透明代理(TPROXY)

原来出过一篇透明代理的教程，但过了许久，V2Ray 也已经迭代了好多个版本。原来的教程依旧可以正常使用，但随着 V2Ray 的更新，V2Ray 推出了新的透明代理方式——TPROXY，原来的叫 REDIRECT。最近测试了一下 TPROXY ，效果还不错，主观感觉比 REDIRECT 好。并且在本文的透明代理中，DNS 服务将由 V2Ray 提供。不过这种方式需要 iptables 的 TPROXY 模块支持，有一些阉割版的系统会精简掉 TPROXY 模块，这种系统是不适用于本文的。

普通家庭大多数是光纤入户接光猫调制解调，一个路由器的 WAN 口接光猫的 LAN 口，要上网的设备（如 PC 、电视盒子、手机）接路由器 LAN 口。本文的透明代理需要一台 Linux 主机接路由器 LAN 口，作为局域网中的网关，为其他接入局域网中的设备提供翻墙功能。这样的方式与我原来的透明代理教程是一样的，都是搭建在一个 Linux 主机上。这样可以透明代理的设备，有的人叫“透明网关”，也有的叫“旁路由”。我觉得这种不是很严肃的场合，叫什么都行，只要不妨碍理解。

很多设备都可以做透明网关，路由器、开发板、个人电脑、虚拟机和 Android 设备等。路由器可能比较特殊点，因为它本身就可以充当网关。上面可能说得太抽象，我就举些实际的，比如说树莓派、香橙派、用 PC 装的 Linux 虚拟机、淘宝的工控机（如 j1900）、NAS、电视盒子（如翻车迅）、你刚配的牙膏厂或农厂的电脑，这些都没问题。至于到底用什么？这得看需求，我觉得网络 200M 以下搞个高性能的类树莓派的 SBC 就够了，200M 以上就得考虑 X86 主机了（如今甚火的软路由）。当然，到底怎么选择还是得看自己。

本文假设你已经有一个设备（就以树莓派举例），将用来作网关（或说旁路由），并且已经安装好 Linux。关于系统，我更推荐 Debian 或 Debian 衍生版。为方面起见，本文均以 root 权限账户执行命令。并且有一台 PC 以便于操作。

设置网关

1. 用网线将树莓派接入路由器 LAN 口，假设分给树莓派的 IP 是 192.168.1.22。
2. 树莓派开启 IP 转发（需要开启 IP 转发才能作为网关）。命令为 `echo net.ipv4.ip_forward=1 >> /etc/sysctl.conf && sysctl -p` 。执行后将出现 net.ipv4.ip_forward=1 的提示。
3. 手动配置 PC 的网络，将默认网关指向树莓派的地址即 `192.168.1.22` 。此时 PC 应当能正常上网（由于还没设置代理，“正常”是指可以上国内的网站）。

树莓派安装配置 V2Ray

1. 安装 V2Ray。可以使用 V2Fly 提供的 fhs-install-v2ray 脚本安装，由于 GFW 会恶化对 GitHub 的访问，直接运行脚本几乎无法安装，建议先下载 V2Ray 的压缩包，然后用安装脚本通过 --local 参数进行安装。
2. 配置 V2Ray。按照前文教程将 V2Ray 配置成客户端形式。然后执行 `curl -so /dev/null -w "%{http_code}" google.com -x socks5://127.0.0.1:1080` 确认 V2Ray 已经可以翻墙(命令中 socks5 指 inbound 协议为 socks，1080 指该 inbound 端口是 1080)。如果执行这个命令出现了 301 或 200 这类数字的话代表可以翻墙，如果长时间没反应或者是 000 的话说明不可以翻墙。

配置透明代理

为 V2Ray 配置透明代理的入站和 DNS 分流

以下是 V2Ray 透明代理的配置示例，配置文件之后有说明。

```json
{
  "inbounds": [
    {
      "tag":"transparent",
      "port": 12345,
      "protocol": "dokodemo-door",
      "settings": {
        "network": "tcp,udp",
        "followRedirect": true
      },
      "sniffing": {
        "enabled": true,
        "destOverride": [
          "http",
          "tls"
        ]
      },
      "streamSettings": {
        "sockopt": {
          "tproxy": "tproxy", // 透明代理使用 TPROXY 方式
          "mark":255
        }
```

