eBPF and XDP seen from the eyes of a meerkat

É. Leblond

Stamus Networks

September 29, 2017

STAMUS NETWORKS

About me

Eric Leblond a.k.a Regit

- Network security expert
- Netfilter core team:
  - Maintainer of ulogd2: Netfilter logging daemon
- Suricata developer:
  - In charge of packet acquisition
  - co-founder of Stamus Networks, a company providing Suricata based network probe appliances.

STAMUS NETWORKS

About me

Eric Leblond a.k.a Regit

- Network security expert
- Netfilter core team:
  - Maintainer of ulogd2: Netfilter logging daemon
- Suricata developer:
  - In charge of packet acquisition
  - co-founder of Stamus Networks, a company providing Suricata based network probe appliances.

Not a Kernel developer

- 42 + 1 patches in Linux
- Mainly hacks

STAMUS NETWORKS

About me

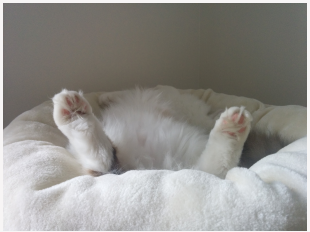
Eric Leblond a.k.a Regit

- Network security expert
- Netfilter core team:
  - Maintainer of ulogd2: Netfilter logging daemon
- Suricata developer:
  - In charge of packet acquisition
  - co-founder of Stamus Networks, a company providing Suricata based network probe appliances.

Not a Kernel developer

- 42 + 1 patches in Linux
- Mainly hacks

No kernel harmed during the making of this talk

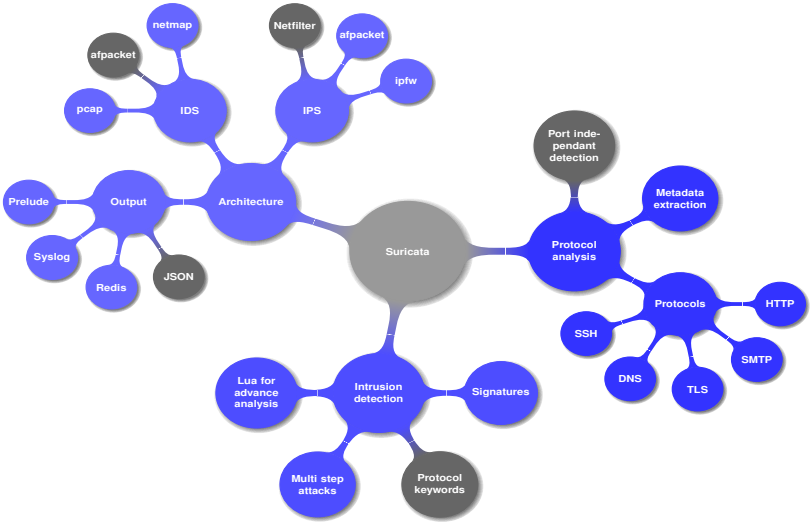


What is Suricata

- IDS and IPS engine
- Get it here: `http://www.suricata-ids.org`
- Open Source (GPLv2)
- Initially publicly funded, now funded by consortium members
- Run by Open Information Security Foundation (OISF)
- More information about OISF at `http://www.openinfosecfoundation.org/`

