mangle 表主要用于修改数据包的 TOS（Type Of Service，服务类型）、TTL（Time To Live，生存周期）指以及为数据包设置 Mark 标记，以实现 Qos(Quality Of Service，服务质量)调整以及策略路由等应用，由于需要相应的路由设备支持，因此应用并不广泛

关于 mangle 模块，内核里主要有三个功能模块: mark match、MARK target 、CONNMARK target。

1）CONNMARK target 的选项

//调用

-j connmark

选项 功能

–set-mark value[/mask] 给链接跟踪记录打标记。

–save-mark [–mask mask] 将数据包上的标记值记录到链接跟踪记录上。

–restore-mark [–mask mask] 重新设置数据包的 nfmark 值。

2）MARK target 的选项

//调用 -j mark

选项 功能

–set-mark value 设置数据包的 nfmark 值。

–and-mark value 数据包的 nfmark 值和 value 进行按位与运算。

–or-mark value 数据包的 nfmark 值和 value 进行按或与运算。

3）MARK match 的选项 //需要调用 -m mark 模块

选项 功能

[!] –mark value[/mask] 数据包的 nfmark 值与 value 进行匹配，其中 mask 的值

为可选的

先说说 CONNMARK 和 MARK 的区别：

同样是打标记，但 CONNMARK 是针对连接的，而 MARK 是针对单一数据包的

这两种机制一般都要和 ip rule 中的 fwmark 联用，实现对满足某一类条件的数据包的策略路由

1.对连接打了标记，只是标记了连接，没有标记连接中的每个数据包。标记单个数据包，也不会对整条连接的标记有影响。二者是相对独立的

2. 路由判定(routing decision)是以单一数据包为单位的。或者说，在 netfilter 框架之外，并没有连接标记的概念。或者说，ip 命令只知道 MARK, 而不知道 CONNMARK 是什么。

3.关键在于：给所有要进行 ip rule 匹配的单一数据包打上标记。方法一般有二：用 MARK 直接打，或者用 CONNMARK –restore-mark 把打在连接上的标记转移到数据包上。

下面就代码分析一下：

mangle 它的模块代码在 iptable_mangle.c 中它的初始化工作和之前的 filter、nat 类似.Mangle 作用在所有的 hook 点

1.首先 iptable_mangle.c 的主要工作就是注册和初始化 mangle 表

注册 mangl 钩子函数.

2.其实我们发现除了 ct 的 hook 是单独处理外，其他的 filter、nat、mangle 都是通过 hook 之后调用 ipt_do_table 来处理，要么重点在 match 里，要么重点在 target 处理中.但是这个基本机制框架没变. 即都是通过 rules.

3.首先我们就看看 –j MARK –set-mark 1 这个 target 的执行

4.其实应该分析下规则的下发.

先看一条命令：

#iptables -t mangle -A PREROUTING -i eth0 -p tcp –dport 80 -j MARK –set-mark 1

即 target MARK

内核里是这样的

```
1    static unsigned int
2    mark_tg(struct sk_buff *skb, const struct xt_action_param *par
3    {
4        const struct xt_mark_tginfo2 *info = par->targinfo;
5
6        skb->mark = (skb->mark & ~info->mask) ^ info->mark;
7        return XT_CONTINUE;
8    }
```

它就是设置了 skb->mark 而已，并没有改变报文内容.

我们看看具体命令怎么配置：

mark match:它依赖 MARK target 先设置标记

#iptables -A POSTROUTING -t mangle -m mark ! –mark 0 -j ACCEPT

MARK target:

