



facebook

INFRASTRUCTURE

eBPF / XDP firewall and packet filtering

Anant Deepak

Software Engineer, Facebook Infrastructure. Nov 2018

Agenda

- iptables
 - Implementation
 - Policy and Network
- Firewall in bpf / XDP
 - Benefits
 - Deployment @Facebook
 - Performance
- Prototype : iptables in bpf

iptables

iptables

Implementation

```
// net/ipv6/netfilter/ip6_tables.c
unsigned int ip6t_do_table(....)
{
    .....
    e = get_entry(.....);

    do {
        .....

        if (!ip6_packet_match(....)) {
no_match:
        e = ip6t_next_entry(e);
        continue;
    }
    .....

} while (....);
.....
}
```

iptables

INPUT Chain

```
Chain INPUT (policy DROP ... packets, ... bytes)
target     prot      in      out      source          dest
ACCEPT     all       lo      *       ::/0           ::/0
.....
ACCEPT     tcp       *       *       ::/0           ::/0      tcp flags:!syn
.....
ACCEPT     udp       *       *       ::/0           ::/0      udp spts:<...>
.....
ACCEPT     icmpv6   *       *       ::/0           ::/0
.....
ACCEPT     tcp       *       *       ::/0           ::/0      tcp dpt:80 flags:syn
ACCEPT     tcp       *       *       ::/0           ::/0      tcp dpt:443 flags:syn
.....
ACCEPT     udp       *       *       ::/0           ::/0      udp dpt:53
ACCEPT     udp       *       *       ::/0           ::/0      udp dpt:443
.....
ACCEPT     tcp       *       *       dead:beef:dead:beef::/64 ::/0      tcp dpt:<...> flags:syn
ACCEPT     tcp       *       *       dead:beef:dead:beef::/64 ::/0      tcp dpt:<...> flags:syn
.....
ACCEPT     tcp       *       *       dead:beef::/32      ::/0      tcp dpt:<...> flags:syn
ACCEPT     tcp       *       *       ff00::/8        ::/0      tcp dpt:<...> flags:syn
.....
ACCEPT     udp       *       *       dead:beef::/32      ::/0      udp dpt:<...>
ACCEPT     udp       *       *       ff00::/8        ::/0      udp spt:<...>
.....
ACCEPT     udp       *       *       ::/0           ::/0      udp spt:53
ACCEPT     udp       *       *       ::/0           ::/0      udp spt:443
.....
DROP      tcp       *       *       ::/0           ::/0      tcp flags:syn
.....
DROP      udp       *       *       ::/0           ::/0
```

iptables

Rules lower in the chain suffer in performance

```
Chain INPUT (policy DROP ... packets, ... bytes)
target     prot      in      out      source          dest
ACCEPT    all       lo      *       ::/0           ::/0
...
ACCEPT    tcp       *       *       ::/0           ::/0      tcp flags:!syn
...
ACCEPT    udp       *       *       ::/0           ::/0      udp spts:<...>
...
ACCEPT    icmpv6   *       *       ::/0           ::/0
...
ACCEPT    tcp       *       *       ::/0           ::/0      tcp dpt:80 flags:syn
ACCEPT    tcp       *       *       ::/0           ::/0      tcp dpt:443 flags:syn
...
ACCEPT    udp       *       *       ::/0           ::/0      udp dpt:53
ACCEPT    udp       *       *       ::/0           ::/0      udp dpt:443
...
ACCEPT    tcp       *       *       dead:beef:dead:beef::/64 ::/0      tcp dpt:<...> flags:syn
ACCEPT    tcp       *       *       dead:beef:dead:beef::/64 ::/0      tcp dpt:<...> flags:syn
...
ACCEPT    tcp       *       *       dead:beef::/32  ::/0      tcp dpt:<...> flags:syn
ACCEPT    tcp       *       *       ff00::/8       ::/0      tcp dpt:<...> flags:syn
...
ACCEPT    udp       *       *       dead:beef::/32  ::/0      udp dpt:<...>
ACCEPT    udp       *       *       ff00::/8       ::/0      udp spt:<...>
...
ACCEPT    udp       *       *       ::/0           ::/0      udp spt:53
ACCEPT    udp       *       *       ::/0           ::/0      udp spt:443
...
DROP     tcp       *       *       ::/0           ::/0      tcp flags:syn
...
DROP     udp       *       *       ::/0           ::/0
```



