Suricata and XDP

É. Leblond

Stamus Networks

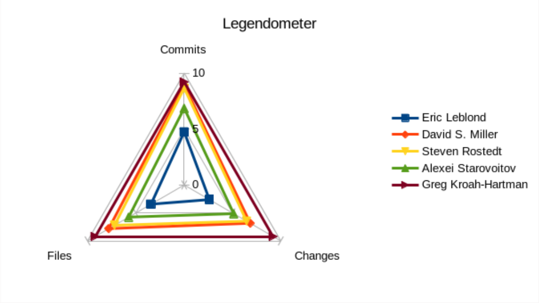
September 27, 2019

About me

Eric Leblond a.k.a Regit

- Network security expert
- Netfilter core team
- Suricata developer:
  - In charge of packet acquisition
- Co-founder of Stamus Networks, a company providing Suricata based appliances.
- @Regiteric on Twitter (#sorry)

Legendometer

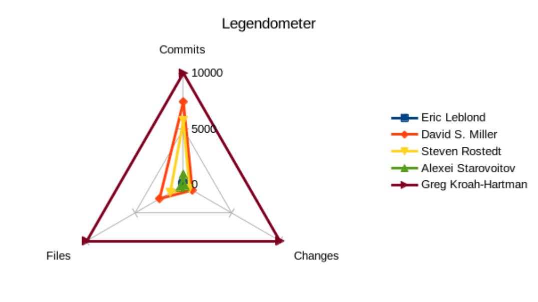


About me

Eric Leblond a.k.a Regit

- Network security expert
- Netfilter core team
- Suricata developer:
  - In charge of packet acquisition
- Co-founder of Stamus Networks, a company providing Suricata based appliances.
- @Regiteric on Twitter (#sorry)

Legendometer (No log scale)

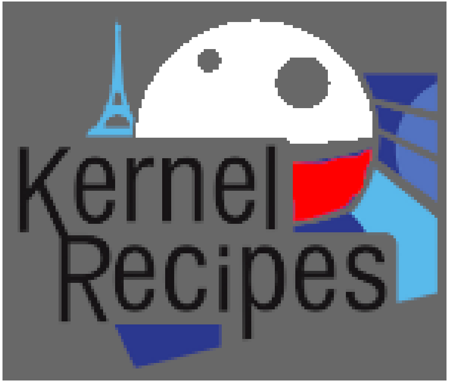


What about Kernel Recipes logo ?

Kernel
Recipes

What about Kernel Recipes logo ?

What about Kernel Recipes logo ?

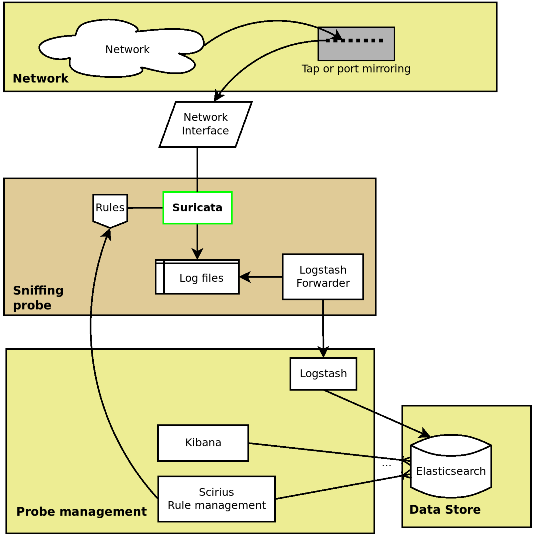


What about Kernel Recipes logo ?

