On this page

EdgeOS

Tips

- EdgeOS User Guide
- 参考
  - Policy-Based Routing
    - 策略路由
  - WAN Load-Balancing
  - 不同网段多出口路由案列
  - IP 策略路由配置
  - Dual wan, port forwarding
    - 使用 DNAT
- switch 0
  - 类似于交换机，ER-PoE-5 有 switch 芯片
    - 创建失败 interface switch switch 0: does not exist
  - 类似于 Linux Bridge
  - 不支持场景可直接接一个 switch
- 注意
  - 默认不允许 ICMP - 公网如果开启 ICMP 建议限制只允许 echo request
  - load-balance 设置后会对 interface 打标进入不同路由表
    - 多 wan 时非常有用 - 就不用自行维护
    - 即便是不用负载能力
    - iptable mark -> ip rule -> ip route table 201

```
# 监听 nat
show nat translations

show ip route
show ip route table 102

show firewall modify PBR statistics
```

配置

ssh key

- `system/login/user/<USER>/authentication/public-keys`
  - user@example.com
    - key=KEY 内容
    - type=ssh-rsa

```
# 上传到路由
scp ~/.ssh/id_rsa.pub ubnt@192.168.1.1:/tmp

# 通过 loadkey 配置
ssh ubnt@192.168.1.1
configure
loadkey ubnt /tmp/id_rsa.pub
commit
save
exit
```

启用 snmp

```
set service snmp community public network 192.168.0.0/16
set service snmp listen-address 192.168.1.1 interface eth0
```

端口转发

- 端口转发默认会添加防火墙规则
- 简化配置 DNAT

```
# 端口转发
# ==========
# 8001 -> 192.168.1.2:80
# DNAT
-A UBNT_PFOR_DNAT_RULES -p tcp -m tcp --dport 8001 -j DNAT --to-destination
192.168.1.2:80
# SNAT
```

```
-A UBNT_PFOR_SNAT_RULES -d 192.168.1.2/32 -o eth2 -p tcp -m set --match-set
NETV4_eth2 src -m tcp --dport 80 -j MASQUERADE
# 防火墙
-A UBNT_PFOR_FW_RULES -d 192.168.1.2/32 -p tcp -m tcp --dport 80 -j ACCEPT

# 防火墙
# ==========
# 端口转发防火墙规则
-A UBNT_PFOR_FW_HOOK -i pppoe0 -j UBNT_PFOR_FW_RULES
# 允许端口转发的目标通过
-A UBNT_PFOR_FW_RULES -d 192.168.1.2/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

PBR

- DNAT - connections not sticky ?

ipatbles

```
# NAT 表处理逻辑
*nat
-A PREROUTING -j MINIUPNPD
# 端口转发
-A PREROUTING -j UBNT_PFOR_DNAT_HOOK
-A PREROUTING -j VYATTA_PRE_DNAT_HOOK
-A PREROUTING -j UBNT_SUSPEND_DNAT_HOOK
# 自定义 DNAT
-A PREROUTING -j VYATTA_DNAT
-A POSTROUTING -j UBNT_VPN_IPSEC_SNAT_HOOK
-A POSTROUTING -j MINIUPNPD-POSTROUTING
-A POSTROUTING -j UBNT_PFOR_SNAT_HOOK
-A POSTROUTING -j VYATTA_PRE_SNAT_HOOK
# 自定义 SNAT
-A POSTROUTING -j VYATTA_SNAT

# 包处理
*mangle
-A PREROUTING -j MINIUPNPD
-A PREROUTING -j VYATTA_FW_IN_HOOK
-A PREROUTING -j UBNT_FW_MSS_CLAMP_I
# 负载均衡
-A INPUT -j UBNT_HOOK_WLBL
-A FORWARD -j UBNT_QOS_FW_IN_HOOK
-A FORWARD -j UBNT_HOOK_WLBI
-A OUTPUT -j UBNT_HOOK_WLBO
-A POSTROUTING -j VYATTA_FW_OUT_HOOK
```