Match traffic at high pps

iptables

Policy and Network

```
Chain INPUT (policy DROP ... packets, ... bytes)
target  prot   in    out   source          dest
ACCEPT  all    lo    *     ::/0            ::/0
.....
ACCEPT  tcp    *     *     ::/0            ::/0      tcp flags:!syn
.....
ACCEPT  udp    *     *     ::/0            ::/0      udp spts:<...>
.....
ACCEPT  icmpv6 *     *     ::/0            ::/0
.....
ACCEPT  tcp    *     *     ::/0            ::/0      tcp dpt:80 flags:syn
ACCEPT  tcp    *     *     ::/0            ::/0      tcp dpt:443 flags:syn
.....
ACCEPT  udp    *     *     ::/0            ::/0      udp dpt:53
ACCEPT  udp    *     *     ::/0            ::/0      udp dpt:443
.....
ACCEPT  tcp    *     *     dead:beef:dead:beef::/64 ::/0  tcp dpt:<...> flags:syn
ACCEPT  tcp    *     *     dead:beef:dead:beef::/64 ::/0  tcp dpt:<...> flags:syn
.....
ACCEPT  tcp    *     *     dead:beef::/32       ::/0  tcp dpt:<...> flags:syn
ACCEPT  tcp    *     *     ff00::/8        ::/0  tcp dpt:<...> flags:syn
.....
ACCEPT  udp    *     *     dead:beef::/32       ::/0  udp dpt:<...>
ACCEPT  udp    *     *     ff00::/8        ::/0  udp spt:<...>
.....
ACCEPT  udp    *     *     ::/0            ::/0      udp spt:53
ACCEPT  udp    *     *     ::/0            ::/0      udp spt:443
.....
DROP   tcp    *     *     ::/0            ::/0      tcp flags:syn
.....
DROP   udp    *     *     ::/0            ::/0
```

Bootstrap / Stateless

Public dest ports

Domain dest ports

Network dest ports

Public source ports

Deny / Log

Firewall in BPF / XDP @ Facebook

BPF Firewall

Benefits

- Performance
 - Lower CPU Utilization for filtering
 - DDoS attacks on closed ports
- Networking in XDP : BPF_MAP_TYPE_PROG_ARRAY
 - Load Balancer (katran)
 - Other filters or Rate-limiters
 - Filter past tunnel headers
 - More flexibility in match criteria (pcap style bytes @ offset)
- Manageability
 - Policy - Enforce a policy with logical mapping of packet attributes
 - Network - Decouple network topology internals (networks, ports, prefixes,..)

Revisiting : iptables

Policy and Network

```
Chain INPUT (policy DROP ... packets, ... bytes)
target  prot   in    out   source          dest
ACCEPT  all    lo    *     ::/0            ::/0
.....
ACCEPT  tcp    *     *     ::/0            ::/0  tcp flags:!syn
.....
ACCEPT  udp    *     *     ::/0            ::/0  udp spts:<...>
.....
ACCEPT  icmpv6 *     *     ::/0            ::/0
.....
ACCEPT  tcp    *     *     ::/0            ::/0  tcp dpt:80 flags:syn
ACCEPT  tcp    *     *     ::/0            ::/0  tcp dpt:443 flags:syn
.....
ACCEPT  udp    *     *     ::/0            ::/0  udp dpt:53
ACCEPT  udp    *     *     ::/0            ::/0  udp dpt:443
.....
ACCEPT  tcp    *     *     dead:beef:dead:beef::/64 ::/0  tcp dpt:<...> flags:syn
ACCEPT  tcp    *     *     dead:beef:dead:beef::/64 ::/0  tcp dpt:<...> flags:syn
.....
ACCEPT  tcp    *     *     dead:beef::/32   ::/0  tcp dpt:<...> flags:syn
ACCEPT  tcp    *     *     ff00::/8      ::/0  tcp dpt:<...> flags:syn
.....
ACCEPT  udp    *     *     dead:beef::/32   ::/0  udp dpt:<...>
ACCEPT  udp    *     *     ff00::/8      ::/0  udp spt:<...>
.....
ACCEPT  udp    *     *     ::/0            ::/0  udp spt:53
ACCEPT  udp    *     *     ::/0            ::/0  udp spt:443
.....
DROP   tcp    *     *     ::/0            ::/0  tcp flags:syn
.....
DROP   udp    *     *     ::/0            ::/0
```

