РЕФЕРАТ

Отчёт содержит 45 стр., 1 рис., 4 источн., 1 прил.

В курсовой работе рассмотрены принципы и методики реализации точечной маршрутизации по доменам на маршрутизаторах с использованием прошивки OpenWrt. Основное внимание уделено настройке правил маршрутизации для отдельных доменов, что позволяет эффективно управлять сетевым трафиком через VPN-туннели (например, WireGuard) или другие интерфейсы. В работе детализируются этапы установки и конфигурации необходимых программных пакетов. Особое внимание уделено настройке DNS через DNSCrypt и устранению возможных проблем, связанных с маршрутизацией и разрешением доменных имен.

Работа основана на опыте использования OpenWrt на роутерах и включает скрипт, позволяющий автоматически настраивать доменную маршрутизаицю автоматически.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные сети требуют гибкого и эффективного управления трафиком, особенно в условиях стремительного роста числа подключенных устройств и использования сложных сетевых топологий. Одним из ключевых инструментов для таких задач является маршрутизация, позволяющая направлять трафик по различным маршрутам в зависимости от его назначения. В частности, точечная маршрутизация по доменам открывает новые возможности для управления потоками данных, позволяя выделять трафик определённых ресурсов или групп доменов и перенаправлять его через заранее заданные маршруты.

В данной курсовой работе рассматривается реализация точечной маршрутизации по доменам на роутерах с прошивкой OpenWrt. OpenWrt, как одна из наиболее популярных и гибких прошивок для роутеров, предоставляет мощные инструменты для настройки сетевых параметров, включая работу с VPN-туннелями, списками доменов и кастомными правилами маршрутизации. Такие решения актуальны как для повышения конфиденциальности (через использование VPN), так и для обеспечения избирательного доступа к ресурсам или обхода сетевых ограничений.

Целью работы является анализ методов настройки маршрутизации по доменам на маршрутизаторах с OpenWrt и разработка оптимального решения, позволяющего пользователям гибко управлять трафиком. В ходе исследования планируется рассмотреть теоретические аспекты маршрутизации и провести практическую реализацию на примере конкретной конфигурации.

ОСНОВНАЯ ЧАСТЬ

1 Маршрутизация

В данной работе рассматривается использование широко распространённых open-source инструментов для реализации точечной маршрутизации по доменам. Основные компоненты данного метода — это dnsmasq и nftables.

Примером служит домен graylog.org, доступ к которому ограничен системой WAF Cloudflare для пользователей с российских IP-адресов. Это означает, что стандартное подключение к сайту через провайдера невозможно.

Процесс маршрутизации начинается с отправки компьютером DNS-запроса к маршрутизатору, чтобы разрешить доменное имя в IP-адрес [1]. В OpenWrt обработкой таких запросов занимается dnsmasq, обладающий функцией, позволяющей сопоставлять домены с IP-адресами и сохранять их в специальные таблицы (sets), поддерживаемые nftables.

Пример конфигурации представлен в листинге 1. Здесь list name задаёт таблицу (set), в которую будут записаны IP-адреса домена graylog.org. При запросе dnsmasq обращается к вышестоящему DNS-серверу, получает IP-адреса и возвращает их клиенту, одновременно добавляя их в указанный set (vpn_domains).

Листинг 1 – Пример конфигурации Dnsmasq

```
config ipset
        list name 'vpn_domains'
        list domain 'graylog.org'
```

В дальнейшем, при поступлении пакета на маршрутизатор, если его IP-адрес совпадает с адресами из таблицы, маршрут определяет его направление через заранее настроенный туннель (например, VPN). Оставшийся трафик продолжает проходить через провайдера без изменений, что изображено на рисунке 1.

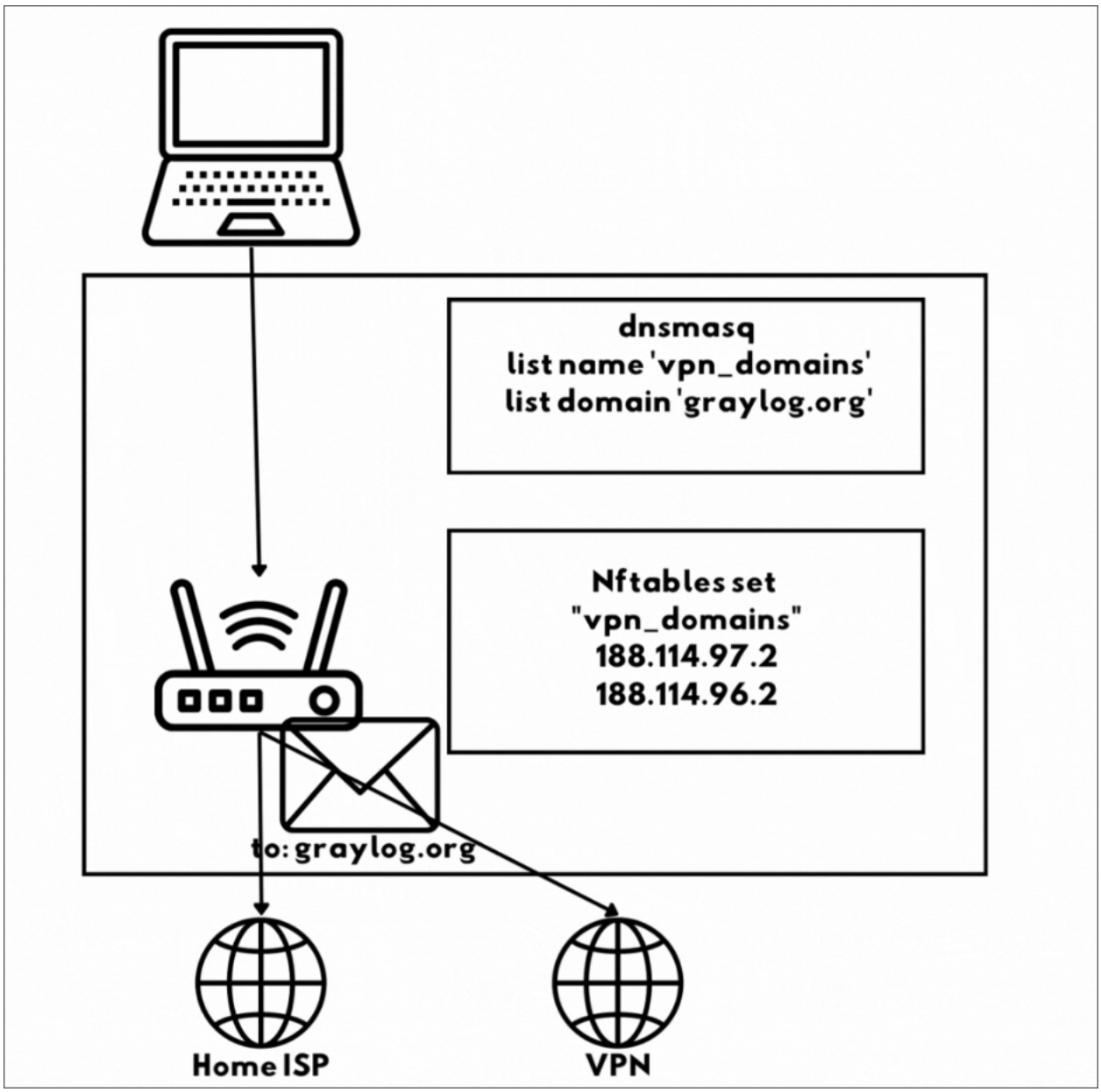

Рисунок 1 – Маршрутизация трафика

2 Настройка

2.1 Установка пакетов

Для настройки точечной маршрутизации по доменам на OpenWrt требуется установка пакета dnsmasq-full, так как по умолчанию в прошивке используется урезанная версия dnsmasq для экономии места. Dnsmasq — это ключевой компонент системы, отвечающий за DNS-обслуживание, и его удаление может нарушить работу маршрутизатора. Поэтому перед установкой нового пакета необходимо выполнить несколько подготовительных шагов.

