一点 ebtables 的心得

项目用到了 ebtables，抽空研究了以下。可能是因为 ebtables 在实际运用中使用得比较少的原因吧，提供的接口没有 iptables 方便。

iptables 提供一个 libiptc 的库，其中提供获取策略列表、修改策略列表、实施策略等相关函数，二次开发者可以很简单使用这些函数运用自定义模块。

两者都提供很好的扩展机制，用于解析自定义的过滤或者目标(ebtables 还有 watch 类型)模块。但很多时候为了效率，并不推荐使用程序里面还使用 shell 命令 ,特别像很多新手还喜欢疯狂的使用 system()函数 - - 当然这个函数本身并没有什么问题，但是我们很难获取运行结果 - - 扯远了。

以前在看 iptables 代码时没有太认真，而且当时自己也很缺乏这方面的知识，因为 ebtables 代码比较少，可以比较快的整理出它的整体结构

关于扩展模块支持，最重要的问题还是怎么注册一个解析器，也就是怎么维护这些加载了的插件呢？以前曾经想写具有扩展功能的程序 ,才发现这的确是个问题。iptables/ebtables 都一样 - - 把管理插件的功能函数放到一个动态库中，在动态库中维护插件列表。在编译是首先编译这个动态库在编译主程序和扩展模块的时候就可以使用这个动态库来编译如此就可以编译通过。在运行期间使用 dlopen 来加载扩展模块，而模块被加载时系统会自动调用该模块中的 void_init(void)函数，只要在这个函数内注册自己就可以达到我们的目的。而在

使用 dlclose 卸载模块时，系统会调用 void_fini(void)函数，在这个函数中注销自己

ebtables 与内核交互格式:

```
    ebt_replace|ebt_entries|ebt_entry|ebt_entry|...|ebt_entries|ebt_entry|...|ebt_entries|...
```

ebtables 中的几个结构说明：

```
    ebt_replace

    name 表名(filternatbroute)

    valid_hooks 掩码形式表示在使用的钩子

    nentries 策略条目总数

    entries_size 策略总大小(包括所有 ebt_entries 和所有 ebt_entry 的大小)

    hook_entry 最重要的部分确定每个 chain(也就是钩子)策略的起始位置

    本以为会使用偏移量最终发现它使用的实际地址位置

    num_counters 用户态希望计数器数量

    counters 内核返回计数器位置

    entries 策略开始指针没有使用 char[0]的标签功能
```

老实说个人对这个结构实在不太恭维总体上来说容错还是很不错的几个地方(hook_entry 和 entries)没有使用偏移量表示而是使用了用户态指针估计是考

虑到整个结构比较庞大可以不必使用完全的连续内存吧。这种做法是仁者见仁智者见智了。

本人认为既然最大内存使用的策略部分已经要求连续内存了又何必对此一举呢？

ebt_entries

distinguisher 兼容部分直接飘过 zeroit

namechain 名称(filter:PREROUTINGINPUTOUTPUTPOSTROUTING

nat:PREROUTINGINPUTOUTPUTPOSTROUTINGbroute:BROUTING)

counter_offset

policychain 默认的策略可以是 acceptdropreturn 就是对待数据包最后方式了

nentries 本 chain 的策略总数

data[0]这地方就是说明需要连续内存地址了

简单明了的结构但是在这我还是犯了困 counter_offset 内核注释是这样的 counteroffsetforthischain 按理说:总的 counter 是 0 这地方怎么会不是 0 呢可惜它就是不是 0 它等于前面所有 chain 的策略总数到现在还是没想明白这是为什么

ebt_entry

bitmask 有效位置掩码

invflags 无效位置掩码

ethproto 以太网协议类型

in 输入设备名

logical_in 逻辑输入设备

out 输出设备名

logical_out 逻辑输出设备名

sourcemac 源 MAC 地址

sourcemask 源 MAC 地址掩码

destmac 目标 MAC 地址

destmask 目标 MAC 地址掩码

watchers_offset 该策略 watcher 的偏移地址(这里是相对偏移量)

target_offset 该策略 target 的偏移地址

next_offset 下一条策略的偏移地址(实际上是本策略的总长度)

每条策略都必须以 ebt_entry 开头策略可以包含多个 match 和多个 watcher 但是只能有一个 target

match 是紧跟着 ebt_entry 没有设置其的起始

watcher 的位置由 watchers_offset 指定如果没有那么它应该与 target_offset 相等

target 任何策略都“必须”有一个 target

这部分与 iptables 相似只是多了 watcher 以及其中的内容不同

还有一点重大区别是 ebtalbes 并没有对 matchwatcher 和 target 的内容长度进行对齐

Linux 下实现桥式防火墙的两个组件：

bridge：http://linux-net.osdl.org/index.php/Bridge

ebtables：http://ebtables.sourceforge.net/