Libnids 快速入门

Libnids 概述

Libnids（Library Network Intrusion Detection System）是网络入侵检测开发的专业编程接口，实现了网络入侵检测系统的基本框架，提供了一些基本的功能。Libnids 是基于 Libpcap 和 Libnet 而开发的，其主要功能包括捕获网络数据包、IP 碎片重组、TCP 数据流重组以及端口扫面攻击检测和异常数据包检测等。

Libnids 的安装

因为 Libnids 必须支持库 Libpcap 和 Libnet,所以在安装 Libnids 之前需要先安装 Libpcap 和 Libnet。

在 Linux 下安装 Libnids 的步骤:

（1） 安装 Libpcap 开发包: 我们安装的版本是 libpcap-0.8.3.tar.gz.

```
tar    xzvf  libpcap-0.8.3.tar.gz
./configure
make
make install
```

（2） 安装 Libnet 开发包: 我们安装的版本是 libnet-1.1.2.1.tar.gz.

```
tar    xzvf  libnet-1.1.2.1.tar.gz
./configure
make
```

make install

（3） 安装Libnids开发包: 我们安装的版本是libnids-1.20.tar.gz.

tar xzvf libnids.1.20.tar.gz

./configure

make

make install

Libnids的数据结构

Libnids的状态主要有如下6种:

#define NIDS_JUST_EST 1(表示TCP连接建立，在此状态下就可以决定是否对此TCP连接进行数据分析,可以决定是否捕获TCP客户端接收的数据、TCP服务器端接收的数据、TCP客户端接收的紧急数据或者TCP服务器端接收的紧急数据);

#define NIDS_DATA 2（表示接受数据，在这个状态下可以判断是否有新的诗句到达，如果有就可以把数据存起来，可以在这个状态之中来分析TCP传输数据,此数据就存储在half_stream数据结构的缓存之中）;

#define NIDS_CLOSE 3（表示TCP连接正常关闭）;

#define NIDS_RESET 4（表示TCP连接被重置关闭）;

#define NIDS_TIMED_OUT 5（表示由于超时TCP连接被关闭）;

#define NIDS_EXITING 6（表示Libnids正在推出，在这个状态下可以最后一次使用存储在half_stream数据结构中的缓存数据）;

tuple4 的数据结构:

struct tuple4

{//下面以 TCP 连接为例

u_short source; //表示源 IP 地址的端口号

u_short dest;//表示目的 IP 地址的端口号

u_int saddr;//表示一个 TCP 连接的一端 IP 地址，称为源 IP 地址

u_int daddr;// 表示一个 TCP 连接的另一端 IP 地址，称为目的 IP 地址

}该数据结构描述的是一个地址端口对，表示发送方 IP 和端口以及接收方 IP 和端口。

half_stream 结构

该结构用来描述在 TCP 连接中一端的所有信息(可以是客户端也可以是服务器端)

struct half_stream

{char state;//表示套接字的状态，即 TCP 连接状态

char collect;//用来表示有数据到达，此数据存放在 data 成员中，当此数据可以忽略时就不需要存储

char collect_urg;//表示有紧急数据到达，此数据存在 urgdata 中

char *data;//存储正常接收到的数据

int offset;//存储在 data 中数据在第一个字节的偏移量

int count;//从 TCP 连接开始已经存储在 data 中的数据的字节数

int count_new;//有多少新的数据存储到 data 中

u_char urgdata; //存储紧急数据

u_char count_new_urg; //有新的紧急数据到达

……

}

tcp_stream 的结构

该结构中成员 client 表示客户端信息，成员 server 表示服务器端信息，都是 half_stream 类型的，所以 tcp_stream 数据结构描述了一个完整的 TCP 连接的所有信息。

```
tcp_stream
{struct tuple4 addr;
char nids_state;
struct lurker_nods *listeners;
struct half_stream client;
struct half_stream server;
struct tcp_stream *next_node;
struct tcp_strem *prev_nods;
int hash_index;
struct tcp_stream *next_time;
struct tcp_stream *prev_time;
int read;
struct tcp_stream *next_free;
};
```

nids_prm 结构描述了 Libnids 的一些全局参数信息，利用它可以对 Libnids 的一些环境参数进行设置。

nids_chksum_ctl 数据结构描述的是计算校验和

```
struct nids_chksum_ctl
{u_int netaddr;//地址
u_int mask;//掩码
```

u_int action;//动作，如果是NIDS_DO_CHKSUM，则表示计算校验和;
如果是NIDS_DONT_CHKSUM，则表示不计算校验和
u_int reserved;
}

Libnids的常用函数

1.基本函数:

（1）int nids_init(void);是对Libnids进行初始化，主要内容包括打开网络接口、打开文件、编译过滤规则、设置过滤规则、判断网络连接层类型、进行必要的初始化工作。

（2）void nids_run(void);试运行Libnids，进入循环捕获数据包状态。实际上是调用Libpcap函数pcap_loop()来循环捕获数据包。