Процесс установки dnsmasq-full выглядит следующим образом [2]:

а) Обновление индекса пакетов и скачивание dnsmasq-full. Для начала нужно обновить список доступных пакетов и выгрузить пакет dnsmasq-full в директорию /tmp:

```
opkg update && cd /tmp/ && opkg download dnsmasq-full
```

б) Удаление стандартного dnsmasq. Поскольку стандартный dnsmasq не поддерживает необходимые функции, его нужно удалить перед установкой полной версии:

```
opkg remove dnsmasq
```

в) Установка dnsmasq-full. Новый пакет устанавливается из кэша в директории /tmp. При этом важно сохранить текущие настройки DHCP, если они необходимы. Если файл /etc/config/dhcp имеет пользовательские изменения, стоит перенести конфигурацию вручную:

```
opkg install dnsmasq-full --cache /tmp/
mv /etc/config/dhcp-opkg /etc/config/dhcp
```

г) Установка curl. Для работы со скриптами обновления списков доменов и проверки подключения потребуется установка утилиты curl:

```
    opkg install curl
```

Эти шаги обеспечивают установку всех необходимых инструментов для реализации точечной маршрутизации по доменам на маршрутизаторе с OpenWrt

2.2 Настройка маршрутизации

Для управления трафиком и перенаправления его через VPN-туннель необходимо настроить специальную таблицу маршрутизации и правила для маркировки пакетов. Все пакеты с заданной маркировкой будут направляться в отдельную таблицу маршрутизации, которая перенаправляет трафик в туннель.

Добавляется новая таблица маршрутизации с уникальным идентификатором [2]:

```
echo '99 vpn' >> /etc/iproute2/rt_tables
```

Для указания, что весь трафик с определённой маркировкой должен использовать таблицу vpn, настраивается правило. Это можно выполнить через UCI или напрямую в конфигурационных файлах (/etc/config/network).

```
uci add network rule
uci set network.@rule[-1].name='mark0x1'
uci set network.@rule[-1].mark='0x1'
uci set network.@rule[-1].priority='100'
uci set network.@rule[-1].lookup='vpn'
uci commit network
```

Добавляется правило, направляющее весь трафик таблицы маршрутизации в туннель. В случае с WireGuard и интерфейсом wg0 выполняется следующая настройка.

```
uci set network.vpn_route=route
uci set network.vpn_route.interface='wg0'
uci set network.vpn_route.table='vpn'
uci set network.vpn_route.target='0.0.0.0/0'
uci commit network
```

Для туннелей, таких как OpenVPN, Sing-box, tun2socks, создающих интерфейсы типа device (например, tun0), настройка через UCI не поддерживается. В этом случае применяется hotplug. Создаётся файл /etc/hotplug.d/iface/30-vpnroute со следующим содержимым:

```
#!/bin/sh

sleep 10
ip route add table vpn default dev tun0
```

2.3 Настройка firewall

Для обеспечения прохождения трафика через туннель требуется создать зону в файерволе, настроить nftables set для IP-адресов доменов, а также правило, которое будет маркировать трафик к этим адресам.

Зона для туннеля настраивается для обеспечения корректного прохождения трафика через него. Настройка зоны может варьироваться в зависимости от используемого туннеля (WireGuard, OpenVPN и т.д.) и описана в соответствующей документации для конкретного типа туннеля. Обычно создаётся зона с интерфейсом туннеля, например, wg0 или tun0, и задаются необходимые правила для передачи трафика между зонами.

Создаётся список vpn_domains, в который будут автоматически добавляться IP-адреса доменов для точечной маршрутизации. Настройка через UCI выглядит следующим образом:

```
uci add firewall ipset
uci set firewall.@ipset[-1].name='vpn_domains'
uci set firewall.@ipset[-1].match='dst_net'
uci commit firewall
```

Для маркировки трафика, идущего к IP-адресам из списка vpn_domains, создаётся правило:

```
uci add firewall rule
uci set firewall.@rule[-1]=rule
uci set firewall.@rule[-1].name='mark_domains'
uci set firewall.@rule[-1].src='lan'
uci set firewall.@rule[-1].dest='*'
uci set firewall.@rule[-1].proto='all'
uci set firewall.@rule[-1].ipset='vpn_domains'
uci set firewall.@rule[-1].set_mark='0x1'
uci set firewall.@rule[-1].target='MARK'
uci set firewall.@rule[-1].family='ipv4'
uci commit firewall
```

2.4 Настройка автоматического обновления списка доменов

Чтобы автоматически обновлять список доменов, которые необходимо перенаправлять в тунель быд написан скрипт, включающий следующие функции:

– Проверка доступности интернета после загрузки роутера.

– Загрузка готового конфигурационного файла в /tmp/dnsmasq.d/.
– Проверка валидности загруженного файла.
– Перезапуск службы dnsmasq при наличии корректной конфигурации.

Для обработки доменов и их резолвинга в списки IP-адресов используется конфигурация для dnsmasq. Она помещается в /tmp/dnsmasq.d/ — директорию для временных конфигураций. Пример конфигурации:

```
nftset=/graylog.org/4#inet#fw4#vpn_domains
nftset=/terraform.io/4#inet#fw4#vpn_domains
```

Каждая строка добавляет домен (включая субдомены) в список vpn_domains. Таким образом, IP-адреса доменов автоматически помещаются в таблицу маршрутизации.

Скрипт обновления списка доменов (/etc/init.d/getdomains):

```
#!/bin/sh /etc/rc.common

START=99

start () {
    DOMAINS=https://raw.githubusercontent.com/itdoginfo/allow-
       domains/main/Russia/inside-dnsmasq-nfset.lst

    count=0
    while true; do
        if curl -m 3 github.com; then
            curl -f $DOMAINS --output /tmp/dnsmasq.d/domains.lst
            break
        else
            echo "GitHub is not available. Check the internet
               availability [$count]"
            count=$((count+1))
        fi
    done

    if dnsmasq --conf-file=/tmp/dnsmasq.d/domains.lst --test 2>&1 |
        grep -q "syntax check OK"; then
        /etc/init.d/dnsmasq restart
    fi
}
```

Скрипт необходимо сделать исполняемым и добавить в автозагрузку:

```
chmod +x /etc/init.d/getdomains
ln -sf ../init.d/getdomains /etc/rc.d/S99getdomains
```

Для регулярного обновления конфигурации используется cron. Добавляется задание на обновление списка каждые 8 часов [3]. В файл crontab добавляется строка:

```
0 */8 * * * /etc/init.d/getdomains start
```

После настройки выполняется перезапуск сети и ручной запуск скрипта:

```
service network restart
service getdomains start
```

Преимущества подхода:

- Ускорение работы: Готовые конфигурации избавляют от необходимости конвертировать списки вручную.
- Удобство обновления: Скрипт автоматически загружает и проверяет новые списки.
- Автоматизация: Настройка автозапуска и периодического обновления обеспечивает бесперебойную работу маршрутизации.

