LINUX PLUMBERS CONFERENCE September 20-24, 2021

# bpfilter – BPF based firewall

Dmitrii Banshchikov

# Agenda

- How iptables works?
- How bpfilter works?

# How iptables works

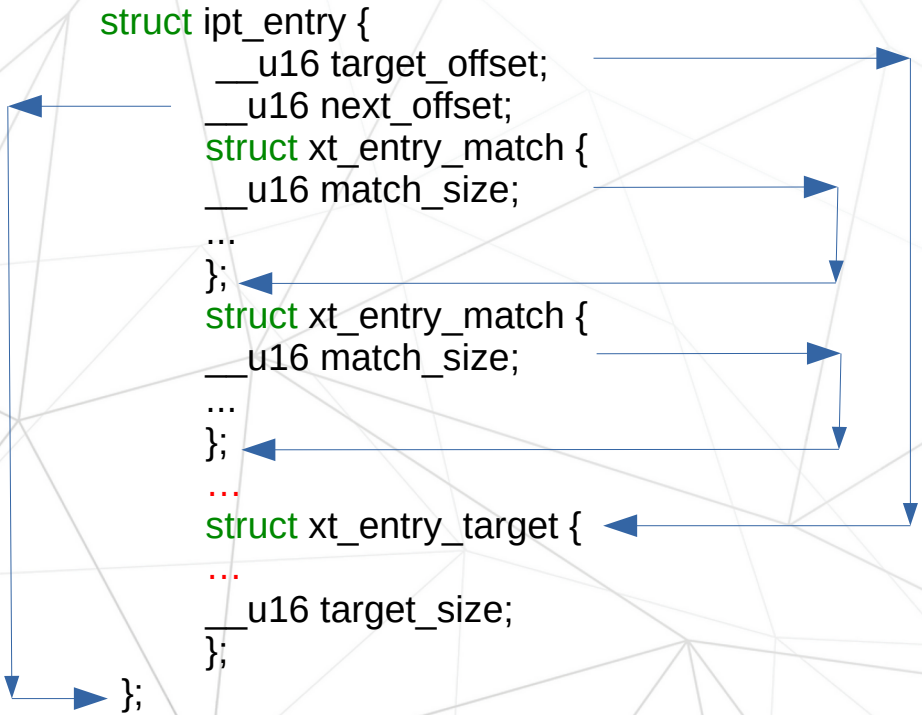
- Tables: filter, nat, mangle, ...
- Hooks: input, output, forward, ...
- Chains: DAG of rules
- Rule: set of matches and a target
- iptables -A INPUT -s 2.2/16 -d 3/8 -p udp –dport 25 -j DROP

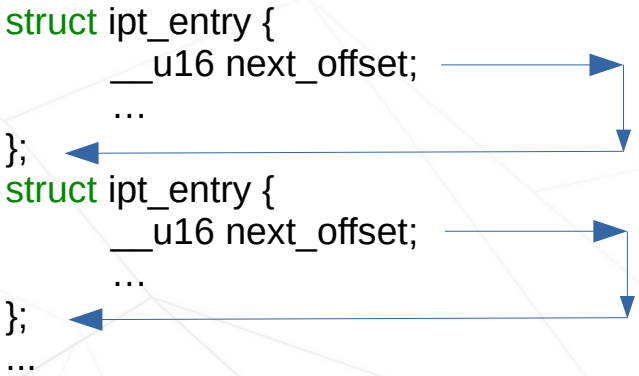
# Data structures

```
struct ipt_entry {
        struct ipt_ip ip;
        unsigned int nfcache;
         __u16 target_offset;
        __u16 next_offset;
        unsigned int comefrom;
        struct xt_counters counters;
        unsigned char elems[0];
};
```

```
struct ipt_ip {
        struct in_addr src, dst;
        struct in_addr smsk, dmsk;
        char iniface[16];
        char outiface[16];
        unsigned char iniface_mask[16];
        unsigned char outiface_mask[16];
        __u16 proto;
         __u8 flags;
        __u8 invflags;
};
```

# Data structures



# Data structures

```
struct xt_entry_match {
        union {
                struct {
                        __u16 match_size;
                        char name[29];
                        __u8 revision;
                } user;
                struct {
                        __u16 match_size;
                        struct xt_match *match;
                } kernel;
                __u16 match_size;
        } u;
        unsigned char data[0];
};
```

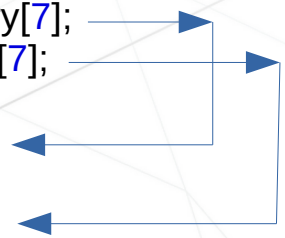
```
struct xt_entry_target {
        union {
                struct {
                        __u16 target_size;
                        char name[29];
                        __u8 revision;
                } user;
                struct {
                        __u16 target_size;
                        struct xt_target *target;
                } kernel;
                __u16 target_size;
        } u;
        unsigned char data[0];
};
```