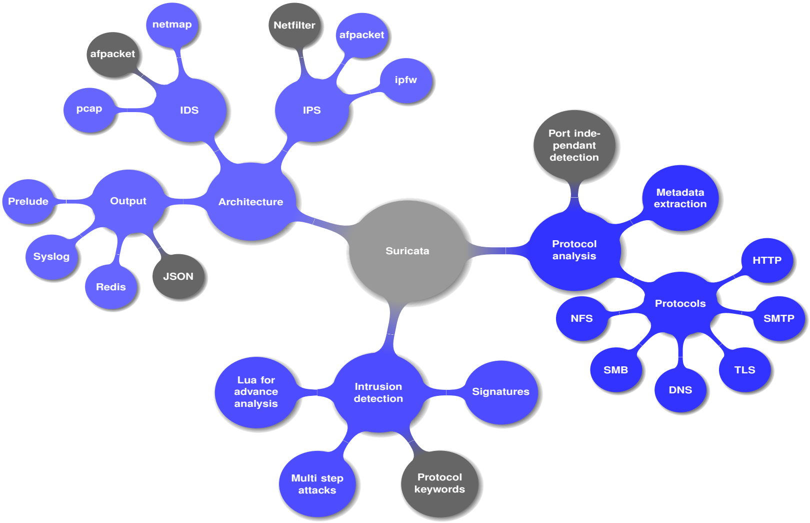
What is Suricata ?

- 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/`

Suricata Ecosystem (example)



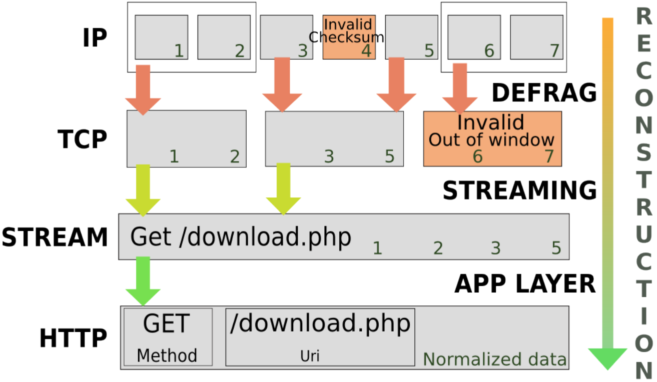
Suricata key points



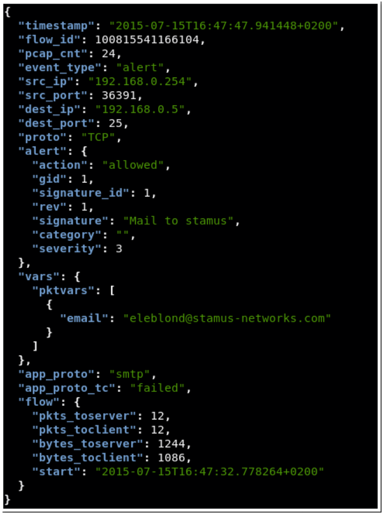
Suricata application layer analysis

Suricata analysis

- Network interface gets copy of traffic
- Aggregated RX and TX of sniffed interface
- Reconstruct flow stream as target host
- Decode application layer
- Extract file (optional)