2.5 Добавление доменов в обход скрипта

Для добавления необходимых доменов в список vpn_domains через конфигурацию dnsmasq требуется изменить файл /etc/config/dhcp. В конце файла добавляется следующий блок:

```
config ipset
        list name 'vpn_domains'
        list domain 'graylog.org'
        list domain 'terraform.io'
```

Каждый домен добавляется новой строкой с ключевым словом list domain. Поскольку dnsmasq поддерживает wildcard, добавлять субдомены отдельно не требуется — они автоматически обрабатываются. После внесения изменений для применения настроек необходимо перезапустить dnsmasq:

```
service dnsmasq restart
```

2.6 Возможные проблемы

В случае точечной маршрутизации через домены на роутере с OpenWrt могут возникнуть проблемы, если провайдер вмешивается в DNS-запросы или блокирует трафик, что нарушает корректную работу маршрутизации. Существуют несколько возможных ситуаций:

а) Подмена IP-адресов на DNS-серверах провайдера. В этом случае достаточно просто сменить DNS-сервер на роутере (например, на 8.8.8.8 или 8.8.4.4), чтобы обойти эту подмену.

б) Провайдер подменяет запросы на своих DNS-серверах, даже если использовать другие DNS-серверы (например, 8.8.8.8), оборудование провайдера может перехватывать и изменять DNS-запросы. Для решения этой проблемы можно настроить DNS over TLS или DNS over HTTPS.

в) Провайдер блокирует весь трафик на порту 53, кроме запросов к своим DNS-серверам. В этой ситуации необходимо использовать зашифрованные каналы для DNS-запросов, такие как DNS over TLS или DNS over HTTPS.

Для обхода блокировки DNS-запросов можно установить один из двух популярных резолверов, поддерживающих шифрование: DNSCrypt-proxy2 или Stubby. Если на роутере ограниченное количество памяти, например, 16 МБ, лучше выбрать Stubby из-за его меньшего размера. В остальных случаях можно использовать DNSCrypt для более широких возможностей.

DNSCrypt-proxy2

DNSCrypt-proxy2 позволяет шифровать DNS-запросы через DNS over TLS или DNS over HTTPS. Этот инструмент требует больше места на роутере (около 10.7 МБ), но предоставляет расширенные функции [4].

```
opkg update && opkg install dnscrypt-proxy2
```

Конфигурация изменяется в файле /etc/dnscrypt-proxy2/dnscrypt-proxy.toml, где можно выбрать серверы в параметре server_names. После этого нужно перезапустить сервис:

```
service dnscrypt-proxy restart
```

Далее настраиваем dnsmasq, чтобы он использовал DNSCrypt как сервер:

```
uci set dhcp.@dnsmasq[0].noresolv="1"
uci -q delete dhcp.@dnsmasq[0].server
uci add_list dhcp.@dnsmasq[0].server="127.0.0.53#53"
uci commit dhcp
service dnsmasq restart
```

Теперь все DNS-запросы будут шифроваться через DNSCrypt.

Stubby

Stubby — это более легковесная альтернатива DNSCrypt, занимающая всего 36K + библиотеки. Он использует DNS over TLS и по умолчанию настроен на сервера Cloudflare.

```
opkg update && opkg install stubby
```

Конфигурация по умолчанию использует 127.0.0.1:5453, но можно настроить другие серверы в /etc/config/stubby. Для настройки dnsmasq нужно изменить сервер на 127.0.0.1#5453:

```
uci set dhcp.@dnsmasq[0].noresolv="1"
uci -q delete dhcp.@dnsmasq[0].server
uci add_list dhcp.@dnsmasq[0].server="127.0.0.1#5453"
uci commit dhcp
service dnsmasq restart
```

3 Автоматическая настройка

Для автоматизации настроек точечной маршрутизации на роутере с OpenWrt, был создан скрипт, который выполняет шаги, описанные в предыдущей части. Полный код скрипта представлен в Приложении 1. Скрипт предполагает свежую установку OpenWrt или минимальные изменения в текущей конфигурации. Если на роутере уже есть другие туннели или маршруты, выполнение скрипта может привести к некорректной настройке, поэтому рекомендуется предварительно проверить текущую конфигурацию.

а) Выбор туннеля: Скрипт предложит выбрать туннель. Для WireGuard (WG) настройка будет выполнена автоматически. Для OpenVPN, sing-box или tun2socks потребуется вручную настроить соответствующие клиенты. Для sing-box скрипт создаст шаблон конфигурации.

б) Подмена DNS-запросов: Если провайдер подменяет DNS-запросы, скрипт предложит установить резолверы, такие как DNSCrypt-proxy2 или Stubby. DNSCrypt-proxy2 более функционален, но занимает больше памяти, в то время как Stubby более легковесен.

в) Выбор доменов: Скрипт позволяет легко выбрать список доменов, для которых будет настроена точечная маршрутизация. Эти списки загружаются автоматически и добавляются в конфигурацию DNS-сервера.

г) Смена туннелей: Скрипт также позволяет легко переключать туннели. Например, при переходе с OpenVPN на sing-box он автоматически обновит файрвол, но для предотвращения конфликтов интерфейсов потребуется вручную остановить старую службу или изменить интерфейс.

Этот скрипт значительно упрощает настройку, но в сложных случаях, когда требуются изменения в существующих туннелях или маршрутах, может потребоваться ручное вмешательство.

ЗАКЛЮЧЕНИЕ

В процессе выполнения работы были изучены и реализованы ключевые подходы к решению задачи точечной маршрутизации интернет-трафика на основе доменов с использованием возможностей операционной системы OpenWrt. Были подробно рассмотрены принципы работы DNS-серверов, механизмы маршрутизации и перенаправления трафика, а также инструменты, обеспечивающие защиту от возможного вмешательства со стороны провайдеров, такие как DNSCrypt-proxy2 и Stubby.

На основании проведённой работы удалось продемонстрировать эффективное решение задачи маршрутизации трафика через туннель для заданных доменов, что позволяет обеспечить стабильную работу даже в условиях ограничений со стороны интернет-провайдеров. Практическая ценность данной работы заключается в возможности автоматизации описанных процессов, что снижает порог сложности для внедрения подобных решений на устройствах конечных пользователей.

