NAME

ipf - packet filtering kernel interface

SYNOPSIS

#include <netinet/ip_compat.h>
#include <netinet/ip_fil.h>

IOCTLS

To add and delete rules to the filter list, three 'basic' ioctls are provided for use. The ioctl's are called as:

ioctl(fd, SIOCADDFR, struct frentry **)
ioctl(fd, SIOCDELFR, struct frentry **)
ioctl(fd, SIOCIPFFL, int *)

However, the full complement is as follows:

```
ioctl(fd, SIOCADAFR, struct frentry **) (same as SIOCADDFR)
ioctl(fd, SIOCRMAFR, struct frentry **) (same as SIOCDELFR)
ioctl(fd, SIOCADIFR, struct frentry **)
ioctl(fd, SIOCRMIFR, struct frentry **)
ioctl(fd, SIOCINAFR, struct frentry **)
ioctl(fd, SIOCINIFR, struct frentry **)
ioctl(fd, SIOCSETFF, u_int *)
ioctl(fd, SIOGGETFF, u_int *)
ioctl(fd, SIOCGETFS, struct friostat **)
ioctl(fd, SIOCIPFFL, int *)
ioctl(fd, SIOCIPFFB, int *)
ioctl(fd, SIOCSWAPA, u_int *)
ioctl(fd, SIOCFRENB, u_int *)
ioctl(fd, SIOCFRSYN, u_int *)
ioctl(fd, SIOCFRZST, struct friostat **)
ioctl(fd, SIOCZRLST, struct frentry **)
ioctl(fd, SIOCAUTHW, struct fr info **)
ioctl(fd, SIOCAUTHR, struct fr_info **)
ioctl(fd, SIOCATHST, struct fr_authstat **)
```

The variations, SIOCADAFR vs. SIOCADIFR, allow operation on the two lists, active and inactive, respectively. All of these ioctl's are implemented as being routing ioctls and thus the same rules for the various routing ioctls and the file descriptor are employed, mainly being that the fd must be that of the

device associated with the module (i.e., /dev/ipl).

The three groups of ioctls above perform adding rules to the end of the list (SIOCAD*), deletion of rules from any place in the list (SIOCRM*) and insertion of a rule into the list (SIOCIN*). The rule place into which it is inserted is stored in the "fr_hits" field, below.

```
typedef struct frentry {
    struct frentry *fr next;
    u_short fr_group;
                          /* group to which this rule belongs */
    u_short fr_grhead;
                          /* group # which this rule starts */
    struct frentry *fr_grp;
                      /* reference count - for grouping */
    int fr_ref;
    void *fr ifa;
#ifdef BSD
    void *fr oifa;
#endif
     /*
     * These are only incremented when a packet matches this rule and
     * it is the last match
     */
     U QUAD T
                      fr hits;
    U_QUAD_T
                      fr_bytes;
     /*
     * Fields after this may not change whilst in the kernel.
     */
    struct fr_ip fr_ip;
    struct fr_ip fr_mip; /* mask structure */
                          /* tcp flags mask */
    u_char fr_tcpfm;
    u_char fr_tcpf;
                         /* tcp flags */
    u_short fr_icmpm;
                           /* data for ICMP packets (mask) */
    u_short fr_icmp;
                          /* data for port comparisons */
    u_char fr_scmp;
    u_char fr_dcmp;
    u_short fr_dport;
    u_short fr_sport;
    u_short fr_stop;
                         /* top port for <> and >< */
                         /* top port for <> and >< */
     u_short fr_dtop;
```

```
u_32_t fr_flags; /* per-rule flags && options (see below) */
u_short fr_skip; /* # of rules to skip */
u_short fr_loglevel; /* syslog log facility + priority */
int (*fr_func)(int, ip_t *, fr_info_t *));
char fr_icode; /* return ICMP code */
char fr_ifname[IFNAMSIZ];
#ifdef BSD
char fr_oifname[IFNAMSIZ];
#endif
struct frdest fr_tif; /* "to" interface */
struct frdest fr_dif; /* duplicate packet interfaces */
} frentry_t;
```

When adding a new rule, all unused fields (in the filter rule) should be initialised to be zero. To insert a rule, at a particular position in the filter list, the number of the rule which it is to be inserted before must be put in the "fr_hits" field (the first rule is number 0).

Flags which are recognised in fr_flags:

FR_BLOCK 0x000001 /* do not allow packet to pass */
FR_PASS 0x000002 /* allow packet to pass */
FR_OUTQUE 0x000004 /* outgoing packets */
FR_INQUE 0x000008 /* ingoing packets */
FR_LOG 0x000010 /* Log */
FR_LOGB 0x000011 /* Log-fail */
FR_LOGP 0x000012 /* Log-pass */
FR_LOGBODY 0x000020 /* log the body of packets too */
FR_LOGFIRST 0x000040 /* log only the first packet to match */
FR_RETRST 0x000080 /* return a TCP RST packet if blocked */
FR_RETICMP 0x000100 /* return an ICMP packet if blocked */
FR_FAKEICMP 0x00180 /* Return ICMP unreachable with fake source */
FR_NOMATCH 0x000200 /* no match occurred */
FR_ACCOUNT 0x000400 /* count packet bytes */
FR_KEEPFRAG 0x000800 /* keep fragment information */
FR_KEEPSTATE 0x001000 /* keep 'connection' state information */
FR_INACTIVE 0x002000
FR_QUICK 0x004000 /* match & stop processing list */
FR_FASTROUTE 0x008000 /* bypass normal routing */
FR_CALLNOW 0x010000 /* call another function (fr_func) if matches */
FR_DUP 0x020000 /* duplicate the packet */