☰ 新 V2Ray 白话文指南

```
    {
      "port": 1080,
      "protocol": "socks", // 入口协议为 SOCKS 5
      "sniffing": {
        "enabled": true,
        "destOverride": ["http", "tls"]
      },
      "settings": {
        "auth": "noauth"
      }
    }
  ],
  "outbounds": [
    {
      "tag": "proxy",
      "protocol": "vmess", // 代理服务器
      "settings": {
        "vnext": [
          ...
        ]
      },
      "streamSettings": {
        "sockopt": {
          "mark": 255
        }
      },
      "mux": {
        "enabled": true
      }
    },
    {
      "tag": "direct",
      "protocol": "freedom",
      "settings": {
        "domainStrategy": "UseIP"
      },
      "streamSettings": {
        "sockopt": {
          "mark": 255
        }
      }
    },
    {
      "tag": "block",
      "protocol": "blackhole",
```

新 V2Ray 白话文指南

```
          "type": "http"
        }
      }
    },
    {
      "tag": "dns-out",
      "protocol": "dns",
      "streamSettings": {
        "sockopt": {
          "mark": 255
        }
      }
    }
  ],
  "dns": {
    "servers": [
      {
        "address": "223.5.5.5", //中国大陆域名使用阿里的 DNS
        "port": 53,
        "domains": [
          "geosite:cn",
          "ntp.org",   // NTP 服务器
          "$myserver.address" // 此处改为你 VPS 的域名
        ]
      },
      {
        "address": "114.114.114.114", //中国大陆域名使用 114 的 DNS (备用)
        "port": 53,
        "domains": [
          "geosite:cn",
          "ntp.org",   // NTP 服务器
          "$myserver.address" // 此处改为你 VPS 的域名
        ]
      },
      {
        "address": "8.8.8.8", //非中国大陆域名使用 Google 的 DNS
        "port": 53,
        "domains": [
          "geosite:geolocation-!cn"
        ]
      },
      {
        "address": "1.1.1.1", //非中国大陆域名使用 Cloudflare 的 DNS
        "port": 53,
        "domains": [
```

新 V2Ray 白话文指南

```
      ]
    },
    "routing": {
      "domainStrategy": "IPOnDemand",
      "rules": [
        { // 劫持 53 端口 UDP 流量，使用 V2Ray 的 DNS
          "type": "field",
          "inboundTag": [
            "transparent"
          ],
          "port": 53,
          "network": "udp",
          "outboundTag": "dns-out"
        },
        { // 直连 123 端口 UDP 流量（NTP 协议）
          "type": "field",
          "inboundTag": [
            "transparent"
          ],
          "port": 123,
          "network": "udp",
          "outboundTag": "direct"
        },
        {
          "type": "field",
          "ip": [
            // 设置 DNS 配置中的国内 DNS 服务器地址直连，以达到 DNS 分流目的
            "223.5.5.5",
            "114.114.114.114"
          ],
          "outboundTag": "direct"
        },
        {
          "type": "field",
          "ip": [
            // 设置 DNS 配置中的国外 DNS 服务器地址走代理，以达到 DNS 分流目的
            "8.8.8.8",
            "1.1.1.1"
          ],
          "outboundTag": "proxy" // 改为你自己代理的出站 tag
        },
        { // 广告拦截
          "type": "field",
          "domain": [
```

新 V2Ray 白话文指南

```
        "outboundTag": "block"
      },
      { // BT 流量直连
        "type": "field",
        "protocol":["bittorrent"],
        "outboundTag": "direct"
      },
      { // 直连中国大陆主流网站 ip 和 保留 ip
        "type": "field",
        "ip": [
          "geoip:private",
          "geoip:cn"
        ],
        "outboundTag": "direct"
      },
      { // 直连中国大陆主流网站域名
        "type": "field",
        "domain": [
          "geosite:cn"
        ],
        "outboundTag": "direct"
      }
    ]
  }
}
```

以上是 V2Ray 透明代理的参考配置，关于配置有一些注意点及说明:

- dokodemo-door 是用来接收透明代理的入站协议，followRedirect 项须为 true 以及 sockopt.tproxy 项须为 tproxy，建议开启 snifing，否则路由无法匹配域名；
- 本节添加了 DNS 配置，用来对国内外域名进行 DNS 分流，需要 `DNS 配置` 、 `DNS 入站` 、 `DNS 出站` 和 `路由` 四者配合，在本例中 DNS 入站直接使用透明代理入站，可参考 DNS 及其应用 ；
- 在 DNS 配置中，依次配置了 Google、Cloudflare、114 和阿里的 DNS，由于在阿里的 DNS 中指定了 domain，所以匹配的域名会用阿里的 DNS 查询，其他的先查询 Google 的 DNS，如果查不到的话再依次查 Cloudflare 及 114 的。所以达到了国内外域名 DNS 分流，以及 DNS 备用。要注意把 NTP 服务器和你自己 VPS 域名也加入到直连的 DNS，否则会导致 V2Ray 无法与 VPS 正常连接；
- DNS 配置只是说明哪些域名查哪个 DNS，至于哪个 DNS 走代理哪个 DNS 直连要在 routing 里设置规则；

代理，也无法自动校准时间；

- freedom 的出站设置 domainStrategy 为 UseIP，以避免直连时因为使用本机的 DNS 出现一些奇怪问题；
- 注意要在 dokodemo inbound 和所有的 outbound 加一个 255 的 mark，这个 mark 与下文 iptables 命令中 `iptables -t mangle -A V2RAY_MASK -j RETURN -m mark --mark 0xff` 配合，以直连 V2Ray 发出的流量（blackhole 可以不配置 mark）。

配置透明代理规则

此部分分为 iptables 和 nftables，两者作用相同，择其一即可。

iptables 规则

执行下面的命令开启透明代理。由于使用了 TPROXY 方式的透明代理，所以 TCP 流量也是使用 mangle 表。以下命令中，以 `#` 开头的为注释。

```
# 设置策略路由
ip rule add fwmark 1 table 100
ip route add local 0.0.0.0/0 dev lo table 100

# 代理局域网设备
iptables -t mangle -N V2RAY
iptables -t mangle -A V2RAY -d 127.0.0.1/32 -j RETURN
iptables -t mangle -A V2RAY -d 224.0.0.0/4 -j RETURN
iptables -t mangle -A V2RAY -d 255.255.255.255/32 -j RETURN
iptables -t mangle -A V2RAY -d 192.168.0.0/16 -p tcp -j RETURN # 直连局域网，避免
iptables -t mangle -A V2RAY -d 192.168.0.0/16 -p udp ! --dport 53 -j RETURN #
iptables -t mangle -A V2RAY -j RETURN -m mark --mark 0xff    # 直连 SO_MARK 为
iptables -t mangle -A V2RAY -p udp -j TPROXY --on-ip 127.0.0.1 --on-port 1234
iptables -t mangle -A V2RAY -p tcp -j TPROXY --on-ip 127.0.0.1 --on-port 1234
iptables -t mangle -A PREROUTING -j V2RAY # 应用规则

# 代理网关本机
iptables -t mangle -N V2RAY_MASK
iptables -t mangle -A V2RAY_MASK -d 224.0.0.0/4 -j RETURN
iptables -t mangle -A V2RAY_MASK -d 255.255.255.255/32 -j RETURN
iptables -t mangle -A V2RAY_MASK -d 192.168.0.0/16 -p tcp -j RETURN # 直连局域网
iptables -t mangle -A V2RAY_MASK -d 192.168.0.0/16 -p udp ! --dport 53 -j RETU
iptables -t mangle -A V2RAY_MASK -j RETURN -m mark --mark 0xff    # 直连 SO_MA
iptables -t mangle -A V2RAY_MASK -p udp -j MARK --set-mark 1   # 给 UDP 打标记,
iptables -t mangle -A V2RAY_MASK -p tcp -j MARK --set-mark 1   # 给 TCP 打标记,
```

```
# 新建 DIVERT 规则，避免已有连接的包二次通过 TPROXY，理论上有一定的性能提升
iptables -t mangle -N DIVERT
iptables -t mangle -A DIVERT -j MARK --set-mark 1
iptables -t mangle -A DIVERT -j ACCEPT
iptables -t mangle -I PREROUTING -p tcp -m socket -j DIVERT
```