Результаты работы подтверждают актуальность применения OpenWrt и сопутствующих инструментов для реализации сложных сценариев маршрутизации и управления трафиком. Полученные выводы и предложенные методы могут быть полезны как для дальнейших исследований в области сетевых технологий, так и для практического использования в целях обеспечения конфиденциальности, обхода блокировок и повышения гибкости настроек сетевой инфраструктуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Таненбаум Э., Уэзеролл Д.* Компьютерные сети. 6-е изд. — Спб.: Питер, 2023. — С. 992.
2. *OpenWrt*. Documentation. — 2024. — URL: `https://openwrt.org/docs/start` (дата обр. 25.10.2024).
3. *GitLab*. Cron syntax. — 2024. — URL: `https://docs.gitlab.com/ee/topics/cron/` (дата обр. 25.10.2024).
4. *OpenWrt*. DNSCrypt with Dnsmasq and dnscrypt-proxy2. — 2024. — URL: `https://openwrt.org/docs/guide-user/services/dns/dnscrypt_dnsmasq_dnscrypt-proxy2` (дата обр. 25.10.2024).

ПРИЛОЖЕНИЕ А

Автоматическая настройка точечной маршрутизации

Листинг А.1 – Автоматическая настройка точечной маршрутизации

```
#!/bin/sh

#set -x

check_repo() {
    printf "\033[32;1mChecking OpenWrt repo availability...\033[0m\
       n"
    opkg update | grep -q "Failed to download" && printf "\033[32;1
       mopkg failed. Check internet or date. Command for force ntp
       sync: ntpd -p ptbtime1.ptb.de\033[0m\n" && exit 1
}

route_vpn () {
    if [ "$TUNNEL" == wg ]; then
cat << EOF > /etc/hotplug.d/iface/30-vpnroute
#!/bin/sh

ip route add table vpn default dev wg0
EOF
    elif [ "$TUNNEL" == awg ]; then
cat << EOF > /etc/hotplug.d/iface/30-vpnroute
#!/bin/sh

ip route add table vpn default dev awg0
EOF
    elif [ "$TUNNEL" == singbox ] || [ "$TUNNEL" == ovpn ] || [ "
       $TUNNEL" == tun2socks ]; then
cat << EOF > /etc/hotplug.d/iface/30-vpnroute
#!/bin/sh

sleep 10
ip route add table vpn default dev tun0
EOF
    fi

    cp /etc/hotplug.d/iface/30-vpnroute /etc/hotplug.d/net/30-
       vpnroute
```

```
}

add_mark() {
    grep -q "99 vpn" /etc/iproute2/rt_tables || echo '99 vpn' >> /
       etc/iproute2/rt_tables

    if ! uci show network | grep -q mark0x1; then
        printf "\033[32;1mConfigure mark rule\033[0m\n"
        uci add network rule
        uci set network.@rule[-1].name='mark0x1'
        uci set network.@rule[-1].mark='0x1'
        uci set network.@rule[-1].priority='100'
        uci set network.@rule[-1].lookup='vpn'
        uci commit
    fi
}

add_tunnel() {
    echo "We can automatically configure only Wireguard and Amnezia
        WireGuard. OpenVPN, Sing-box(Shadowsocks2022, VMess, VLESS,
        etc) and tun2socks will need to be configured manually"
    echo "Select a tunnel:"
    echo "1) WireGuard"
    echo "2) OpenVPN"
    echo "3) Sing-box"
    echo "4) tun2socks"
    echo "5) wgForYoutube"
    echo "6) Amnezia WireGuard"
    echo "7) Amnezia WireGuard For Youtube"
    echo "8) Skip this step"

    while true; do
    read -r -p '' TUNNEL
        case $TUNNEL in

        1)
            TUNNEL=wg
            break
            ;;

        2)
            TUNNEL=ovpn
```

```
            break
            ;;

        3)
            TUNNEL=singbox
            break
            ;;

        4)
            TUNNEL=tun2socks
            break
            ;;

        5)
            TUNNEL=wgForYoutube
            break
            ;;

        6)
            TUNNEL=awg
            break
            ;;

        7)
            TUNNEL=awgForYoutube
            break
            ;;

        8)
            echo "Skip"
            TUNNEL=0
            break
            ;;

        *)
            echo "Choose from the following options"
            ;;
        esac
    done

    if [ "$TUNNEL" == 'wg' ]; then
        printf "\033[32;1mConfigure WireGuard\033[0m\n"
```

```
if opkg list-installed | grep -q wireguard-tools; then
    echo "Wireguard already installed"
else
    echo "Installed wg..."
    opkg install wireguard-tools
fi

route_vpn

read -r -p "Enter the private key (from [Interface]):"$'\n'
    WG_PRIVATE_KEY

while true; do
    read -r -p "Enter internal IP address with subnet,
       example 192.168.100.5/24 (from [Interface]):"$'\n'
       WG_IP
    if echo "$WG_IP" | egrep -oq '^([0-9]{1,3}\.)
       {3}[0-9]{1,3}/[0-9]+$'; then
        break
    else
        echo "This IP is not valid. Please repeat"
    fi
done

read -r -p "Enter the public key (from [Peer]):"$'\n'
   WG_PUBLIC_KEY
read -r -p "If use PresharedKey, Enter this (from [Peer]).
   If your don't use leave blank:"$'\n' WG_PRESHARED_KEY
read -r -p "Enter Endpoint host without port (Domain or IP)
    (from [Peer]):"$'\n' WG_ENDPOINT

read -r -p "Enter Endpoint host port (from [Peer]) [51820]:
   "$'\n' WG_ENDPOINT_PORT
WG_ENDPOINT_PORT=${WG_ENDPOINT_PORT:-51820}
if [ "$WG_ENDPOINT_PORT" = '51820' ]; then
    echo $WG_ENDPOINT_PORT
fi

uci set network.wg0=interface
uci set network.wg0.proto='wireguard'
uci set network.wg0.private_key=$WG_PRIVATE_KEY
uci set network.wg0.listen_port='51820'
```

```
        uci set network.wg0.addresses=$WG_IP

        if ! uci show network | grep -q wireguard_wg0; then
            uci add network wireguard_wg0
        fi
        uci set network.@wireguard_wg0[0]=wireguard_wg0
        uci set network.@wireguard_wg0[0].name='wg0_client'
        uci set network.@wireguard_wg0[0].public_key=$WG_PUBLIC_KEY
        uci set network.@wireguard_wg0[0].preshared_key=
           $WG_PRESHARED_KEY
        uci set network.@wireguard_wg0[0].route_allowed_ips='0'
        uci set network.@wireguard_wg0[0].persistent_keepalive='25'
        uci set network.@wireguard_wg0[0].endpoint_host=
           $WG_ENDPOINT
        uci set network.@wireguard_wg0[0].allowed_ips='0.0.0.0/0'
        uci set network.@wireguard_wg0[0].endpoint_port=
           $WG_ENDPOINT_PORT
        uci commit
    fi

    if [ "$TUNNEL" == 'ovpn' ]; then
        if opkg list-installed | grep -q openvpn-openssl; then
            echo "OpenVPN already installed"
        else
            echo "Installed openvpn"
            opkg install openvpn-openssl
        fi
        printf "\033[32;1mConfigure route for OpenVPN\033[0m\n"
