XDP(eXpress Data Path) Overview

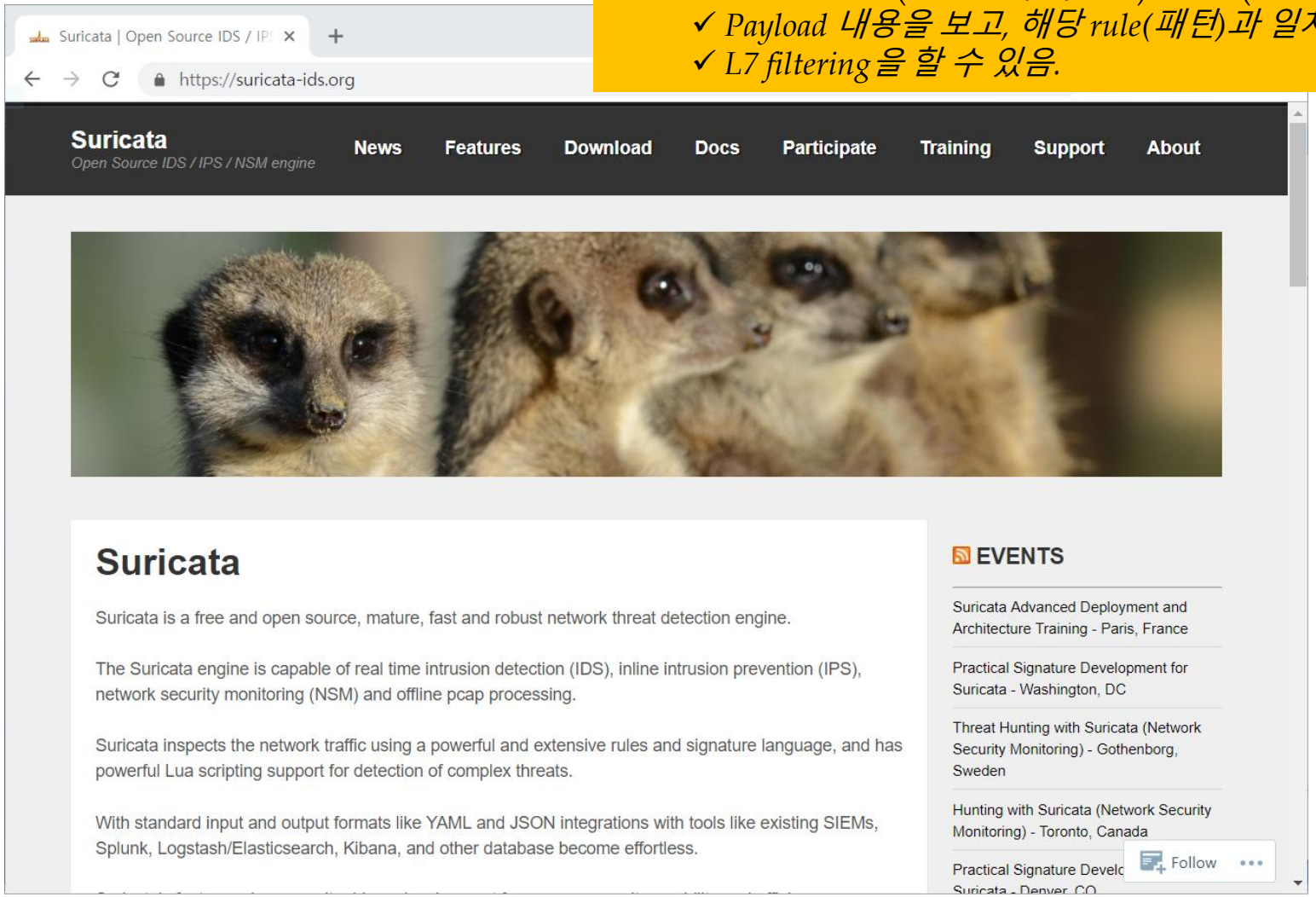
Slowboot(chunghan.yi@gmail.com)

Doc. Revision: 0.1

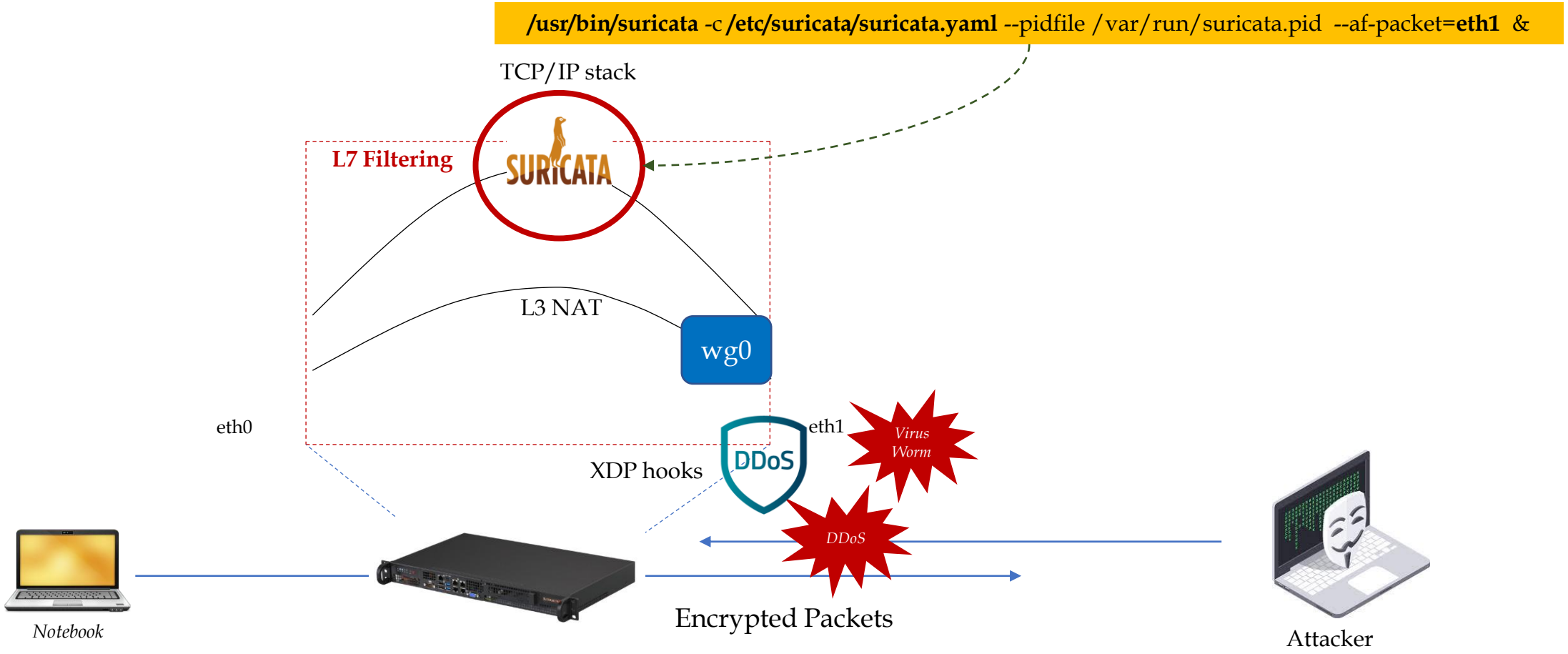
Contents

1. Suricata IDS/IPS(1) - Overview(1)

Suricata: 고성능 IDS(침입탐지 시스템) or IPS(침입 방지/차단 시스템)
- ✓ *Payload 내용을 보고, 해당 rule(패턴)과 일치하는지 여부를 찾아냄.*
- ✓ *L7 filtering을 할 수 있음.*



1. Suricata IDS/IPS(1) – Overview(2)



1. Suricata IDS/IPS(1) - Overview(3)

https://suricata.readthedocs.io/en/latest/rules/intro.html

Suricata
latest

Search docs

1. What is Suricata
2. Installation
3. Command Line Options
4. Suricata Rules
   - 4.1. Rules Format
     - 4.1.1. Action
     - 4.1.2. Protocol
     - 4.1.3. Source and destination
     - 4.1.4. Ports (source and destination)
     - 4.1.5. Direction
     - 4.1.6. Rule options
   - 4.2. Meta Keywords
   - 4.3. IP Keywords
   - 4.4. TCP keywords
   - 4.5. ICMP keywords
   - 4.6. Payload Keywords
   - 4.7. Transformations
   - 4.8. Prefiltering Keywords
   - 4.9. Flow Keywords
   - 4.10. Bypass Keyword

Read the Docs v: latest

Docs » 4. Suricata Rules » 4.1. Rules Format

Edit on GitH

4.1. Rules Format

Signatures play a very important role in Suricata. In most occasions people are using existing rulesets.

The official way to install rulesets is described in Rule Management with Suricata-Update.

This Suricata Rules document explains all about signatures; how to read, adjust and create them

A rule/signature consists of the following:

- The **action**, that determines what happens when the signature matches
- The **header**, defining the protocol, IP addresses, ports and direction of the rule.
- The **rule options**, defining the specifics of the rule.

An example of a rule is as follows:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA
+..)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK ";
pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124;
classtype:trojan-activity; sid:2008124; rev:2;)
```

In this example, red is the action, green is the header and blue are the options.

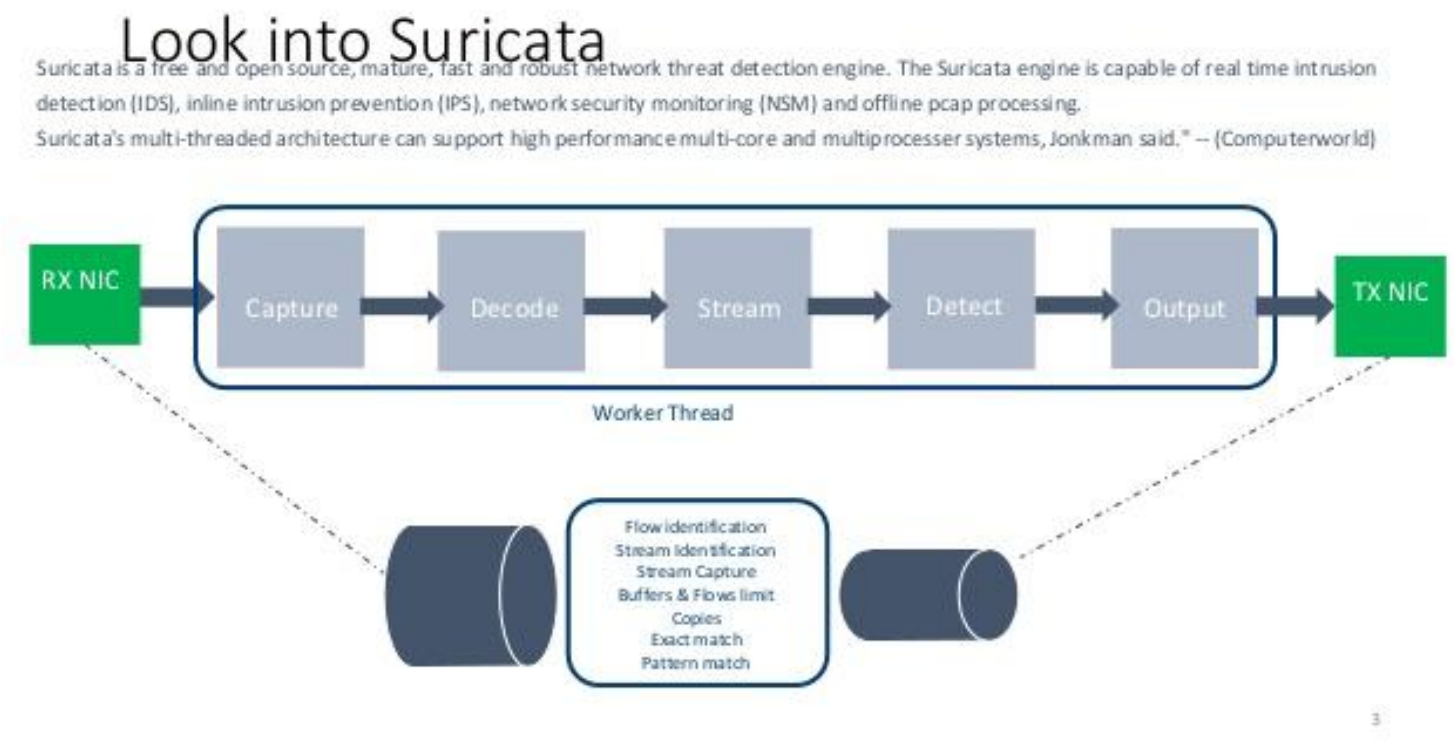
We will be using the above signature as an example throughout this section, highlighting the different parts of the signature. It is a signature taken from the database of Emerging Threats, an open database featuring lots of rules that you can freely download and use in your Suricata instance.

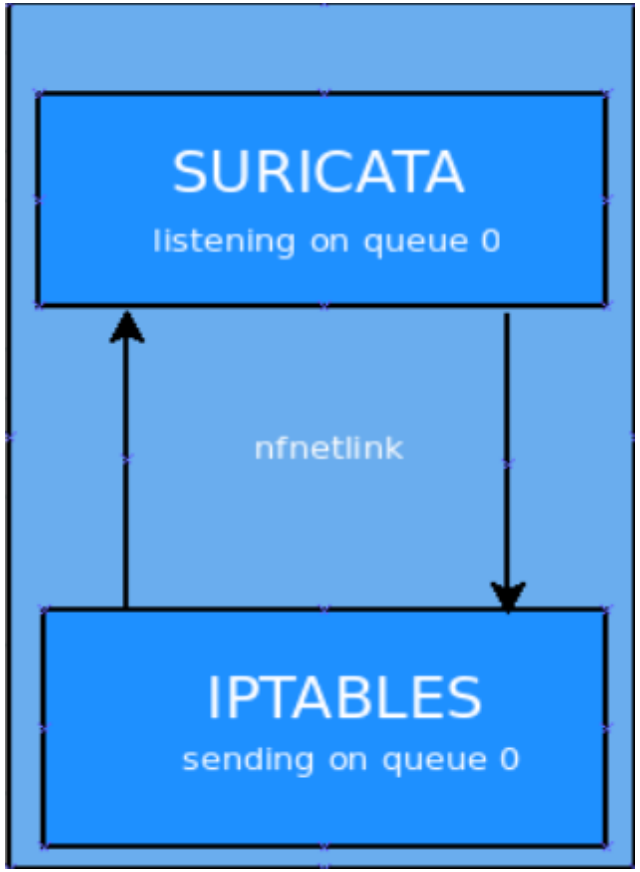
<Suricata rule 형식>

