。
第二个参数	我们将存储 过滤器编译版本的位置引用。
第三个参数	表达式本身；格式：常规字符串
第四个参数	是一个整数，它决定表达式是否应该“优化”（0 为假，1 为真——标准内容）。
第五个参数	我们必须指定过滤器应用到的网络的网络掩码。
返回值	失败返回-1，其他值都是成功

设置过滤器 pcap_setfilter

表达式编译完成后，就可以应用它了。输入 `pcap_setfilter()` 。按照我们解释 pcap 的格式，我们来看看原型：

```
int pcap_setfilter(pcap_t *p, struct bpf_program *fp)
```

参数说明	描述
第一个参数	我们的会话处理程序
第二个参数	是对表达式的编译版本的引用（可能与 pcap_compile() 的第二个参数相同的变量）

https://www.tcpdump.org/manpages/libpcap-1.10.1/pcap-filter.7.html

通过句柄实际抓包

抓包函数 pcap_next

从 pcap_t 读取下一个数据包

一次捕获一个数据包

```
# u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)
const u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h);
```

参数具体说明	说明
第一个参数	session handler
第二个参数	是一个指向结构的指针，该结构包含有关数据包的一般信息，特别是它被嗅探的时间、该数据包的长度以及该特定部分的长度（例如，如果它被分段）。
返回值	返回指向此结构描述的数据包的 u_char 指针

pcap_loop() – 处理来自实时捕获或保存文件的数据包

进入循环

```
typedef void (*pcap_handler)(u_char *user, const struct pcap_pkthdr *h, const u_char *bytes);
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
```

参数具体说明	说明
第一个参数	第一个参数是我们会话句柄
第二个参数	是一个整数，它告诉 pcap_loop() 在返回之前它应该嗅探多少数据包（负值意味着它应该嗅探直到发生错误）
第三个参数	是回调函数的名称（只是它的标识符，没有括号）
第四个参数	最后一个参数在某些应用程序中很有用，但很多时候只是简单地设置为 NULL。假设除了 pcap_loop() 发送的参数之外，我们还有自己希望发送给回调函数的参数

说明

pcap_loop() 处理来自实时捕获或“保存文件”的数据包，直到处理完 **cnt** 数据包，从“保存文件”读取时到达“保存文件”的末尾，调用 pcap_breakloop(3PCAP)，或发生错误。当实时数据包缓冲区超时发生时，它不会返回。 **cnt** 的值 -1 或 0 等价于无穷大，因此数据包将一直处理，直到出现另一个结束条件。

> 实践中，当这个方法被执行调用时，会进入循环，这个循环是本地库的循环，因为接口是同步调用，所以不会一直卡在这里，知道循环结束或者抛异常。

pcap_dispatch()

`pcap_dispatch()` 的用法几乎和 `pcap_loop` 相同。这两个函数之间的唯一区别是 pcap_dispatch() 将只处理它从系统接收到的第一批数据包，而 pcap_loop() 将继续处理数据包或批量数据包，直到数据包计数用完。有关它们之间差异的更深入讨论，请参见手册页。

获取错误信息

pcap_geterr, pcap_perror - 获取或打印 libpcap 错误消息文本

说明

```
#include <pcap/pcap.h>

char *pcap_geterr(pcap_t *p);
void pcap_perror(pcap_t *p, const char *prefix);
```

描述

pcap_geterr() 返回与最后一个 pcap 库错误有关的错误文本。
注意：在传递给它的 pcap_t 关闭后，它返回的指针将不再指向有效的错误消息字符串；您必须在关闭 pcap_t 之前使用或复制该字符串。

pcap_perror() 在 stderr 上打印最后一个 pcap 库错误的文本，前缀为 prefix。

其他

pcap_datalink - 获取链路层标头类型

说明

```
#include <pcap/pcap.h>

int pcap_datalink(pcap_t *p);
```

描述

`pcap_datalink()` 返回由 p 指定的实时捕获或“保存文件”的链路层标头类型。
它不能在 `pcap_create(3PCAP)` 创建的 `pcap_activate(3PCAP)` 尚未激活的 pcap 描述符上调用。
https://www.tcpdump.org/linktypes.html 列出了 `pcap_datalink()` 可以返回的值，并描述了与这些值对应的数据包格式。

不要假设给定捕获或“保存文件”的数据包将具有任何给定的链路层标头类型，例如以太网的 DLT_EN10MB。例如，Linux 上的“any”设备将具有 DLT_LINUX_SLL 或 DLT_LINUX_SLL2 的链路层标头类型，即使在打开“any”设备时系统上的所有设备都具有一些其他数据链路类型，例如 DLT_EN10MB 用于以太网。

返回值

`pcap_datalink()` 成功时返回链路层标头类型，如果在已创建但未激活的捕获句柄上调用会抛出 `PCAP_ERROR_NOT_ACTIVATED` 。

参考地址：https://www.tcpdump.org/manpages/libpcap-1.10.1/pcap_datalink.3pcap.html

参考地址：

pcap
https://www.tcpdump.org/manpages/libpcap-1.10.1/