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"
  }
}
```

Packet loss drama (1/2)

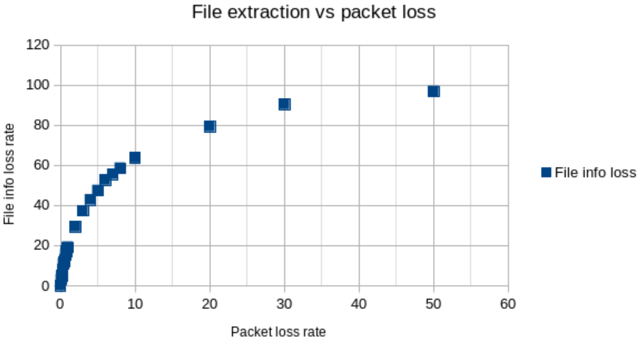
Suricata as a passive sniffer

- Work on traffic duplication
- No influence retransmission
- No influence on bandwitdth throttling

Need to minimize packet loss

- Accuracy of reconstruction drop when packet are lost
- Packets drop means
  - Missed IDS alerts
  - Missed file extraction

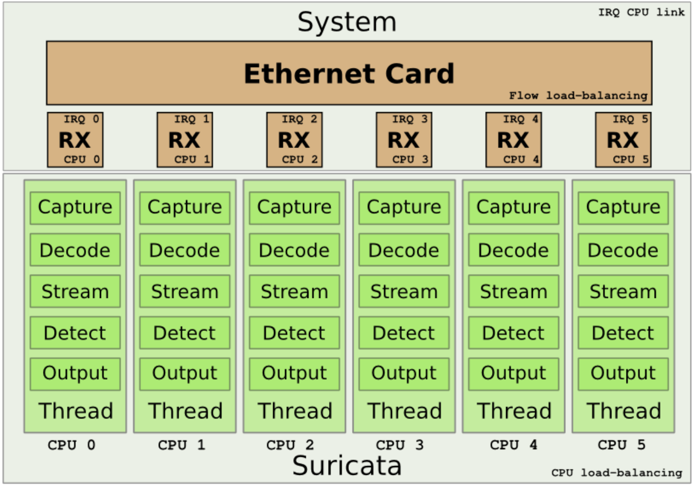
Packet loss drama (2/2)



Some numbers

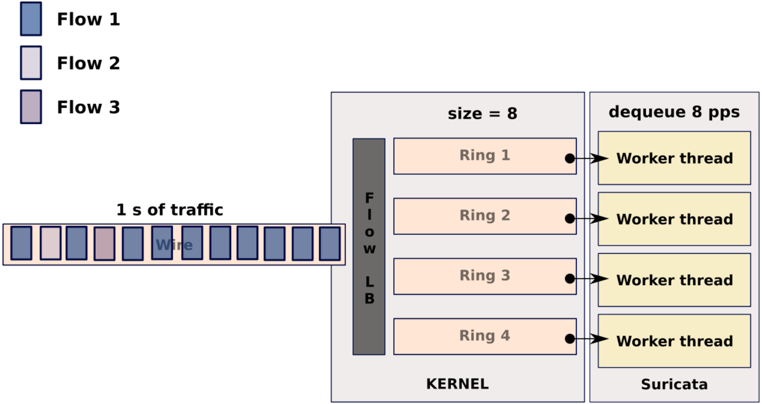
- 10% missed alerts with 3% packets loss
- 50% failed file extraction with 5.5% packets loss

Suricata load balancing

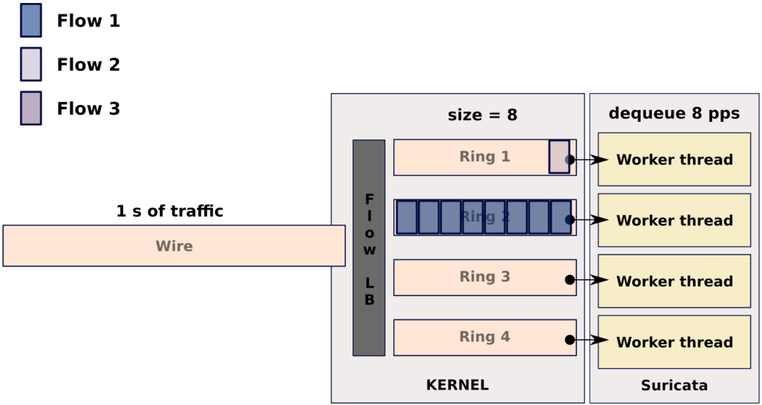


1. Bypass support in Suricata
2. XDP bypass
3. XDP in hardware mode