```
alert udp $EXTERNAL_NET 389 -> $HOME_NET 389 (msg:"ET DOS CLDAP Amplification Reflection (PoC based)"; d
size:52; content:"|30 84 00 00 00 2d 02 01 01 63 84 00 00 00 24 04 00 0a 01 00|"; fast_pattern; threshol
d:type both, count 100, seconds 60, track by_src; metadata: former_category DOS; reference:url,www.akama
i.com/us/en/multimedia/documents/state-of-the-internet/cldap-threat-advisory.pdf; reference:url,packetst
ormsecurity.com/files/139561/LDAP-Amplication-Denial-Of-Service.html; classtype:attempted-dos; sid:20245
84; rev:1; metadata:affected_product Windows_XP_Vista_7_8_10_Server_32_64_Bit, affected_product Linux, a
ttack_target Server, deployment Perimeter, signature_severity Major, created_at 2017_08_16, performance_
impact Significant, updated_at 2017_08_16;)

alert udp $EXTERNAL_NET any -> $HOME_NET 389 (msg:"ET DOS Potential CLDAP Amplification Reflection"; con
tent:"objectclass0"; fast_pattern; threshold:type both, count 200, seconds 60, track by_src; metadata: f
ormer_category DOS; reference:url,www.akamai.com/us/en/multimedia/documents/state-of-the-internet/cldap-
threat-advisory.pdf; reference:url,packetstormsecurity.com/files/139561/LDAP-Amplication-Denial-Of-Servi
ce.html; classtype:attempted-dos; sid:2024585; rev:1; metadata:affected_product Windows_XP_Vista_7_8_10_
Server_32_64_Bit, affected_product Linux, attack_target Client_and_Server, deployment Perimeter, signatu
re_severity Major, created_at 2017_08_16, performance_impact Significant, updated_at 2017_08_16;)

alert udp $EXTERNAL_NET any -> any 11211 (msg:"ET DOS Possible Memcached DDoS Amplification Query (set)"
; content:"|00 00 00 00 00 01 00|"; depth:7; fast_pattern; content:"|0d 0a|"; distance:0; within:20; isd
ataat:!1,relative; threshold: type both, count 100, seconds 60, track by_dst; flowbits:set,ET.memcached.
ddos; metadata: former_category DOS; reference:url,blog.cloudflare.com/memcrashed-major-amplification-at
tacks-from-port-11211/; classtype:attempted-dos; sid:2025401; rev:2; metadata:affected_product Windows_X
P_Vista_7_8_10_Server_32_64_Bit, affected_product Linux, attack_target Server, deployment Perimeter, sig
nature_severity Major, created_at 2018_03_01, performance_impact Low, updated_at 2018_03_01;)

alert udp any 11211 -> $EXTERNAL_NET any (msg:"ET DOS Possible Memcached DDoS Amplification Response Out
bound"; flowbits:isset,ET.memcached.ddos; content:"STATS|20|pid"; depth:9; fast_pattern; threshold: type
 both, count 100, seconds 60, track by_dst; metadata: former_category DOS; reference:url,blog.cloudflare
.com/memcrashed-major-amplification-attacks-from-port-11211/; classtype:attempted-dos; sid:2025402; rev:
1; metadata:affected_product Windows_XP_Vista_7_8_10_Server_32_64_Bit, affected_product Linux, attack_ta
rget Server, deployment Perimeter, signature_severity Major, created_at 2018_03_01, performance_impact L
ow, updated_at 2018_03_01;)
```

<Suricata rule 예>

1. Suricata IDS/IPS(1) - Overview(4)





1. Suricata IDS/IPS(2) – NFQ Integration(1)

<How to build suricata with nfqueue>

$ ./configure --prefix=/usr/ --sysconfdir=/etc/ --localstatedir=/var/ --enable-nfqueue

$ make clean && make

$ sudo make install-full

$ sudo ldconfig

<suricata.yaml 파일 편집>

$ sudo vi /etc/suricata/suricata.yaml

…

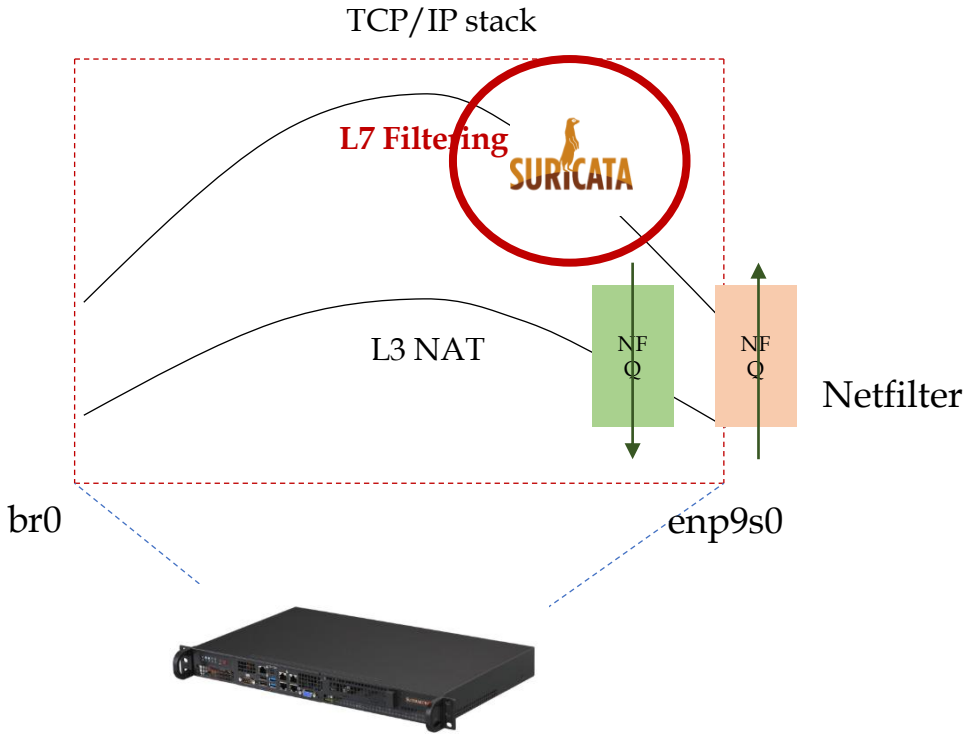
nfq:

  mode: **repeat** # 이 option이 중요함. L7 filtering 후, NAT/SPN 처리 가능하게 함.

  repeat-mark: 1

  repeat-mask: 1

…



이 방식으로도 L7 filtering이 가능하지만, XDP를 사용해야 대용량 트래픽 처리가 가능해진다.

1. Suricata IDS/IPS(2) – NFQ Integration(2)

<netfilter 규칙 추가하기>

$ sudo iptables -I INPUT -i enp9s0 -j NFQUEUE # enp9s0: spnbox WAN port

$ sudo iptables -I OUTPUT -o enp9s0 -j NFQUEUE

<suricata rule 추가하기>

$ sudo vi /etc/suricata/rules/drop.rules

…

drop tcp any any -> any 22

drop tcp any any -> any any (msg:"Detect test string"; content:"security_gateway"; nocase;)

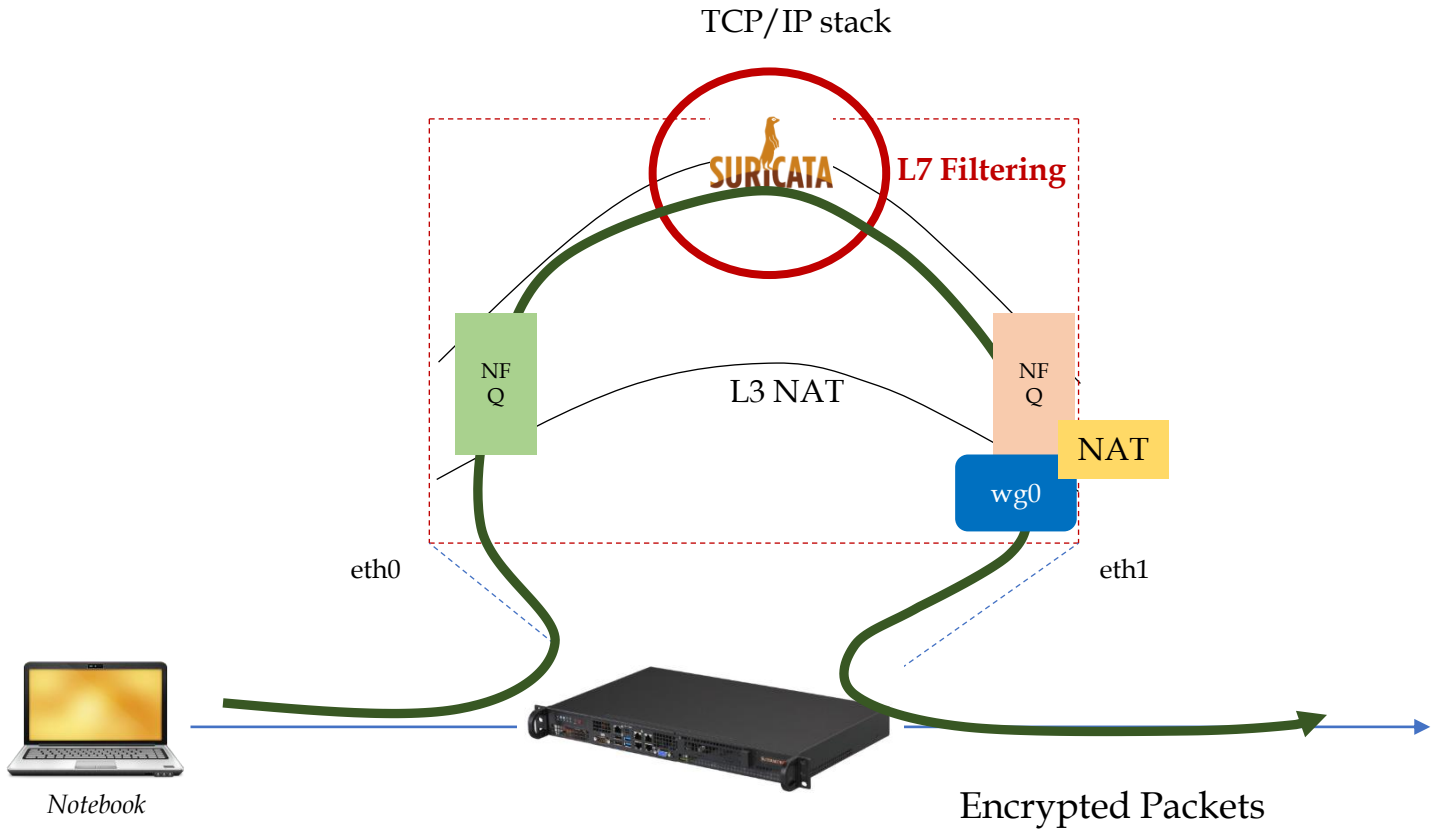
drop tcp any any -> any any (msg:"Detect test string"; content:"path=cmd&msg"; nocase;)

…

<suricata daemon 실행하기>

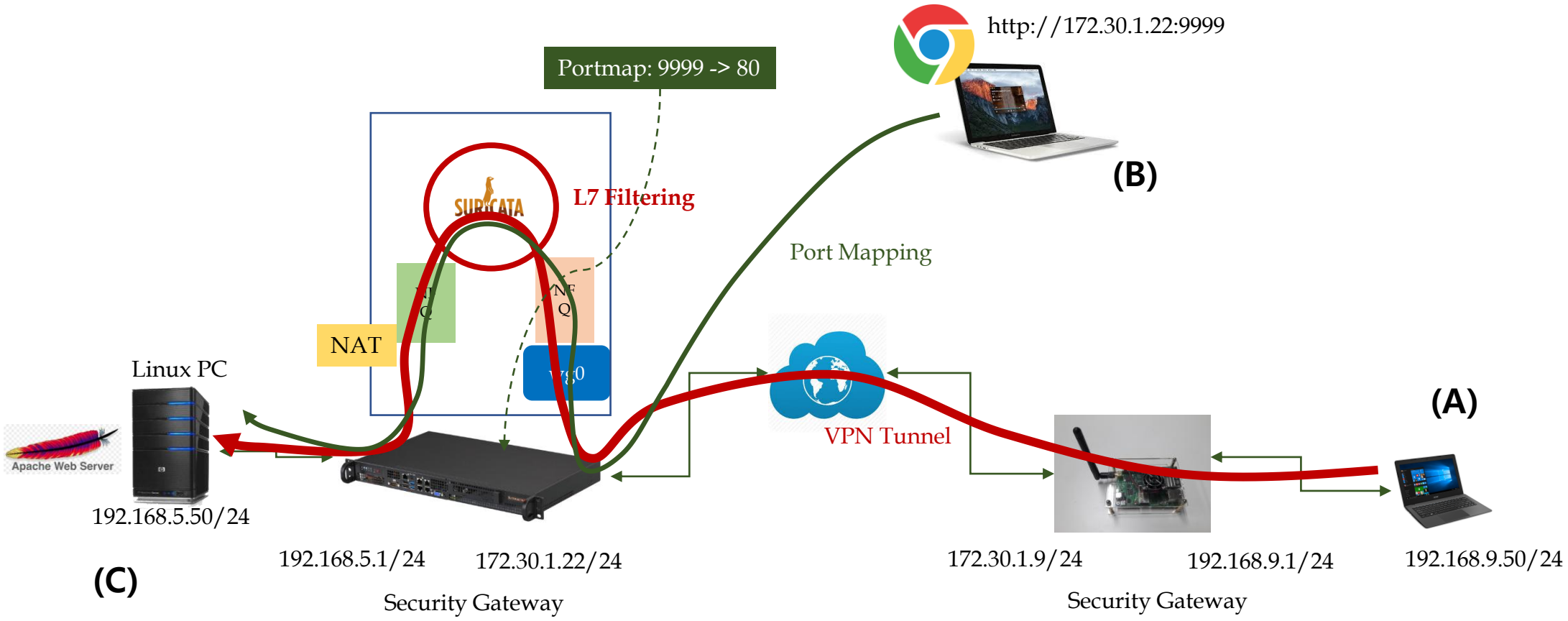
$ sudo suricata -c /etc/suricata/suricata.yaml -q 0 &

1. Suricata IDS/IPS(2) – NFQ Integration(3)




```
$ sudo iptables -I FORWARD -j NFQUEUE          # forwarding packet L7 filtering
$ sudo iptables -I INPUT -i br0 -j NFQUEUE     # localhost L7 filtering
$ sudo iptables -I OUTPUT -o br0 -j NFQUEUE    # localhost L7 filtering
$ sudo suricata -c /etc/suricata/suricata.yaml -q 0 &
```

1. Suricata IDS/IPS(2) – NFQ Integration(4)



1. Suricata IDS/IPS(3) – XDP Integration(1)

<Suricata build 시 설치해야 하는 Ubuntu packages>

Recommended:

```
apt-get install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev   \
                libnet1-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev \
                libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev        \
                libnss3-dev libgeoip-dev liblua5.1-dev libhiredis-dev libevent-dev \
                python-yaml rustc cargo
```

Extra for iptables/nftables IPS integration:

```
apt-get install libnetfilter-queue-dev libnetfilter-queue1  \
                libnetfilter-log-dev libnetfilter-log1      \
                libnfnetlink-dev libnfnetlink0
```

1. Suricata IDS/IPS(3) - XDP Integration(2)

sudo apt-get install clang llvm

17.4.3.4. libbpf

Suricata uses libbpf to interact with eBPF and XDP. This library is available in the Linux tree. Before Linux 4.16, a patched libbpf library is also needed:

```
git clone -b libbpf-release  https://github.com/regit/linux.git
```

If you have a recent enough kernel, you can skip this part.

Now, you can build and install the library

```
cd linux/tools/lib/bpf/
make && sudo make install

sudo make install_headers
sudo ldconfig
```

17.4.4. Compile and install Suricata

To get Suricata source, you can use the usual

```
git clone  https://github.com/OISF/suricata.git
cd suricata && git clone https://github.com/OISF/libhtp.git -b 0.5.x

