Netfilter

By
V.R.Sundar
&
Karthik Dantu

What is Netfilter

- netfilter is a framework for packet mangling, outside the normal Berkeley socket interface.
- Using this framework various modules have been written including an extensible NAT system and an extensible packet filtering system

Topics

- netfilter overview
- iptables overview
- netfilter & iptable implementation details with ipv4 & packet filtering as examples

netfilter has three parts :

I. Each protocol defines "hooks"
- Hooks are well defined points in a packet's traversal of that protocol's stack.
- The protocol code jumps into netfilter when it hits the hook-point during a packet's traversal through the protocol

II. Kernel modules can register to listen at any of the different hooks for each protocol.
- When registering, module must set the priority of the function within the hook
- netfilter hook is called from the core networking code
  - The corresponding packet is passed to it
  - Each module registered for that hook is called in the order of priorites, and is free to manipulate the packet.
- The module can tell netfilter to do one of

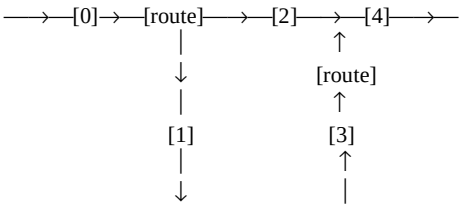
five things:

1. NF_ACCEPT: continue traversal as normal
2. NF_DROP: drop the packet; don't continue traversal.
3. NF_STOLEN: I've taken over the packet; don't continue traversal.
4. NF_QUEUE: queue the packet (usually for userspace handling).
5. NF_REPEAT: call this hook again.

III. Packets that have been queued are collected (by the ip_queue driver) for sending to userspace; these packets are handled asynchronously.

Example

IPV4 defines 5 hooks. A packet traverses the netfilter system as shown



Hook No	Name
0	NF_IP_PRE_ROUTING
1	NF_IP_LOCAL_IN
2	NF_IP_FORWARD
3	NF_IP_LOCAL_OUT
4	NF_IP_POST_ROUTING

To summarize:

netfilter is the “glue” code between the protocol hooks and the kernel modules that want to process packets at these hooks. Control jumps from the protocol code to the netfilter code to the various modules registered and then flows back

IPTables overview

- IPTables is a packet selection system that has been built over the netfilter framework
- Provides a named array of rules in memory along with information as to where packets for a particular hook should begin traversal and various helper routines to traverse them
- What is a rule?

  A rule is the set of conditions to be matched and the corresponding actions to be taken
- Each hook has a list of rules called chains

- Kernel modules can register a new table, and ask for a packet to traverse a given table
- Example tables:
  - “filter” for packet filtering
  - ”mangle” for packet mangling
  - “nat” for network address translation.

These are defined by default and cannot be removed. (see net/ipv4/netfilter/ip_tables.c)

iptable – netfilter interaction

- iptables does not register with any netfilter hooks. It relies on the modules that use it to do that and feed it the packets as appropriate. A module must register the netfilter hooks and iptables separately, and provide the mechanism to call iptables when the hook is reached.

There is a iptables tool which can be used to control the packet filtering.

Example:

$iptables –A INPUT –s 127.0.0.1 –p icmp –j drop

netfilter implementation

- Main data structures
  - struct list_head nf_hooks[NPROTO][NF_MAX_HOOKS];

    A 2-dim array where each element is the head of a circular list. Uses the list implementation of linux to manage circular lists.

```
struct list_head {
  struct list_head *next, *prev;
};
```

- Objects of type nf_hook_ops are hooked into the chain using the list_head element at the head of it through the list.h helper functions

```
struct nf_hook_ops {
    struct list_head list;
    /* User fills in from here down. */
    nf_hookfn *hook;
    int pf;
    int hooknum;
    /* Hooks are ordered in ascending priority. */
    int priority;
};
```

- This creates a 2-dimensional array of circular linked lists indexed by
  - (protocol number , hook number)
- Example
  - nf_hooks[PF_INET][NF_IP_LOCAL_IN] is the head of the circualar list for hook 1 (local input hook).
  - Traverse this list using element list of nf_hook_ops structure
  - Access registered hook functions through element hook