        route_vpn
    fi

    if [ "$TUNNEL" == 'singbox' ]; then
        if opkg list-installed | grep -q sing-box; then
            echo "Sing-box already installed"
        else
            AVAILABLE_SPACE=$(df / | awk 'NR>1 { print $4 }')
            if  [[ "$AVAILABLE_SPACE" -gt 2000 ]]; then
                echo "Installed sing-box"
                opkg install sing-box
            else
                printf "\033[31;1mNo free space for a sing-box.
                   Sing-box is not installed.\033[0m\n"
```

```
                exit 1
            fi
        fi
        if grep -q "option enabled '0'" /etc/config/sing-box; then
            sed -i "s/  option enabled \'0\'/ option enabled \'1\'/
               " /etc/config/sing-box
        fi
        if grep -q "option user 'sing-box'" /etc/config/sing-box;
           then
            sed -i "s/  option user \'sing-box\'/ option user \'
               root\'/" /etc/config/sing-box
        fi
        if grep -q "tun0" /etc/sing-box/config.json; then
        printf "\033[32;1mConfig /etc/sing-box/config.json already
           exists\033[0m\n"
        else
cat << 'EOF' > /etc/sing-box/config.json
{
  "log": {
    "level": "debug"
  },
  "inbounds": [
    {
      "type": "tun",
      "interface_name": "tun0",
      "domain_strategy": "ipv4_only",
      "inet4_address": "172.16.250.1/30",
      "auto_route": false,
      "strict_route": false,
      "sniff": true
    }
  ],
  "outbounds": [
    {
      "type": "$TYPE",
      "server": "$HOST",
      "server_port": $PORT,
      "method": "$METHOD",
      "password": "$PASS"
    }
  ],
  "route": {
```

```
    "auto_detect_interface": true
  }
}
EOF
        printf "\033[32;1mCreate template config in /etc/sing-box/
           config.json. Edit it manually. Official doc: https://
           sing-box.sagernet.org/configuration/outbound/\033[0m\n"
        printf "\033[32;1mOfficial doc: https://sing-box.sagernet.
           org/configuration/outbound/\033[0m\n"
        printf "\033[32;1mManual with example SS: https://cli.co/
           Badmn3K \033[0m\n"

        fi
        printf "\033[32;1mConfigure route for Sing-box\033[0m\n"
        route_vpn
    fi

    if [ "$TUNNEL" == 'wgForYoutube' ]; then
        add_internal_wg Wireguard
    fi

    if [ "$TUNNEL" == 'awgForYoutube' ]; then
        add_internal_wg AmneziaWG
    fi

    if [ "$TUNNEL" == 'awg' ]; then
        printf "\033[32;1mConfigure Amnezia WireGuard\033[0m\n"

        install_awg_packages

        route_vpn

        read -r -p "Enter the private key (from [Interface]):"$'\n'
            AWG_PRIVATE_KEY

        while true; do
            read -r -p "Enter internal IP address with subnet,
               example 192.168.100.5/24 (Address from [Interface]):
               "$'\n' AWG_IP
            if echo "$AWG_IP" | egrep -oq '^([0-9]{1,3}\.)
               {3}[0-9]{1,3}/[0-9]+$'; then
                break
```

```
        else
            echo "This IP is not valid. Please repeat"
        fi
    done

    read -r -p "Enter Jc value (from [Interface]):"$'\n' AWG_JC
    read -r -p "Enter Jmin value (from [Interface]):"$'\n'
       AWG_JMIN
    read -r -p "Enter Jmax value (from [Interface]):"$'\n'
       AWG_JMAX
    read -r -p "Enter S1 value (from [Interface]):"$'\n' AWG_S1
    read -r -p "Enter S2 value (from [Interface]):"$'\n' AWG_S2
    read -r -p "Enter H1 value (from [Interface]):"$'\n' AWG_H1
    read -r -p "Enter H2 value (from [Interface]):"$'\n' AWG_H2
    read -r -p "Enter H3 value (from [Interface]):"$'\n' AWG_H3
    read -r -p "Enter H4 value (from [Interface]):"$'\n' AWG_H4

    read -r -p "Enter the public key (from [Peer]):"$'\n'
       AWG_PUBLIC_KEY
    read -r -p "If use PresharedKey, Enter this (from [Peer]).
       If your don't use leave blank:"$'\n' AWG_PRESHARED_KEY
    read -r -p "Enter Endpoint host without port (Domain or IP)
        (from [Peer]):"$'\n' AWG_ENDPOINT

    read -r -p "Enter Endpoint host port (from [Peer]) [51820]:
       "$'\n' AWG_ENDPOINT_PORT
    AWG_ENDPOINT_PORT=${AWG_ENDPOINT_PORT:-51820}
    if [ "$AWG_ENDPOINT_PORT" = '51820' ]; then
        echo $AWG_ENDPOINT_PORT
    fi

    uci set network.awg0=interface
    uci set network.awg0.proto='amneziawg'
    uci set network.awg0.private_key=$AWG_PRIVATE_KEY
    uci set network.awg0.listen_port='51820'
    uci set network.awg0.addresses=$AWG_IP

    uci set network.awg0.awg_jc=$AWG_JC
    uci set network.awg0.awg_jmin=$AWG_JMIN
    uci set network.awg0.awg_jmax=$AWG_JMAX
    uci set network.awg0.awg_s1=$AWG_S1
    uci set network.awg0.awg_s2=$AWG_S2
```

```
        uci set network.awg0.awg_h1=$AWG_H1
        uci set network.awg0.awg_h2=$AWG_H2
        uci set network.awg0.awg_h3=$AWG_H3
        uci set network.awg0.awg_h4=$AWG_H4

        if ! uci show network | grep -q amneziawg_awg0; then
            uci add network amneziawg_awg0
        fi

        uci set network.@amneziawg_awg0[0]=amneziawg_awg0
        uci set network.@amneziawg_awg0[0].name='awg0_client'
        uci set network.@amneziawg_awg0[0].public_key=
           $AWG_PUBLIC_KEY
        uci set network.@amneziawg_awg0[0].preshared_key=
           $AWG_PRESHARED_KEY
        uci set network.@amneziawg_awg0[0].route_allowed_ips='0'
        uci set network.@amneziawg_awg0[0].persistent_keepalive='25
           '
        uci set network.@amneziawg_awg0[0].endpoint_host=
           $AWG_ENDPOINT
        uci set network.@amneziawg_awg0[0].allowed_ips='0.0.0.0/0'
        uci set network.@amneziawg_awg0[0].endpoint_port=
           $AWG_ENDPOINT_PORT
        uci commit
    fi

}

dnsmasqfull() {
    if opkg list-installed | grep -q dnsmasq-full; then
        printf "\033[32;1mdnsmasq-full already installed\033[0m\n"
    else
        printf "\033[32;1mInstalled dnsmasq-full\033[0m\n"
        cd /tmp/ && opkg download dnsmasq-full
        opkg remove dnsmasq && opkg install dnsmasq-full --cache /
           tmp/

        [ -f /etc/config/dhcp-opkg ] && cp /etc/config/dhcp /etc/
           config/dhcp-old && mv /etc/config/dhcp-opkg /etc/config/
           dhcp
fi
}
```

```
remove_forwarding() {