./autogen.sh
```

Then you need to add the ebpf flags to configure

```
CC=clang ./configure --prefix=/usr/ --sysconfdir=/etc/ --localstatedir=/var/ \
--enable-ebpf --enable-ebpf-build

make clean && make
sudo  make install-full
sudo ldconfig
sudo mkdir /etc/suricata/ebpf/
```

1. Suricata IDS/IPS(3) - XDP Integration(3)

$ **sudo vi /etc/suricata/suricata.yaml**

…

af-packet:

- interface: enp9s0

cluster-id: 97

cluster-type: cluster_qm

defrag: yes

xdp-mode: driver

xdp-filter-file: /etc/suricata/ebpf/xdp_filter.bpf

bypass: yes

use-mmap: yes

ring-size: 200000

…

<suricata config file 편집>

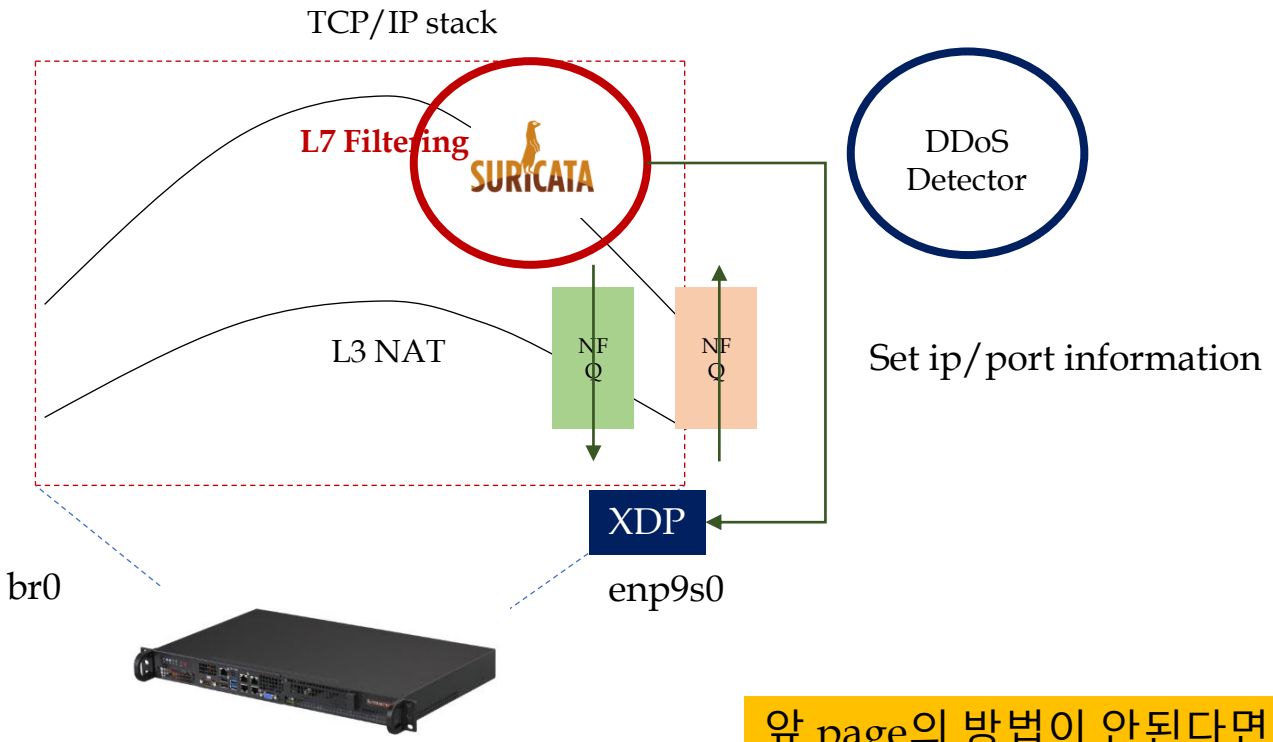
<suricata 실행하기>

$ LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib64/
$ export LD_LIBRARY_PATH

$ **/usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid --af-packet=enp9s0 -vvv &**

문제는 XDP enable 과정에서 에러 발생함.

1. Suricata IDS/IPS(3) - XDP Integration(4)



앞 page의 방법이 안된다면, 이런 형태로 구현하자.
즉, detection은 suricata에서 하고, 차단은 XDP에서 하도록 만들자.

2. BPF & eBPF Overview(1) – Classical BPF(1)

```
# tcpdump -i eth0 tcp dst port 22 -d

(000) ldh      [12]                            # Ethertype
(001) jeq      #0x86dd          jt 2    jf 6   # is IPv6?
(002) ldb      [20]                            # IPv6 next header field
(003) jeq      #0x6             jt 4    jf 15  # is TCP?
(004) ldh      [56]                            # TCP dst port
(005) jeq      #0x16            jt 14   jf 15  # is port 22?
(006) jeq      #0x800           jt 7    jf 15  # is IPv4?
(007) ldb      [23]                            # IPv4 protocol field
(008) jeq      #0x6             jt 9    jf 15  # is TCP?
(009) ldh      [20]                            # IPv4 flags + frag. offset
(010) jset     #0x1fff          jt 15   jf 11  # fragment offset is != 0?
(011) ldxb     4*([14]&0xf)                    # x := 4 * header_length (words)
(012) ldh      [x + 16]                        # TCP dest port
(013) jeq      #0x16            jt 14   jf 15  # is port 22?
(014) ret      #262144                         # trim to 262144 bytes, return packet
(015) ret      #0                              # drop packet
```

tcpdump → libpcap → BPF bytecode → kernel interpreter / JIT

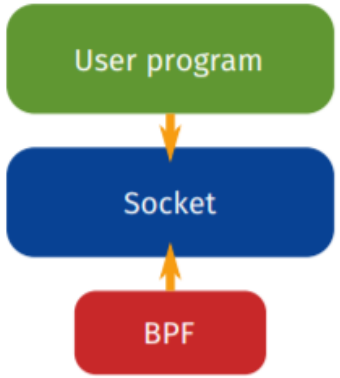
BPF filter attached to socket to filter packets and avoid useless copies

2. BPF & eBPF Overview(1) – Classical BPF(2)

BPF == Lightweight VM

BPF is an assembly-like language with registers and stack, integer arithmetic, conditional branches. JIT-compilable, for performances.

Usage: filter packets **in the kernel** with programs coming **from user space**

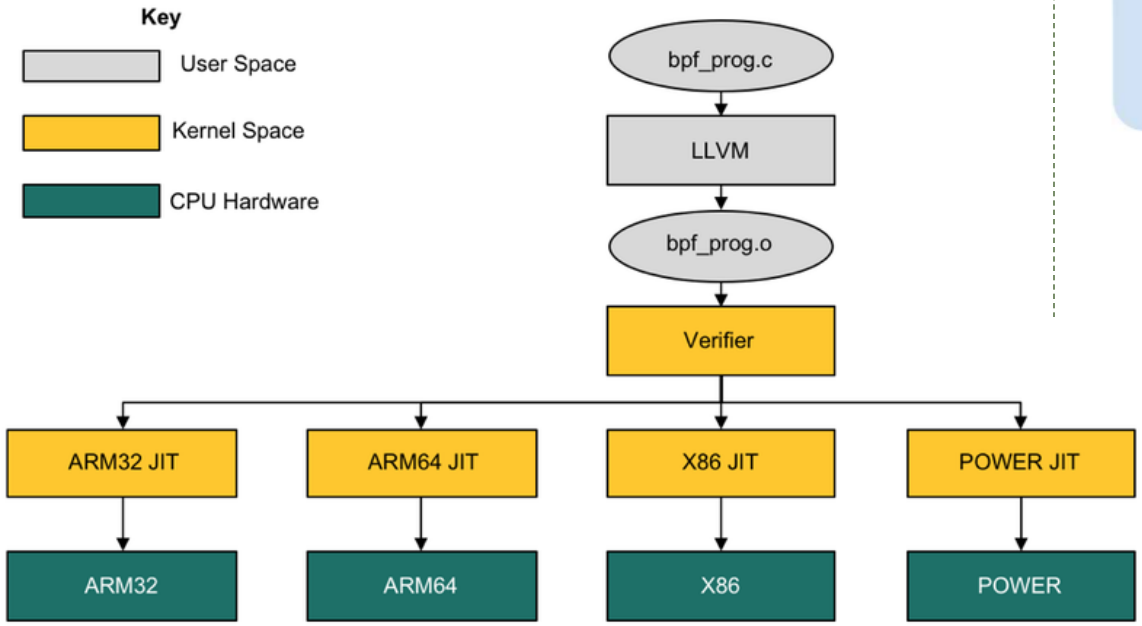


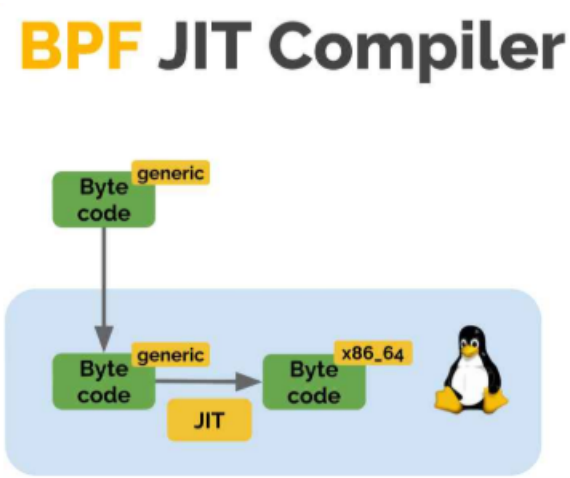