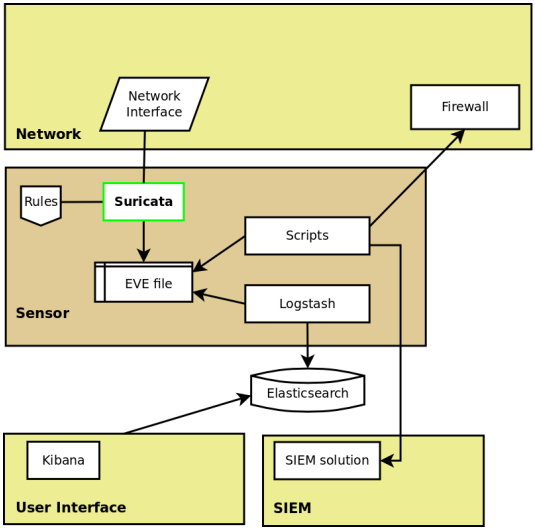
STAMUS NETWORKS

Suricata key points

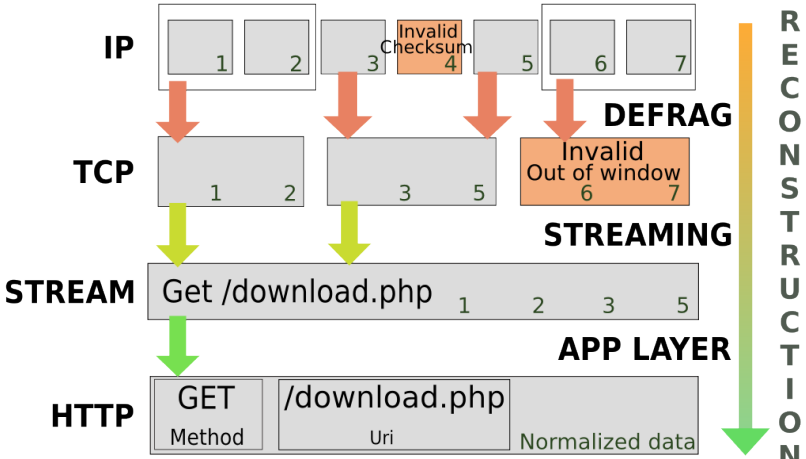


STAMUS NETWORKS

Suricata Ecosystem



Suricata application layer analysis

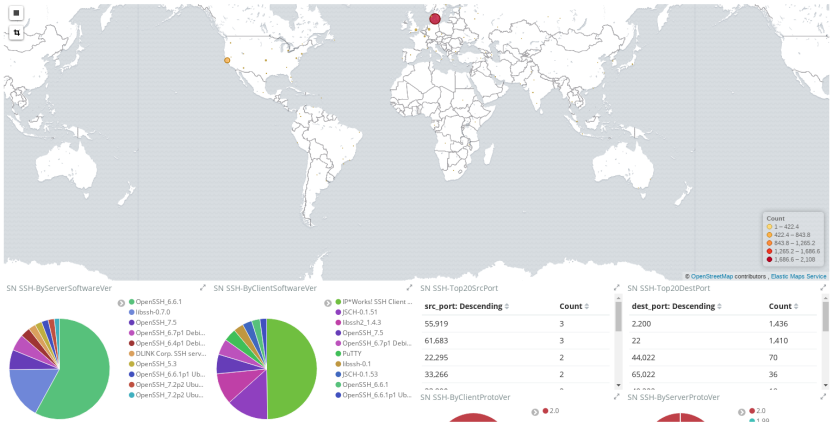


Suricata EVE JSON event


```
{
  "timestamp": "2015-07-15T16:47:47.941448+0200",
  "flow_id": 100815541166104,
  "pcap_cnt": 24,
  "event_type": "alert",
  "src_ip": "192.168.0.254",
  "src_port": 36391,
  "dest_ip": "192.168.0.5",
  "dest_port": 25,
  "proto": "TCP",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 1,
    "rev": 1,
    "signature": "Mail to stamus",
    "category": "",
    "severity": 3
  },
  "vars": {
    "pktvars": [
      {
        "email": "eleblond@stamus-networks.com"
      }
    ]
  },
  "app_proto": "smtp",
  "app_proto_tc": "failed",
  "flow": {
    "pkts_toserver": 12,
    "pkts_toclient": 12,
    "bytes_toserver": 1244,
    "bytes_toclient": 1086,
    "start": "2015-07-15T16:47:32.778264+0200"
  }
}
```


STAMUS NETWORKS

Suricata Dashboard



STAMUS NETWORKS

AF_PACKET

Linux raw socket

- Raw packet capture method
- Socket based or mmap based

STAMUS NETWORKS

AF_PACKET

Linux raw socket

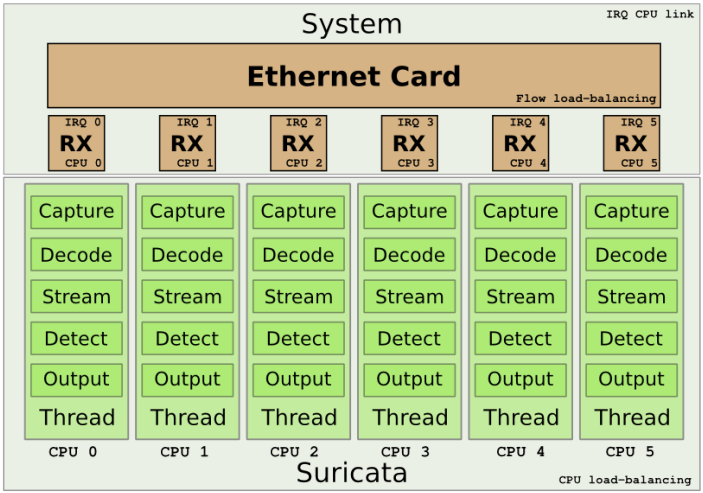
- Raw packet capture method
- Socket based or mmap based

Fanout mode

- Load balancing over multiple sockets
- Multiple load balancing functions
  - Flow based
  - CPU based
  - RSS based