    if [ ! -z "$forward_id" ]; then
        while uci -q delete firewall.@forwarding[$forward_id]; do
           :; done
    fi
}

add_zone() {
    if  [ "$TUNNEL" == 0 ]; then
        printf "\033[32;1mZone setting skipped\033[0m\n"
    elif uci show firewall | grep -q "@zone.*name='$TUNNEL'"; then
        printf "\033[32;1mZone already exist\033[0m\n"
    else
        printf "\033[32;1mCreate zone\033[0m\n"

        # Delete exists zone
        zone_tun_id=$(uci show firewall | grep -E '@zone.*tun0' |
           awk -F '[][{}]' '{print $2}' | head -n 1)
        if [ "$zone_tun_id" == 0 ] || [ "$zone_tun_id" == 1 ]; then
            printf "\033[32;1mtun0 zone has an identifier of 0 or
               1. That's not ok. Fix your firewall. lan and wan
               zones should have identifiers 0 and 1. \033[0m\n"
            exit 1
        fi
        if [ ! -z "$zone_tun_id" ]; then
            while uci -q delete firewall.@zone[$zone_tun_id]; do :;
                done
        fi

        zone_wg_id=$(uci show firewall | grep -E '@zone.*wg0' | awk
            -F '[][{}]' '{print $2}' | head -n 1)
        if [ "$zone_wg_id" == 0 ] || [ "$zone_wg_id" == 1 ]; then
            printf "\033[32;1mwg0 zone has an identifier of 0 or 1.
                That's not ok. Fix your firewall. lan and wan zones
                should have identifiers 0 and 1. \033[0m\n"
            exit 1
        fi
        if [ ! -z "$zone_wg_id" ]; then
            while uci -q delete firewall.@zone[$zone_wg_id]; do :;
               done
        fi
```

```
    zone_awg_id=$(uci show firewall | grep -E '@zone.*awg0' |
       awk -F '[][{}]' '{print $2}' | head -n 1)
    if [ "$zone_awg_id" == 0 ] || [ "$zone_awg_id" == 1 ]; then
        printf "\033[32;1mawg0 zone has an identifier of 0 or
           1. That's not ok. Fix your firewall. lan and wan
           zones should have identifiers 0 and 1. \033[0m\n"
        exit 1
    fi
    if [ ! -z "$zone_awg_id" ]; then
        while uci -q delete firewall.@zone[$zone_awg_id]; do :;
            done
    fi

    uci add firewall zone
    uci set firewall.@zone[-1].name="$TUNNEL"
    if [ "$TUNNEL" == wg ]; then
        uci set firewall.@zone[-1].network='wg0'
    elif [ "$TUNNEL" == awg ]; then
        uci set firewall.@zone[-1].network='awg0'
    elif [ "$TUNNEL" == singbox ] || [ "$TUNNEL" == ovpn ] || [
        "$TUNNEL" == tun2socks ]; then
        uci set firewall.@zone[-1].device='tun0'
    fi
    if [ "$TUNNEL" == wg ] || [ "$TUNNEL" == awg ] || [ "
       $TUNNEL" == ovpn ] || [ "$TUNNEL" == tun2socks ]; then
        uci set firewall.@zone[-1].forward='REJECT'
        uci set firewall.@zone[-1].output='ACCEPT'
        uci set firewall.@zone[-1].input='REJECT'
    elif [ "$TUNNEL" == singbox ]; then
        uci set firewall.@zone[-1].forward='ACCEPT'
        uci set firewall.@zone[-1].output='ACCEPT'
        uci set firewall.@zone[-1].input='ACCEPT'
    fi
    uci set firewall.@zone[-1].masq='1'
    uci set firewall.@zone[-1].mtu_fix='1'
    uci set firewall.@zone[-1].family='ipv4'
    uci commit firewall
fi

if [ "$TUNNEL" == 0 ]; then
    printf "\033[32;1mForwarding setting skipped\033[0m\n"
```

```
elif uci show firewall | grep -q "@forwarding.*name='$TUNNEL-
   lan'"; then
    printf "\033[32;1mForwarding already configured\033[0m\n"
else
    printf "\033[32;1mConfigured forwarding\033[0m\n"
    # Delete exists forwarding
    if [[ $TUNNEL != "wg" ]]; then
        forward_id=$(uci show firewall | grep -E "@forwarding.*
           dest='wg'" | awk -F '[][{}]' '{print $2}' | head -n
           1)
        remove_forwarding
    fi

    if [[ $TUNNEL != "awg" ]]; then
        forward_id=$(uci show firewall | grep -E "@forwarding.*
           dest='awg'" | awk -F '[][{}]' '{print $2}' | head -n
            1)
        remove_forwarding
    fi

    if [[ $TUNNEL != "ovpn" ]]; then
        forward_id=$(uci show firewall | grep -E "@forwarding.*
           dest='ovpn'" | awk -F '[][{}]' '{print $2}' | head -
           n 1)
        remove_forwarding
    fi

    if [[ $TUNNEL != "singbox" ]]; then
        forward_id=$(uci show firewall | grep -E "@forwarding.*
           dest='singbox'" | awk -F '[][{}]' '{print $2}' |
           head -n 1)
        remove_forwarding
    fi

    if [[ $TUNNEL != "tun2socks" ]]; then
        forward_id=$(uci show firewall | grep -E "@forwarding.*
           dest='tun2socks'" | awk -F '[][{}]' '{print $2}' |
           head -n 1)
        remove_forwarding
    fi

    uci add firewall forwarding
```

```
        uci set firewall.@forwarding[-1]=forwarding
        uci set firewall.@forwarding[-1].name="$TUNNEL-lan"
        uci set firewall.@forwarding[-1].dest="$TUNNEL"
        uci set firewall.@forwarding[-1].src='lan'
        uci set firewall.@forwarding[-1].family='ipv4'
        uci commit firewall
    fi
}

show_manual() {
    if [ "$TUNNEL" == tun2socks ]; then
        printf "\033[42;1mZone for tun2socks cofigured. But you
           need to set up the tunnel yourself.\033[0m\n"
        echo "Use this manual: https://cli.co/VNZISEM"
    elif [ "$TUNNEL" == ovpn ]; then
        printf "\033[42;1mZone for OpenVPN cofigured. But you need
           to set up the tunnel yourself.\033[0m\n"
    fi
}

add_set() {
    if uci show firewall | grep -q "@ipset.*name='vpn_domains'";
       then
        printf "\033[32;1mSet already exist\033[0m\n"
    else
        printf "\033[32;1mCreate set\033[0m\n"
        uci add firewall ipset
        uci set firewall.@ipset[-1].name='vpn_domains'
        uci set firewall.@ipset[-1].match='dst_net'
        uci commit
    fi
    if uci show firewall | grep -q "@rule.*name='mark_domains'";
       then
        printf "\033[32;1mRule for set already exist\033[0m\n"
    else
        printf "\033[32;1mCreate rule set\033[0m\n"
        uci add firewall rule
        uci set firewall.@rule[-1]=rule
        uci set firewall.@rule[-1].name='mark_domains'
        uci set firewall.@rule[-1].src='lan'
        uci set firewall.@rule[-1].dest='*'
        uci set firewall.@rule[-1].proto='all'
```

```
        uci set firewall.@rule[-1].ipset='vpn_domains'
