程序创坊

Author: cugriver@163.com

目录

第一章　使用说明

一、系统环境

Ubuntu >= 20.10

- 安装依赖库 elf & zlib

```
apt install -y libelf1 libelf-dev
apt install -y zlib1g  zlib1g-dev
apt install  -y libcap2 libcap-dev
```

- 安装编译器 g++ & g++ & cmake & git

```
apt install -y git cmake g++ gcc pkg-config

gcc --version
gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0

g++ --version
g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0

git --version
git version 2.25.1

cmake --version
cmake version 3.16.3

pkg-config --version
0.29.1
```

- 源码安装 libbpf

```
git clone https://github.com/libbpf/libbpf.git
cd libbpf/src
make
```


```
make install
```

- 二进制安装 libbpf

```
apt install libbpf0 libbpf-dev
```

Debian >= 11

- 安装依赖库 elf & zlib

```
apt install -y libelf1 libelf-dev
apt install -y  zlib1g  zlib1g-dev
apt install  -y libcap libcap-dev
```

- 安装编译器 g++ & g++ & cmake & git

```
apt install -y git cmake g++ gcc pkg-config

gcc --version
gcc (Debian 10.2.1-6) 10.2.1 20210110

g++ --version
g++ (Debian 10.2.1-6) 10.2.1 20210110

git --version
git version 2.30.2

cmake --version
cmake version 3.18.4
```

- 源码安装 libbpf

```
git clone https://github.com/libbpf/libbpf.git
cd libbpf/src
make
make install
```


- 二进制安装 libbpf

```
apt install libbpf0 libbpf-dev
```

Fedora >= 35

- 安装依赖库 elf & zlib

```
dnf install elfutils elfutils-devel
dnf install zlib zlib-devel
dnf install libcap libcap-devel

zlib-1.2.11-30.fc35.x86_64
zlib-devel-1.2.11-30.fc35.x86_64
elfutils-libelf-0.186-1.fc35.x86_64
elfutils-libelf-devel-0.186-1.fc35.x86_64
```

- 安装编译器 g++ & g++ & cmake & git

```
dnf install git cmake gcc gcc-c++ git

gcc --version
gcc (GCC) 11.2.1 20211203 (Red Hat 11.2.1-7)

g++ --version
g++ (GCC) 11.2.1 20211203 (Red Hat 11.2.1-7)
cmake --version
cmake version 3.22.1

git --version
git version 2.33.1
```

- 源码安装 libbpf

```
git clone https://github.com/libbpf/libbpf.git
```


```
cd libbpf/src
make
make install
```

- 二进制安装 libbpf

```
dnf install libbpf libbpf-devel
```

二、必备工具

bpftool 调试和验证

```
uname -r                                          //查看内核版本
5.15.8-200.fc35.x86_64
wget https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/linux-
`uname -r | awk -F'-' '{print $1}'`.tar.gz   //下载内核源代码

tar -zxf linux-5.15.8.tar.gz
cd linux-5.15.8/tools/bpf/bpftool/
make
mv bpftool /usr/bin/

bpftool --version
bpftool v5.15.8
```

hping3 测试和分析

```
dnf install hping3
apt install hping3

随机源 IP 向 10.10.2.4 发送 SYN 包
hping3 -I enp0s3 --rand-source -S 10.10.2.4 -p 22 -i u100 -c 10
指定源 IP 向 10.10.2.4 发送 SYN 包
hping3 -I enp0s3 -a 10.12.2.4 -S 10.10.2.4 -p 80 -I u100 -c 10
```

bpffs 持久化[1]

```
mount | grep bpf
none on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)

如果不存在则创建 bpffs
mount bpffs/sys/fs/bpf -t bpf
```

三、安装程序

启动停止

```
dpkg -i   fw-0.1.0-0-x86_64.deb        //安装
rpm -ivh fw-0.1.0-0-x86_64.deb

systemctl start fw                     //启动或停止
systemctl status fw
systemctl stop fw

dpkg -r fw                             // 卸载
rpm -e --nodeps fw
```

接口指南

- UpdateFirewall

```
POST http://192.168.56.4:3333
{
    "action":"UpdateFirewall",
    "traceId":"86475a74-c5a3-4632-9214-20bca82d433d",
    "version": 10,
    "rules":[
        {
            "protocol":"tcp",
            "srcIp":"119.119.119.119/32",
```


```
            "srcPort":"4000-5000",
            "dstIp":"10.10.2.1/24",
            "dstPort":"22-100",
            "action":"accept"
        },
        {
            "protocol":"icmp",
            "srcIp":"10.10.2.1/24",
            "dstIp":"10.10.2.1/24",
            "action":"accept"
        }
    ]
}
```