```
-A POSTROUTING -j UBNT_FW_MSS_CLAMP
-A POSTROUTING -j UBNT_QOS_FW_OUT_HOOK

# RAW
*raw
-A PREROUTING -j UBNT_PREROUTING_HOOK
-A PREROUTING -j VYATTA_CT_IGNORE
-A PREROUTING -j UBNT_CT_BRIDGE
-A PREROUTING -j VYATTA_CT_PREROUTING_HOOK
-A PREROUTING -j UBNT_WLB
-A PREROUTING -j NAT_CONNTRACK
-A PREROUTING -j PFOR_CONNTRACK
-A PREROUTING -j FW_CONNTRACK
-A PREROUTING -j QOS_CONNTRACK
-A PREROUTING -j NOTRACK
-A OUTPUT -j VYATTA_CT_IGNORE
-A OUTPUT -j VYATTA_CT_OUTPUT_HOOK
-A OUTPUT -j UBNT_WLB
-A OUTPUT -j NAT_CONNTRACK
-A OUTPUT -j PFOR_CONNTRACK
-A OUTPUT -j FW_CONNTRACK
-A OUTPUT -j QOS_CONNTRACK
-A OUTPUT -j NOTRACK
-A FW_CONNTRACK -j ACCEPT
-A NAT_CONNTRACK -j ACCEPT
-A PFOR_CONNTRACK -j ACCEPT
-A QOS_CONNTRACK -j RETURN
-A UBNT_WLB -j ACCEPT
-A VYATTA_CT_IGNORE -j RETURN
-A VYATTA_CT_OUTPUT_HOOK -j RETURN
-A VYATTA_CT_PREROUTING_HOOK -j RETURN

# 防火墙
*filter
-A INPUT -j UBNT_VPN_IPSEC_FW_HOOK
-A INPUT -j VYATTA_FW_LOCAL_HOOK
-A INPUT -j VYATTA_POST_FW_IN_HOOK
-A FORWARD -j MINIUPNPD
-A FORWARD -j UBNT_VPN_IPSEC_FW_IN_HOOK
-A FORWARD -j UBNT_PFOR_FW_HOOK
-A FORWARD -j UBNT_FW_IN_SUSPEND_HOOK
-A FORWARD -j VYATTA_FW_IN_HOOK
-A FORWARD -j VYATTA_FW_OUT_HOOK
-A FORWARD -j VYATTA_POST_FW_FWD_HOOK
-A OUTPUT -j VYATTA_POST_FW_OUT_HOOK
```

负载均衡

- https://community.ui.com/questions/ 5 0 3 d 5 b 8 1 - 7 cb 5 - 4 fe 5 -ba 3 9 - 7 2 df 1 f 5 eb 9 8 f

```
*nat
# 规则 HOOK 位置
# 标记
-A INPUT -j UBNT_HOOK_WLBL
-A FORWARD -j UBNT_QOS_FW_IN_HOOK
# IN
-A FORWARD -j UBNT_HOOK_WLBI
# OUT
-A OUTPUT -j UBNT_HOOK_WLBO

-A UBNT_HOOK_WLBI -j UBNT_WLBI_LB
-A UBNT_HOOK_WLBL -j UBNT_WLBL_LB
-A UBNT_HOOK_WLBO -j UBNT_WLBO_LB

# 后端
-A UBNT_WLBE_LB -o eth1 -j RETURN
-A UBNT_WLBE_LB -o pppoe0 -j RETURN
-A UBNT_WLBE_LB -j ACCEPT

# 恢复 mark
-A UBNT_WLBI_LB -j CONNMARK --restore-mark --nfmask 0x7f800000 --ctmask
0x7f800000
# mark 不匹配 返回
-A UBNT_WLBI_LB -m mark ! --mark 0x0/0x7f800000 -j RETURN
# 设置 mark
-A UBNT_WLBI_LB -i eth1 -m state --state NEW -j MARK --set-xmark
0x33000000/0x7f800000
-A UBNT_WLBI_LB -i pppoe0 -m state --state NEW -j MARK --set-xmark
0x32800000/0x7f800000
-A UBNT_WLBI_LB -m mark ! --mark 0x0/0x7f800000 -j CONNMARK --save-mark --
nfmask 0x7f800000 --ctmask 0x7f800000
-A UBNT_WLBI_LB -j RETURN