STAMUS NETWORKS

Suricata workers mode



Load balancing and hash symmetry

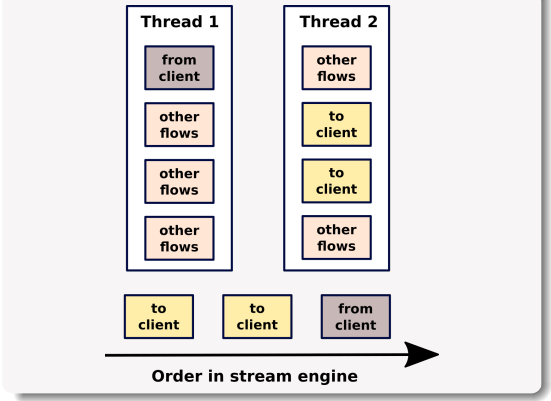
Stream reconstruction

- Using packets sniffed from network
- to reconstruct TCP stream as seen by remote application

Non symmetrical hash break

- Out of order packets

Effect of non symmetrical hash



STAMUS NETWORKS

Broken symmetry

History

- T. Herbert introduce asymmetrical hash function in flow
  - Kernel 4.2
- Users did start to complain
- And our quest did begin
- Fixed in 4.6 and pushed to stable by David S. Miller

STAMUS NETWORKS

Broken symmetry

History

- T. Herbert introduce asymmetrical hash function in flow
  - Kernel 4.2
- Users did start to complain
- And our quest did begin
- Fixed in 4.6 and pushed to stable by David S. Miller

Intel NIC RSS hash

- XL510 hash is not symmetrical
- XL710 could be symmetrical
  - Hardware is capable
  - Driver does not allow it
  - Patch proposed by Victor Julien

eBPF cluster

Userspace to the rescue

- Program your own hash function in userspace
- Available since Linux 4.3
- Developed by Willem de Bruijn
- Using eBPF infrastructure by Alexei Storovoitov

eBPF cluster: ippair

- IP pair load balancing
- Perfect for xbit
- `ebpf-lb-file` variable in af-packet iface configuration

STAMUS NETWORKS

eBPF code for ippair

```
static __always_inline int ipv4_hash(struct __sk_buff *skb)
{
    uint32_t nhoff;
    uint32_t src, dst;
    nhoff = skb->cb[0];
    src = load_word(skb, nhoff + offsetof(struct iphdr, saddr));
    dst = load_word(skb, nhoff + offsetof(struct iphdr, daddr));
    return  src + dst;
}

int  __section("loadbalancer") lb(struct __sk_buff *skb) {
    __u32 nhoff = BPF_LL_OFF + ETH_HLEN;
    skb->cb[0] = nhoff;
    switch (skb->protocol) {
        case __constant_htons(ETH_P_IP):
            return ipv4_hash(skb);
        case __constant_htons(ETH_P_IPV6):
            return ipv6_hash(skb);
        default:
            break;
    }
    return skb->protocol; /* hash on proto by default */
```

eBPF cluster: prospective

Custom tunnelled traffic

- Tunneling protocol not known by kernel and card
  - L2TP
  - GTP: 4G protocol
- Shared by different flows
- Result in poor load balancing