- QueryFirewall

```
POST http://192.168.56.4:3333
{
    "action":"QueryFirewall",
    "traceId":"b7be689f-0a1b-481b-bbb1-2cc987d3e809"
}
```

错误码表

错误码	错误信息	错误含义	解决方式
1000	read request body error	读取请求错误	重试
1100	json format error	请求参数格式错误	检查请求发送格式 JSON
1101	not support action error	请求参数错误	检查接口拼写
1200	update firewall error	更新防火墙失败	权限不足 需要 ROOT 用户启动
1201	query firewall error	查询防火墙失败	权限不足 需要 ROOT 用户启动

注意事项

⚠ 防火墙规则最多 1000 条

⚠ 支持 TCP UDP ICMP GRE

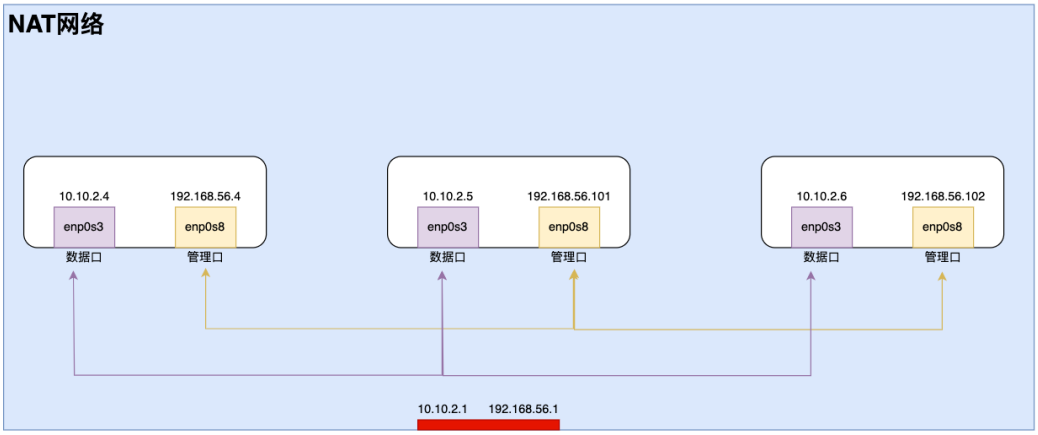
⚠ 不支持分片报文和 IPv6

⚠ 防火墙规则系统重启消失

⚠ bpftool 加载 EBPF 程序，ip 命令绑定到网络设备

四、使用示例

网络拓扑



外网

执行步骤

- 目录树介绍

```
/usr/local/fw/
├── bin
│   └── fw                    服务程序
├── conf
│   └── fw.json               配置文件
├── systemd
│   └── fw.service            SYSTEMD
└── xdp
└── fw.bpf.o                  eBPF 程序
```

- 启动服务程序

```
{
        "logCfg": {
         "level": "info"
    },
    "httpCfg": {
```

```
        "host": "192.168.56.4",
        "port": 3333
    }
}
systemctl start  fw
```

- 初始化防火墙规则(避免将自己拦截)

```
POST http://192.168.56.4:3333
{
    "action":"UpdateFirewall",
    "traceId":"9b8f92fe-356a-4ac7-bb7f-35ff68104987",
    "version":10,
    "rules":[
        {
            "protocol":"tcp",
            "srcIp":"0.0.0.0/0",
            "dstIp":"10.10.2.4/32",
            "action":"accept"
        },
        {
            "protocol":"udp",
            "srcIp":"0.0.0.0/0",
            "dstIp":"10.10.2.4/32",
            "action":"accept"
        },
        {
            "protocol":"icmp",
            "srcIp":"0.0.0.0/0",
            "dstIp":"10.10.2.4/32",
            "action":"accept"
        }
    ]
}
```