```
int s = socket (PF_INET, SOCK_RAW, IPPROTO_TCP);
setsockopt(s, SOL_SOCKET, SO_ATTACH_FILTER, &bpf_prog, sizeof(bpf_prog));
```

Safety ensured by in-kernel verifier:

- No backward jumps
- Program limited to 4096 instructions
- Dynamic packet-boundary checks
- Etc.

2. BPF & eBPF Overview(1) – Classical BPF(3)





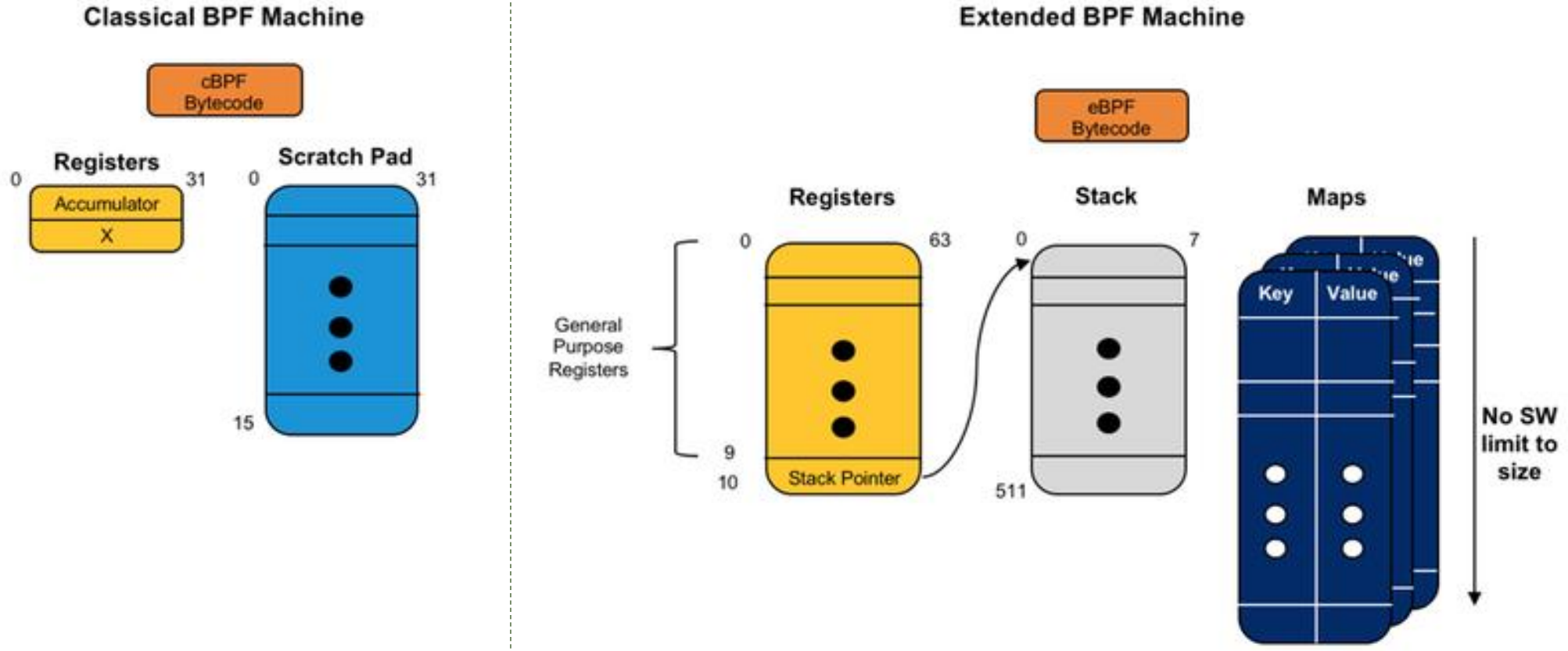
JIT Compiler

- Ensures native execution performance without requiring to understand CPU
- Compiles BPF bytecode to CPU architecture specific instruction set

Supported architectures:

- X86_64, arm64, ppc64, s390x, mips64, sparc64, arm

2. BPF & eBPF Overview(2) – Extended BPF(1)



eBPF == Virtual Machine in Kernel

2. BPF & eBPF Overview(2) – Extended BPF(2)

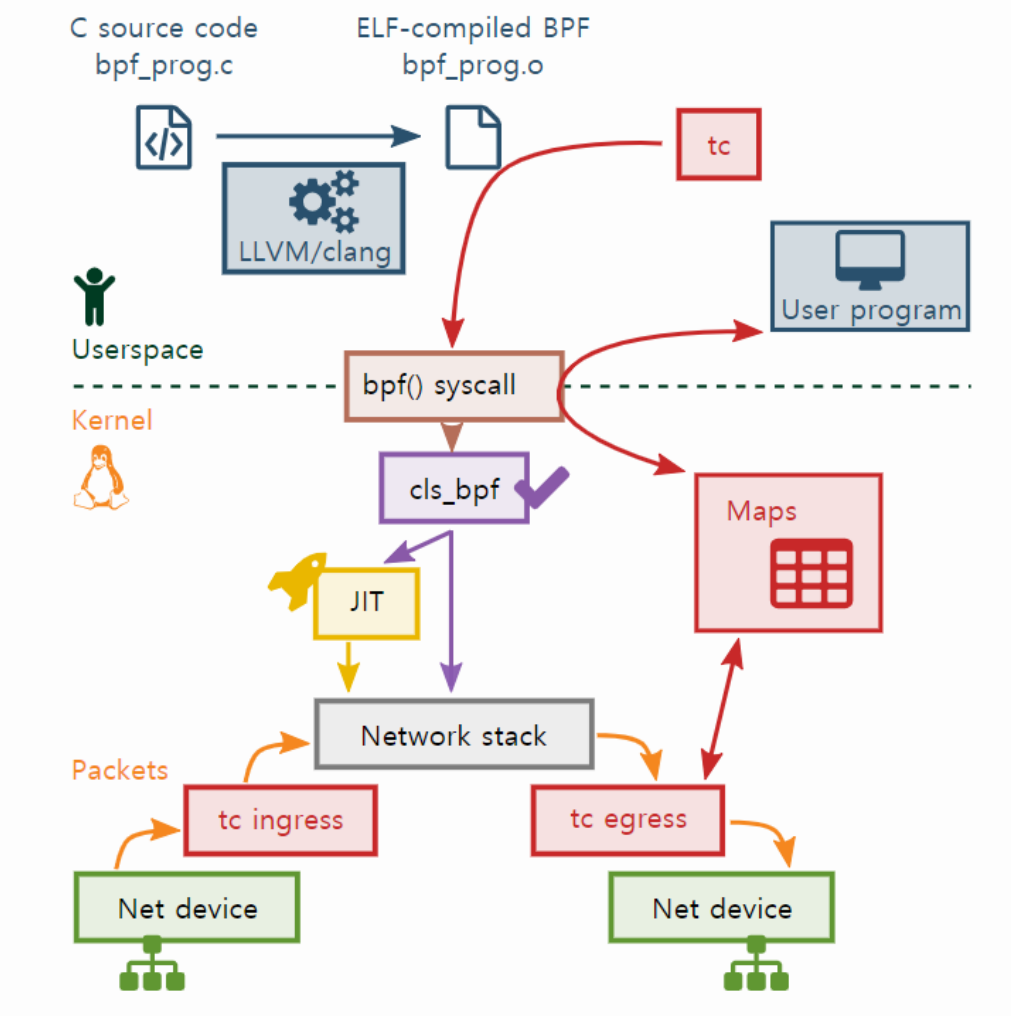
eBPF is an assembly-like language that allows writing userspace program and running it in the Linux kernel.

It is based on a former version (BPF or cBPF) and has been under heavy development over the last couple of years.

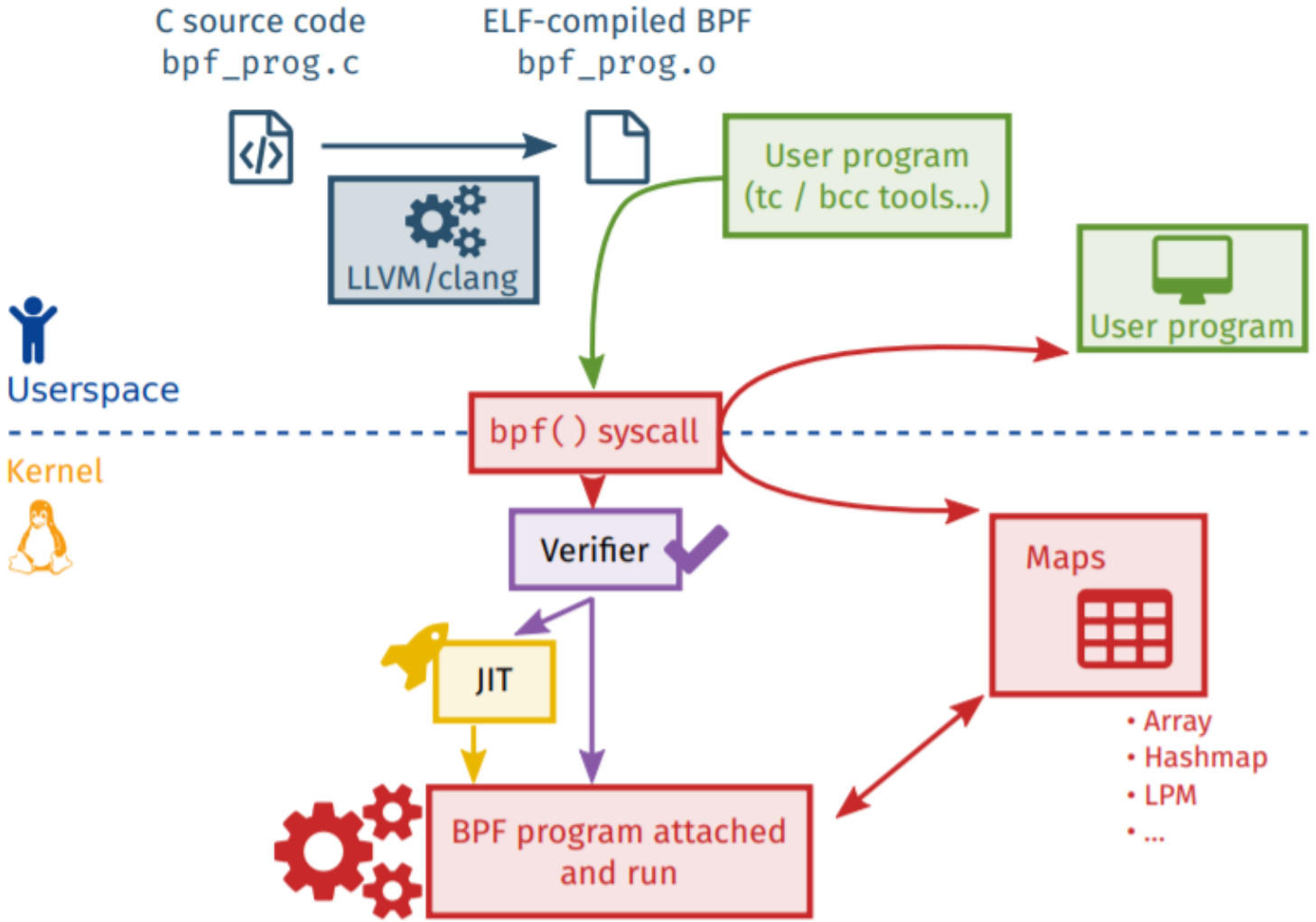
We work with eBPF tc hook, meaning that the program will be:

1.Compiled with clang/LLVM from C,
2.Injected into the kernel with tc (Linux tool for traffic control),
that relies on the bpf() syscall to do that, and sent to the cls_bpf classifier
3.Checked by the kernel verifier, to ensure that it is safe and secure