eBPF solution

- Strip tunnel headers
- Load balance on inner packets
- Get fair balancing

STAMUS NETWORKS

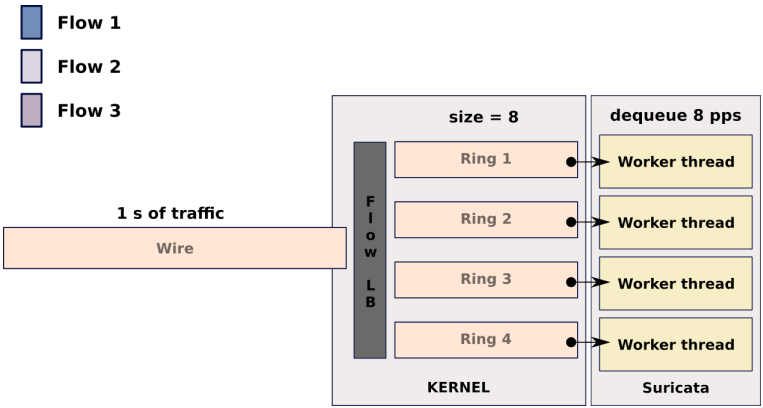
1 Suricata meets eBPF

2 AF_PACKET bypass via eBPF

3 XDP

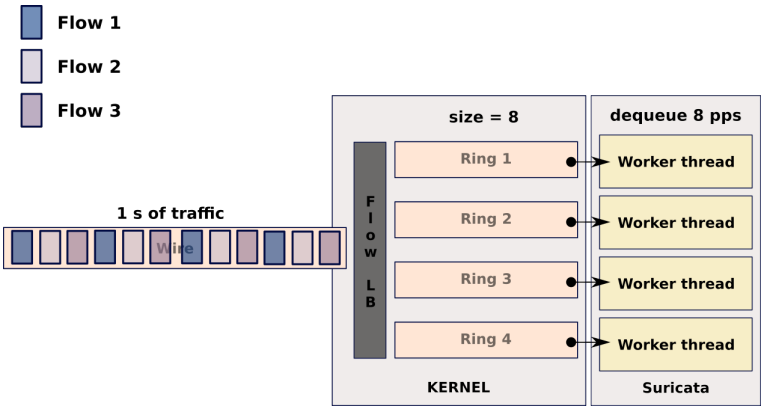
STAMUS NETWORKS

The big flow problem: load balancing

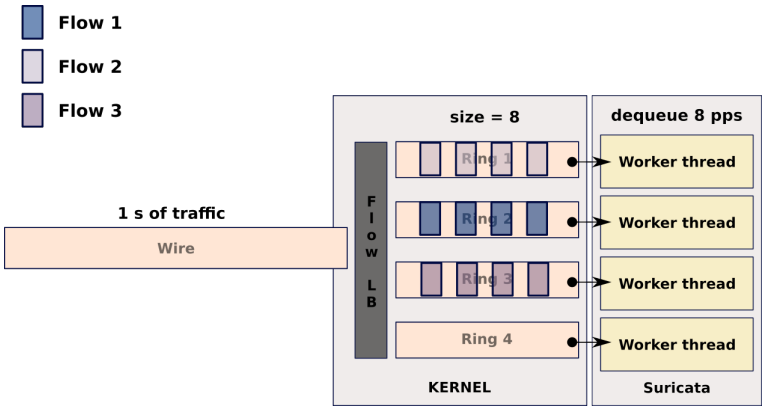


STAMUS NETWORKS

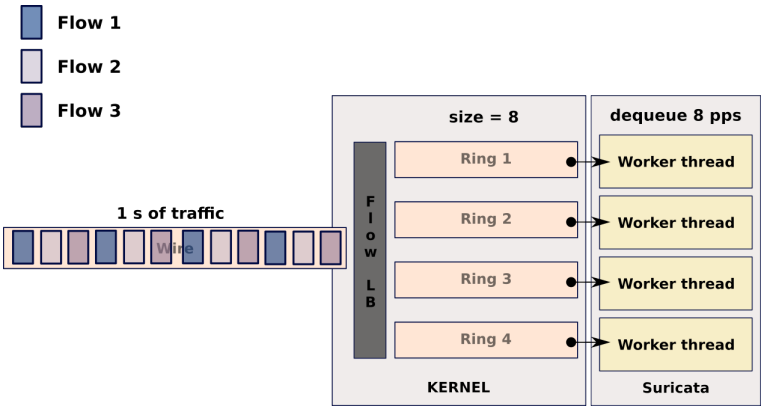
The big flow problem: load balancing



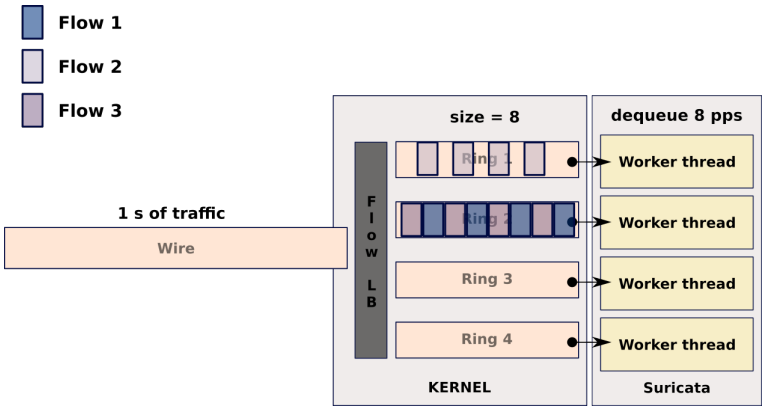
The big flow problem: load balancing



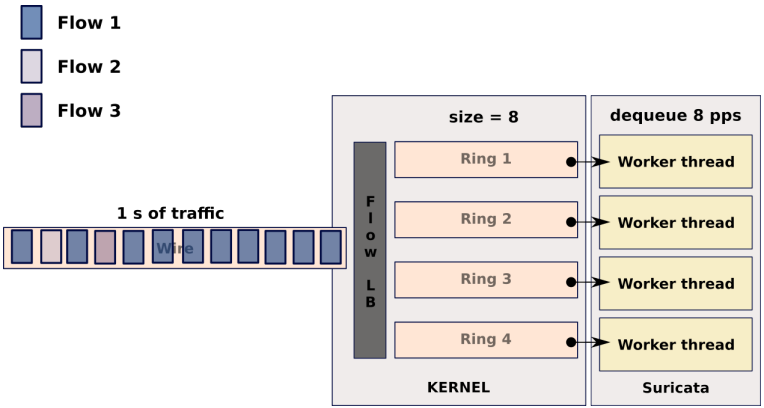
The big flow problem: unfair balancing



The big flow problem: unfair balancing

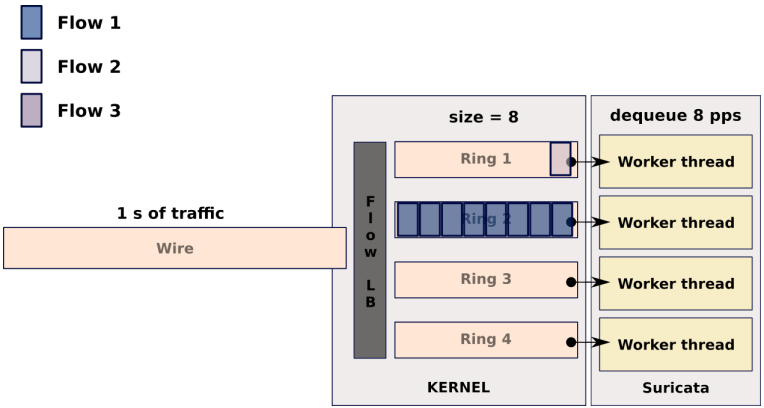


The big flow problem: elephant flow

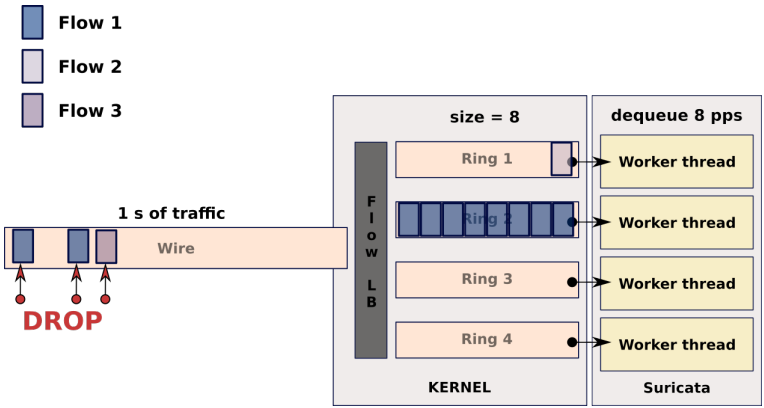