- 加载 eBPF 程序

```
unlink /sys/fs/bpf/fw

bpftool prog load /usr/local/fw/xdp/fw.bpf.o /sys/fs/bpf/fw \
```


```
    map name metadata pinned /sys/fs/bpf/metadata \
    map name ipv4_proto pinned /sys/fs/bpf/ipv4_proto \
    map name ipv4_nw_src pinned /sys/fs/bpf/ipv4_nw_src \
    map name ipv4_nw_dst pinned /sys/fs/bpf/ipv4_nw_dst \
    map name ipv4_tp_src pinned /sys/fs/bpf/ipv4_tp_src \
    map name ipv4_tp_dst pinned /sys/fs/bpf/ipv4_tp_dst \
        map name ipv4_action pinned /sys/fs/bpf/ipv4_action

bpftool prog show pinned /sys/fs/bpf/fw

359: xdp  name xpd_handle_fw  tag 5b2d17efc34ca615  gpl
loaded_at 2022-01-02T16:20:17+0800  uid 0
xlated 5336B  jited 2982B  memlock 8192B  map_ids 35,5,10,15,20,25,30
btf_id 235
```

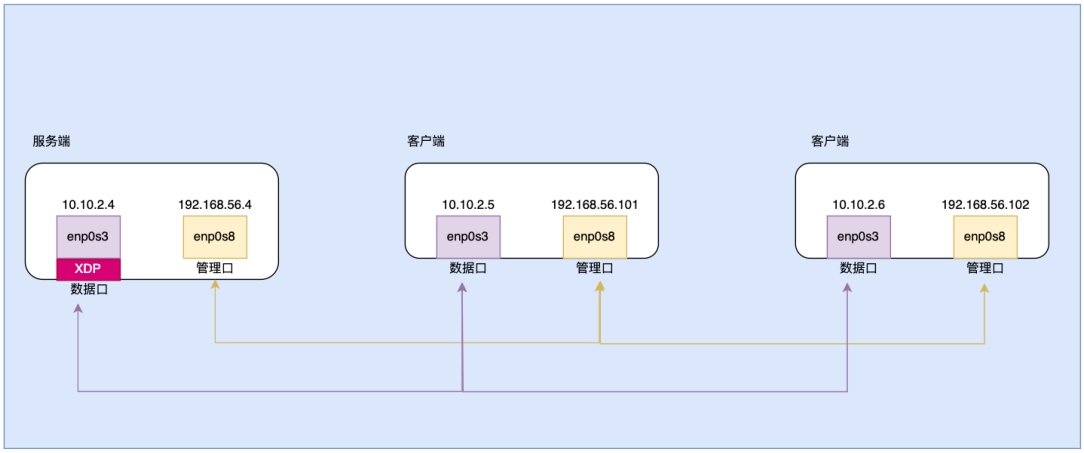
- 网卡设备关联 eBPF

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
qlen 1000
    link/ether 08:00:27:f0:1e:41 brd ff:ff:ff:ff:ff:ff
    inet 10.10.2.4/24 brd 10.10.2.255 scope global dynamic enp0s3
       valid_lft 415sec preferred_lft 415sec
    inet6 fe80::a00:27ff:fef0:1e41/64 scope link
       valid_lft forever preferred_lft forever

ip link set dev enp0s3 xdp pinned /sys/fs/bpf/fw

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdpgeneric/id:359 qdisc pfifo_fast state
 UP group default qlen 1000
    link/ether 08:00:27:f0:1e:41 brd ff:ff:ff:ff:ff:ff
    inet 10.10.2.4/24 brd 10.10.2.255 scope global dynamic enp0s3
       valid_lft 371sec preferred_lft 371sec
    inet6 fe80::a00:27ff:fef0:1e41/64 scope link
       valid_lft forever preferred_lft forever
```

- 测试和验证



ICMP 测试和验证

10.10.2.4 防火墙规则：icmp 0.0.0.0/0 10.10.2.4/32 accept

10.10.2.5 长 PING 10.10.2.4

10.10.2.6 长 PING 10.10.2.4

10.10.2.4 防火墙规则：icmp 10.10.2.6/32 10.10.2.4/32 accept

10.10.2.5 长 PING 10.10.2.4

10.10.2.6 长 PING 10.10.2.4