4. Other usages of XDP
5. AF_XDP
6. Challenge of XDP for a generic software

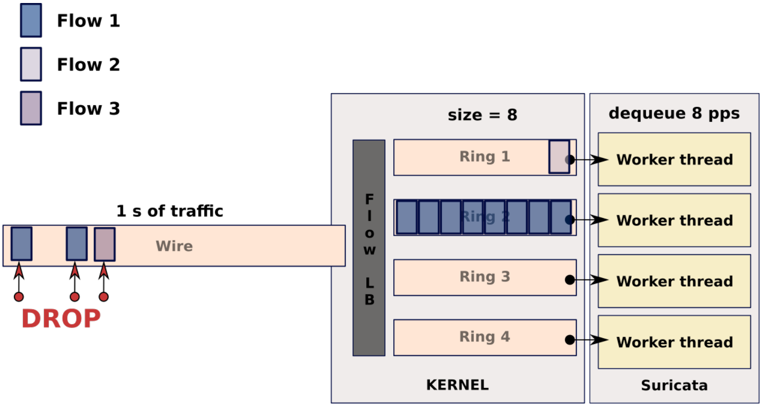
The big flow problem: elephant flow



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

Introducing bypass

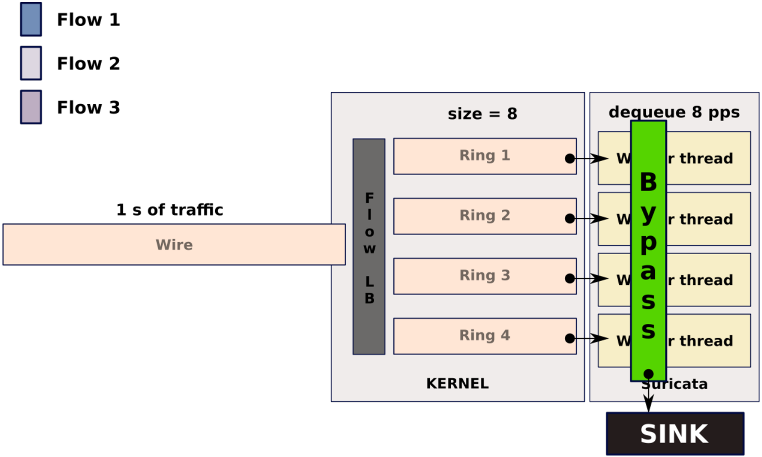
Stop packet handling as soon as possible

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

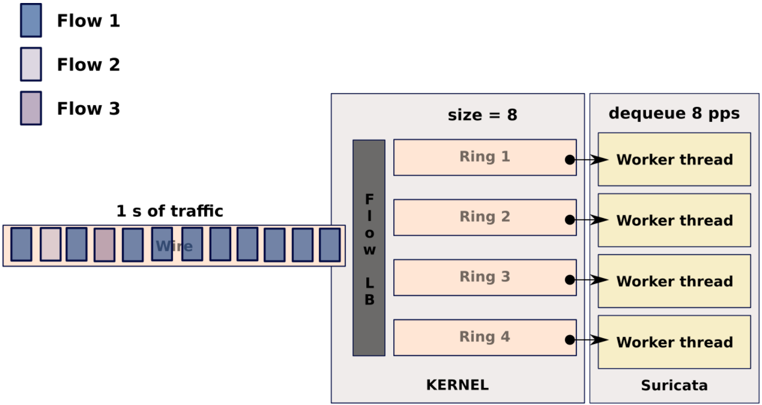
Bypass method

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

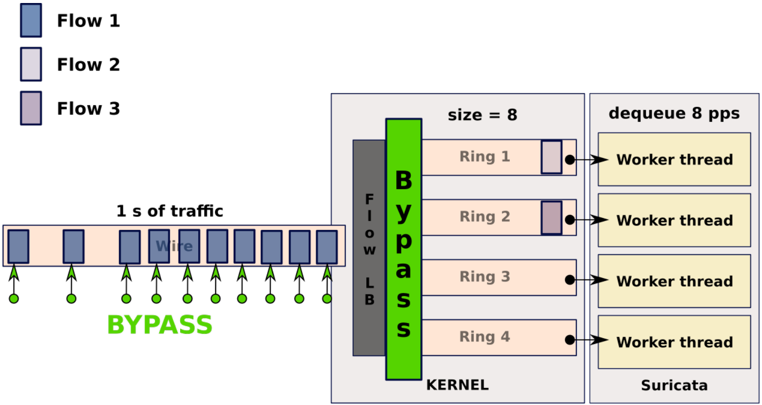
Bypassing big flow: local bypass



Bypassing big flow: capture bypass



Bypassing big flow: capture bypass



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

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;)
```