        uci set firewall.@rule[-1].set_mark='0x1'
        uci set firewall.@rule[-1].target='MARK'
        uci set firewall.@rule[-1].family='ipv4'
        uci commit
    fi
}

add_dns_resolver() {
    echo "Configure DNSCrypt2 or Stubby? It does matter if your ISP
        is spoofing DNS requests"
    DISK=$(df -m / | awk 'NR==2{ print $2 }')
    if [[ "$DISK" -lt 32 ]]; then
        printf "\033[31;1mYour router a disk have less than 32MB.
           It is not recommended to install DNSCrypt, it takes 10MB
           \033[0m\n"
    fi
    echo "Select:"
    echo "1) No [Default]"
    echo "2) DNSCrypt2 (10.7M)"
    echo "3) Stubby (36K)"

    while true; do
    read -r -p '' DNS_RESOLVER
        case $DNS_RESOLVER in

        1)
            echo "Skiped"
            break
            ;;

        2)
            DNS_RESOLVER=DNSCRYPT
            break
            ;;

        3)
            DNS_RESOLVER=STUBBY
            break
            ;;

        *)
```

```
            echo "Choose from the following options"
            ;;
        esac
    done

    if [ "$DNS_RESOLVER" == 'DNSCRYPT' ]; then
        if opkg list-installed | grep -q dnscrypt-proxy2; then
            printf "\033[32;1mDNSCrypt2 already installed\033[0m\n"
        else
            printf "\033[32;1mInstalled dnscrypt-proxy2\033[0m\n"
            opkg install dnscrypt-proxy2
            if grep -q "# server_names" /etc/dnscrypt-proxy2/
               dnscrypt-proxy.toml; then
                sed -i "s/^# server_names =.*/server_names = [\'
                   google\', \'cloudflare\', \'scaleway-fr\', \'
                   yandex\']/g" /etc/dnscrypt-proxy2/dnscrypt-proxy
                   .toml
            fi

            printf "\033[32;1mDNSCrypt restart\033[0m\n"
            service dnscrypt-proxy restart
            printf "\033[32;1mDNSCrypt needs to load the relays
               list. Please wait\033[0m\n"
            sleep 30

            if [ -f /etc/dnscrypt-proxy2/relays.md ]; then
                uci set dhcp.@dnsmasq[0].noresolv="1"
                uci -q delete dhcp.@dnsmasq[0].server
                uci add_list dhcp.@dnsmasq[0].server="127.0.0.53#53
                   "
                uci add_list dhcp.@dnsmasq[0].server='/use-
                   application-dns.net/'
                uci commit dhcp

                printf "\033[32;1mDnsmasq restart\033[0m\n"

                /etc/init.d/dnsmasq restart
            else
                printf "\033[31;1mDNSCrypt not download list on /
                   etc/dnscrypt-proxy2. Repeat install DNSCrypt by
                   script.\033[0m\n"
            fi
```

```
    fi

    fi

    if [ "$DNS_RESOLVER" == 'STUBBY' ]; then
        printf "\033[32;1mConfigure Stubby\033[0m\n"

        if opkg list-installed | grep -q stubby; then
            printf "\033[32;1mStubby already installed\033[0m\n"
        else
            printf "\033[32;1mInstalled stubby\033[0m\n"
            opkg install stubby

            printf "\033[32;1mConfigure Dnsmasq for Stubby\033[0m\n
               "
            uci set dhcp.@dnsmasq[0].noresolv="1"
            uci -q delete dhcp.@dnsmasq[0].server
            uci add_list dhcp.@dnsmasq[0].server="127.0.0.1#5453"
            uci add_list dhcp.@dnsmasq[0].server='/use-application-
               dns.net/'
            uci commit dhcp

            printf "\033[32;1mDnsmasq restart\033[0m\n"

            /etc/init.d/dnsmasq restart
        fi
    fi
}

add_packages() {
    for package in curl nano; do
        if opkg list-installed | grep -q "^$package "; then
            printf "\033[32;1m$package already installed\033[0m\n"
        else
            printf "\033[32;1mInstalling $package...\033[0m\n"
            opkg install "$package"

            if "$package" --version >/dev/null 2>&1; then
                printf "\033[32;1m$package was successfully
                   installed and available\033[0m\n"
            else
```

```
                printf "\033[31;1mError: failed to install $package
                    \033[0m\n"
                exit 1
            fi
        fi
    done
}

add_getdomains() {
    echo "Choose you country"
    echo "Select:"
    echo "1) Russia inside. You are inside Russia"
    echo "2) Russia outside. You are outside of Russia, but you
       need access to Russian resources"
    echo "4) Skip script creation"

    while true; do
    read -r -p '' COUNTRY
        case $COUNTRY in

        1)
            COUNTRY=russia_inside
            break
            ;;

        2)
            COUNTRY=russia_outside
            break
            ;;

        4)
            echo "Skiped"
            COUNTRY=0
            break
            ;;

        *)
            echo "Choose from the following options"
            ;;
        esac
    done
```

```
    if [ "$COUNTRY" == 'russia_inside' ]; then
        EOF_DOMAINS=DOMAINS=https://raw.githubusercontent.com/
           itdoginfo/allow-domains/main/Russia/inside-dnsmasq-nfset
           .lst
    elif [ "$COUNTRY" == 'russia_outside' ]; then
        EOF_DOMAINS=DOMAINS=https://raw.githubusercontent.com/
           itdoginfo/allow-domains/main/Russia/outside-dnsmasq-
           nfset.lst
    fi

    if [ "$COUNTRY" != '0' ]; then
        printf "\033[32;1mCreate script /etc/init.d/getdomains
           \033[0m\n"

cat << EOF > /etc/init.d/getdomains
#!/bin/sh /etc/rc.common

START=99

start () {
    $EOF_DOMAINS
EOF
cat << 'EOF' >> /etc/init.d/getdomains
    count=0
    while true; do
        if curl -m 3 github.com; then
            curl -f $DOMAINS --output /tmp/dnsmasq.d/domains.lst
            break
        else
            echo "GitHub is not available. Check the internet
               availability [$count]"
            count=$((count+1))
        fi
    done

    if dnsmasq --conf-file=/tmp/dnsmasq.d/domains.lst --test 2>&1 |
        grep -q "syntax check OK"; then
        /etc/init.d/dnsmasq restart
    fi
}
EOF
```

```
        chmod +x /etc/init.d/getdomains
        /etc/init.d/getdomains enable

        if crontab -l | grep -q /etc/init.d/getdomains; then
            printf "\033[32;1mCrontab already configured\033[0m\n"