and will not crash the kernel or leak memory,
4.Possibly Just-In-Time (JIT)-compiled,
5.And then attached to the traffic control interface.

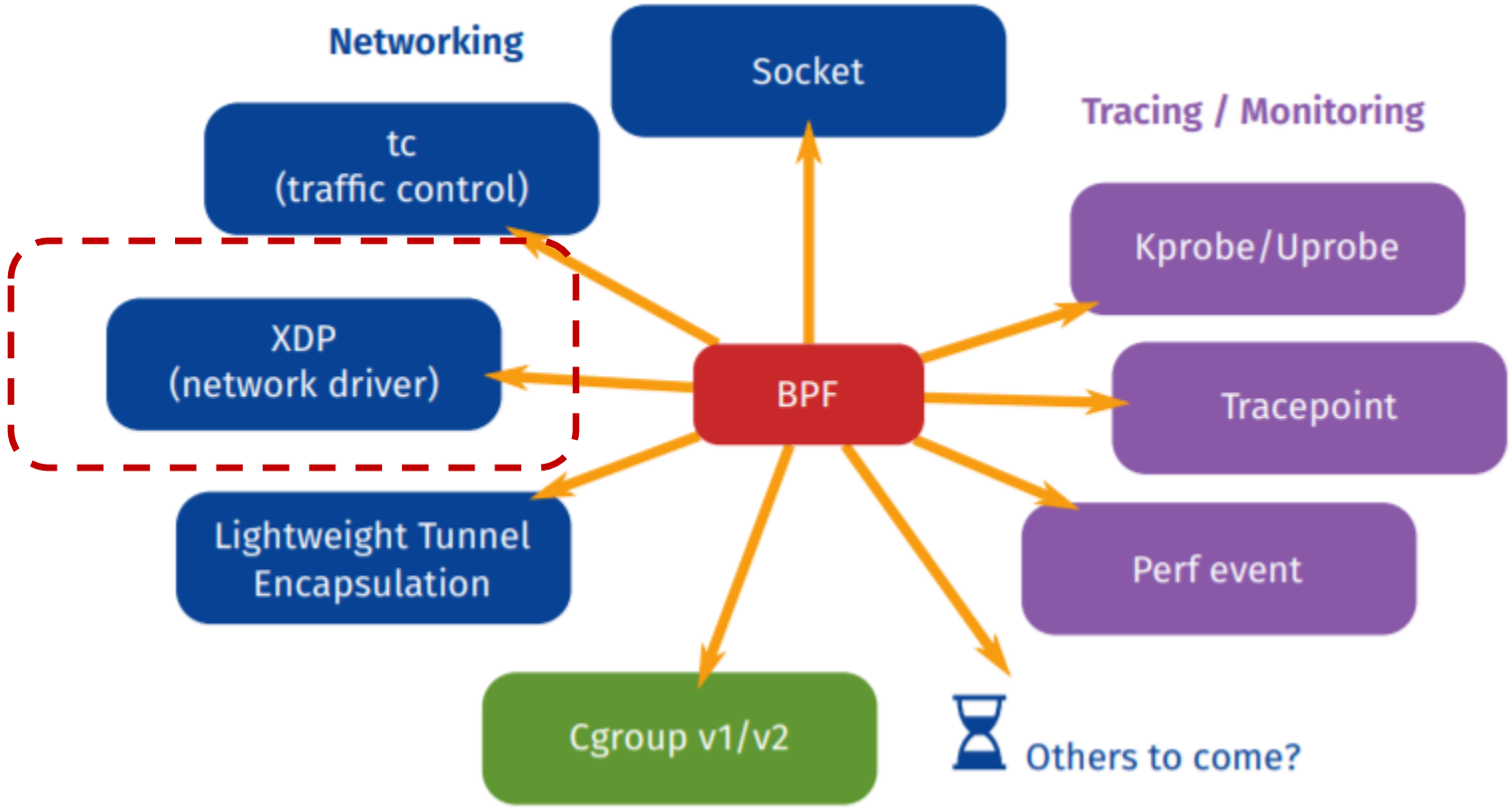
eBPF : extended Berkeley Packet Filter



2. BPF & eBPF Overview(2) – Extended BPF(3)

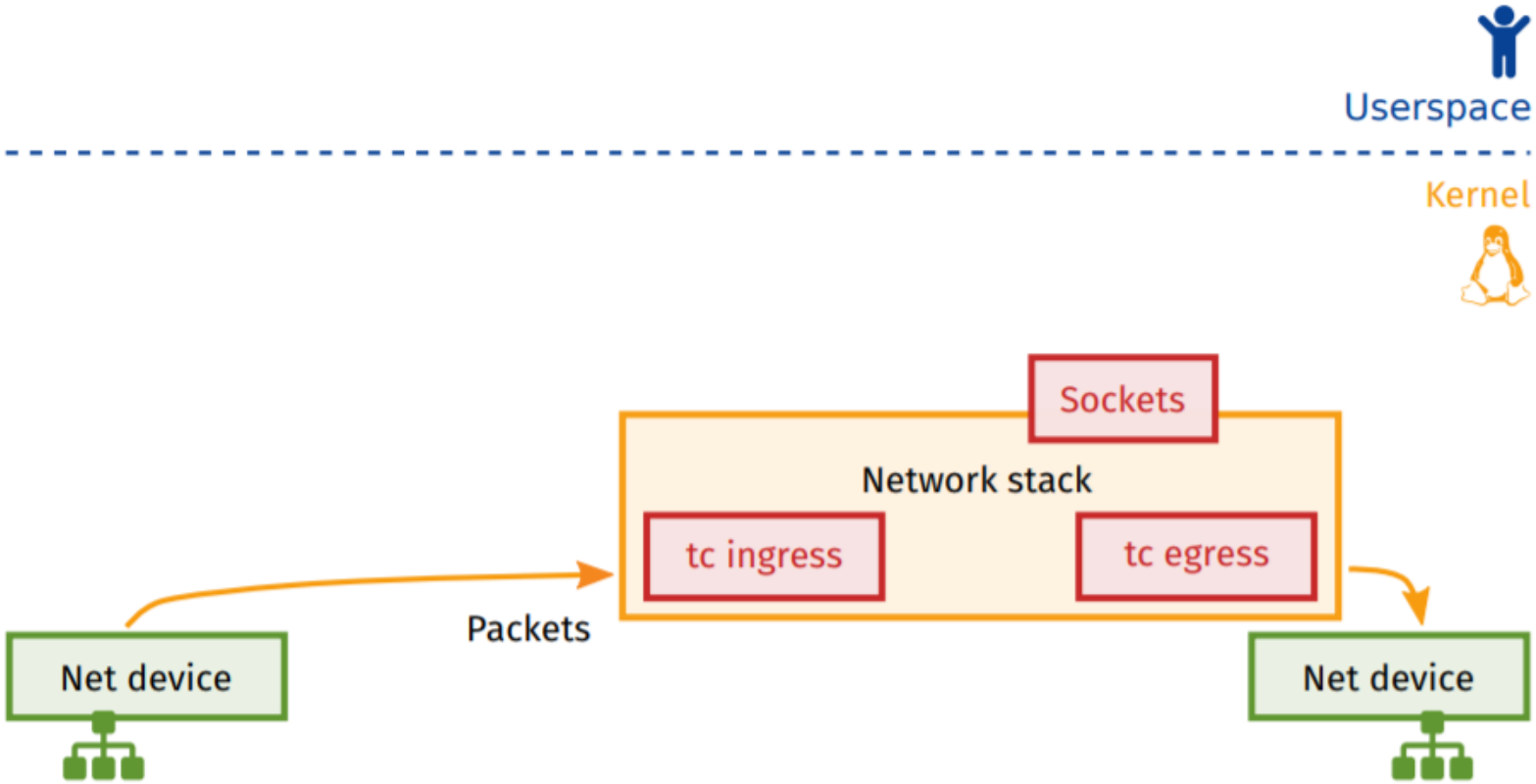


2. BPF & eBPF Overview(2) – Extended BPF(4)

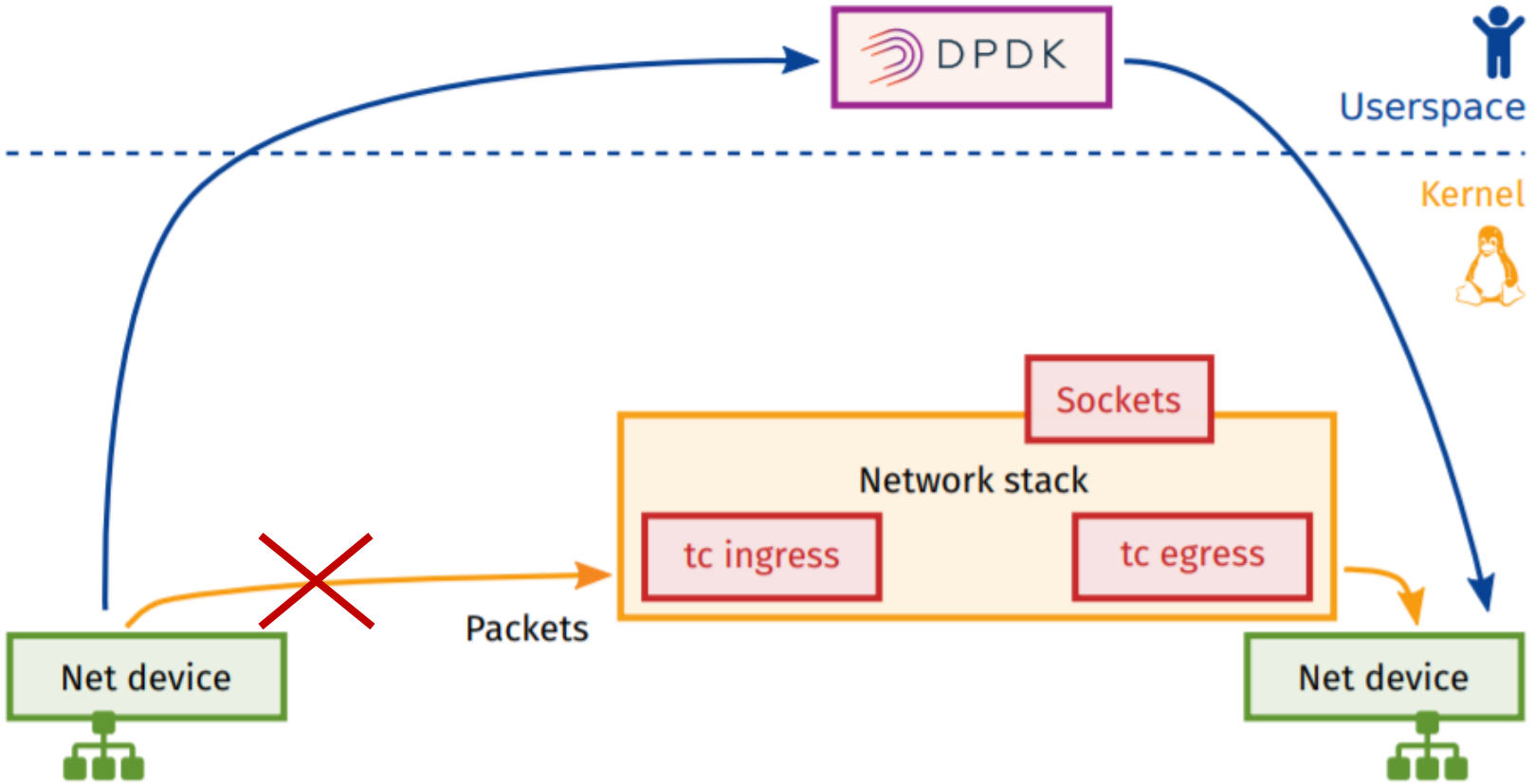


eBPF를 사용하기 위해서는 kernel 내에 eBPF를 사용하도록 작업된 부분이 있어야만 한다.

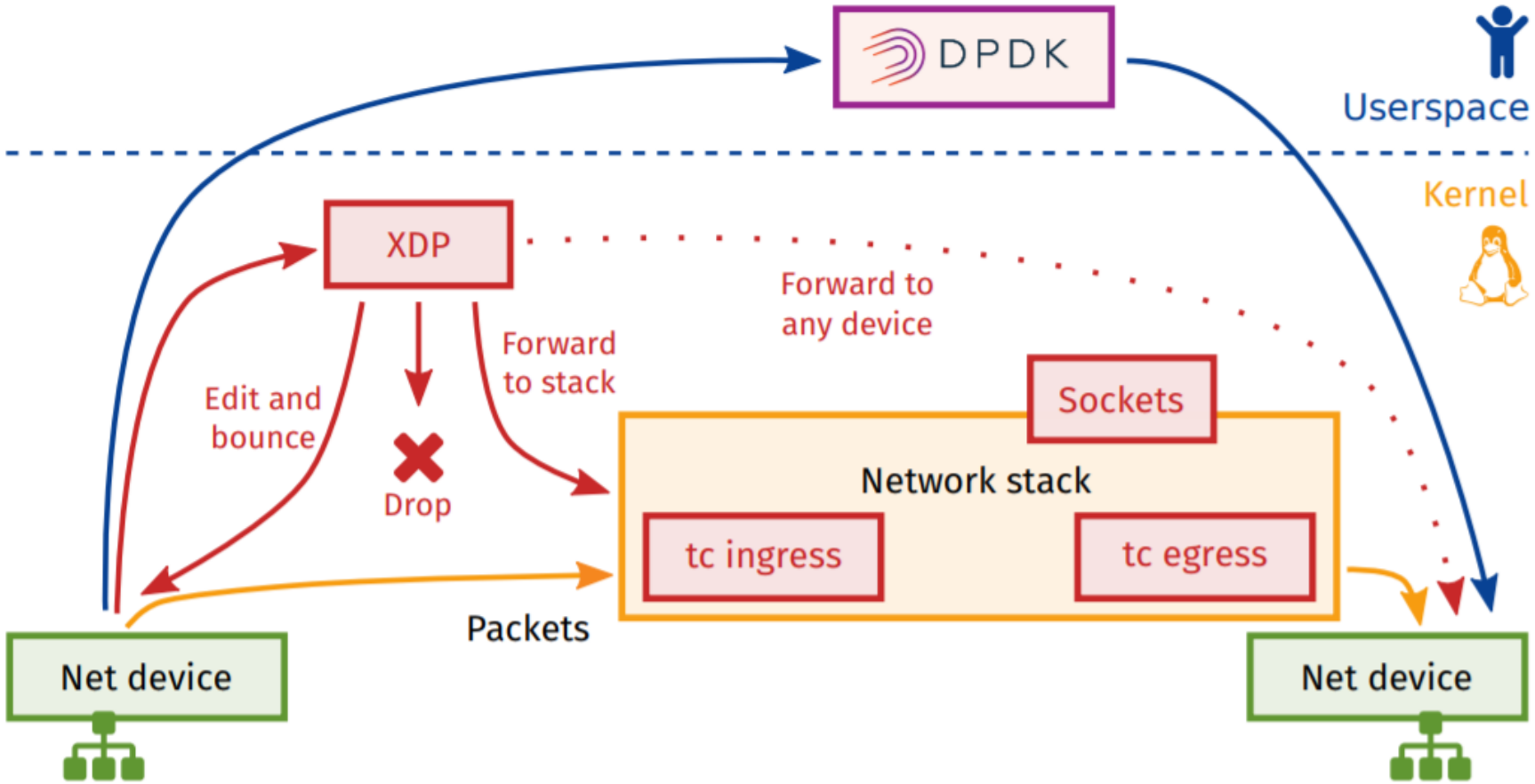
3. XDP Overview(1) – XDP vs DPDK(1)



3. XDP Overview(1) – XDP vs DPDK(2)



3. XDP Overview(1) – XDP vs DPDK(3)

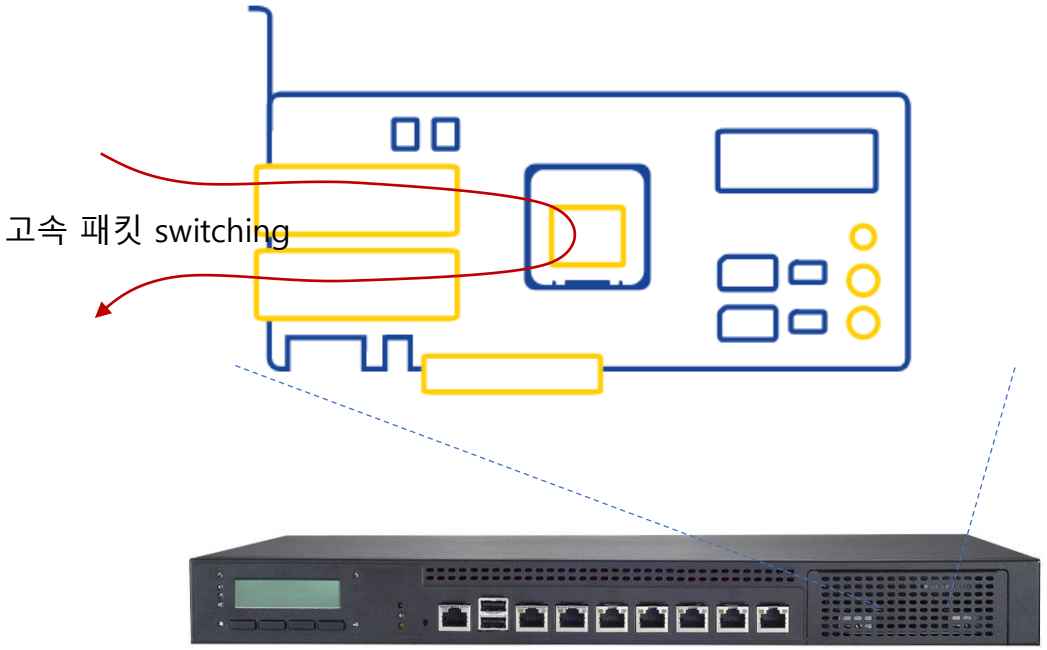


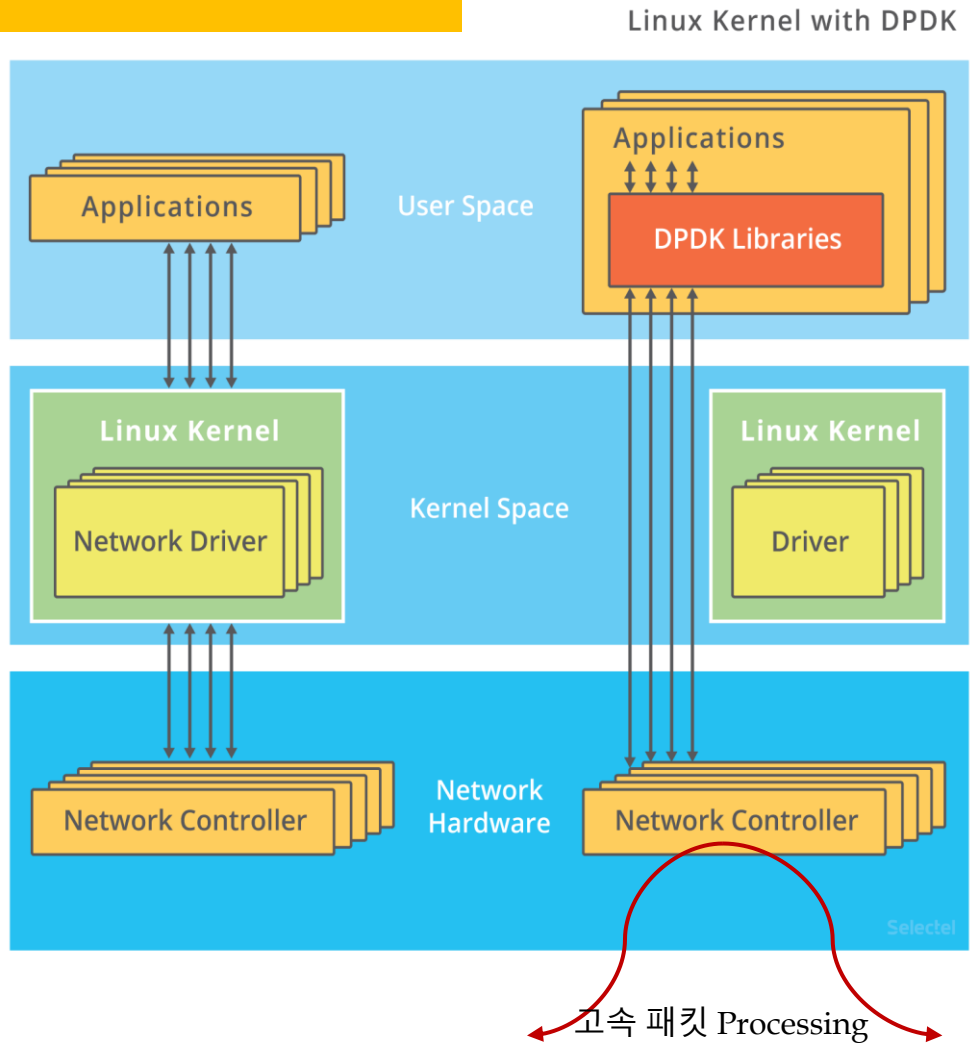
3. XDP Overview(2) – DPDK Overview(1)

DPDK: Data Plane Development Kit

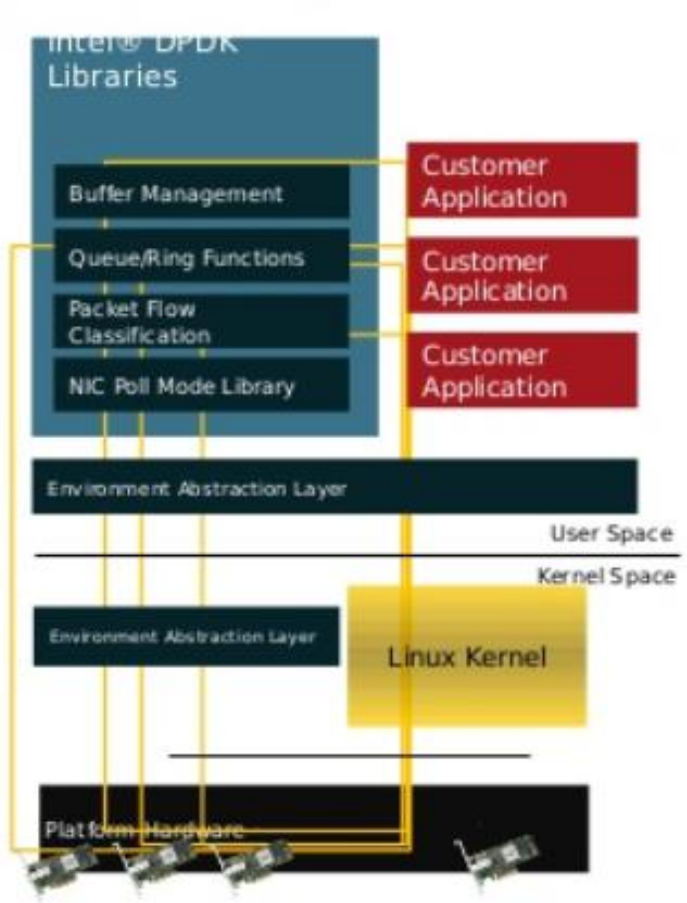
- ✓ *엄청난 속도 향상을 얻을 수는 있으나, Kernel 기능을 전혀 사용할 수 없는 단점 있음.*
- ✓ *Intel, Mellanox, Netronome, Broadcom 등이 주축되어 개발*

DPDK enabled Smart NIC(FPGA)

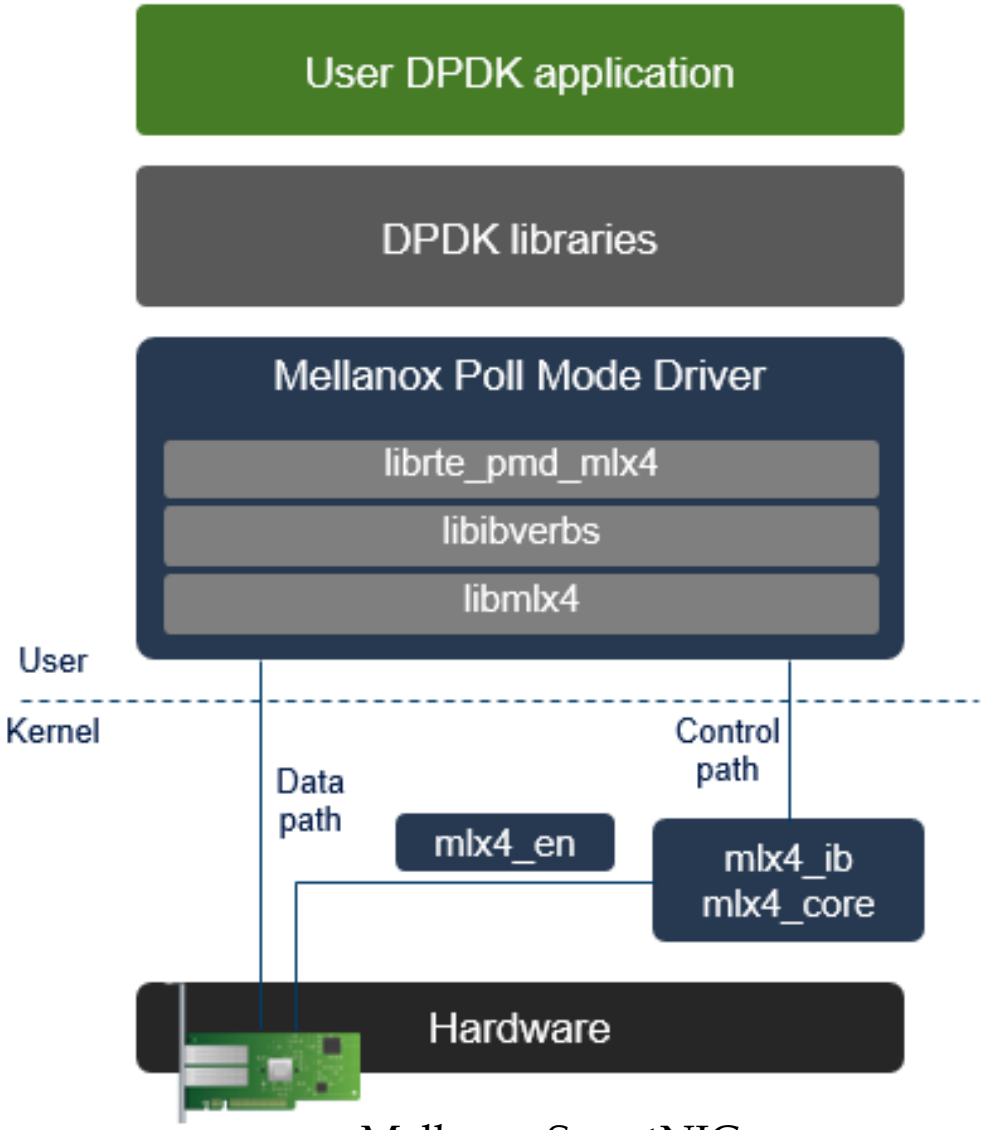




3. XDP Overview(2) – DPDK Overview(2)

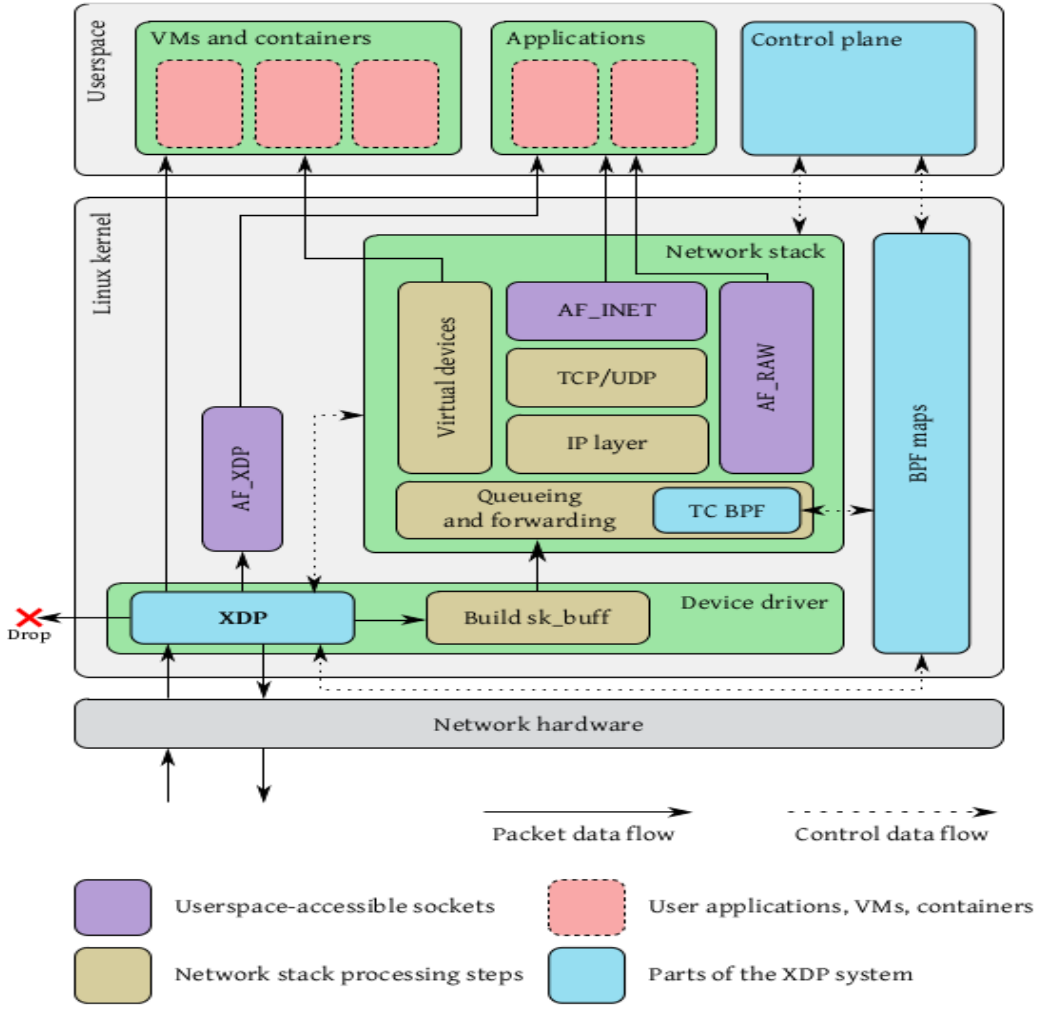


Intel SmartNIC



Mellanox SmartNIC

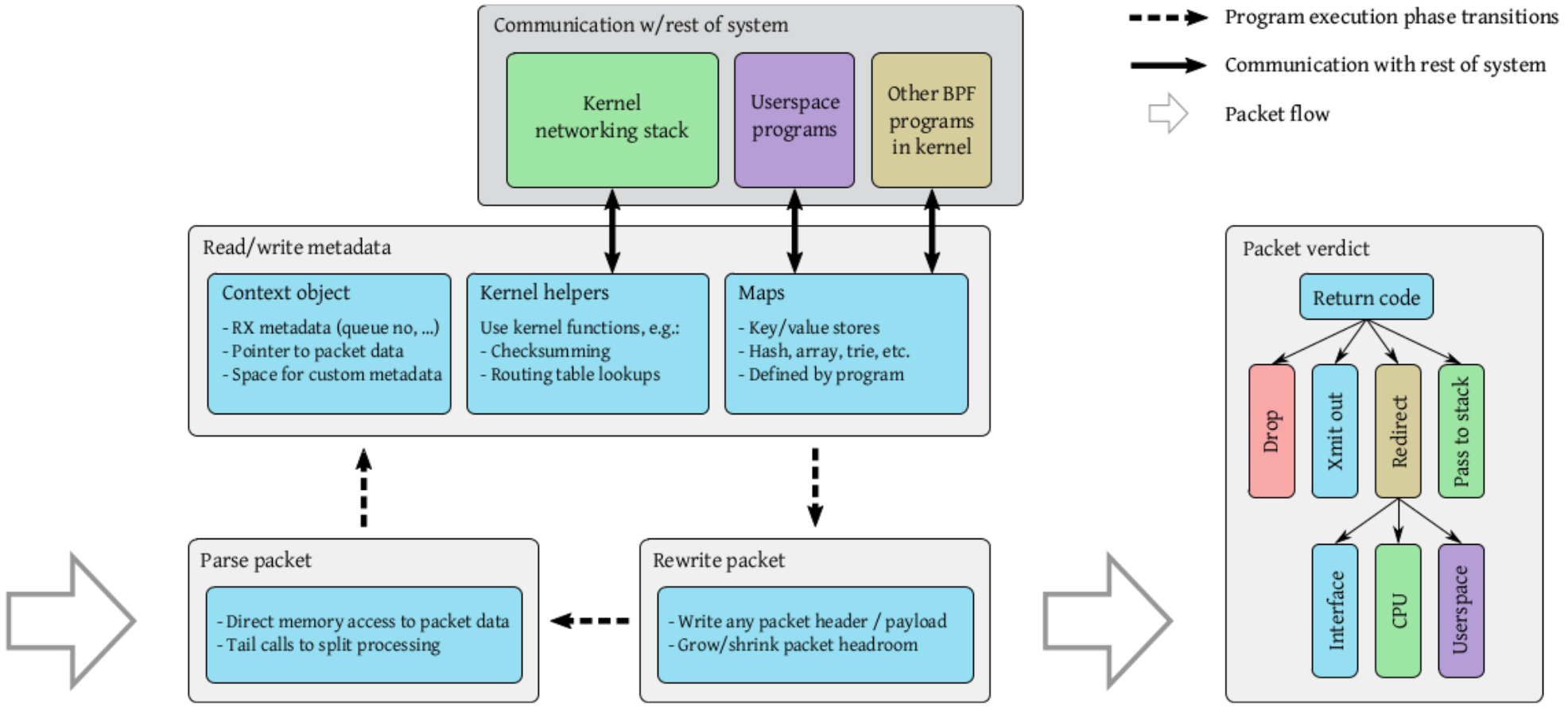
3. XDP Overview(3) – XDP(1)



XDP: eXpress Data Path

✓ *엄청난 속도 향상도 얻으면서 kernel 기능도 사용할 수 있다.*

3. XDP Overview(3) – XDP(2)

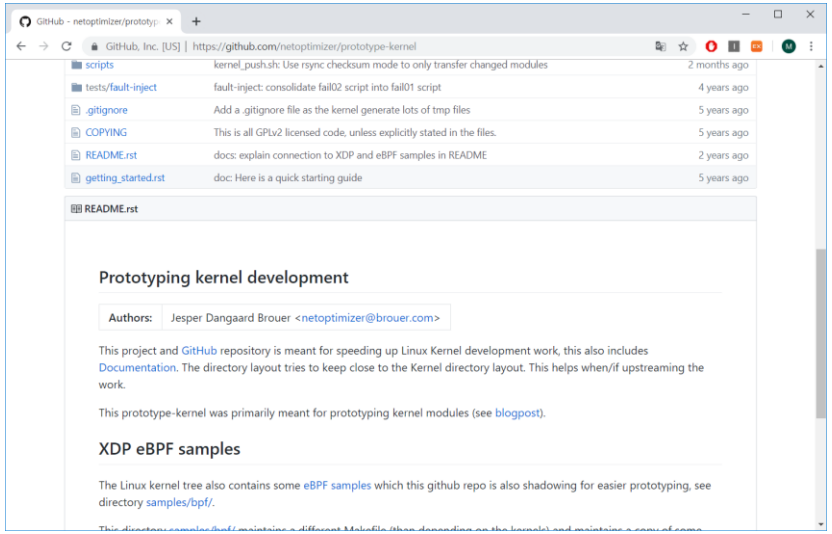


3. XDP Overview(3) – XDP(3)