STAMUS NETWORKS

The big flow problem: elephant flow



The big flow problem: elephant flow



The big flow problem

Ring buffer overrun

- Limited sized ring buffer
- Overrun cause packets loss
- that cause streaming malfunction

Ring size increase

- Work around
- Use memory
- Fail for non burst
  - Dequeue at N
  - Queue at speed N+M

STAMUS NETWORKS

Introducing bypass

Stop packet handling as soon as possible

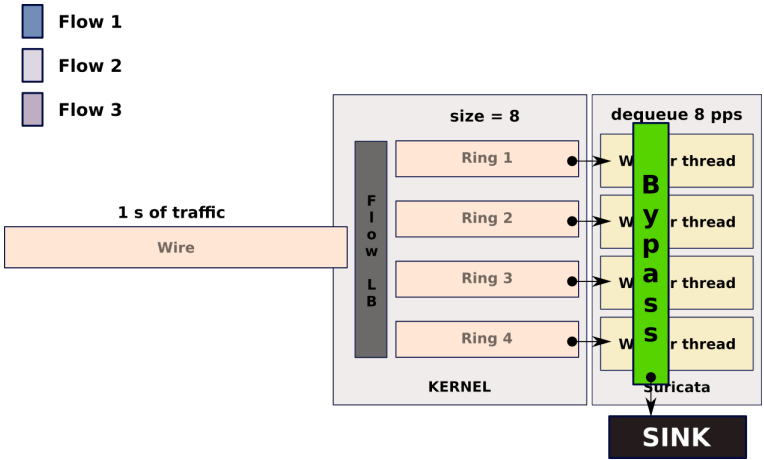
- Tag flow as bypassed
- Maintain table of bypassed flows
- Discard packet if part of a bypassed flow

Bypass method

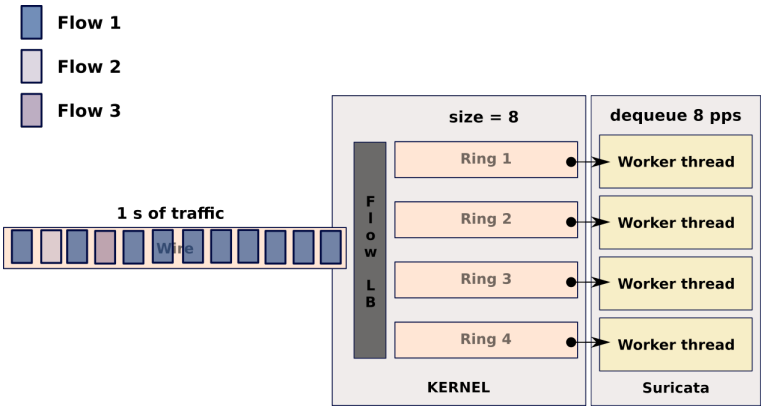
- Local bypass: Suricata discard packet after decoding
- Capture bypass: capture method maintain flow table and discard packets of bypassed flows