Bypass: a long running story

- Suricata 3.2.1 (Feb. 2017)
    - Suricata bypass API
    - NFQ implementation
- Suricata 4.1 (Nov. 2018)
    - Pfring HW bypass for Accolade NIC (Alfredo Cardigliano)
    - AF_PACKET eBPF socket filtering bypass
    - AF_PACKET XDP bypass
- Suricata 5.0 (Oct. 2019)
    - Netronome hardware bypass
    - Tunnel decapsulation
    - Pattern based bypass for TLS

1 Bypass support in Suricata

2 XDP bypass

3 XDP in hardware mode

4 Other usages of XDP

5 AF_XDP

6 Challenge of XDP for a generic software

Implementation

Libbpf based

- Suricata loads and install the eBPF filter
- Set up the maps and pinned them if asked

Principle

- Flow table maps in eBPF
- eBPF filter drop packet belonging to the flow in the flow table
- Suricata maintains the flow table maps

Flow table maintenance (Suricata 4.1)

eBPF update Flow table

- Pass packet if not in a bypassed flow
- Update the last seen timestamp and do accounting

Flow table dump

1. Suricata iterate on Flow table
2. Check entry with expired timeout
3. Remove them for the Flow table

Polling limitation

This is slow

- 2 syscall per item
  - Up to 30 seconds to dump a 300000 entries table
  - And we need big table

Accounting dead flow

- Long bypassed flow get accounted at expiration
- Wrong performance stats
  - Estimating bypass efficiency with flow data fails

1 Bypass support in Suricata

2 XDP bypass

3 XDP in hardware mode

4 Other usages of XDP

5 AF_XDP

6 Challenge of XDP for a generic software

XDP in hardware mode with Netronome cards

Netronome card can run XDP eBPF code

- eBPF bytecode is loaded by the card
- Maps are available
- A true offloading

Usage

- A specific function call in libbpf at eBPF installation
- That's all.
- If hardware support the code

Hardware constraints

Costly time function

- Netronome NIC CPUs get time via kernel
- Costly to get it to update last seen
- We need an algorithm update

Some minor constraints

- Limited key+value size: fixed by compressing some fields in the keys
- No per-cpu maps

Some XDP features can't be offloaded

- Some make no sense in hardware (CPU redirect)
- Some are in the roadmap
- Fixed by #ifdef in the code

Flow key compression

```
 struct flowv4_keys {
         __u32 src;
         __u32 dst;
         union {
                 __u32 ports;
                 __u16 port16[2];
         };
-        __u32 ip_proto;
-        __u16 vlan_id[2];
+        __u8 ip_proto:1;
+        __u16 vlan0:15;
+        __u16 vlan1;
 };
```

Flow key compression

```
 struct flowv4_keys {
        __u32 src;
        __u32 dst;
        union {
                __u32 ports;
                __u16 port16[2];
        };
-       __u32 ip_proto;
-       __u16 vlan_id[2];
+       __u8 ip_proto:1;
+       __u16 vlan0:15;
+       __u16 vlan1;
 };
```

u32 for ip_proto was like

Flow timeout logic update

Algorithm update

- Keep Flow in Suricata internal flow table
- Fetch eBPF flow entries when flow timeout
- Increase timeout if traffic has been seen
- Update bypassed counters

Benefit

- Work on Netronome card
- Avoid stressing system with a full dump
- Intermediate accounting for flow
- Exact per-flow accounting of bypassed traffic

Netronome RSS load balancing

Programmable Receive Side Scaling

- RSS distributes packets on multiple queues to share load
- Netronome supports RSS
- RSS load balancing can be done in eBPF code

Netronome RSS load balancing

Programmable Receive Side Scaling

- RSS distributes packets on multiple queues to share load
- Netronome supports RSS
- RSS load balancing can be done in eBPF code

Code is #KISS

```c
/* IP-pairs + protocol (UDP/TCP/ICMP) hit same CPU */
__u32 xdp_hash = tuple.src + tuple.dst;
xdp_hash = SuperFastHash((char *) &xdp_hash, 4, INITVAL + iph->protocol);
ctx->rx_queue_index = xdp_hash % RSS_QUEUE_NUMBERS;
```

1. Bypass support in Suricata

2. XDP bypass

3. XDP in hardware mode

4. Other usages of XDP

5. AF_XDP

6. Challenge of XDP for a generic software

Improving Suricata restart

Feeling like the falling whale in H2G2

- Flow taken in the middle can't be properly analyzed
- Suricata restart reset the in kernel Flow table
- Big trouble at restart and bypassed flow striking hard

Improving Suricata restart

Feeling like the falling whale in H2G2

- Flow taken in the middle can't be properly analyzed
- Suricata restart reset the in kernel Flow table
- Big trouble at restart and bypassed flow striking hard

Pinned maps for flow table

- Keep maps between Suricata run
- Previously bypassed flows are not seen again
- Suricata is not overwhelmed at restart

Flow restoration

- Bypassed flows kept in the map need to timeout
- Need to restore the flow from the eBPF map to Suricata

Tunnel Decapsulation

Minify the elephant

- Fow reconstruction implies all packets of a flow on a single thread
- IP transport tunnel reach one single thread