```
richard@ubuntu:~$ sudo hping3 -I  enp0s3 -1 10.10.2.4 -a 10.10.2.5
HPING 10.10.2.4 (enp0s3 10.10.2.4): icmp mode set, 28 headers + 0 data bytes
len=46 ip=10.10.2.4 ttl=64 id=35649 icmp_seq=0 rtt=6.6 ms
len=46 ip=10.10.2.4 ttl=64 id=35816 icmp_seq=1 rtt=6.1 ms
len=46 ip=10.10.2.4 ttl=64 id=36010 icmp_seq=2 rtt=5.7 ms
len=46 ip=10.10.2.4 ttl=64 id=36104 icmp_seq=3 rtt=5.0 ms
len=46 ip=10.10.2.4 ttl=64 id=36105 icmp_seq=4 rtt=4.2 ms
len=46 ip=10.10.2.4 ttl=64 id=36318 icmp_seq=5 rtt=4.5 ms
len=46 ip=10.10.2.4 ttl=64 id=36328 icmp_seq=6 rtt=3.9 ms
^C
--- 10.10.2.4 hping statistic ---
22 packets transmitted, 7 packets received, 69% packet loss
round-trip min/avg/max = 3.9/5.1/6.6 ms
richard@ubuntu:~$
```

```
[root@10 ~]# hping3 -I  enp0s3 -1 10.10.2.4 -a 10.10.2.6
HPING 10.10.2.4 (enp0s3 10.10.2.4): icmp mode set, 28 headers + 0 data bytes
len=46 ip=10.10.2.4 ttl=64 id=64391 icmp_seq=0 rtt=1.3 ms
len=46 ip=10.10.2.4 ttl=64 id=64522 icmp_seq=1 rtt=2.0 ms
len=46 ip=10.10.2.4 ttl=64 id=64667 icmp_seq=2 rtt=1.6 ms
len=46 ip=10.10.2.4 ttl=64 id=64709 icmp_seq=3 rtt=1.2 ms
len=46 ip=10.10.2.4 ttl=64 id=64951 icmp_seq=4 rtt=1.0 ms
len=46 ip=10.10.2.4 ttl=64 id=65135 icmp_seq=5 rtt=2.1 ms
len=46 ip=10.10.2.4 ttl=64 id=65298 icmp_seq=6 rtt=2.7 ms
len=46 ip=10.10.2.4 ttl=64 id=65495 icmp_seq=7 rtt=2.3 ms
len=46 ip=10.10.2.4 ttl=64 id=119 icmp_seq=8 rtt=2.1 ms
len=46 ip=10.10.2.4 ttl=64 id=272 icmp_seq=9 rtt=2.0 ms
len=46 ip=10.10.2.4 ttl=64 id=405 icmp_seq=10 rtt=1.6 ms
len=46 ip=10.10.2.4 ttl=64 id=571 icmp_seq=11 rtt=2.3 ms
len=46 ip=10.10.2.4 ttl=64 id=790 icmp_seq=12 rtt=2.1 ms
len=46 ip=10.10.2.4 ttl=64 id=917 icmp_seq=13 rtt=1.7 ms
len=46 ip=10.10.2.4 ttl=64 id=931 icmp_seq=14 rtt=1.4 ms
len=46 ip=10.10.2.4 ttl=64 id=1160 icmp_seq=15 rtt=2.3 ms
len=46 ip=10.10.2.4 ttl=64 id=1283 icmp_seq=16 rtt=2.2 ms
len=46 ip=10.10.2.4 ttl=64 id=1394 icmp_seq=17 rtt=1.8 ms
len=46 ip=10.10.2.4 ttl=64 id=1501 icmp_seq=18 rtt=1.5 ms
len=46 ip=10.10.2.4 ttl=64 id=1744 icmp_seq=19 rtt=3.1 ms
^C
--- 10.10.2.4 hping statistic ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max = 1.0/1.9/3.1 ms
[root@10 ~]#
```

TCP 连接测试

10.10.2.4 防火墙规则：tcp 0.0.0.0/0 10.10.2.4/32 accept

10.10.2.5：nc -nvv 10.10.2.4 3389

10.10.2.6：nc -nvv 10.10.2.4 3389

```
root@debian:~# tcpdump -i enp0s3 -ennn -tttt -c 10 port 3389
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes


2022-01-02 23:30:25.544786 08:00:27:99:cf:ce > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 74: 10.10.2.6.57356 > 10.10.2.4.
3389: Flags [S], seq 468089055, win 64240, options [mss 1460,sackOK,TS val 1642884064 ecr 0,nop,wscale 7], length 0
2022-01-02 23:30:25.544832 08:00:27:f0:1e:41 > 08:00:27:99:cf:ce, ethertype IPv4 (0x0800), length 74: 10.10.2.4.3389 > 10.10.2.6.5
7356: Flags [S.], seq 2471574168, ack 468089056, win 65160, options [mss 1460,sackOK,TS val 1782302129 ecr 1642884064,nop,wscale 7
], length 0
2022-01-02 23:30:25.545718 08:00:27:99:cf:ce > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 66: 10.10.2.6.57356 > 10.10.2.4.
3389: Flags [.], ack 1, win 502, options [nop,nop,TS val 1642884065 ecr 1782302129], length 0
2022-01-02 23:30:27.347524 08:00:27:99:cf:ce > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 66: 10.10.2.6.57356 > 10.10.2.4.
3389: Flags [F.], seq 1, ack 1, win 502, options [nop,nop,TS val 1642885867 ecr 1782302129], length 0
2022-01-02 23:30:27.347594 08:00:27:f0:1e:41 > 08:00:27:99:cf:ce, ethertype IPv4 (0x0800), length 67: 10.10.2.4.3389 > 10.10.2.6.5
7356: Flags [P.], seq 1:2, ack 2, win 510, options [nop,nop,TS val 1782303932 ecr 1642885867], length 1
2022-01-02 23:30:27.347631 08:00:27:f0:1e:41 > 08:00:27:99:cf:ce, ethertype IPv4 (0x0800), length 66: 10.10.2.4.3389 > 10.10.2.6.5
7356: Flags [F.], seq 2, ack 2, win 510, options [nop,nop,TS val 1782303932 ecr 1642885867], length 0
2022-01-02 23:30:27.347991 08:00:27:99:cf:ce > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 60: 10.10.2.6.57356 > 10.10.2.4.
3389: Flags [R], seq 468089057, win 0, length 0
2022-01-02 23:30:27.347996 08:00:27:99:cf:ce > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 60: 10.10.2.6.57356 > 10.10.2.4.
3389: Flags [R], seq 468089057, win 0, length 0
```

```
[root@10 ~]# nc -nvv 10.10.2.4 3389
Ncat: Version 7.91 ( https://nmap.org/ncat )
NCAT DEBUG: Using system default trusted CA certificates and those in /usr/share/ncat/ca-bundle.crt.
NCAT DEBUG: Unable to load trusted CA certificates from /usr/share/ncat/ca-bundle.crt: error:02001002:system library:fopen:No such
 file or directory
libnsock nsock_iod_new2(): nsock_iod_new (IOD #1)
libnsock nsock_connect_tcp(): TCP connection requested to 10.10.2.4:3389 (IOD #1) EID 8
libnsock nsock_trace_handler_callback(): Callback: CONNECT SUCCESS for EID 8 [10.10.2.4:3389]
Ncat: Connected to 10.10.2.4:3389.
libnsock nsock_iod_new2(): nsock_iod_new (IOD #2)
libnsock nsock_read(): Read request from IOD #1 [10.10.2.4:3389] (timeout: -1ms) EID 18
libnsock nsock_readbytes(): Read request for 0 bytes from IOD #2 [peer unspecified] EID 26
^C
[root@10 ~]#
```


10.10.2.4 防火墙规则：tcp 10.10.2.5/32 10.10.2.4/32 accept

10.10.2.5：nc -nvv 10.10.2.4 3389

10.10.2.6：nc -nvv 10.10.2.4 3389

```
[root@10 ~]# nc -nvv 10.10.2.4 3389
Ncat: Version 7.91 ( https://nmap.org/ncat )
NCAT DEBUG: Using system default trusted CA certificates and those in /usr/share/ncat/ca-bundle.crt.
NCAT DEBUG: Unable to load trusted CA certificates from /usr/share/ncat/ca-bundle.crt: error:02001002:system library:fopen:No suc
 file or directory
libnsock nsock_iod_new2(): nsock_iod_new (IOD #1)
libnsock nsock_connect_tcp(): TCP connection requested to 10.10.2.4:3389 (IOD #1) EID 8
libnsock nsock_trace_handler_callback(): Callback: CONNECT TIMEOUT for EID 8 [10.10.2.4:3389]
Ncat: TIMEOUT.
```

```
richard@ubuntu:~$ nc -nvv 10.10.2.4 3389
Connection to 10.10.2.4 3389 port [tcp/*] succeeded!
^C
```

```
root@debian:~# tcpdump -i enp0s3 -ennn -tttt -c 10 port 3389
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes


2022-01-02 23:33:49.394851 08:00:27:75:8b:ff > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 74: 10.10.2.5.47126 > 10.10.2.4.
3389: Flags [S], seq 1127576475, win 64240, options [mss 1460,sackOK,TS val 2643745914 ecr 0,nop,wscale 7], length 0
2022-01-02 23:33:49.395403 08:00:27:f0:1e:41 > 08:00:27:75:8b:ff, ethertype IPv4 (0x0800), length 74: 10.10.2.4.3389 > 10.10.2.5.4
7126: Flags [S.], seq 3251294427, ack 1127576476, win 65160, options [mss 1460,sackOK,TS val 1933546366 ecr 2643745914,nop,wscale
7], length 0
2022-01-02 23:33:49.395831 08:00:27:75:8b:ff > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 66: 10.10.2.5.47126 > 10.10.2.4.
3389: Flags [.], ack 1, win 502, options [nop,nop,TS val 2643745915 ecr 1933546366], length 0
2022-01-02 23:33:58.972441 08:00:27:75:8b:ff > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 66: 10.10.2.5.47126 > 10.10.2.4.
3389: Flags [F.], seq 1, ack 1, win 502, options [nop,nop,TS val 2643755492 ecr 1933546366], length 0
2022-01-02 23:33:58.972523 08:00:27:f0:1e:41 > 08:00:27:75:8b:ff, ethertype IPv4 (0x0800), length 67: 10.10.2.4.3389 > 10.10.2.5.4
7126: Flags [P.], seq 1:2, ack 2, win 510, options [nop,nop,TS val 1933555944 ecr 2643755492], length 1