iptables -t mangle -A PREROUTING -i eth0 -p tcp –dport 80 -j MARK –set-mark 1

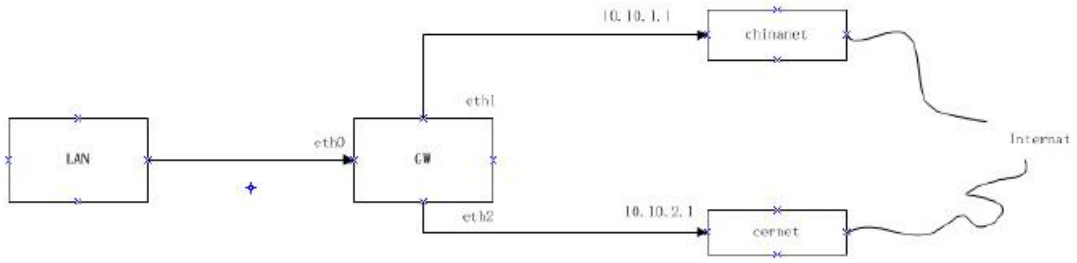
CONNMARK target：

#iptables -A POSTROUTING -t mangle -j CONNMARK –save-mark

然后我们看看几种应用场景：

1. 策略路由

现要求对内网进行策略路由，所有通过 TCP 协议访问 80 端口的数据包都从 ChinaNet 线路出去，而所有访问 UDP 协议 53 号端口的数据包都从 Cernet 线路出去




打标记:

iptables -t mangle -A PREROUTING -i eth0 -p tcp --dport 80 -j MARK --set-mark 1

iptables -t mangle -A PREROUTING -i eth0 -p udp --dprot 53 -j MARK --set-mark 2

建表：

ip rule add from all fwmark 1 table 10

ip rule add from all fwmark 2 table 20

1

2

ipruleaddfromallfwmark1table10

ipruleaddfromallfwmark2table20

策略路由:

```
1        ip route add default via 10.10.1.1 dev eth1 table 10
2        ip route add default via 10.10.2.1 dev eth2 table 20
```

2. CONNMARK 和 MARK 结合:

```
1.iptables-APOSTROUTING-tmangle-jCONNMARK--restore-mark
1
2
3
2.iptables-APOSTROUTING-tmangle-mmark!--mark0-jACCEPT
4
5
6
3.iptables-APOSTROUTING-mmark--mark0-ptcp--dport21-tmangle-jMARK--set-ma
```

4.iptables-APOSTROUTING-mmark--mark0-ptcp--dport80-tmangle-jMARK--set-ma

5.iptables-APOSTROUTING-mmark--mark0-tmangle-ptcp-jMARK--set-mark3

6.iptables-APOSTROUTING-tmangle-jCONNMARK--save-mark

1）第 1 条规则就是完成了将链接跟踪上的标记记录到该连接上的每一个数据包中；

2）第 2 条规则判断数据包的标记，如果不为 0，则直接 ACCEPT。如果数据包上没有被打标记，则交由后面的规则进行匹配并打标记。这里为什么会出现经过了 CONNMARK 模块，数据包仍未被打标记呢？可以想到，如果一条链接的第 1 个数据包经过第 1 条规则处理之后，由于 ct->mark 为 0，所以其数据包的标记 skb->nfmark 应该仍旧为 0，就需要进行后面规则的匹配。

3）第 3~5 条规则，则按照匹配选项对符合规则的数据包打上不同的标记

4)第 6 条规则，则是把之前打的标记信息保存到 ct 里.

3.结合 tc 的应用：

#tc fi add dev eth0 parent 1: pref 1000 protocol ip handle 13 fw flowid 1:1

1

#iptables -t mangle -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST ACK -m leng

2

13

题外扯点其他东西：

mangle 可以修改 IP 包头的 TTL 值,修改 ip 报头 dscp 值(QOS)

Iptables 下 MSS 数据调整模块 TCPMSS 使用

比如说 string 模块其实挺强大的，本身已经支持过来报文 data 里特征码，但是功能依然有缺陷所以才有了 l7 的插件.

```
1
        Iptables–AOUTPUT–mstring–string“www.baidu.com”–algobm–jDROP
```

关于 algo 参数的查找算法 它在内核里是通过 textsearch_register 注册的

内核默认注册了名字为 fsm、bm 、kmp

简单的就说这么多，具体应用需求还需要我们自己去挖掘和开发.