（3）int nids_getfd(void);获得文件描述符号。

（4）int nids_dispatch(int cnt);调用Libpcap中的捕获数据包函数pcap_dispatch()。

（5）int nids_next(void);调用Libpcap中捕获数据包函数pcap_next()。

（6）void nids_register_chksum_ctl(struct nids_chksum_ctl *ptr,int nr);根据数据结构中的nids_chksum_ctl中的action进行决定是否计算校验和。

2.IP碎片函数

（1）void nids_register_if_frag(void(*));注册一个能够检测所有 IP 数据包的回调函数，其中的参数就是一个回调函数。

（2）void nids)register_if(void(*));此回调函数可以接收正常的 IP 数据包。

3.TCP 数据流重组函数

（1）void nids_register_tcp(void(*));注册一个 TCP 连接的回调函数，此回调函数接收的 TCP 数据存放在 half_stream 的缓存中。

（2）void nids_killtcp(struct tcp_stream *a_tcp);该函数实际上是调用 Libnet 的函数进行构造数据包并发送出去，使得 TCP 连接终止。

（3）void nids_discard(struct tcp_stream * a_tcp,int num);丢弃 num 字节 TCP 数据，以便存储更多的数据。

4. UDP 注册函数

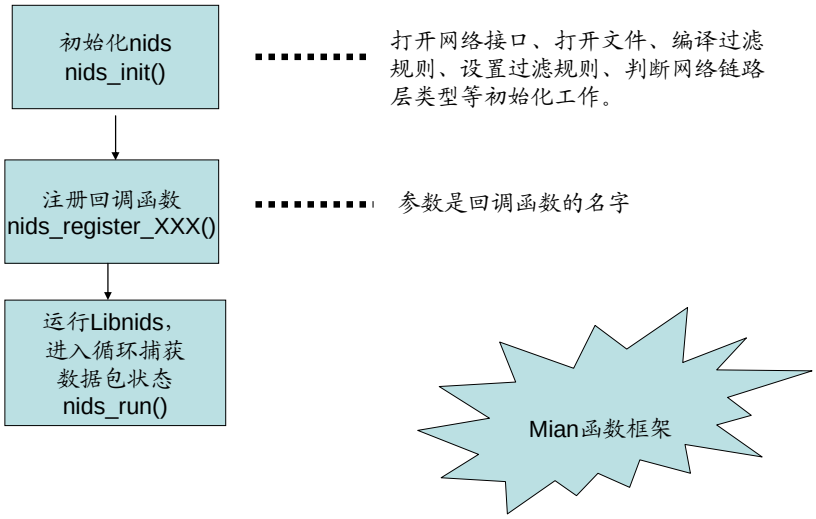
Libnids 不仅提供对 TCP 协议的分析，也提供对 UDP 协议的分析，注册函数的定义为 void nids_register_udp(void(*)),参数为 UDP 的回调函数，类型定义为:

void udp_callback(struct tuple4 * addr,char * buf,int len,struct ip * iph);

参数 tuple4 表示地址端口信息，buf 表示 UDP 协议负载数据内容，len 表示 UDP 负载数据的长度，iph 表示一个 IP 数据包（包括 IP 首部、UDP 首部以及 UDP 负载内容）。

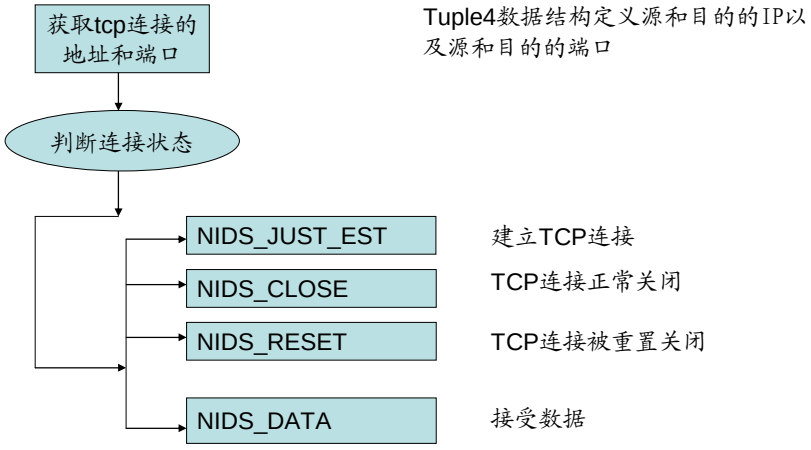
Libnids 的应用

Libnids应用的基本框架



callback()的框架

- 以TCP为例：



这里如果 Libnids 的状态为 NIDS_DATA 时，就进行如下的操作:

（1）先创建一个 half_stream 类型的对象;

（2）然后用 tcp_stream 判断服务器/客户端是否有新的数据到达?

是紧急数据还是普通数据?

(3)如果有,就把数据存储起来,并在这里可以对这些数据进行分析。

应用程序的编译和运行

在 Linux 下编译将会非常简单，使用 gcc 编译命令，使用到的开发包是 libnids、libpcap 和 libnet，它们的库名分别为 nids、pcap 和 net，执行命令如下:

```
gcc -o file_name.c -lnids -lpcap -lnet
```

！注意：在链接编译时，要注意先后顺序，越是底层库，位置越后。