```c
/* map used to count packets; key is IP protocol, value is pkt count */
struct bpf_map_def SEC("maps") rxcnt = {
    .type = BPF_MAP_TYPE_PERCPU_ARRAY,
    .key_size = sizeof(u32),
    .value_size = sizeof(long),
    .max_entries = 256,
};

/* swaps MAC addresses using direct packet data access */
static void swap_src_dst_mac(void *data)
{
    unsigned short *p = data;
    unsigned short dst[3];
    dst[0] = p[0];  dst[1] = p[1];  dst[2] = p[2];
    p[0]   = p[3];  p[1]   = p[4];  p[2]   = p[5];
    p[3]   = dst[0]; p[4]  = dst[1]; p[5]  = dst[2];
}

static int parse_ipv4(void *data, u64 nh_off, void *data_end)
{
    struct iphdr *iph = data + nh_off;
    if (iph + 1 > data_end)
        return 0;
    return iph->protocol;
}

SEC("xdp1") /* marks main eBPF program entry point */
int xdp_prog1(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_DROP;
    long *value; u16 h_proto; u64 nh_off; u32 ipproto;

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

    h_proto = eth->h_proto;

    /* check VLAN tag; could be repeated to support double-tagged VLAN */
    if (h_proto == htons(ETH_P_8021Q) || h_proto == 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;
        h_proto = vhdr->h_vlan_encapsulated_proto;
    }

    if (h_proto == htons(ETH_P_IP))
        ipproto = parse_ipv4(data, nh_off, data_end);
    else if (h_proto == htons(ETH_P_IPV6))
        ipproto = parse_ipv6(data, nh_off, data_end);
    else
        ipproto = 0;

    /* lookup map element for ip protocol, used for packet counter */
    value = bpf_map_lookup_elem(&rxcnt, &ipproto);
    if (value)
        *value += 1;

    /* swap MAC addrs for UDP packets, transmit out this interface */
    if (ipproto == IPPROTO_UDP) {
        swap_src_dst_mac(data);
        rc = XDP_TX;
    }

    return rc;
}
```

4. XDP Programming(1) – Reference Site

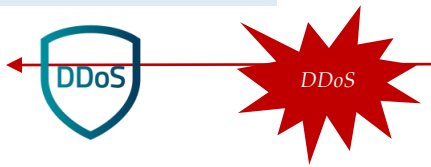
https://github.com/netoptimizer/prototype-kernel



$ sudo ./xdp_ddos01_blacklist –d enp2s0

$ hping3 172.30.1.57 --fast

Security Gateway with XDP



Attacker

4. XDP Programming(2) – DDoS Mitigation(1)

- 1) **Ubuntu 18.10 server 설치**
  - Linux kernel 4.18.x 이상의 환경이 필요함.

  - $ sudo apt install clang llvm libelf-dev
  - $ sudo apt-get install libpcap-dev
  - $ sudo apt install make
  - $ sudo apt install git build-essential

- 2) **prototype-kernel download & build**
  - $ git clone https://github.com/netoptimizer/prototype-kernel
    - ✓ XDP sample code가 있음.
  - $ cd prototype-kernel
  - $ cd kernel; $ make
  - $ cd samples/bpf; $ make
  - $ **sudo ./xdp_ddos01_blacklist –d enp2s0** #enp2s0는 network interface name 임

4. XDP Programming(2) – DDoS Mitigation(2)

- **<Target 장비 – SPNBox>**
- $ **sudo ./xdp_ddos01_blacklist –d enp2s0**

```
spnbox@spnbox-c1037:~/workspace/prototype-kernel/kernel/samples/bpf$ sudo ./xdp_ddos01_blacklist -d
enp2s0
Documentation:
 XDP: DDoS protection via IPv4 blacklist

This program loads the XDP eBPF program into the kernel.
Use the cmdline tool for add/removing source IPs to the blacklist
and read statistics.

 - Attached to device:enp2s0 (ifindex:2)
 - Export bpf-map:blacklist                      to   file:/sys/fs/bpf/ddos_blacklist
 - Export bpf-map:verdict_cnt                    to   file:/sys/fs/bpf/ddos_blacklist_stat_verdict
 - Export bpf-map:port_blacklist                 to   file:/sys/fs/bpf/ddos_port_blacklist
 - Export bpf-map:port_blacklist_drop_count_tcp  to   file:/sys/fs/bpf/ddos_port_blacklist_count_tcp
 - Export bpf-map:port_blacklist_drop_count_udp  to   file:/sys/fs/bpf/ddos_port_blacklist_count_udp
blacklist_modify() IP:198.18.50.3 key:0x33212C6
blacklist_port_modify() dport:80 key:0x50
```

- **<공격용 PC>**
  - ✓ $ sudo apt-get install hping3
  - ✓ $ **hping3 172.30.1.57 --fast**
  - ✓ 이 공격이 먹히지 않아야 함.

이 주소는 172.30.1.57로 수정(xdp_ddos01_blacklist_user.c)하여 아래 테스트를 진행함.

4. XDP Programming(2) – DDoS Mitigation(3)

<Security Gateway에서 xdp ddos blacklist program을 실행한 모습>

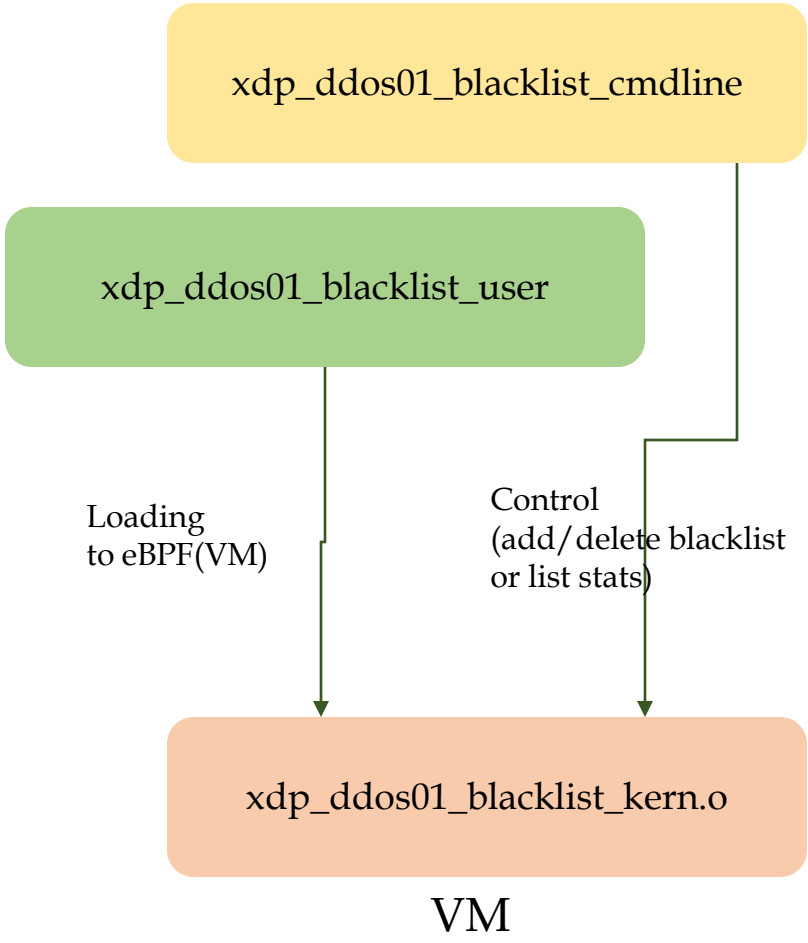
✓blacklist ip 추가/삭제는 xdp_ddos01_blacklist_cmdline 명령을 사용하면 됨.