STAMUS NETWORKS

Bypassing big flow: local bypass

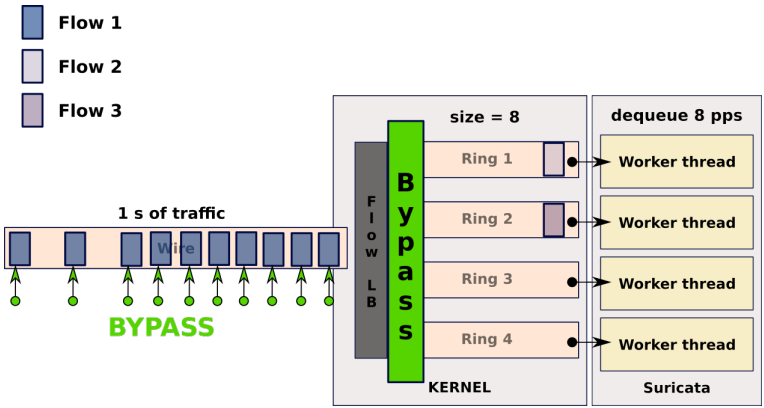


Bypassing big flow: capture bypass



STAMUS NETWORKS

Bypassing big flow: capture bypass



STAMUS NETWORKS

Stream depth bypass

Attacks characteristic

- In most cases attack is done at start of TCP session
- Generation of requests prior to attack is not common
- Multiple requests are often not even possible on same TCP session

Stream reassembly depth

- Reassembly is done till `stream.reassembly.depth` bytes.
- Stream is not analyzed once limit is reached

Activating stream depth bypass

- Set `stream.bypass` to yes in YAML

Selective bypass

Ignore some traffic

- Ignore intensive traffic like Netflix
- Can be done independently of stream depth
- Can be done using generic or custom signatures

STAMUS NETWORKS

Selective bypass

Ignore some traffic

- Ignore intensive traffic like Netflix
- Can be done independently of stream depth
- Can be done using generic or custom signatures

The bypass keyword

- A new `bypass` signature keyword
- Trigger bypass when signature match
- Example of signature

```
pass http any any -> any any (content:"suricata.io"; \\
          http_host; bypass; sid:6666; rev:1;)
```

Implementation

Suricata update

- Add callback function
- Capture method register itself and provide a callback
- Suricata calls callback when it wants to offload

STAMUS NETWORKS

Implementation

Suricata update

- Add callback function
- Capture method register itself and provide a callback
- Suricata calls callback when it wants to offload

NFQ bypass in Suricata 3.2

- Update capture register function
- Written callback function
  - Set a mark with respect to a mask on packet
  - Mark is set on packet when issuing the verdict

STAMUS NETWORKS

And now AF_PACKET

What's needed

- Suricata to tell kernel to ignore flows
- Kernel system able to
  - Maintain a list of flow entries
  - Discard packets belonging to flows in the list
  - Update from userspace
- nftables is too late even in ingress

STAMUS NETWORKS

And now AF_PACKET

What's needed

- Suricata to tell kernel to ignore flows
- Kernel system able to
  - Maintain a list of flow entries
  - Discard packets belonging to flows in the list
  - Update from userspace
- nftables is too late even in ingress

eBPF filter using maps

- eBPF introduce maps
- Different data structures
  - Hash, array, ...
  - Update and fetch from userspace
- Looks good!

Using libbpf

Library Linux source in tools/lib/bpf directory

- Provide high level function to load eBPF elf file
- Create maps for user
- Do the relocation

Sample usage

```
struct bpf_object *bpfobj = bpf_object__open(path);
bpf_object__load(bpfobj);
pfd = bpf_program__fd(bpfprog);
/* store the map in our array */
bpf_map__for_each(map, bpfobj) {
    map_array[last].fd = bpf_map__fd(map);
    map_array[last].name = strdup(bpf_map__name(map));
    last++;
}
```

Kernel code and exchange structure

```
struct pair {
    uint64_t time;
    uint64_t packets;
    uint64_t bytes;
};

struct bpf_map_def SEC("maps") flow_table_v4 = {
    .type = BPF_MAP_TYPE_HASH,
    .key_size = sizeof(struct flowv4_keys),
    .value_size = sizeof(struct pair),
    .max_entries = 32768,
};

value = bpf_map_lookup_elem(&flow_table_v4, &tuple);
if (value) {
    __sync_fetch_and_add(&value->packets, 1);
    __sync_fetch_and_add(&value->bytes, skb->len);
    value->time = bpf_ktime_get_ns();
    return 0;
}
return -1;
```