执行了以上 ip 和 iptables 命令后，局域网同网段的设备以及网关本身就可以直接翻墙了。

关于 iptables 规则，比较容易理解，如果不太理解的话也可以 Google 搜索其他相关文章资料对比学习。在类 ss-redir 透明代理中，有两个观点非常深入人心：

```
1. UDP 只能 TPROXY
2. TPROXY 不能用于 OUTPUT 链
```

然后我们从这两个观点很容易得出一个推论：**无法在提供透明代理的本机(即本例中的网关)上对 UDP 透明代理**。 这个结论好像并没有什么问题，对吧？但实际上，在本例的配置中无论是 TCP 还是 UDP，都可以实现在本机上的透明代理，而且都是用 TPROXY。那好像又跟前面的结论矛盾了？其实关键在于这三句命令：

```
iptables -t mangle -A V2RAY_MASK -p udp -j MARK --set-mark 1
iptables -t mangle -A V2RAY_MASK -p tcp -j MARK --set-mark 1
iptables -t mangle -A OUTPUT -j V2RAY_MASK
```

这几句是说给 OUTPUT 链的 TCP 和 UDP 打个标记 1(OUTPUT 应用 V2RAY_MASK 链)。由于 Netfilter 的特性，在 OUTPUT 链打标记会使相应的包重路由到 PREROUTING 链上，在已经配置好了 PREROUTING 相关的透明代理的情况下，OUTPUT 链也可以透明代理了，也就是网关对自身的 UDP 流量透明代理自身（当然 TCP 也不在话下）。因为这是 netfilter 本身的特性，Shadowsocks 应该也可以用同样的方法对本机的 UDP 透明代理，但我没有实际测试过效果。

nftables 规则

nftables 与 iptables 同样基于 netfilter 框架，早在 2014 年就引入 Linux 内核中，旨在改进 iptables 的一些问题并且将之替换。目前有不少 Linux 发行版默认网络过滤以 nftables 替换了 iptables，但是直到 4.19 的 Linux 内核才有 nft_tproxy 模块，这个模块是透明代理所必须的。如果使用 nftables 配置透明代理，必须具备 nft_tproxy 和 nft_socket 模块，可通过命令

以下是 nftables 规则语句，本质与 iptables 没什么差别。

```
# 设置策略路由
ip rule add fwmark 1 table 100
ip route add local 0.0.0.0/0 dev lo table 100

#代理局域网设备
nft add table v2ray
nft add chain v2ray prerouting { type filter hook prerouting priority 0 \; }
nft add rule v2ray prerouting ip daddr {127.0.0.1/32, 224.0.0.0/4, 255.255.25
nft add rule v2ray prerouting meta l4proto tcp ip daddr 192.168.0.0/16 return
nft add rule v2ray prerouting ip daddr 192.168.0.0/16 udp dport != 53 return
nft add rule v2ray prerouting mark 0xff return # 直连 0xff 流量
nft add rule v2ray prerouting meta l4proto {tcp, udp} mark set 1 tproxy to 12

# 代理网关本机
nft add chain v2ray output { type route hook output priority 0 \; }
nft add rule v2ray output ip daddr {127.0.0.1/32, 224.0.0.0/4, 255.255.255.25
nft add rule v2ray output meta l4proto tcp ip daddr 192.168.0.0/16 return
nft add rule v2ray output ip daddr 192.168.0.0/16 udp dport != 53 return
nft add rule v2ray output mark 0xff return # 直连 0xff 流量
nft add rule v2ray output meta l4proto {tcp, udp} mark set 1 accept # 重路由至

# DIVERT 规则
nft add table filter
nft add chain filter divert { type filter hook prerouting priority -150 \; }
nft add rule filter divert meta l4proto tcp socket transparent 1 meta mark se
```

开机自动运行透明代理规则

由于策略路由以及 iptables/nftables 有重启会失效的特性，所以当测试配置没有问题之后，需要再弄个服务在开机时自动配置策略路由和 iptables，否则每次开机的时候就要手动执行一遍。

1. 由于 iptables 命令有点多，所以先将 iptables 规则保存到 /etc/iptables/rules.v4 中。

```
mkdir -p /etc/iptables && iptables-save > /etc/iptables/rules.v4
```

```
mkdir -p /etc/nftables && nft list ruleset > /etc/nftables/rules.v4
```