```
root@spnbox-j1900:~/workspace/prototype-kernel/kernel/samples/bpf# sudo ./xdp_ddos01_blacklist -d enp4s0
Documentation:
 XDP: DDoS protection via IPv4 blacklist

This program loads the XDP eBPF program into the kernel.
Use the cmdline tool for add/removing source IPs to the blacklist
and read statistics.

 - Attached to device:enp4s0 (ifindex:5)
 - Export bpf-map:blacklist                      to   file:/sys/fs/bpf/ddos_blacklist
 - Export bpf-map:verdict_cnt                    to   file:/sys/fs/bpf/ddos_blacklist_stat_verdict
 - Export bpf-map:port_blacklist                 to   file:/sys/fs/bpf/ddos_port_blacklist
 - Export bpf-map:port_blacklist_drop_count_tcp  to   file:/sys/fs/bpf/ddos_port_blacklist_count_tcp
 - Export bpf-map:port_blacklist_drop_count_udp  to   file:/sys/fs/bpf/ddos_port_blacklist_count_udp
blacklist_modify() IP:172.30.1.43 key:0x2B011EAC
blacklist_port_modify() dport:80 key:0x50
root@spnbox-j1900:~/workspace/prototype-kernel/kernel/samples/bpf# sudo ./xdp_ddos01_blacklist_cmdline --add --ip 172.30.1.37
blacklist_modify() IP:172.30.1.37 key:0x25011EAC
root@spnbox-j1900:~/workspace/prototype-kernel/kernel/samples/bpf# sudo ./xdp_ddos01_blacklist_cmdline --add --ip 172.30.1.44
blacklist_modify() IP:172.30.1.44 key:0x2C011EAC
root@spnbox-j1900:~/workspace/prototype-kernel/kernel/samples/bpf# sudo ./xdp_ddos01_blacklist_cmdline --del --ip 172.30.1.37
blacklist_modify() IP:172.30.1.37 key:0x25011EAC
root@spnbox-j1900:~/workspace/prototype-kernel/kernel/samples/bpf# sudo ./xdp_ddos01_blacklist_cmdline --add --ip 172.30.1.37
blacklist_modify() IP:172.30.1.37 key:0x25011EAC
root@spnbox-j1900:~/workspace/prototype-kernel/kernel/samples/bpf#
```

4. XDP Programming(3) – Sample Codes(1)

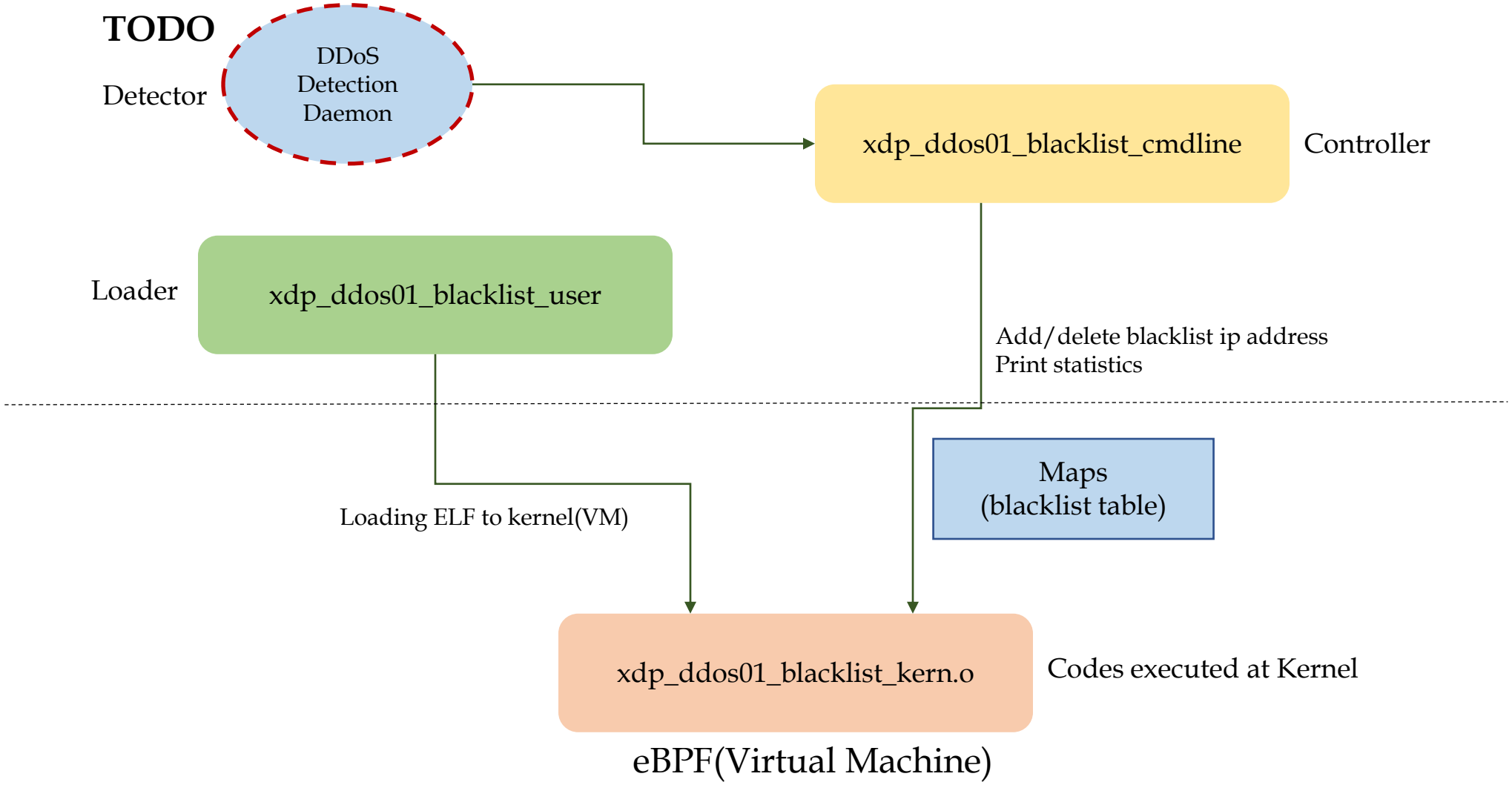
File split similar to samples/bpf in kernel source

- **xdp_ddos01_blacklist_kern.c** - map definitions and restricted-C that is compiled into eBPF
- **xdp_ddos01_blacklist_user.c** - userspace code that loads xdp_ddos01_blacklist_kern.o, sets up sysfs links, and exits
- **xdp_ddos01_blacklist_common.h** - a few definitions and blacklist modify operations
- **xdp_ddos01_blacklist_cmdline.c** - code that is compiled into the userspace program used to configure blacklists



https://people.netfilter.org/hawk/presentations/NetDev2.1_2017/XDP_for_the_Rest_of_Us_Part1.pdf

4. XDP Programming(3) – Sample Codes(2)



4. XDP Programming(4) – Kernel Code(1)

Create per CPU Hash Map

```
01 struct bpf_map_def SEC("maps") blacklist = {
02         .type        = BPF_MAP_TYPE_PERCPU_HASH,
03         .key_size    = sizeof(u32),
04         .value_size  = sizeof(u64), /* Drop counter */
05         .max_entries = 100000,
06         .map_flags   = BPF_F_NO_PREALLOC,
07 };
```

4. XDP Programming(4) – Kernel Code(2)

Create Function that Kernel Will Call

```
01 SEC("xdp_prog")
02 int xdp_program(struct xdp_md *ctx)
03 {
04         void *data_end = (void *)(long)ctx->data_end;
05         void *data     = (void *)(long)ctx->data;
06         struct ethhdr *eth = data;
07         u16 eth_proto, l3_offset = 0;
08         u32 action;
09
10         if (!(parse_eth(eth, data_end, &eth_proto, &l3_offset)))
11                 bpf_debug("Cannot parse L2: L3off:%llu proto ...
12                           l3_offset, eth_proto);
13                 return XDP_PASS;
14         }
15         action = handle_eth_protocol(ctx, eth_proto, l3_offset);
16         stats_action_verdict(action);
17         return action;
18 }
```

4. XDP Programming(4) – Kernel Code(3)

Start Parsing Packets

```
01 static __always_inline
02 bool parse_eth(struct ethhdr *eth, void *data_end,
03                 u16 *eth_proto, u64 *l3_offset)
04 {
05         u16 eth_type;
06         u64 offset;
07
08         offset = sizeof(*eth);
09         if ((void *)eth + offset > data_end)
10                 return false;
11
12         eth_type = eth->h_proto;
13
```

4. XDP Programming(4) – Kernel Code(4)

Continue Parsing Packets

```
01 static __always_inline
02 u32 handle_eth_protocol(struct xdp_md *ctx, u16 eth_proto, ...
03 {
04         switch (eth_proto) {
05         case ETH_P_IP:
06                 return parse_ipv4(ctx, l3_offset);
07                 break;
08         case ETH_P_IPV6: /* Not handler for IPv6 yet*/
09         case ETH_P_ARP:  /* Let OS handle ARP */
10                 break;
11         default:
12                 bpf_debug("Not handling eth_proto:0x%x\n", eth_proto);
13                 return XDP_PASS;
14         }
15         return XDP_PASS;
16 }
```

4. XDP Programming(4) – Kernel Code(5)

Parse IPv4 address and Lookup in Map

```
01 static __always_inline
02 u32 parse_ipv4(struct xdp_md *ctx, u64 l3_offset)
03 {
04         void *data_end = (void *)(long)ctx->data_end;
05         void *data     = (void *)(long)ctx->data;
06         struct iphdr *iph = data + l3_offset;
07         u64 *value;
08         u32 ip_src; /* type need to match map */
09
10         if (iph + 1 > data_end)
11                 return XDP_ABORTED;
12         ip_src = iph->saddr;
13
14         value = bpf_map_lookup_elem(&blacklist, &ip_src);
15         if (value) {
17                 *value += 1; /* Keep a counter for drop matches */
18                 return XDP_DROP;
19         }
```

4. XDP Programming(5) – User Code(1)

The Magic of Interacting with eBPF Maps

- BPF library in tools/lib/bpf of kernel tree is extremely helpful for userspace applications, but feels bit magical until you dig into it:
  - `prog_fd[]` entries are populated with each call to `load_bpf_file()`
  - `map_fd[]` entries align with maps declared in program loaded via `load_bpf_file()` (e.g. `foo_kern.o`)

4. XDP Programming(5) – User Code(2)

The importance of prog_fd[]

- File descriptors for a BPF program are loaded with **`load_bpf_file(foo_kern.o)`**, stored in **`prog_fd[]`**, and attached to netdev with call to **`set_link_xdp_fd()`**

```
01        if (!prog_fd[0]) {
02                printf("load_bpf_file: %s\n", strerror(errno));
03                return 1;
04        }
05
06        if (set_link_xdp_fd(ifindex, prog_fd[0]) < 0) {
07                printf("link set xdp fd failed\n");
08                return EXIT_FAIL_XDP;
09        }
```

4. XDP Programming(5) – User Code(3)

The importance of map_fd[]

- File descriptors for each BPF map are loaded with **`load_bpf_file(foo_kern.o)`**, and stored in **`map_fd[]`**, array. Those fds are used as first argument to operate on the map with library calls (from **`tools/lib/bpf/bpf.h`**)

```
01 int bpf_map_update_elem(int fd, const void *key, const void *value,
                           __u64 flags);
02
03 int bpf_map_lookup_elem(int fd, const void *key, void *value);
04 int bpf_map_delete_elem(int fd, const void *key);
05 int bpf_map_get_next_key(int fd, const void *key, void *next_key);
06 int bpf_obj_pin(int fd, const char *pathname);
```

4. XDP Programming(5) – User Code(4)

Simple coding for an interactive program (`xdp1_user.c`)

```
01        if (load_bpf_file(filename)) {
02                printf("%s", bpf_log_buf);
03                return 1;
04        }
05
06        if (!prog_fd[0]) {
07                printf("load_bpf_file: %s\n", strerror(errno));
08                return 1;
09        }
10
11        signal(SIGINT, int_exit);
12
13        if (set_link_xdp_fd(ifindex, prog_fd[0]) < 0) {
14                printf("link set xdp fd failed\n");
15                return 1;
16        }
17
18        poll_stats(2);
```

4. XDP Programming(6) – Cmdline Code(1)

Command Line Tool Operations

- Print blacklist entries and historical stats
- Add/Delete IPv4 address from blacklist
- Add/Delete UDP and TCPs port from blacklist
- Print real-time XDP verdict stats

4. XDP Programming(6) – Cmdline Code(2)

Read Blacklist Entries

```
01              fd_blacklist = open_bpf_map(file_blacklist);
02              blacklist_list_all_ipv4(fd_blacklist);
03              close(fd_blacklist);
```

Get file descriptor for map file

```
01 int open_bpf_map(const char *file)
02 {
03         int fd;
04
05         fd = bpf_obj_get(file);
06         if (fd < 0) {
07                 printf("ERR: Failed to open bpf map file:%s err(%d):%s\n",
08                        file, errno, strerror(errno));
09                 exit(EXIT_FAIL_MAP_FILE);
10         }
11         return fd;
12 }
```

4. XDP Programming(6) – Cmdline Code(3)

Find Each Key in the Hash