# Data structures

```
struct xt_match {
        ...
        bool (*match)(const struct sk_buff *skb, struct xt_action_param *);
        int (*checkentry)(const struct xt_mtchk_param *);
        void (*destroy)(const struct xt_mtdtor_param *);
        ...
};

struct xt_target {
        ...
        unsigned int (*target)(struct sk_buff *skb, const struct xt_action_param *);
        int (*checkentry)(const struct xt_tgchk_param *);
        void (*destroy)(const struct xt_tgdtor_param *);
        ...
};
```

# Data structures

```
struct xt_table_info {
        ...
        unsigned int hook_entry[7];
        unsigned int underflow[7];
        ...
        struct ipt_entry entry;
        ...
        ...
        ...
        struct ipt_entry entry;
 };
```



# Rules interpretation

```
do {
        const struct xt_entry_target *t;
        const struct xt_entry_match *ematch;

        if (!ip_packet_match(ip, indev, outdev, &e->ip, acpar.fragoff)) {
no_match:
            e = ipt_next_entry(e);
            continue;
        }

        xt_ematch_foreach(ematch, e) {
                if (!ematch->u.kernel.match->match(skb, ...))
                        goto no_match;
        }

        t = ipt_get_target_c(e);
...
```

# Rules interpretation

```
/* Returns whether matches rule or not. */
/* Performance critical - called for every packet */
static inline bool
ip_packet_match(const struct iphdr *ip,
                const char *indev,
                const char *outdev,
                const struct ipt_ip *ipinfo,
                int isfrag)
{
        unsigned long ret;

        if (NF_INVF(ipinfo, IPT_INV_SRCIP,
                    (ip->saddr & ipinfo->smsk.s_addr) != ipinfo->src.s_addr) ||
            NF_INVF(ipinfo, IPT_INV_DSTIP,
                    (ip->daddr & ipinfo->dmsk.s_addr) != ipinfo->dst.s_addr))
                return false;

        …
```

# Kernel API

- setsockopt(2)
- IPT_SO_GET_INFO
- IPT_SO_GET_ENTRIES
- IPT_SO_SET_REPLACE
- others

# Userspace API

- libiptc, libxtables
- iptables
- iptables-save, iptables-restore

# BPFilter

- Introduced in 2018 by Daniel Borkman and David S. Miller
- Transparent replacement for iptables

