**Linux下使用libiptc库编程下发规则**

原创 2016年11月06日 09:59:58

* 标签：
* [iptables](http://so.csdn.net/so/search/s.do?q=iptables&t=blog) /
* [libiptc](http://so.csdn.net/so/search/s.do?q=libiptc&t=blog) /
* [netfilter](http://so.csdn.net/so/search/s.do?q=netfilter&t=blog) /
* [snat](http://so.csdn.net/so/search/s.do?q=snat&t=blog) /
* [dnat](http://so.csdn.net/so/search/s.do?q=dnat&t=blog)
* 1087

一、背景需求

近期编写tcp透明代理时，考虑了透明方式下需要使用 iptables 进行 DNAT/SNAT 规则下发，在程序中可以生成命令后通过 system 调用执行；

那么，iptables 是否有提供外部接口来代替上述这种 system() 执行命令的方式？答案是可以使用libiptc库进行编程实现。

二、相关知识

2.1 Linux/Netfilter

《Linux网络安全技术与实现》提到，Linux系统的防火墙功能主要为Netfilter模块进行实现，从功能上又细分了四大表：Filter、Nat、Mangle、Raw；

1）\*Filter表\*：数据包过滤功能；

2）\*Nat表\*：网络地址转换功能，IP分享器（一对一、一对多、多对多）功能；

3）Mangle表：修改经过防火墙的数据包功能；

4）Raw表：负责加快数据包穿越防火墙的速度，用于提高性能；

2.2 Netfilter/iptables

上述简单阐述了Netfilter在Linux内核防火墙的地位，而Netfilter工作在系统内核，规则是存在于内存中的，那么管理人员如何管理防火墙的规则呢？

这样的规则管理工具就是 iptables / ip6tables，比如下发一个规则到 Netfilter规则数据库中：

*iptables -t filter -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT*

三、实例

3.1 准备工作

内核版本：linux-3.10.25\_x64

源码版本：iptables-1.4.14(https://www.netfilter.org/projects/iptables/files/iptables-1.4.14.tar.bz2)

工具准备：gcc、automake、libtools(编译时需要把本机的libtools 放到 iptables 源码目录下)

3.2 接口说明

1) 获取当前防火墙规则快照 iptc\_init

**[html]** [view plain](http://blog.csdn.net/stayneckwind2/article/details/53053109) [copy](http://blog.csdn.net/stayneckwind2/article/details/53053109)

1. Name: iptc\_init
3. Usage: Takes a snapshot of the rules.
5. Description: This function must be called as initiator before any other function can be called.
7. Parameters: tablename is the name of the table we need to query and/or modify; this could be filter, mangle, nat, etc.
9. Returns: Pointer to a structure of type iptc\_handle\_t that must be used as main parameter for the rest of functions we will call from libiptc.
10. iptc\_init returns the pointer to the structure or NULL if it fails.
11. If this happens you can invoke iptc\_strerror to get information about the error.

2) 追加防火墙规则 iptc\_append\_entry

**[html]** [view plain](http://blog.csdn.net/stayneckwind2/article/details/53053109) [copy](http://blog.csdn.net/stayneckwind2/article/details/53053109)

1. Name: iptc\_append\_entry
3. Usage: Append a new rule in a chain.
5. Description: This function append a rule defined in structure type ipt\_entry in chain chain (equivalent to insert with rulenum = length of chain).
7. Parameters: chain is a char pointer to the name of the chain to be modified;
8. e is a pointer to a structure of type ipt\_entry that contains information about the rule to be appended.
9. The programmer must fill the fields of this structure with values required to define his or her rule before passing the pointer as parameter to the function.
10. handle is a pointer to a structure of type iptc\_handle\_t that was obtained by a previous call to iptc\_init.
12. Returns: Returns integer value 1 (true) if successful; returns integer value 0 (false) if fails.
13. In this case errno is set to the error number generated. Use iptc\_strerror to get a meaningful information about the problem.
14. If errno == 0, it means there was a version error (ie. upgrade libiptc).

3) 匹配删除防火墙规则 iptc\_delete\_entry

**[html]** [view plain](http://blog.csdn.net/stayneckwind2/article/details/53053109) [copy](http://blog.csdn.net/stayneckwind2/article/details/53053109)

1. Name: iptc\_delete\_entry
3. Usage: Delete the first rule in chain matched.
5. Description: This function delete the entry rule in chain chain matches rules. Rule numbers start at 1 for the first rule.
7. Parameters: chain is a char pointer to the name of the chain to be modified;
8. handle is a pointer to a structure of type iptc\_handle\_t that was obtained by a previous call to iptc\_init.
10. Returns: Returns integer value 1 (true) if successful;
11. returns integer value 0 (false) if fails. In this case errno is set to the error number generated.
12. Use iptc\_strerror to get a meaningful information about the problem.
13. If errno == 0, it means there was a version error (ie. upgrade libiptc).