NETWORK

Bootstrap / Stateless

Public dest ports

Domain dest ports

Network dest ports

Public source ports

Deny / Log

POLICY

Now in bpf ..

Policy and Network

```
if (udp_sp_match & UDP_DHCP_OUT) {  
    PASS;  
}  
  
if ((tcp_dp_match & TCP_PUBLIC) || (udp_dp_match & UDP_PUBLIC)) {  
    PASS;  
}  
  
if (v6_src_prefix_match & V6NET_SAME_SUBNET) {  
    if (tcp_dp_match & TCP_SAME_NET) {  
        PASS;  
    }  
}  
  
if (v6_src_prefix_match & V6NET_PROD_NETBLOCK) {  
    if (udp_dp_match & UDP_PROD) {  
        PASS;  
    }  
}  
  
if (udp_sp_match & UDP_PUBLIC_OUT) {  
    PASS;  
}  
  
stats_map.increment(DEFAULT_DROP);  
DROP;
```

NETWORK

Bootstrap / Stateless

Public dest ports

Domain dest ports

Network dest ports

Public source ports

Deny / Log

POLICY

Now in bpf ..

C Program for policy . Maps for network topology

```
if (udp_sp_match & UDP_DHCP_OUT) {  
    PASS;  
}  
  
if ((tcp_dp_match & TCP_PUBLIC) || (udp_dp_match & UDP_PUBLIC)) {  
    PASS;  
}  
  
if (v6_src_prefix_match & V6NET_SAME_SUBNET) {  
    if (tcp_dp_match & TCP_SAME_NET) {  
        PASS;  
    }  
}  
  
if (v6_src_prefix_match & V6NET_PROD_NETBLOCK) {  
    if (udp_dp_match & UDP_PROD) {  
        PASS;  
    }  
}  
  
if (udp_sp_match & UDP_PUBLIC_OUT) {  
    PASS;  
}  
  
stats_map.increment(DEFAULT_DROP);  
DROP;
```

```
tcp_port_map.lookup(&port);  
udp_port_map.lookup(&port);  
v4_addr_map.lookup(&addr);  
v6_addr_map.lookup(&addr);  
v4_lpm_map.lookup(&v4_key);  
v6_lpm_map.lookup(&v6_key);
```

BPF Array - TCP ports

Key	Value
443	TCP_PUBLIC
8080	TCP_SAMENET
<...>	TCP_PROD
<...>	TCP_SAMENET TCP_PROD

BPF LPM - IP6 Prefix

Key	Value
dead:beef::/32	IP6_PRODNET
dead:beef:dead:beef::/64	IP6_PRODNET IP6_SAMENET

NETWORK

BPF Firewall

Deployment @ Facebook

- Stateless
- Deployed on our Edge InfraStructure
- Runs before our L4 Load balancer (katran)
 - Infact: PASS => bpf_tail_call -> katran
- Tooling for loading and verifying bpf Map contents
- C program (policy) rarely changes
- Atomic Swap : Loaders load new maps and reattach a new program
- BCC Helpers

BPF Firewall

Deployment @ Facebook

- Prefix Matching
 - Loader handles overlapping prefixes
- Sampling : BPF_PERF_OUTPUT
- Stats : PERCPU_ARRAY
- Look beyond tunneled headers

BPF LPM - IP6 Prefix

Key	Value
dead:beef::/32	IP6_PRODNET
dead:beef:dead:beef::/64	IP6_PRODNET IP6_SAMENET