Sharing data

- Data is updated with stats
- Getting last flow activity time allow Suricata to handle timeout

STAMUS NETWORKS

Userspace code

```
struct flowv4_keys {
    __be32 src;
    __be32 dst;
    union {
        __be32 ports;
        __be16 port16[2];
    };
    __u32 ip_proto;
};

while (bpf_map__get_next_key(mapfd, &key, &next_key) == 0) {
    bpf_map__lookup_elem(mapfd, &key, &value);
    clock_gettime(CLOCK_MONOTONIC, &curtime);
    if (curtime->tv_sec * 1000000000 - value.time > BYPASSED_FLOW_TIMEOUT) {
        flowstats->count++;
        flowstats->packets += value.packets;
        flowstats->bytes += value.bytes;
        bpf_map__delete_elem(fd, key);
    }
    key = next_key;
}
```

Test methodology

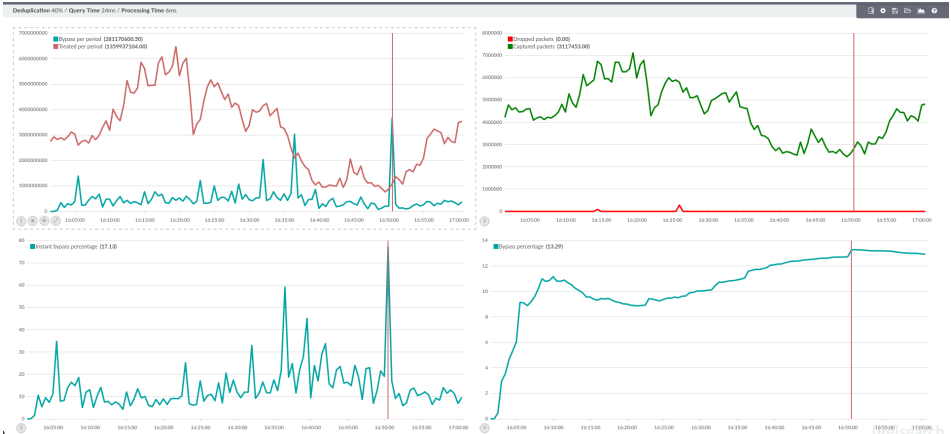
Test setup

- Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
- Intel Corporation 82599ES 10-Gigabit SFI/SFP+
- Live traffic:
  - Around 1Gbps to 2Gbps
  - Real users so not reproducible

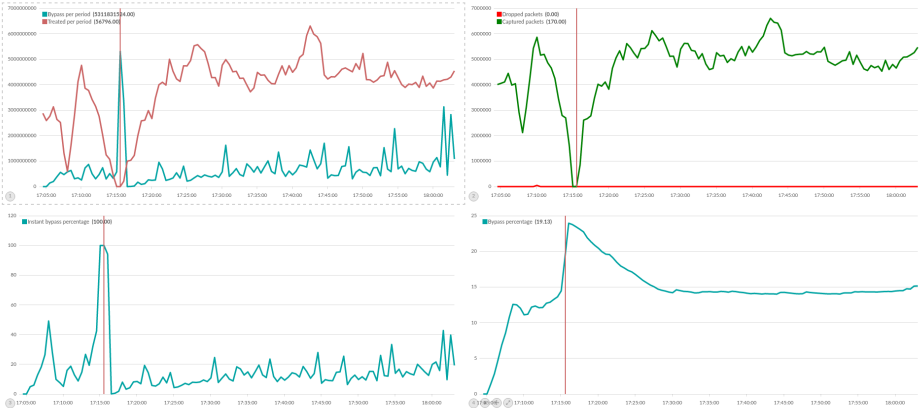
Tests

- One hour long run
- Different stream depth values
- Collected Suricata statistics counters (JSON export)
- Graphs done via Timelion (`https://www.elastic.co/blog/timelion-timeline`)

Results: stream bypass at 1mb



Results: stream bypass at 512kb



{Re}search

STAMUS NETWORKS

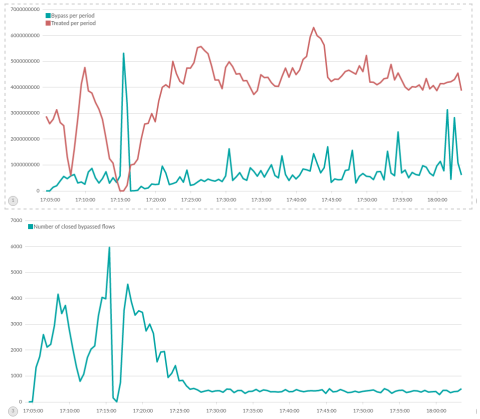
A few words on graphics

Tests at 1mb

- Mark show some really high rate bypass
- Potentialy a big high speed flow

Tests at 512kb

- We have on big flow that kill the bandwidth
- Capture get almost null
- Even number of closed bypassed flows is low