3.3 编程实例

实例这块以SNAT、DNAT为例

1）生成规则 “iptables -p tcp -s src -d dst -j (SNAT|DNAT) --to to”，代码比较繁琐，需要设置好 ipt\_entry 结构体各个成员值；

**[cpp]** [view plain](http://blog.csdn.net/stayneckwind2/article/details/53053109) [copy](http://blog.csdn.net/stayneckwind2/article/details/53053109)

1. **struct** ipt\_entry \*api\_iptc\_entry\_get(**struct** sockaddr\_in src,
2. **struct** sockaddr\_in dst, **struct** sockaddr\_in nto, **const** **char** \*option)
3. {
4. **struct** ipt\_entry \*fw = NULL;
6. **struct** ipt\_entry\_match \*match = NULL;
7. **struct** ipt\_tcp \*tcpinfo = NULL;
9. **struct** ipt\_entry\_target \*target = NULL;
10. **struct** nf\_nat\_multi\_range \*mr = NULL;
12. u32 size1 = XT\_ALIGN(**sizeof**(**struct** ipt\_entry));
13. u32 size2 = XT\_ALIGN(**sizeof**(**struct** ipt\_entry\_match) + **sizeof**(**struct** ipt\_tcp));
14. u32 size3 = XT\_ALIGN(**sizeof**(**struct** ipt\_entry\_target) + **sizeof**(**struct** nf\_nat\_multi\_range));
16. **if** ( !option ) {
17. LOGW("NULL\n");
18. **return** NULL;
19. }
21. fw = calloc(1, size1 + size2 + size3);
22. **if** ( !fw ) {
23. LOGE("Malloc failure");
24. **return** NULL;
25. }
27. /\* Offsets to the other bits \*/
28. fw->target\_offset = size1 + size2;
29. fw->next\_offset = size1 + size2 + size3;
31. /\* Set up packet matching rules \*/
32. **if** ( (fw->ip.src.s\_addr = src.sin\_addr.s\_addr) == INADDR\_ANY ) {
33. fw->ip.smsk.s\_addr = 0;
34. }
35. **else** {
36. fw->ip.smsk.s\_addr = inet\_addr("255.255.255.255");
37. }
39. **if** ( (fw->ip.dst.s\_addr = dst.sin\_addr.s\_addr) == INADDR\_ANY ) {
40. fw->ip.dmsk.s\_addr = 0;
41. }
42. **else** {
43. fw->ip.dmsk.s\_addr = inet\_addr("255.255.255.255");
44. }
46. fw->ip.proto = IPPROTO\_TCP;
47. fw->nfcache = NFC\_UNKNOWN;  /\*Think this stops caching. \*/

50. /\* TCP specific matching(ie. ports) \*/
51. match = (**struct** ipt\_entry\_match \*)fw->elems;
52. match->u.match\_size = size2;
53. strcpy(match->u.user.name, "tcp");

56. tcpinfo = (**struct** ipt\_tcp \*)match->data;

59. **if** ( src.sin\_port == 0 ) {
60. tcpinfo->spts[0] = ntohs(0);
61. tcpinfo->spts[1] = ntohs(0xFFFF);
62. }
63. **else** {
64. tcpinfo->spts[0] = tcpinfo->spts[1] = ntohs(src.sin\_port);
65. }

68. **if**( dst.sin\_port == 0 ) {
69. tcpinfo->dpts[0] = ntohs(0);
70. tcpinfo->dpts[1] = ntohs(0xFFFF);
71. }
72. **else** {
73. tcpinfo->dpts[0] = tcpinfo->dpts[1] = ntohs(dst.sin\_port);
74. }

77. /\* And the target \*/
78. target = (**struct** ipt\_entry\_target \*)(fw->elems + size2);
79. target->u.target\_size = size3;
80. strcpy(target->u.user.name, option);

83. mr = (**struct** nf\_nat\_multi\_range \*)target->data;
84. mr->rangesize = 1;
86. mr->range[0].flags = IP\_NAT\_RANGE\_PROTO\_SPECIFIED | IP\_NAT\_RANGE\_MAP\_IPS;
87. mr->range[0].min.tcp.port = mr->range[0].max.tcp.port = nto.sin\_port;
88. mr->range[0].min\_ip = mr->range[0].max\_ip = nto.sin\_addr.s\_addr;    **return** fw;
89. }

2）追加规则到 Netfilter，需要指定 chain名 “PREROUTING” 或者 “POSTROUTING”