- nf_register_hook registers a module
- nf_unregister_hook unregisters a module
  - These two are called by the modules that want to use netfilter
- nf_hook_slow is the interface between networking code and netfilter. (Through macro NF_HOOK)
  - Example: In file net/ipv4/ip_input.c

```
int ip_local_deliver(struct sk_buff *skb) {
  ……
  return
NF_HOOK(PF_INET,NF_IP_LOCAL_IN,
skb, skb->dev, NULL,ip_local_deliver_finish);
  }
```

- nf_iterate iterates through the modules registered . Called from nf_hook_slow

iptables

- iptables is a collection of tables along with the utility functions to manipulate those
- Each table is a collection of rules for the various hooks that table is used for and is represented by

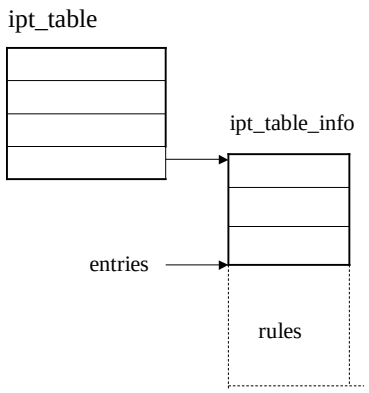
```
struct ipt_table {
    struct list_head list;
    char name[IPT_TABLE_MAXNAMELEN];
    struct ipt_replace *table;
    unsigned int valid_hooks;
    rwlock_t lock;
    struct ipt_table_info *private;
};
```

- ipt_table does not store rules directly
- Uses the “private” element

```
struct ipt_table_info {
    unsigned int size;
    unsigned int number;
unsigned int hook_entry[NF_IP_NUMHOOKS];
unsigned int underflow[NF_IP_NUMHOOKS];
    char entries[0]
__attribute__((aligned(SMP_CACHE_BYTES)));
};
```

- ipt_table_info too does not have any space for entried
- Some interesting members are
  - hook_entry: offsets to the start of the rules for the various hooks
  - entries: a zero size element at the end essentially pointing to the end of the structure
- These two together are used to created and traverse the rules.

- translate_table copies the list of rules immediately after the ipt_table_info structure
- The top of the list is now available through the “entries” element
- hook_entry array elements are set up with the offset to the beginning of the corresponding hook’s rules



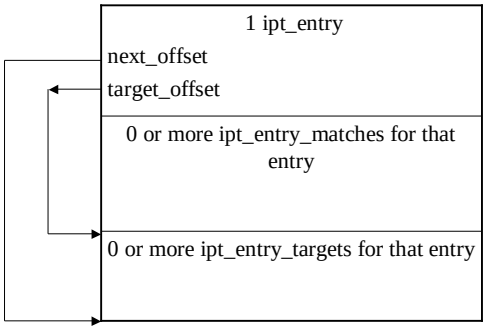
- A rule is combination of 3 data structures
  - struct ipt_entry
  - struct ipt_entry_match
  - struct ipt_entry_target
- The ipt_entry structure contains the criteria that is to be matched
- The ipt_entry_match and ipt_entry_target structures are similar and contain match/target functions to be called

```
struct ipt_entry
{
    struct ipt_ip ip;
    unsigned int nfcache;
    u_int16_t target_offset;
    u_int16_t next_offset;
    unsigned int comefrom;
    struct ipt_counters counters;
    unsigned char elems[0];
};
```

- struct ipt_entry_match {
  union {
  struct {
  u_int16_t match_size;
  char name[IPT_FUNCTION_MAXNAMELEN];
  } user;
  struct {
  u_int16_t match_size;
  struct ipt_match *match;
  } kernel;
  u_int16_t match_size;
  } u;
  unsigned char data[0];
  };

- struct ipt_entry_target {
  union {
  struct {
  u_int16_t target_size;
  char name[IPT_FUNCTION_MAXNAMELEN];
  } user;
  struct {
  u_int16_t target_size;
  struct ipt_target *target;
  } kernel;
  u_int16_t target_size;
  } u;
  unsigned char data[0];
  };

- A rule looks like



- A table is registered through a call to ipt_register_table.
- This in turn calls translate_table which
  - sets up the ipt_table_info structure after error checking
  - Makes one copy of the rules per cpu.
- To iterate through a table helper macros are defined
  - IPT_ENTRY_ITERATE
  - IPT_MATCH_ITERATE
  - IPT_TARGET_ITERATE

http://netfilter.samba.org