Let's use bpf_xdp_adjust_head

```
nh_off += 4;
proto = grhdr->proto; /* parse GRE protocol to get offset to start of inner data */
/* ... some parsing skipped */
if (grhdr->flags & GRE_CSUM)
    nh_off += 4;
if (data + nh_off > data_end) /* pass in case of error */
    return XDP_PASS;
if (bpf_xdp_adjust_head(ctx, 0 + nh_off)) /* move head of data to inner data */
    return XDP_PASS; /* pass in case of error */
/* continue treatment, data start is now inner data of GRE tunnel */
```

TLS bypass improvement

Suricata TLS bypass

- Can do TLS handshake analysis but nothing to be done on encrypted traffic
- Suricat triggers bypass when TLS session switch to encrypted

Issue due to ring buffer

- All packets of short living sessions are in ring buffer
- Bypass is not efficient

XDP pattern based bypass

```
if (app_data[0] == 0x17 /* TLS 1.2 */
        && app_data[1] == 0x3 && app_data[2] == 0x3) { /* and encrypted packet */
    tls_count = bpf_map_lookup_elem(&tls_bypass_count, &key1);
    if (tls_count)
        tls_count++;
    return XDP_DROP;
}
```

1 Bypass support in Suricata

2 XDP bypass

3 XDP in hardware mode

4 Other usages of XDP

5 AF_XDP

6 Challenge of XDP for a generic software

AF_XDP a new raw packet capture method

Principle

- eBPF filter send packet to a shared buffer
- packet reach userspace
  - Before skb creation
  - In a efficient hole compliant buffer structure

Implementation

- New capture method in Suricata (like AF_PACKET or NFQUEUE)
- Code using libbpf

I’m API with my life

Libbpf XSK API

- High level API helps a lot
  - Setup the complex data structure
  - Start without even an eBPF file
- Low level API also available

Know your hardware issue

- Bind to a queue
- Scalibility depends of hardware
  - No CPU based load balancing
  - But do we need that ?

Give me some time

Initial implementation

- libbpf is easy to use
- Suricata part was the most complex

Where is my timestamp ?

- No hardware timestamp available
- Mandatory in Suricata case
  - We are getting copy of packets
  - Case of splitted RX TX can’t be fixed

1 Bypass support in Suricata

2 XDP bypass

3 XDP in hardware mode

4 Other usages of XDP

5 AF_XDP

6 Challenge of XDP for a generic software

Easier now

libbpf

- No de facto standard for eBPF handling in 2015
- Had to patch libbpf to get it working
- libbpf is now available in distribution

Kernel side stabilization

- Less breakage when changing version

Packet decoding

No commodity decoding

- Suricata needs to handle all networks case
- Decoding in eBPF for a lot of common protocols
- Examples exist but are too simple

Would love a decoding library

- Reusable blocks
- For main IP layers and layer 2 protocols

Dealing with distributions

Distributing libbpf

- Available in Debian (sid with backport)
- Available Fedora 30
- Available in Mageia

Shipping eBPF files

- What if we need to tune feature
- Possible solutions
  - Use #ifdef and build eBPF file on prod system
    - Need to have compiler on production system
    - Security implication
  - Use (pinned) maps to setup the XDP filter
    - Need some tooling
    - Or code in Suricata

Conclusion

Suricata and XDP

- It was a long journey
- XDP toolkit has improved over time
- Features and performance are there
- AF_XDP is promising

More information

- Stamus Networks: `https://www.stamus-networks.com/`
- Suricata and XDP whitepaper: `https://tinyurl.com/y6nqhalu`
- Suricata code: `https://github.com/oisf/suricata`
- Libbpf code: `https://github.com/libbpf/libbpf`

Questions ?

Thanks for their help

- Alexei Starovoitov
- Daniel Borkmann
- Jesper Dangaard Brouer
- And Netronome Team
  - David Beckett
  - Jakub Kicinski
  - Jiong Wang

Contact me

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

More information

- Suricata: `https://www.suricata-ids.org/`
- Stamus Networks: `https://www.stamus-networks.com/`
- Suricata and XDP whitepaper: `https://tinyurl.com/y6nqhalu`
- Suricata code: `https://github.com/oisf/suricata`
- Libbpf code: `https://github.com/libbpf/libbpf`