BPF Firewall

Deployment @ Facebook

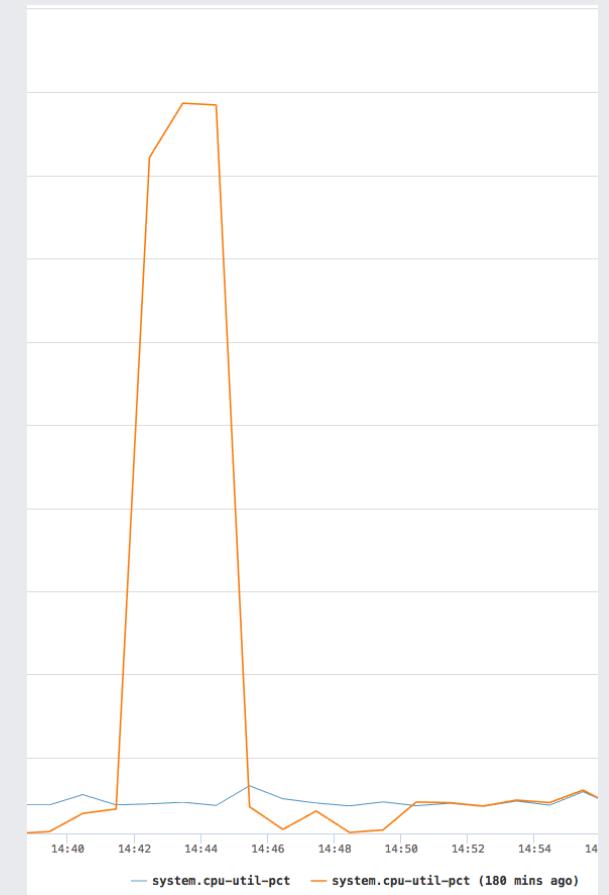
- No per rule stats granularity like in iptables
 - But we have stats for agg. policy (eg : PASS same network traffic)
- Keeping it simple : ACCEPT and DROP
- REJECTS, custom chains possible via bpf_tail_call

```
stats_map.increment(ICMP6_PASS);
stats_map.increment(TCP6_NON_SYN_PASS)
stats_map.increment(TCP_NON_SYN_PASS);
stats_map.increment(UDP6_SP_DHCP_OUT);
stats_map.increment(TCP6_DP_PUBLIC);
stats_map.increment(TCP6_SAMENET);
stats_map.increment(UDP6_PROD);
stats_map.increment(DEFAULT_DROP);
stats_map.increment(DEFAULT_PIPE);
stats_map.increment(DEFAULT_PASS);
```

BPF Firewall

Performance

- iptables has a linearly increasing cpu-util as packets hit lower rules
 - Best when packets match earlier rules
 - Worst for default policy (match attempted for each rule)
- Our BPF firewall performance remains practically constant irrespective of rule being matched or default drop
 - Packet tuple lookup is efficient w/ only 1 BPF map per tuple
 - Location of matching rule now only matters to the extent of few branching instructions



Prototype : iptables in bpf

iptables in bpf

Motivation

- A drop in replacement for iptables in kernel
- User space helper to translate rules and load bpf maps
- bpf program to do minimal work
- Same functionality with:
 - Better performance
 - Customization for attach points (containers ?)

Revisiting : firewall in bpf ..

Policy and Network

```
if (udp_sp_match & UDP_DHCP_OUT) {  
    PASS;  
}  
  
if ((tcp_dp_match & TCP_PUBLIC) || (udp_dp_match & UDP_PUBLIC)) {  
    PASS;  
}  
  
if (v6_src_prefix_match & V6NET_SAME_SUBNET) {  
    if (tcp_dp_match & TCP_SAME_NET) {  
        PASS;  
    }  
}  
  
if (v6_src_prefix_match & V6NET_PROD_NETBLOCK) {  
    if (udp_dp_match & UDP_PROD) {  
        PASS;  
    }  
}  
  
if (udp_sp_match & UDP_PUBLIC_OUT) {  
    PASS;  
}  
  
stats_map.increment(DEFAULT_DROP);  
DROP;
```

```
tcp_port_map.lookup(&port);  
udp_port_map.lookup(&port);  
v4_addr_map.lookup(&addr);  
v6_addr_map.lookup(&addr);  
v4_lpm_map.lookup(&v4_key);  
v6_lpm_map.lookup(&v6_key);
```