STAMUS NETWORKS

AF_PACKET bypass and your CPU is peaceful

STAMUS NETWORKS

1 Suricata meets eBPF

2 AF_PACKET bypass via eBPF

3 XDP

STAMUS NETWORKS

eXtreme Data Path

raw packet-page inside driver

- Before allocating SKBs
- Inside device drivers RX function
- Operate directly on RX DMA packet-pages
- Run eBPF program at hook point

eBPF decision

- XDP_PASS: pass to normal network stack (can be modified)
- XDP_DROP: drop packet
- XDP_TX: bounce trafic
- XDP_REDIRECT: redirect packet to another interface using port map

Talks by Jesper Dangaard Brouer

`http://people.netfilter.org/hawk/presentations/LLC2017/XDP_DDoS_protecting_LLC2017.pdf`
`https://people.netfilter.org/hawk/presentations/NetConf2017/xdp_work_ahead_NetConf_April_2017.pdf`

STAMUS NETWORKS

XDP Support and Performance

Need modified drivers

- Supported drivers
  - Mellanox: mlx4 + mlx5
  - Netronome: nfp
  - Cavium/Qlogic: qede
  - virtio-net
  - Broadcom: bnxt_en
  - Intel: ixgbe, i40e
- Kernel 4.12 introduce generic drivers

Performances

Single CPU on Mellanox 40Gbit/s NICs (mlx4)

- 28 Mpps – Filter drop all (actually read/touch data)
- 12 Mpps – TX-bounce forward (TX bulking)
- 10 Mpps – TX-bounce with udp+mac rewrite

STAMUS NETWORKS

Suricata XDP bypass

Convert eBPF code

- No more access to skb
- Direct access to data means parsing

From filter socket to device

- Filter is attached to device
- Code to attach needs to be run
- And added to libbpf (?)
- per CPU structure for flow table

STAMUS NETWORKS

Suricata XDP capture

Use perf event

- Use perf event system
  - Memory mapped ring buffer
  - Per CPU structure

Architecture constraints

- Load balancing per CPU
- Done by the network card
  - Symetrical hash needed
  - CPU pinning

STAMUS NETWORKS

Implementation

```
int SEC("xdp") xdp_hashfilter(struct xdp_md *ctx)
{
    void *data_end = (void *)(long)ctx->data_end;
    void *data = (void *)(long)ctx->data;
    struct ethhdr *eth = data;
    int rc = XDP_PASS;
    uint16_t h_proto;
    uint64_t nh_off;

    nh_off = sizeof(*eth);
    if (data + nh_off > data_end)
            return rc;

    h_proto = eth->h_proto;

    if (h_proto == __constant_htons(ETH_P_8021Q) || h_proto == __constant_htons(ETH_P_8021AD)
            struct vlan_hdr *vhdr;

            vhdr = data + nh_off;
            nh_off += sizeof(struct vlan_hdr);
            if (data + nh_off > data_end)
                    return rc;
```

Conclusion

Suricata, eBPF and XDP

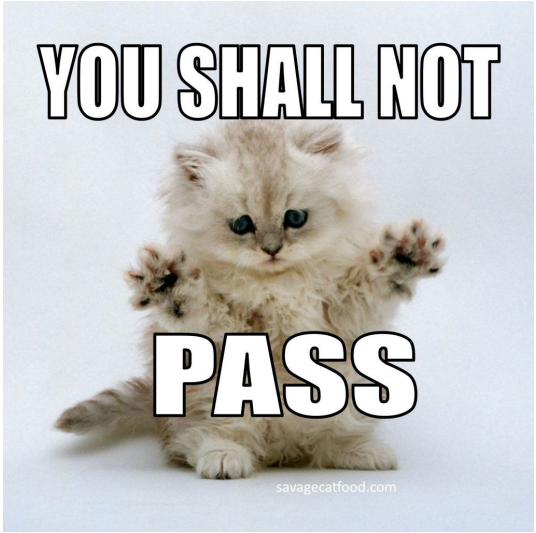
- A fresh but interesting method
- Feedback welcome

More information

- Stamus Networks: `https://www.stamus-networks.com/`
- Suricata eBPF code: `https://github.com/regit/suricata/tree/ebpf-3.16`
- Libbpf update: `https://github.com/regit/linux/tree/libbpf-xdp`

STAMUS NETWORKS

Questions ?



Thanks to

- Alexei Storovoitov
- Daniel Borkmann

Contact me

- Mail: eleblond@stamus-networks.com
- Twitter: @regiteric

More information

- Suricata eBPF and XDP code:
  `https://github.com/regit/suricata/tree/ebpf-3.16`