2. 在 /etc/systemd/system/ 目录下创建一个名为 tproxyrule.service 的文件，然后添加以下内容并保存。

```
[Unit]
Description=Tproxy rule
After=network.target
Wants=network.target

[Service]

Type=oneshot
RemainAfterExit=yes
# 注意分号前后要有空格
ExecStart=/sbin/ip rule add fwmark 1 table 100 ; /sbin/ip route add local 0.0
ExecStop=/sbin/ip rule del fwmark 1 table 100 ; /sbin/ip route del local 0.0.
# 如果是 nftables，则改为以下命令
# ExecStart=/sbin/ip rule add fwmark 1 table 100 ; /sbin/ip route add local 0
# ExecStop=/sbin/ip rule del fwmark 1 table 100 ; /sbin/ip route del local 0.

[Install]
WantedBy=multi-user.target
```

3. 执行下面的命令使 tproxyrule.service 可以开机自动运行。

```
systemctl enable tproxyrule
```

其他

解决 too many open files 问题

对 UDP 透明代理比较容易出现“卡住”的情况，这个时候细心的朋友可能会发现日志中出现了非常多 "too many open files" 的语句，这主要是受到最大文件描述符数值的限制，把这个数值往大调就好了。设置步骤如下。

```
[Unit]
Description=V2Ray Service
Documentation=https://www.v2fly.org/
After=network.target nss-lookup.target

[Service]
User=nobody
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
AmbientCapabilities=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
NoNewPrivileges=true
ExecStart=/usr/local/bin/v2ray -config /usr/local/etc/v2ray/config.json
Restart=on-failure
RestartPreventExitStatus=23
LimitNPROC=500
LimitNOFILE=1000000

[Install]
WantedBy=multi-user.target
```

2. 执行 `systemctl daemon-reload && systemctl restart v2ray` 生效。

设定网关为静态 IP

最好给网关设成静态 IP，以免需要重启的时 IP 发生变化。如何设置请自行探究。 提示一下，如果你用 nmcli 命令设置静态 IP，最好先另外添加一个 connection 进行配置，配置好之后在切换到新添加的这个 connection 来。因为如果在原有的 connection 上直接修改成静态 IP **可能**会导致**无法透明代理**。

设定 DHCP

在路由器上设定 DHCP，将网关地址指向网关设备，在本文的举例中即为树莓派的 IP 192.168.1.22； DNS 随意，因为已经配置了劫持 53 端口的 UDP，当然填常规的 DNS 也更是没有问题的。

备注

TPROXY 并且使用了 V2Ray 的 DNS。但我没有 IPV6 环境，无法进行测试，所以本文只适用于 IPV4。

2. 据了解，到目前（2020.12）为止，在所知的具备透明代理功能的翻墙工具中，TCP 透明代理方式可以使用的 TPROXY 的除 V2Ray 外只有 Trojan-Go、Trojan-rs 和 Brook。所以你要找其他资料参考的话，要注意透明代理方式，因为大多数是 REDIRECT 模式的（包括 V2Ray 官网给的示例）。
3. 由于设计原因，V2Ray 不支持 Full Cone 类型的 NAT，详情见此 Issue 。

参考资料

- DNS 及其应用
- 漫谈各种黑科技式 DNS 技术在代理环境中的应用
- Linux transparent proxy support
- V2Ray 透明代理样例
- iptables - Wikipedia
- Nftables - Archwiki
- Man page of NFT
- Transparent proxy support
- Nftables wiki
- Nftables - Debian Wiki
- 包的路由转圈圈

更新历史

- 2019-10-19 初版
- 2019-10-25 关于配置的说明
- 2019-10-26 改善 DNS 配置
- 2019-10-27 改进
- 2019-10-28 解释重路由
- 2020-08-31 添加 DIVERT 规则
- 2020-09-29 添加 iptables 规则，解决 V2Ray 占用大量 CPU 的问题
- 2020-11-27 新增 nftables
- 2020-11-29 修改 DNS 配置以适应 v4.27.4 修改的 DNS 匹配顺序
- 2020-12-04 补充支持 TPROXY 的工具
- 2020-12-06 添加 dokodemo mark 和 --on-ip 参数

新 V2Ray 白话文指南

在 GitHub 上编辑此页

上次更新: 5/27/2020, 7:40:13 AM

← 透明代理 反向代理 →