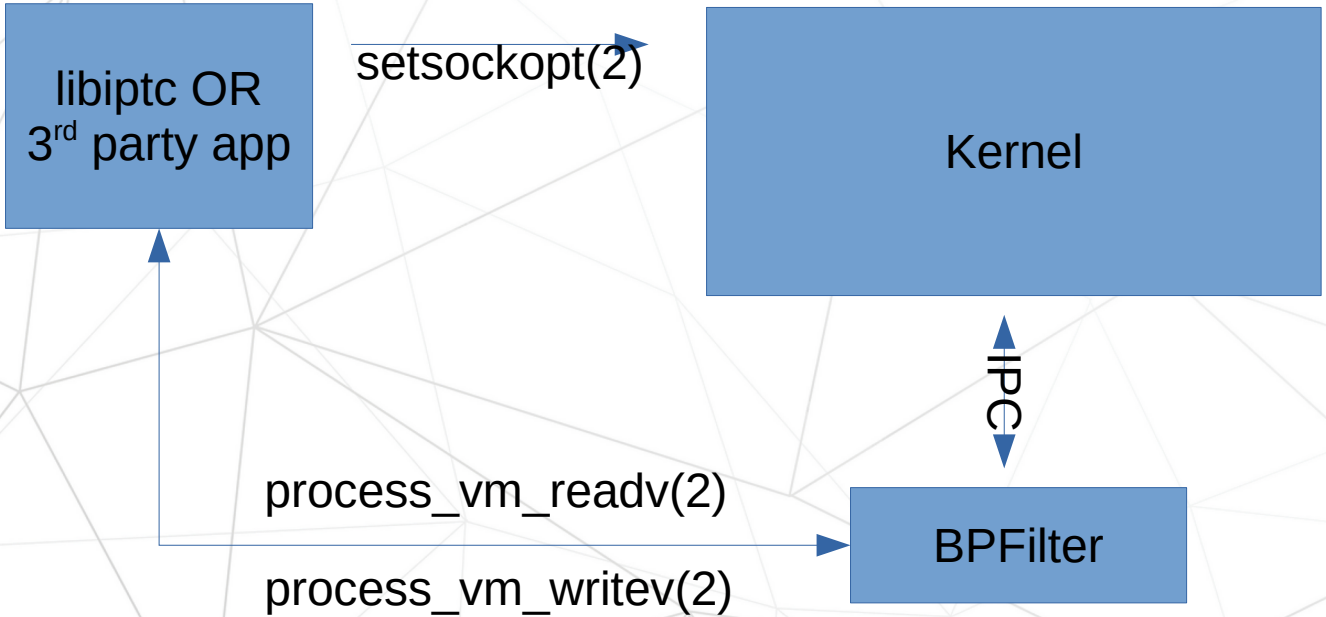
# User Mode Helper

- Userspace process
- Communication channel
- IPC

# Hooking iptables API

- Pass control to bpfilter kernel module
- Send RPC request to userspace process
- process_vm_readv/process_vm_writev
- Process RPC response

# Hooking iptables API



# BPF programs

- IPT_SO_REPLACE
- Generate BPF programs
- Load and attach generated programs
- IPT_SO_GET_INFO, IPT_SO_GET_ENTRIES

# BPF programs

- Table filter
- INPUT hook: XDP BPF
- OUTPUT hook: TC BPF
- FORWARD hook: TBD

# Code generation

- Approach 1: one rule – one set of instructions
- Approach 2: many rules – one set of instructions
- A mix of approaches

# Code generation

```
# cat rules.test
# Generated by iptables-save v1.8.2 on Sat May  8 05:22:41 2021
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -s 1.1.1.1/32 -d 2.2.2.2/32 -j DROP
-A INPUT -s 2.2.0.0/16 -d 3.0.0.0/8 -j DROP
-A INPUT -p udp -m udp --sport 100 --dport 500 -j DROP
COMMIT
# /usr/sbin/iptables-legacy-restore ./rules.test >/dev/null
# ip l show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdpgeneric qdisc pfifo_fast state UP mode \
    DEFAULT group default qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
    prog/xdp id 19 tag 9bb12a1a4b616d6d jited
```

# Code generation

```
# bpftool prog dump xlated id 19
   0: (bf) r9 = r1
   1: (bf) r8 = r10
   2: (b4) (u32) r0 = (u32) 0
   3: (79) r6 = *(u64 *)(r9 +0)
   4: (79) r5 = *(u64 *)(r9 +8)
   5: (bf) r2 = r5
   6: (1f) r2 -= r6
   7: (63) *(u32 *)(r8 -4) = r2
   8: (07) r6 += 14
   9: (2d) if r6 > r5 goto pc+129
  10: (bf) r1 = r6
  11: (07) r1 += 20
  12: (2d) if r1 > r5 goto pc+126
  13: (61) r1 = *(u32 *)(r6 +12)
  14: (54) (u32) r1 &= (u32) -1
  15: (55) if r1 != 0x1010101 goto pc+27
  ....
```

prologue

rule

jump to the next rule

# Code generation

```
static bool udp_mt(const struct sk_buff *skb, struct xt_action_param *par)
{
...
        return port_match(udpinfo->spts[0], udpinfo->spts[1],
                          ntohs(uh->source),
                          !!(udpinfo->invflags & XT_UDP_INV_SRCPT))
                && port_match(udpinfo->dpts[0], udpinfo->dpts[1],
                              ntohs(uh->dest),
                              !!(udpinfo->invflags & XT_UDP_INV_DSTPT));
...

static inline bool
port_match(u_int16_t min, u_int16_t max, u_int16_t port, bool invert)
{
        return (port >= min && port <= max) ^ invert;
}
```