2022-01-02 23:33:58.972561 08:00:27:f0:1e:41 > 08:00:27:75:8b:ff, ethertype IPv4 (0x0800), length 66: 10.10.2.4.3389 > 10.10.2.5.4
7126: Flags [F.], seq 2, ack 2, win 510, options [nop,nop,TS val 1933555944 ecr 2643755492], length 0
2022-01-02 23:33:58.972853 08:00:27:75:8b:ff > 08:00:27:f0:1e:41, ethertype IPv4 (0x0800), length 60: 10.10.2.5.47126 > 10.10.2.4.
3389: Flags [R], seq 1127576477, win 0, length 0
```

五、性能分析

VirtualBox Debian 1CPU 2GMEM

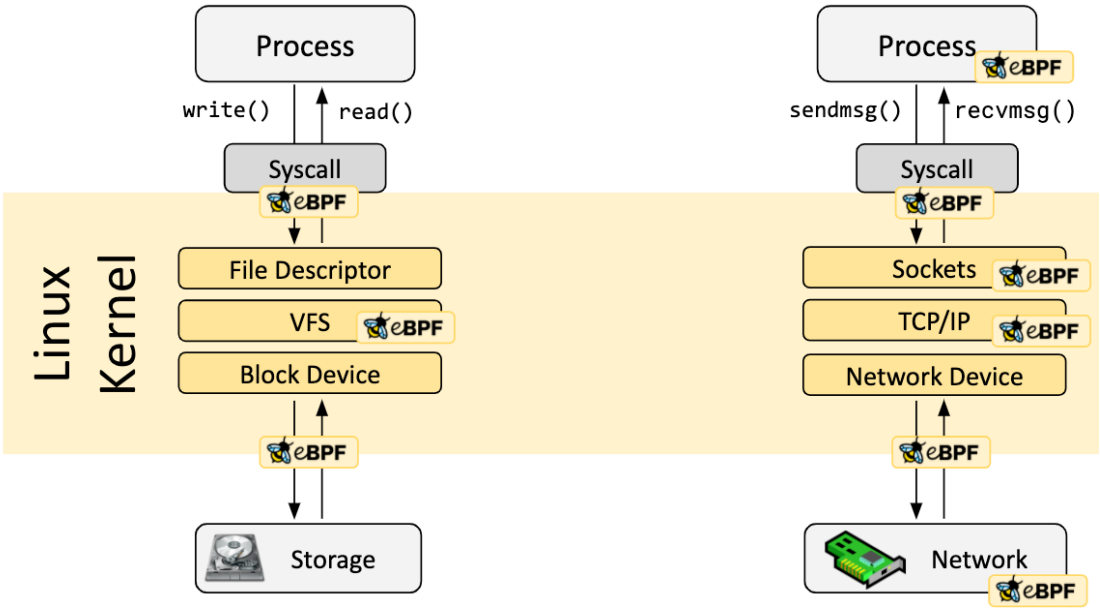
规则 接口秒	100	200	300	400	500	600	700	800	900	1000
更新规则	1.67	2.46	3.40	3.59	4.45	5.38	3.89	7.23	7.44	7.97
查询规则	1.13	1.46	1.23	1.39	2.38	1.56	3.64	4.05	5.21	5.80

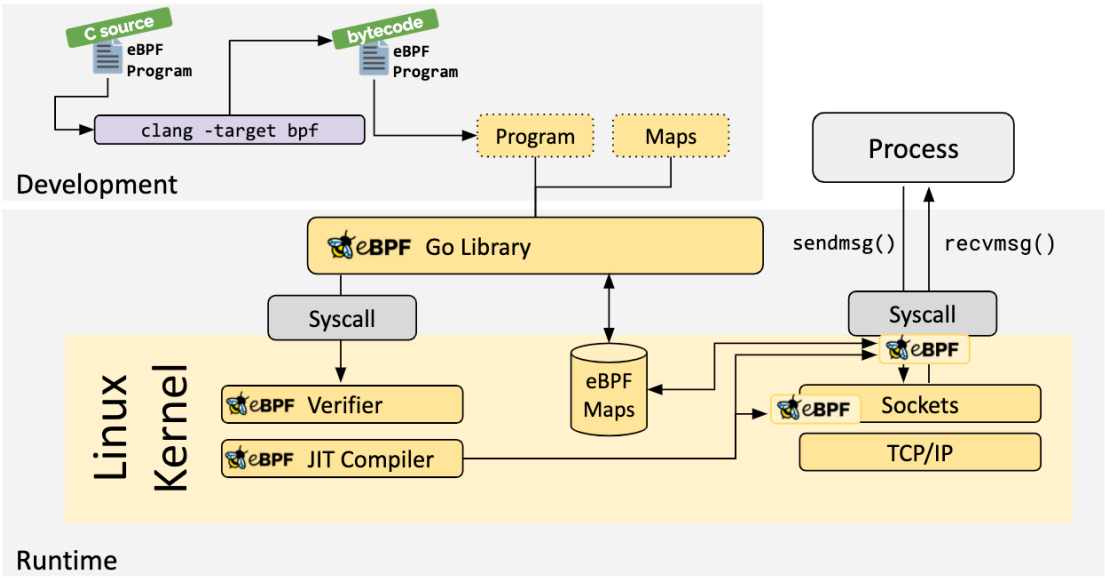
第二章 基础知识

eBPF

eBPF 一种快速，稳定和安全运行在内核的革命性技术。在 Linux 3.15 ，2014 年[2]。有效扩展内核功能而无需更改内核代码或加载内核模块。在高性能网络、监控和统计和安全防护方面有用例使用。eBPF 有以下重要几点内容：

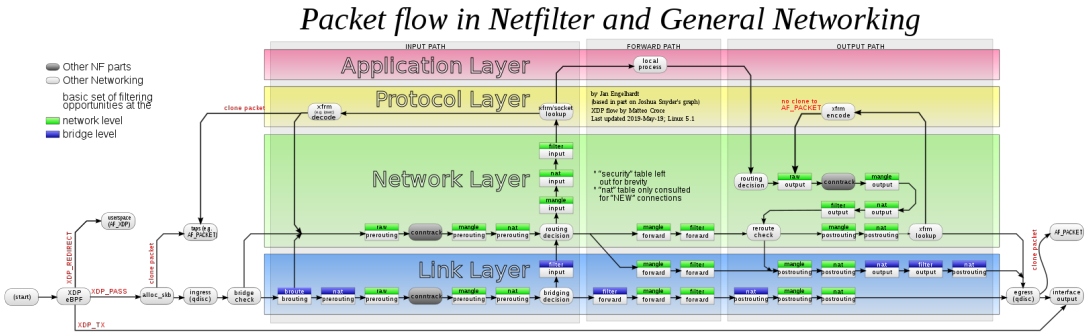
- 10 64-bit registers
- helpers function
- eBPF map
- eBPF verifier





XDP

XDP（eXpress Data Path）是 Linux 内核提供的高性能、可编程的网络数据路径 HOOK。



BPFTOOL

bpftool 是一个可以检查和操作 eBPF 的 prog 和 map 对象的工具集，基于 libbpf 库。它可以查询、创建、修改 map 对象的 key 和 value，还可以对 prog 申明定义的 map 对象进行复用。

