## 基于 netfilter 修改 HTTP 数据包(插入、修改、删除)

### 一、 设计实现

#### 1. 概述:

测试内核: 3.13.0-32-generic (ubuntu14.04/centos7)

网关服务器上的内核模块,实现功能:

修改经过网关的 HTTP 数据包,修改前后数据长度可变。

#### tcp 数据包: next\_seq = 当前 seq + 当前数据包长度

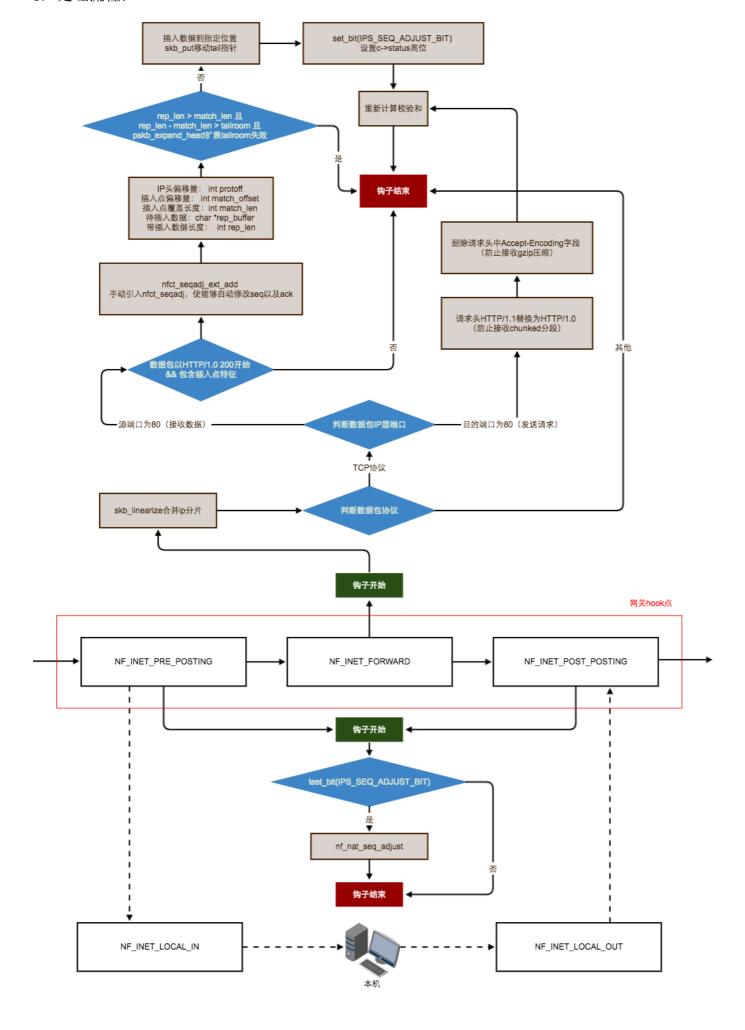
由于修改数据包长度后后续 tcp 连接 seq 以及 ack 都会发生变化,需要调用 nf\_conntrack 会话跟踪模块中的方法跟踪修改整个 tcp 连接后续的 seq 和 ack。

模块基于 netfilter hook 编写,在 NF\_INET\_FORWARD 数据包转发处挂在模块,hook 点信息:

### 2. 主要函数接口:

```
* Generic function for mangling variable-length address changes inside
* NATed TCP connections (like the PORT XXX,XXX,XXX,XXX,XXX,XXX
* command in FTP).
* */
static inline int nf_nat_mangle_tcp_packet(
             struct sk_buff *skb,
              struct nf_conn *ct,
              enum ip conntrack info ctinfo,
             // IP 头偏移量
              unsigned int protoff,
             // 插入点偏移量(距离 payload 首部)
             unsigned int match_offset,
             // 插入点覆盖长度
             unsigned int match_len,
             // 待插入数据
              const char *rep_buffer,
              // 带插入数据长度
              unsigned int rep_len)
```

#### 3. 处理流程:



### 二、注意事项

- 1. 在调用 nf\_nat\_mangle\_tcp\_packet 前需要调用 nfct\_seqadj\_ext\_add 将当前连接跟踪信息加入 seq 修正。
- 2. 修改数据包长度后 3.13 内核可自动修改 seq 和 ack

```
原理: nf_nat_mangle_tcp_packet 时会做以下处理 set_bit(IPS_SEQ_ADJUST_BIT, &ct->status)
后续内核跟踪钩子会修正 test_bit(IPS_SEQ_ADJUST_BIT, &ct->status)的 tcp 数据包的 seq 和 ack)
```

3. 内核版本 3.2 以及 2.68 以下只是在 ct->status 中高几位做了标记(IPS\_SEQ\_ADJUST\_BIT),还需要后续手动 hook 修改。

```
unsigned int fix_seq(unsigned int hooknum, struct sk_buff *skb,
  const struct net_device *in, const struct net_device *out, int(*okfn)(struct sk_buff
*))
{
   enum (ip_conntrack_info) ctinfo;
   struct nf_conn *ct = nf_ct_get(skb, &ctinfo);
   if (ct && test_bit(IPS_SEQ_ADJUST_BIT, &ct->status)
   && (ctinfo != IP_CT_RELATED + IP_CT_IS_REPLY) ) {
       nf_nat_seq_adjust(skb, ct, ctinfo);
   return NF_ACCEPT;
}
// 3.2 以及 2.68 以下内核适用
 {
     .hook
                   = fix seq,
                    = PF_INET,
     .pf
                   = NF_INET_PRE_ROUTING,
     .hooknum
     .priority
                   = NF_IP_PRI_CONNTRACK_CONFIRM,
     .owner
                    = THIS_MODULE
 },
     .hook
                   = fix seq,
     .pf
                   = PF_INET,
                   = NF_INET_POST_ROUTING,
     .hooknum
     .priority
                   = NF_IP_PRI_CONNTRACK_CONFIRM,
                    = THIS MODULE
     .owner
 },
```

- 4. 处理 gzip 压缩: 发送请求时删除请求头中 Accept-Encoding 字段, 防止收到 gzip 压缩包
- 5. 处理 chunked 分段传输: 发送请求时将 HTTP/1.1 修改为 HTTP/1.0 防止收到 chunked 数据包(Transfer-Encoding: chunked 是 HTTP 1.1 中特有的)

- 6. 对数据包进行修改需要重新计算 checksum
- 7. 处理完整数据包需要合并 ip 分片

```
// IP 数据包 frag 合并
if (0 != skb_linearize(skb)) {
  return NF_ACCEPT;
}
```

# 三、 使用方法

• 查看模块信息

# modinfo hook\_ipv4.ko

• 加载模块

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -j MASQUERADE
# cd hook_ipv4
# make
# insmod hook_ipv4.ko
```

• 卸载模块

# rmmod hook\_ipv4

• 查看日志

```
# tail /var/log/kern.log (ubuntu)
# tail /var/log/message.log (centos)
```