```
01 static void blacklist_list_all_ipv4(int fd)
02 {
03         __u32 key = 0, next_key;
04         __u64 value;
05
06         while (bpf_map_get_next_key(fd, &key, &next_key) == 0) {
07                 printf("%s", key ? "," : "" );
08                 key = next_key;
09                 value = get_key32_value64_percpu(fd, key);
10                 blacklist_print_ipv4(key, value);
11         }
12         printf("%s", key ? "," : "");
13 }
```

4. XDP Programming(7) – attach to netdev

Attaching eBPF Program to an Interface

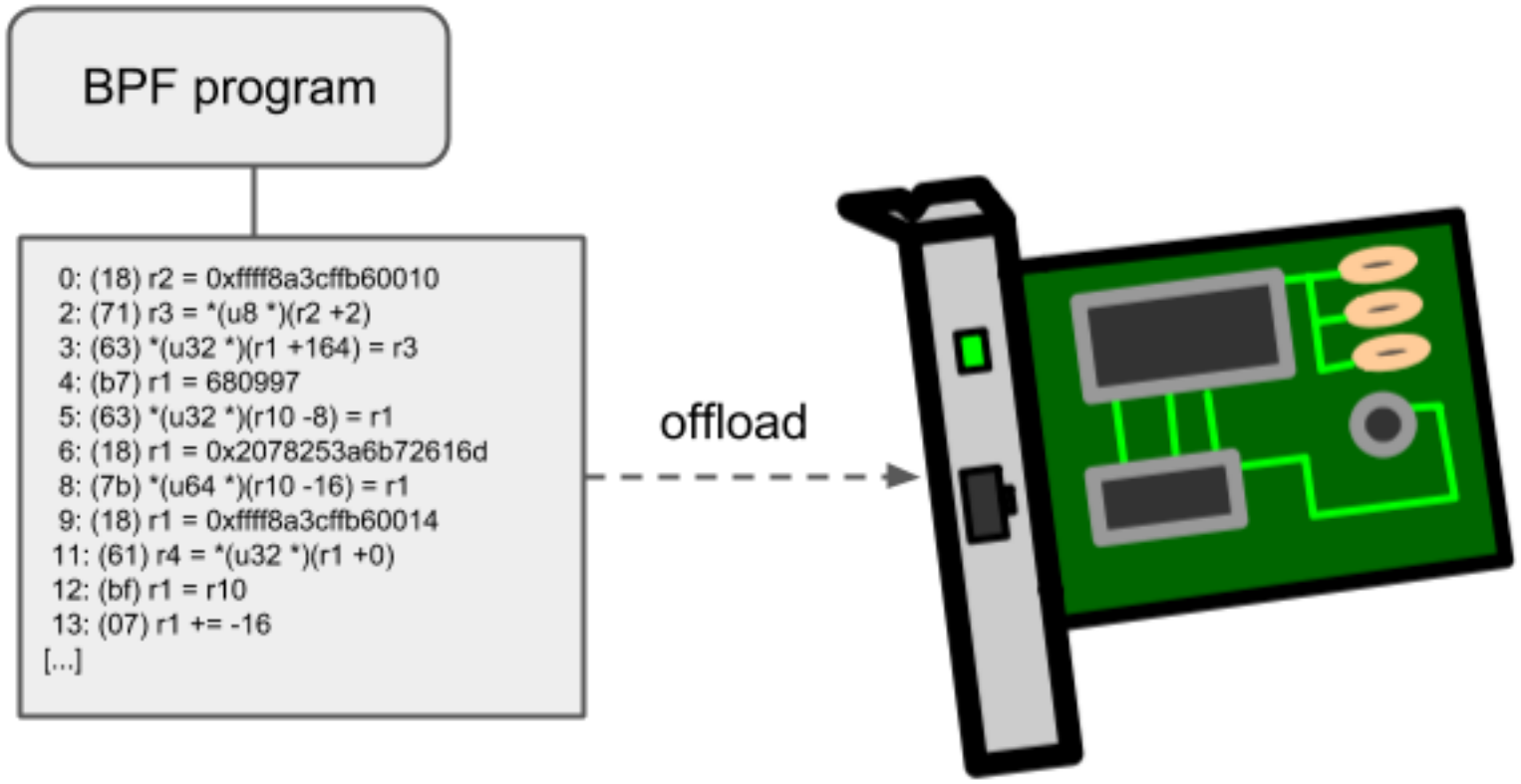
```
01 # ./xdp_ddos01_blacklist --dev enp1s0f1d1
02 Documentation:
03  XDP: DDoS protection via IPv4 blacklist
04
05 This program loads the XDP eBPF program into the kernel.
06 Use the cmdline tool for add/removing source IPs to the blacklist
07 and read statistics.
08
09  - Attached to device:enp1s0f1d1 (ifindex:3)
10  - Blacklist     map file: /sys/fs/bpf/ddos_blacklist
11  - Verdict stats map file: /sys/fs/bpf/ddos_blacklist_stat_verdict
12  - Blacklist Port map file: /sys/fs/bpf/ddos_port_blacklist
13  - Verdict port stats map file: /sys/fs/bpf/ddos_port_blacklist_count_tcp
14  - Verdict port stats map file: /sys/fs/bpf/ddos_port_blacklist_count_udp
15 load_bpf_file: Success
16 # ip link show enp1s0f1d1
17 3: enp1s0f1d1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdp qdisc mq state ...
18     link/ether 00:0a:f7:94:ef:2d brd ff:ff:ff:ff:ff:ff
```

4. XDP Programming(8) – add/dev blacklist

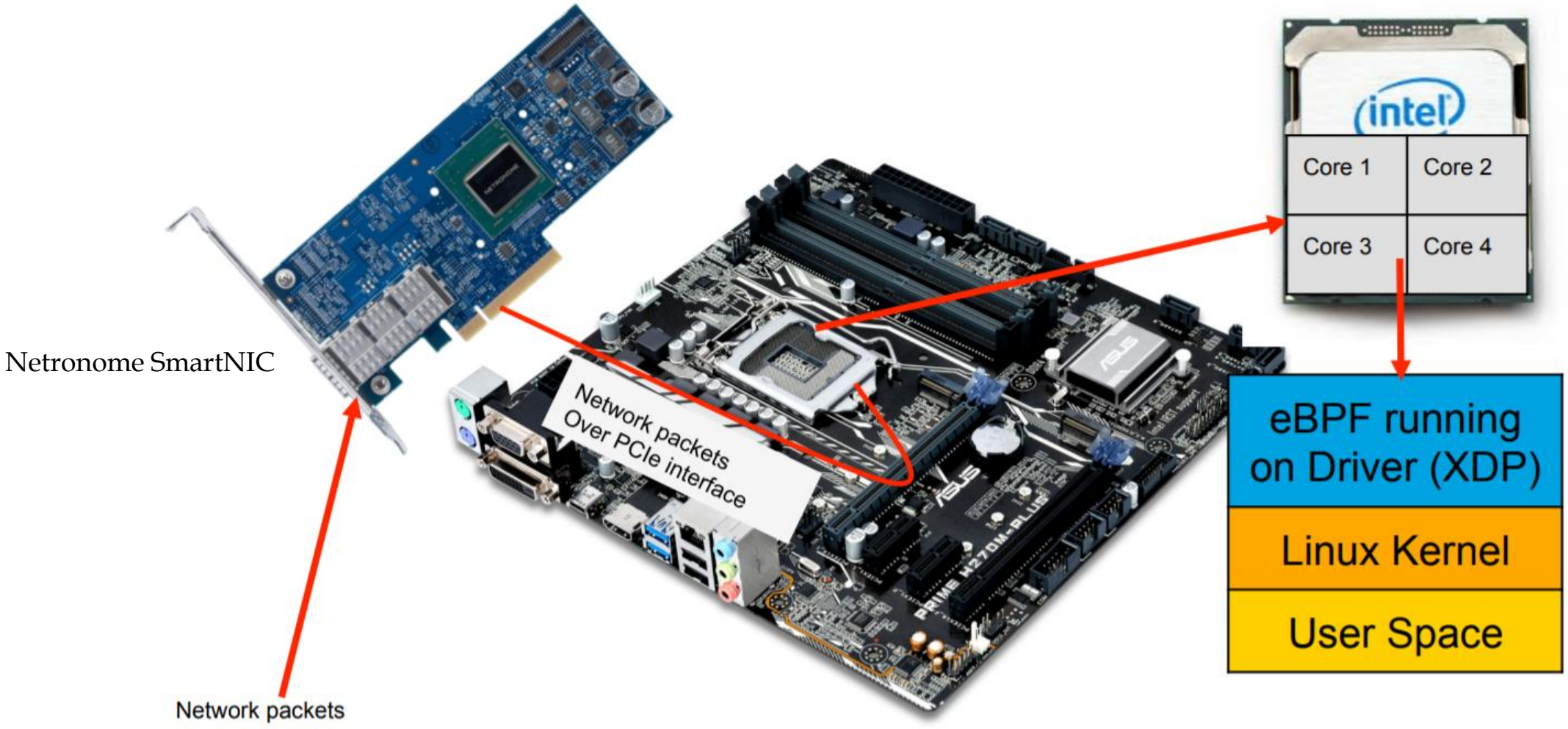
Blacklist configuration

```
01 # ./xdp_ddos01_blacklist_cmdline --add --ip 10.10.10.10
02 blacklist_modify() IP:10.10.10.10 key:0xA0A0A0A
03 # ./xdp_ddos01_blacklist_cmdline --list
04 {
05  "10.10.10.10" : 0,
06 }
07 # ./xdp_ddos01_blacklist_cmdline --del --ip 10.10.10.10
08 blacklist_modify() IP:10.10.10.10 key:0xA0A0A0A
09 # ./xdp_ddos01_blacklist_cmdline --list
10 {
11 }
```

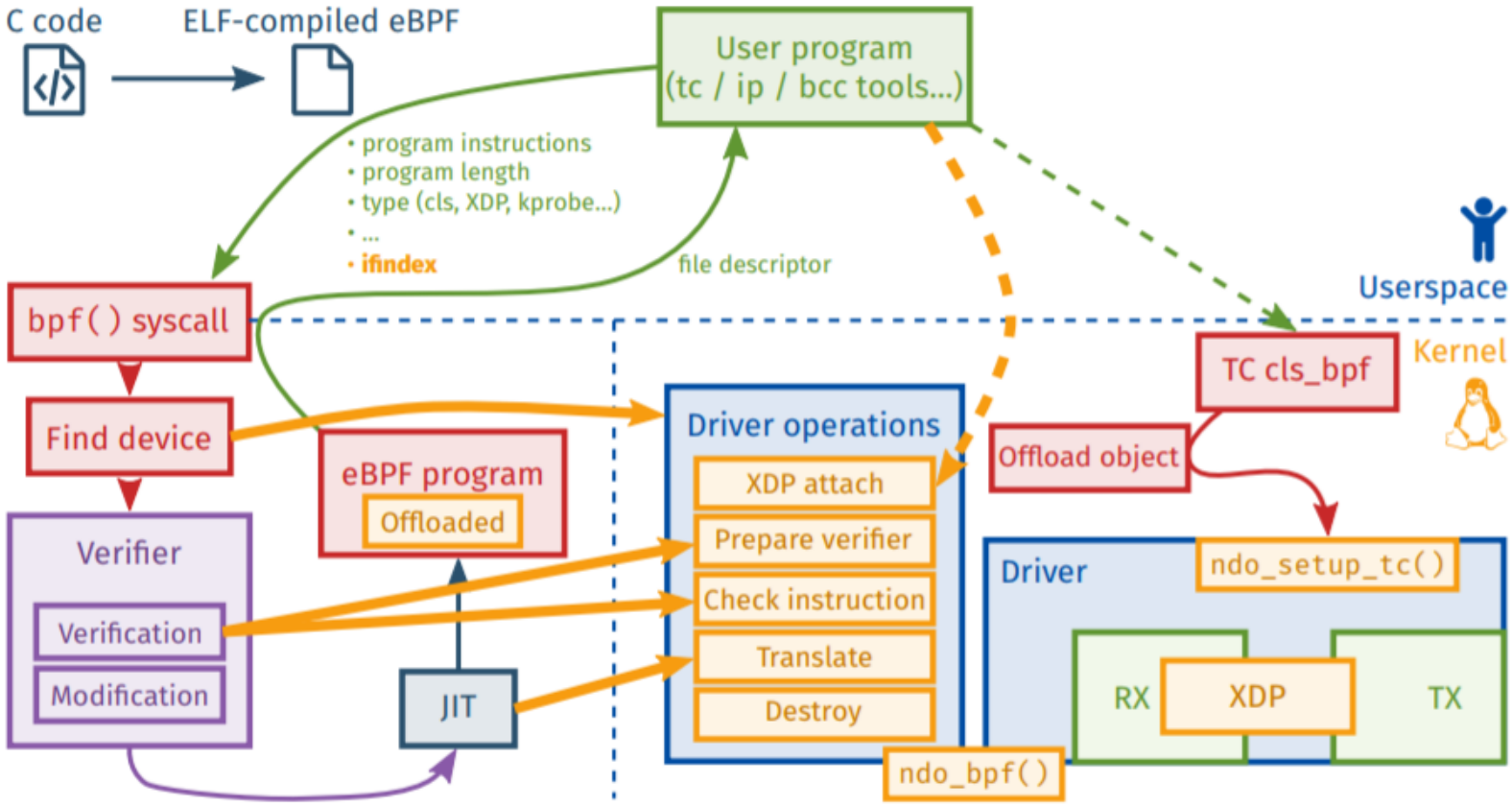
5. XDP H/W Offloading(1)



5. XDP H/W Offloading(2)



5. XDP H/W Offloading(3)



5. XDP H/W Offloading(4)

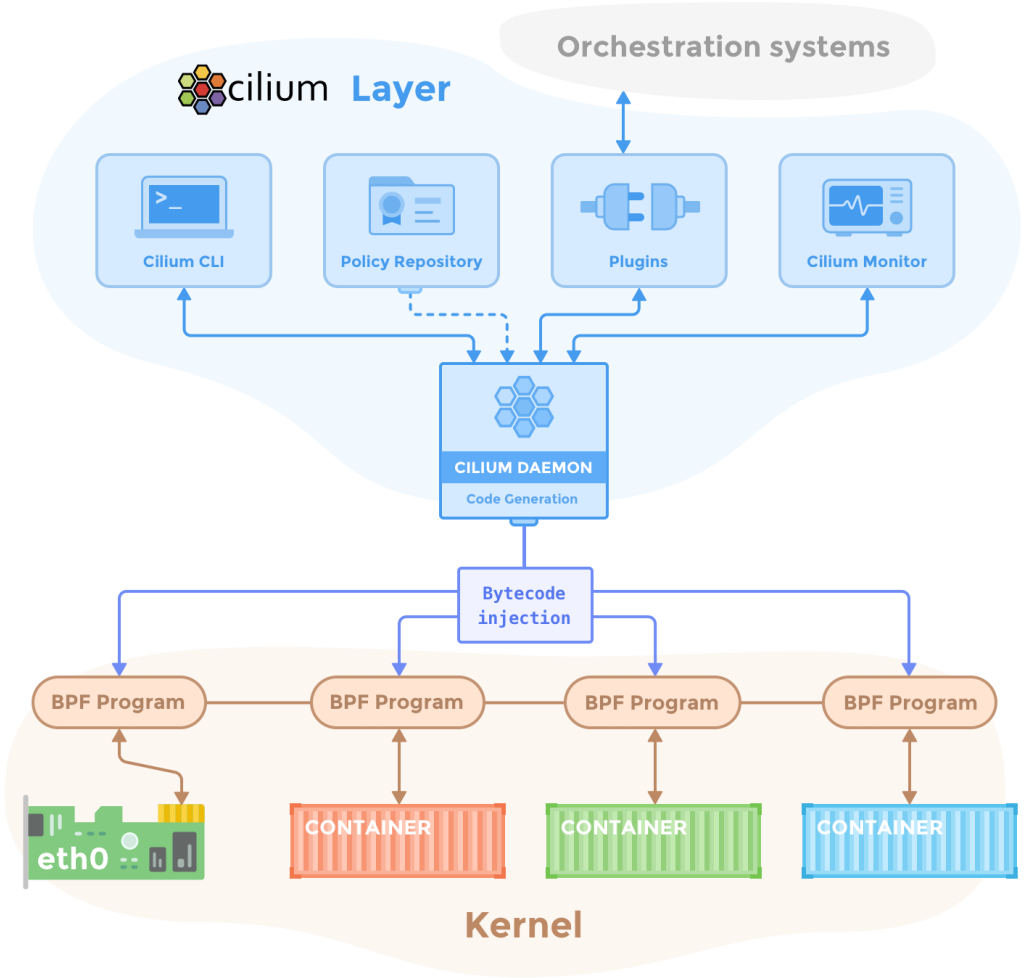
Device drivers with Native XDP support

- Mellanox: mlx4 (v4.10) + mlx5 (v4.9)
- Netronome: nfp (v4.10)
- Virtio-net (v4.10)
- Cavium/Qlogic: qede (v4.10)
- Cavium: thunder/nicvf (v4.12)
- Broadcom: bnxt (v4.12)
- Intel: ixgbe (v4.12) + i40e (net-next)

SmartNIC

https://www.amazon.com/Intel-Converged-Network-Adapter-X540T2/dp/B0077CS9UM

6. Cilium - L7 MicroService Filtering



eBPF & XDP Technology가 앞으로 대세가 될 것이다.

References

- [1] http://media.frnog.org/FRnOG_28/FRnOG_28-3.pdf
- [2] https://people.netfilter.org/hawk/presentations/NetDev2.1_2017/XDP_for_the_Rest_of_Us_Part1.pdf
- [3] https://github.com/xdp-project/xdp-paper/blob/master/xdp-the-express-data-path.pdf
- [4] http://people.netfilter.org/hawk/presentations/OpenSourceDays2017/XDP_DDoS_protecting_osd2017.pdf
- [5] https://www.netronome.com/media/documents/UG_Getting_Started_with_eBPF_Offload.pdf
- [6] https://qmo.fr/docs/talk_20180203_xdp_hw_offload.pdf
- [7] eBPF_HW_OFFLOAD_HNiMne8.pdf
- [8] Lecture-1-Johann-SmartNIC-Programming-Models-Overview.pdf
- [9] https://github.com/OISF/suricata
- [10] https://buildmedia.readthedocs.org/media/pdf/suricata/latest/suricata.pdf
- [11] https://openisf.files.wordpress.com/2015/11/suricata_mixed_mode_g-longo.pdf

We Secure the Internet of Things with vIoTSec !

Thank You