```
Usage: bpftool [OPTIONS] OBJECT { COMMAND | help }
       bpftool batch file FILE
       bpftool version
```

```
       OBJECT := { prog | map | link | cgroup | perf | net | feature |
btf | gen | struct_ops | iter }
       OPTIONS := { {-j|--json} [{-p|--pretty}] | {-f|--bpffs} |
             {-m|--mapcompat} | {-n|--nomount} }
```

第三部分 设计与实现

基于 eBPF/XDP 实现防火墙功能，支持四层网络，可以接受或者丢弃 IP+Port，或 IP/Mask+Port/Mask 的组合。DDoS 以大量无效报文来浪费 CPU 资源来达到攻击目的。所以在数据路径上，越早处理越安全。该服务在 XDP 位置挂载 eBPF 程序，在执行效率上明显高于 IpTables。

在程序设计方面，使用 eBPF LPM Map 在 eBPF 程序和用户态程序共享储存。用户态程序将防火墙规则构造成 bitmap 数据结构并存储到 map 中，eBPF 程序在数据路径上读取 map 数据并进行数据包过滤，将满足条件的数据报文放行，否则丢包拒绝。防火墙规则如下：

协议	源 IP	源端口	目的 IP	目的端口	动作

协议：	TCP UDP ICMP GRE
源 IP：	101.101.102.0/24 10.10.10.10/32
源端口：	1-65535 22-444
目的 IP：	101.101.102.0/24 10.10.10.10/32
目的端口：	1-65535 22-444
动作	accept drop

Bitmap 数据结构的引入目的提高查询性能，缩短查询时间。构造 bitmap 需要 O(n)时间，查询需要 O(Logn)时间。

	协议	源 IP	源端口	目 IP	目端口	动作
#1	tcp	10.10.2.5	1-65535	10.10.2.4	22	accept
#2	tcp	10.10.2.5	1-65535	10.10.2.4	80	accept
#3	udp	10.10.2.6	4000	10.10.2.4	3389	accept

协议表	
tcp:	110
udp:	001

源 IP 表	
10.10.2.5:	110
10.10.2.6:	001

源端口表	
1-65535:	110
4000:	001

目的端口	
22:	100
80:	010
3389:	001

目的 IP 表	
10.10.2.4:	111

动作	
accept:	111

- udp 10.10.2.6 4000 10.10.2.4 3389

b = 001 & 001 & 001 & 001 & 111 & 111 = 001 ✅

- udp 10.10.2.6 4001 10.10.2.4 3389

b = 001 & 001 & 000 & 111 & 001 & 111 = 000 ❌

- tcp 10.10.2.5 4001 10.10.2.4 3 81

b = 110 & 110 & 110 & 000 & 111 & 111 = 000 ❌

端口掩码生成规则和 IP 掩码使用相似，IP 是 32bit，Port 是 16bit。

命令说明

```
ip -c -p addr      显示当前网络设备，ip 命令非常常用，可以 man ip 学习细节

unlink /sys/fs/bpf/fw    取消 ebpf fd 持久化

bpftool prog load /usr/local/fw/xdp/fw.bpf.o /sys/fs/bpf/fw \
    map name metadata pinned /sys/fs/bpf/metadata \
    map name ipv4_proto pinned /sys/fs/bpf/ipv4_proto \
    map name ipv4_nw_src pinned /sys/fs/bpf/ipv4_nw_src \
    map name ipv4_nw_dst pinned /sys/fs/bpf/ipv4_nw_dst \
    map name ipv4_tp_src pinned /sys/fs/bpf/ipv4_tp_src \
    map name ipv4_tp_dst pinned /sys/fs/bpf/ipv4_tp_dst \
    map name ipv4_action pinned /sys/fs/bpf/ipv4_action

通过 bpftool 命令加载 ebpf 程序注入内核;  map name 参数目的将已有的 map 和 ebpf 程序使用的 map 进行关联;

bpftool prog show pinned /sys/fs/bpf/fw  显示挂载到 bpffs 的 ebpf 程序

ip link set dev enp0s3 xdp pinned /sys/fs/bpf/fw  通过 ip link 命令将 ebpf 程序和网络设备关联，使流过网络设备的流量受程序控制

tcpdump -i  dev -ennn -tttt -c 10 port 22  网络抓包 man tcpdump 学习细节
```

附录

[1]https://facebookmicrosites.github.io/bpf/blog/2018/08/31/object-lifetime.html

[2]https://ebpf.io/what-is-ebpf