BPF Array - TCP ports

Key	Value
443	TCP_PUBLIC
8080	TCP_SAMENET
<...>	TCP_PROD
<...>	TCP_SAMENET TCP_PROD

BPF LPM - IP6 Prefix

Key	Value
dead:beef::/32	IP6_PRODNET
dead:beef:dead:beef::/64	IP6_PRODNET IP6_SAMENET

NETWORK

Maps hold matching rules ids

1 Map per attribute : 5 tuples

```
v6_dst_ip_map.lookup(&dst_addr);  
v6_src_ip_map.lookup(&src_addr);  
v6_dst_lpm_map.lookup(&v6_dst_key);  
v6_src_lpm_map.lookup(&v6_src_key);  
proto_map.lookup(&proto);  
src_port_map.lookup(&src_port);  
dst_port_map.lookup(&dst_port);
```

BPF Array - dest ports

Key	Value
443	100111111....1110
8080	010111111....1110
<...>	001111111....1110
<...>	000011111....1111

BPF LPM - src IP6 Prefix

Key	Value
dead:beef::/32	1100111111.....111111
dead:beef:dead:beef::/64	1110111111.....111111
abcd:abcd::/32	1001111111.....111111
* (key not found)	1000111111.....111111

Algorithm

- Value: 1 bit per rule (1K rules = 16 * u64 / 128Bytes)
- wild-card attribute for a rule: mark as "1"
- Logically AND values for each attribute
- First bit set, is the highest order rule matched

BPF program

```
#define RULE_IDS_MAX_WORDS      (8)    // 64 * 8 = 512 rules
int rule_word = 0;
#pragma clang loop unroll(full)
for (rule_word = 0; rule_word < RULE_IDS_MAX_WORDS; rule_word++) {
    u64 rule_id = ip4_src_addr_res->rule_ids[rule_word];
    rule_id &= ip4_src_lpm_res->rule_ids[rule_word];
    .....
    rule_id &= ip6_dst_lpm_res->rule_ids[rule_word];
    .....
    rule_id &= dst_port_res->rule_ids[rule_word];
    .....
    rule_id &= tcp_flag_res->rule_ids[rule_word];
    if (rule_id) {
        int rule_num = (rule_word * 64) + get_msb_set(rule_id);
        int action = lookup_action(rule_num);
        if (action == ACTION_PASS) {
            PASS;
        }
        if (action == ACTION_DENY) {
            DROP;
        }
        break;
    }
}
DROP; // Default action is drop
```

BPF Firewall prototype

Performance

- Similar to our implementation with custom BPF program
- Performance remains practically constant irrespective of packet stream
 - Packet tuple lookup is efficient w/ only 1 BPF map per tuple
 - Location of matching rule now only matters to the extent of finding the first bit set in an array of integers.

BPF Firewall prototype

Implementation

- iptables-save output used as input by python loader w/ BCC
- Simple parser for INPUT chain
 - 5 Tuples and TCP Flags
 - Reject rules not yet supported
- Stateless firewall

bpfilter

Lessons learnt

- Drive adoption by retaining iptables as control interface
- `bpfilter` option to iptables commands
 - iptables-save for configuration
 - iptables-restore : Allow XDP / tc mode attach point
 - iptables : Display rules as loaded in bpfilter
 - stats read and reset support
- Pitch : An optional high performant firewall implementation with some configuration restrictions

References

- **bpfilter** : <https://lwn.net/Articles/747551>
- **katran** : <https://github.com/facebookincubator/katran>
- **droplet** : <https://netdevconf.org/2.1/session.html?zhou>
- **SIGCOMM 2018 - Accelerating Linux Security with eBPF**
iptables : <https://dl.acm.org/citation.cfm?id=3234228>

facebook



Questions

facebook



Thank you

facebook