**[cpp]** [view plain](http://blog.csdn.net/stayneckwind2/article/details/53053109) [copy](http://blog.csdn.net/stayneckwind2/article/details/53053109)

1. **int** api\_iptc\_entry\_add(**const** **struct** ipt\_entry \*fw, **const** **char** \*chain)
2. {
3. **int** ret = FAILURE;
4. **struct** xtc\_handle \*phandle = NULL;
6. **if** ( !fw || !chain ) {
7. LOGW("NULL\n");
8. **return** FAILURE;
9. }
11. **if** ( (phandle = iptc\_init("nat")) &&
12. iptc\_append\_entry(chain, fw, phandle) &&
13. iptc\_commit(phandle) ) {
14. ret = SUCCESS;
15. }
16. **else** {
17. LOGW("%s\n", iptc\_strerror(errno));
18. ret = FAILURE;
19. }
20. **return** ret;
21. }

3）删除规则

**[cpp]** [view plain](http://blog.csdn.net/stayneckwind2/article/details/53053109) [copy](http://blog.csdn.net/stayneckwind2/article/details/53053109)

1. **int** api\_iptc\_entry\_del(**const** **struct** ipt\_entry \*fw, **const** **char** \*chain)
2. {
3. **int** ret = FAILURE;
5. unsigned **char** \*matchmask = NULL;
7. **struct** xtc\_handle \*phandle = NULL;
9. **if** ( !fw || !chain ) {
10. LOGW("NULL\n");
11. **return** FAILURE;
12. }
14. matchmask = calloc(1, fw->next\_offset);
15. **if** ( !matchmask ) {
16. **return** FAILURE;
17. }
19. **if** ( !phandle ) {
20. phandle = iptc\_init("nat");
21. }
23. **if** ( (phandle = iptc\_init("nat")) &&
24. iptc\_delete\_entry(chain, fw, matchmask, phandle) &&
25. iptc\_commit(phandle) ) {
26. ret = SUCCESS;
27. }
28. **else** {
29. LOGW("%s\n", iptc\_strerror(errno));
30. ret = FAILURE;
31. }
33. FREE\_POINTER(matchmask);
34. **return** ret;
35. }

4）使用实例

**[cpp]** [view plain](http://blog.csdn.net/stayneckwind2/article/details/53053109) [copy](http://blog.csdn.net/stayneckwind2/article/details/53053109)

1. **int** test\_snat(**int** off, u16 times)
2. {
3. **int** ret = FAILURE;
4. **int** ix = 0;
6. **struct** ipt\_entry \*fw = NULL;
8. **struct** sockaddr\_in src;
9. **struct** sockaddr\_in dst;
10. **struct** sockaddr\_in sto;
12. src.sin\_addr.s\_addr = inet\_addr("1.1.1.1") + off;
13. dst.sin\_addr.s\_addr = inet\_addr("1.1.1.2") + off;
14. sto.sin\_addr.s\_addr = inet\_addr("1.1.1.3") + off;
16. **for** ( ix = 0; ix < times; ix++ ) {
17. src.sin\_port = htons(1 + ix);
18. dst.sin\_port = htons(2 + ix);
19. sto.sin\_port = htons(3 + ix);
21. ASSERT\_FAIL(NULL, fw = api\_iptc\_entry\_get(src, dst, sto, "SNAT"));
22. ASSERT(SUCCESS, api\_iptc\_entry\_add(fw, "POSTROUTING"));
24. usleep(300);
25. ASSERT(SUCCESS, api\_iptc\_entry\_del(fw, "POSTROUTING"));
26. }
28. FREE\_POINTER(fw);
29. ret = SUCCESS;
30. \_E1:
31. **return** ret;
32. }

4 总结

  Linux c/c++编程使用 libiptc库下发规则可以代替system调用 iptables 命令方法，可以满足程序中批量地创建连接所需的批量增加、删除iptables规则；

使用步骤确实比较繁琐，得注意iptables版本的问题，编译相关库需要“-lip4tc” “-lxtables”，头文件需要“libiptc/libiptc.h” “net/netfilter/nf\_nat.h” “iptables.h”；

然而实验结果表明，当多进程高频率得操作iptables时，system 和 libiptc 均会出现 “Resource temporarily unavailable” 错误，建议进行返回值判定和多进程的加锁保护；

参考文章：

[1] Linux网络安全技术与实现

[2] http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html\_single/Querying-libiptc-HOWTO.html

[3] http://wiki.tldp.org/iptc%20library%20HOWTO

[4] http://blog.chinaunix.net/uid-9950859-id-98280.html

[5] http://blog.chinaunix.net/uid-23069658-id-3223404.html