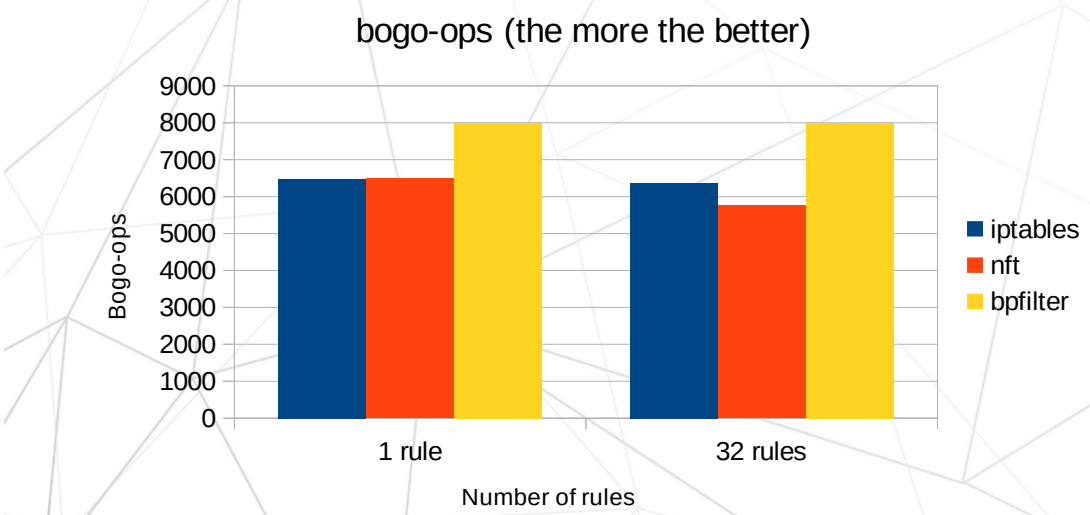
# Code generation

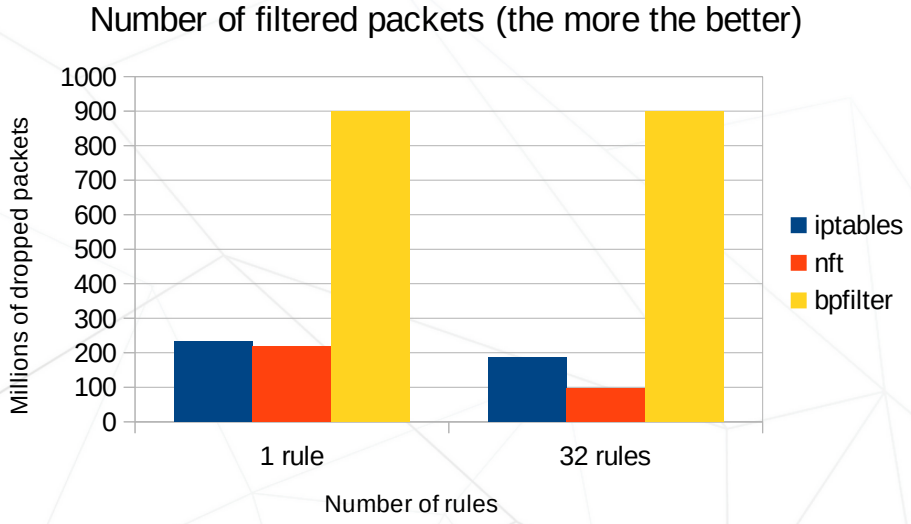
```
static int xt_udp_gen_inline_ports(struct codegen *ctx, int regno, bool inv, const u16 (*ports)[2])
 {
        if ((*ports)[0] == 0 && (*ports)[1] == 65535) {
                if (inv)
                        EMIT_FIXUP(ctx, CODEGEN_FIXUP_NEXT_RULE, BPF_JMP_IMM(BPF_JA, 0, 0, 0));
        } else if ((*ports)[0] == (*ports)[1]) {
                const u16 port = htons((*ports)[0]);

                EMIT_FIXUP(ctx, CODEGEN_FIXUP_NEXT_RULE,
                           BPF_JMP_IMM((inv ? BPF_JEQ : BPF_JNE), regno, port, 0));
        } else {
                EMIT_LITTLE_ENDIAN(ctx, BPF_ENDIAN(BPF_TO_BE, regno, 16));
                EMIT_FIXUP(ctx, CODEGEN_FIXUP_NEXT_RULE,
                           BPF_JMP_IMM(inv ? BPF_JGT : BPF_JLT, regno, (*ports)[0], 0));
                EMIT_FIXUP(ctx, CODEGEN_FIXUP_NEXT_RULE,
                           BPF_JMP_IMM(inv ? BPF_JLT : BPF_JGT, regno, (*ports)[1], 0));
        }
        ...
```

# Code generation

```
struct codegen_ops {
        int (*gen_inline_prologue)(struct codegen *codegen);
        int (*load_packet_data)(struct codegen *codegen, int dst_reg);
        int (*load_packet_data_end)(struct codegen *codegen, int dst_reg);
        int (*emit_ret_code)(struct codegen *codegen, int ret_code);
        int (*gen_inline_epilogue)(struct codegen *codegen);
        int (*load_img)(struct codegen *codegen);
        void (*unload_img)(struct codegen *codegen);
 };
```

# Performance Testing



# Limitations

- 100% compatibility with iptables isn’t the goal
- Concerns about security model

# Future

- New matches and targets
- From nft to bpfilter
- Own interface
- Containers – CGROUP_SKB?
- Inplace upgrade
- Privilege separation
- BPF code optimization