# 恢复 mark
-A UBNT_WLBL_LB -j CONNMARK --restore-mark --nfmask 0x7f800000 --ctmask
0x7f800000
# mark 不匹配 返回
-A UBNT_WLBL_LB -m mark ! --mark 0x0/0x7f800000 -j RETURN
# 设置 mark
-A UBNT_WLBL_LB -i eth1 -m state --state NEW -j MARK --set-xmark
0x33000000/0x7f800000
-A UBNT_WLBL_LB -i pppoe0 -m state --state NEW -j MARK --set-xmark
0x32800000/0x7f800000
# 无 mark 则设置
-A UBNT_WLBL_LB -m mark ! --mark 0x0/0x7f800000 -j CONNMARK --save-mark
```

```
#fmask 0x7f800000 --ctmask 0x7f800000
# DONE
-A UBNT_WLBL_LB -j RETURN

# 后端处理
-A UBNT_WLBO_LB -j UBNT_WLBE_LB
# mark 处理
-A UBNT_WLBO_LB -j CONNMARK --restore-mark --nfmask 0x7f800000 --ctmask
0x7f800000
-A UBNT_WLBO_LB -m mark ! --mark 0x0/0x7f800000 -j RETURN
-A UBNT_WLBO_LB -o eth1 -p icmp -j MARK --set-xmark 0x33000000/0x7f800000
-A UBNT_WLBO_LB -o pppoe0 -p icmp -j MARK --set-xmark 0x32800000/0x7f800000
-A UBNT_WLBO_LB -m mark ! --mark 0x0/0x7f800000 -j CONNMARK --save-mark --
nfmask 0x7f800000 --ctmask 0x7f800000
-A UBNT_WLBO_LB -m mark ! --mark 0x0/0x7f800000 -j RETURN
# 随机选择
-A UBNT_WLBO_LB -m state --state NEW -m mark --mark 0x0/0x7f800000 -m
dyn_random --prob-name "LB_0"  -j MARK --set-xmark 0x33000000/0x7f800000
-A UBNT_WLBO_LB -m state --state NEW -m mark --mark 0x0/0x7f800000 -j MARK
--set-xmark 0x32800000/0x7f800000
-A UBNT_WLBO_LB -m mark ! --mark 0x0/0x7f800000 -j CONNMARK --save-mark --
nfmask 0x7f800000 --ctmask 0x7f800000
-A UBNT_WLBO_LB -j RETURN

# 负载逻辑
-A UBNT_WLB_LB -j CONNMARK --restore-mark --nfmask 0x7f800000 --ctmask
0x7f800000
-A UBNT_WLB_LB -m mark ! --mark 0x0/0x7f800000 -j RETURN
-A UBNT_WLB_LB -m state --state NEW -m mark --mark 0x0/0x7f800000 -m
dyn_random --prob-name "LB_0"  -j MARK --set-xmark 0x33000000/0x7f800000
-A UBNT_WLB_LB -m state --state NEW -m mark --mark 0x0/0x7f800000 -j MARK -
-set-xmark 0x32800000/0x7f800000
-A UBNT_WLB_LB -j CONNMARK --save-mark --nfmask 0x7f800000 --ctmask
0x7f800000
-A UBNT_WLB_LB -j RETURN
```

笔记

- 路由功能主要通过配置树进行全局控制
- 调用 iproute 2 和 iptables 进行实际配置
- 通过 iptables 的 mark + iproute 的 rule 实现复杂的策略路由
- /
  - custom-attribute
  - firewall
  - interfaces

    - load-balance
    - policy
    - port-forward
    - protocols
    - service
    - system
    - traffic-control
    - traffic-policy
    - vpn
    - zone-policy

系统

- EdgeOS file system layout and firmware images
- /proc/mtd

版本

2.0 - 2019-1-7

- v2.0.0
    - ER-X/ER-X-SFP/EP-R 6
- Debian 9

✏️ Edit this page