FR_LOGORBLOCK0x040000/* block the packet if it can't be logged */FR_NOTSRCIP0x080000/* not the src IP# */FR_NOTDSTIP0x100000/* not the dst IP# */FR_AUTH0x200000/* use authentication */FR_PREAUTH0x400000/* require preauthentication */

Values for fr_scomp and fr_dcomp (source and destination port value comparisons) :

FR_NONE 0 FR_EQUAL 1 2 FR_NEQUAL FR LESST 3 FR_GREATERT 4 FR LESSTE 5 FR GREATERTE 6 FR_OUTRANGE 7 FR_INRANGE 8

The third ioctl, SIOCIPFFL, flushes either the input filter list, the output filter list or both and it returns the number of filters removed from the list(s). The values which it will take and recognise are FR_INQUE and FR_OUTQUE (see above). This ioctl is also implemented for **/dev/ipstate** and will flush all state tables entries if passed 0 or just all those which are not established if passed 1.

General Logging Flags

There are two flags which can be set to log packets independently of the rules used. These allow for packets which are either passed or blocked to be logged. To set (and clear)/get these flags, two ioctls are provided:

SIOCSETFF	Takes an unsigned integer as the parameter. The flags are then set to those provided (clearing/setting all in one).		
	FF_LOGPASS 0x100	00000	
	FF_LOGBLOCK	0x20000000	
	FF_LOGNOMATCH	0x40000000	
	FF_BLOCKNONIP	0x80000000	/* Solaris 2.x only */
SIOCGETFF	Takes a pointer to an unsigned currently in used is copied to us	integer as the p ser space.	parameter. A copy of the flags

Filter statistics

Statistics on the various operations performed by this package on packets is kept inside the kernel. These statistics apply to packets traversing through the kernel. To retrieve this structure, use this ioctl:

```
ioctl(fd, SIOCGETFS, struct friostat *)
```

```
struct friostat {
```

```
struct filterstats
                   f_st[2];
struct frentry
                   *f_fin[2];
struct frentry
                   *f_fout[2];
struct frentry
                   *f_acctin[2];
struct frentry
                   *f_acctout[2];
struct frentry
                   *f_auth;
u_long f_froute[2];
                   /* 1 or 0 - active rule set */
     f active;
int
                   /* default pass - from fr_pass */
int
     f_defpass;
int
     f_running;
                    /* 1 if running, else 0 */
int
    f_logging;
                    /* 1 if enabled, else 0 */
char f_version[32]; /* version string */
```

};

struct filterstats {	
u_long fr_pass;	/* packets allowed */
u_long fr_block;	/* packets denied */
u_long fr_nom;	/* packets which don't match any rule */
u_long fr_ppkl;	/* packets allowed and logged */
u_long fr_bpkl;	/* packets denied and logged */
u_long fr_npkl;	/* packets unmatched and logged */
u_long fr_pkl;	/* packets logged */
u_long fr_skip;	/* packets to be logged but buffer full */
u_long fr_ret;	/* packets for which a return is sent */
u_long fr_acct;	/* packets for which counting was performed */
u_long fr_bnfr;	/* bad attempts to allocate fragment state */
u_long fr_nfr;	/* new fragment state kept */
u_long fr_cfr;	/* add new fragment state but complete pkt */
u_long fr_bads;	/* bad attempts to allocate packet state */
u_long fr_ads;	/* new packet state kept */
u_long fr_chit;	/* cached hit */
u_long fr_pull[2];	/* good and bad pullup attempts */
#if SOLARIS	

u_longfr_notdata;/* PROTO/PCPROTO that have no data */u_longfr_nodata;/* mblks that have no data */u_longfr_bad;/* bad IP packets to the filter */u_longfr_notip;/* packets passed through no on ip queue */u_longfr_drop;/* packets dropped - no info for them! */

#endif

};

If we wanted to retrieve all the statistics and reset the counters back to 0, then the ioctl() call would be made to SIOCFRZST rather than SIOCGETFS. In addition to the statistics above, each rule keeps a hit count, counting both number of packets and bytes. To reset these counters for a rule, load the various rule information into a frentry structure and call SIOCZRLST.

Swapping Active lists

IP Filter supports two lists of rules for filtering and accounting: an active list and an inactive list. This allows for large scale rule base changes to be put in place atomically with otherwise minimal interruption. Which of the two is active can be changed using the SIOCSWAPA ioctl. It is important to note that no passed argument is recognised and that the value returned is that of the list which is now inactive.

FILES

/dev/ipauth /dev/ipl /dev/ipnat /dev/ipstate

SEE ALSO

ipl(4), ipnat(4), ipf(5), ipf(8), ipfstat(8)