        else
            crontab -l | { cat; echo "0 */8 * * * /etc/init.d/
               getdomains start"; } | crontab -
            printf "\033[32;1mIgnore this error. This is normal for
                a new installation\033[0m\n"
            /etc/init.d/cron restart
        fi

        printf "\033[32;1mStart script\033[0m\n"

        /etc/init.d/getdomains start
    fi
}

add_internal_wg() {
    PROTOCOL_NAME=$1
    printf "\033[32;1mConfigure ${PROTOCOL_NAME}\033[0m\n"
    if [ "$PROTOCOL_NAME" = 'Wireguard' ]; then
        INTERFACE_NAME="wg1"
        CONFIG_NAME="wireguard_wg1"
        PROTO="wireguard"
        ZONE_NAME="wg_internal"

        if opkg list-installed | grep -q wireguard-tools; then
            echo "Wireguard already installed"
        else
            echo "Installed wg..."
            opkg install wireguard-tools
        fi
    fi

    if [ "$PROTOCOL_NAME" = 'AmneziaWG' ]; then
        INTERFACE_NAME="awg1"
        CONFIG_NAME="amneziawg_awg1"
        PROTO="amneziawg"
        ZONE_NAME="awg_internal"
```

```
    install_awg_packages
fi

read -r -p "Enter the private key (from [Interface]):"$'\n'
   WG_PRIVATE_KEY_INT

while true; do
    read -r -p "Enter internal IP address with subnet, example
       192.168.100.5/24 (from [Interface]):"$'\n' WG_IP
    if echo "$WG_IP" | egrep -oq '^([0-9]{1,3}\.)
       {3}[0-9]{1,3}/[0-9]+$'; then
        break
    else
        echo "This IP is not valid. Please repeat"
    fi
done

read -r -p "Enter the public key (from [Peer]):"$'\n'
   WG_PUBLIC_KEY_INT
read -r -p "If use PresharedKey, Enter this (from [Peer]). If
   your don't use leave blank:"$'\n' WG_PRESHARED_KEY_INT
read -r -p "Enter Endpoint host without port (Domain or IP) (
   from [Peer]):"$'\n' WG_ENDPOINT_INT

read -r -p "Enter Endpoint host port (from [Peer]) [51820]:"$'\
   n' WG_ENDPOINT_PORT_INT
WG_ENDPOINT_PORT_INT=${WG_ENDPOINT_PORT_INT:-51820}
if [ "$WG_ENDPOINT_PORT_INT" = '51820' ]; then
    echo $WG_ENDPOINT_PORT_INT
fi

if [ "$PROTOCOL_NAME" = 'AmneziaWG' ]; then
    read -r -p "Enter Jc value (from [Interface]):"$'\n' AWG_JC
    read -r -p "Enter Jmin value (from [Interface]):"$'\n'
       AWG_JMIN
    read -r -p "Enter Jmax value (from [Interface]):"$'\n'
       AWG_JMAX
    read -r -p "Enter S1 value (from [Interface]):"$'\n' AWG_S1
    read -r -p "Enter S2 value (from [Interface]):"$'\n' AWG_S2
    read -r -p "Enter H1 value (from [Interface]):"$'\n' AWG_H1
    read -r -p "Enter H2 value (from [Interface]):"$'\n' AWG_H2
```

```
        read -r -p "Enter H3 value (from [Interface]):"$'\n' AWG_H3
        read -r -p "Enter H4 value (from [Interface]):"$'\n' AWG_H4
    fi

    uci set network.${INTERFACE_NAME}=interface
    uci set network.${INTERFACE_NAME}.proto=$PROTO
    uci set network.${INTERFACE_NAME}.private_key=
       $WG_PRIVATE_KEY_INT
    uci set network.${INTERFACE_NAME}.listen_port='51821'
    uci set network.${INTERFACE_NAME}.addresses=$WG_IP

    if [ "$PROTOCOL_NAME" = 'AmneziaWG' ]; then
        uci set network.${INTERFACE_NAME}.awg_jc=$AWG_JC
        uci set network.${INTERFACE_NAME}.awg_jmin=$AWG_JMIN
        uci set network.${INTERFACE_NAME}.awg_jmax=$AWG_JMAX
        uci set network.${INTERFACE_NAME}.awg_s1=$AWG_S1
        uci set network.${INTERFACE_NAME}.awg_s2=$AWG_S2
        uci set network.${INTERFACE_NAME}.awg_h1=$AWG_H1
        uci set network.${INTERFACE_NAME}.awg_h2=$AWG_H2
        uci set network.${INTERFACE_NAME}.awg_h3=$AWG_H3
        uci set network.${INTERFACE_NAME}.awg_h4=$AWG_H4
    fi

    if ! uci show network | grep -q ${CONFIG_NAME}; then
        uci add network ${CONFIG_NAME}
    fi

    uci set network.@${CONFIG_NAME}[0]=$CONFIG_NAME
    uci set network.@${CONFIG_NAME}[0].name="${INTERFACE_NAME}
       _client"
    uci set network.@${CONFIG_NAME}[0].public_key=
       $WG_PUBLIC_KEY_INT
    uci set network.@${CONFIG_NAME}[0].preshared_key=
       $WG_PRESHARED_KEY_INT
    uci set network.@${CONFIG_NAME}[0].route_allowed_ips='0'
    uci set network.@${CONFIG_NAME}[0].persistent_keepalive='25'
    uci set network.@${CONFIG_NAME}[0].endpoint_host=
       $WG_ENDPOINT_INT
    uci set network.@${CONFIG_NAME}[0].allowed_ips='0.0.0.0/0'
    uci set network.@${CONFIG_NAME}[0].endpoint_port=
       $WG_ENDPOINT_PORT_INT
    uci commit network
```

```
grep -q "110 vpninternal" /etc/iproute2/rt_tables || echo '110
   vpninternal' >> /etc/iproute2/rt_tables

if ! uci show network | grep -q mark0x2; then
    printf "\033[32;1mConfigure mark rule\033[0m\n"
    uci add network rule
    uci set network.@rule[-1].name='mark0x2'
    uci set network.@rule[-1].mark='0x2'
    uci set network.@rule[-1].priority='110'
    uci set network.@rule[-1].lookup='vpninternal'
    uci commit
fi

if ! uci show network | grep -q vpn_route_internal; then
    printf "\033[32;1mAdd route\033[0m\n"
    uci set network.vpn_route_internal=route
    uci set network.vpn_route_internal.name='vpninternal'
    uci set network.vpn_route_internal.interface=
       $INTERFACE_NAME
    uci set network.vpn_route_internal.table='vpninternal'
    uci set network.vpn_route_internal.target='0.0.0.0/0'
    uci commit network
fi

if ! uci show firewall | grep -q "@zone.*name='${ZONE_NAME}'";
   then
    printf "\033[32;1mZone Create\033[0m\n"
    uci add firewall zone
    uci set firewall.@zone[-1].name=$ZONE_NAME
    uci set firewall.@zone[-1].network=$INTERFACE_NAME
    uci set firewall.@zone[-1].forward='REJECT'
    uci set firewall.@zone[-1].output='ACCEPT'
    uci set firewall.@zone[-1].input='REJECT'
    uci set firewall.@zone[-1].masq='1'
    uci set firewall.@zone[-1].mtu_fix='1'
    uci set firewall.@zone[-1].family='ipv4'
    uci commit firewall
fi

if ! uci show firewall | grep -q "@forwarding.*name='${
   ZONE_NAME}'"; then
```

```
        printf "\033[32;1mConfigured forwarding\033[0m\n"
        uci add firewall forwarding
        uci set firewall.@forwarding[-1]=forwarding
        uci set firewall.@forwarding[-1].name="${ZONE_NAME}-lan"
        uci set firewall.@forwarding[-1].dest=${ZONE_NAME}
        uci set firewall.@forwarding[-1].src='lan'
        uci set firewall.@forwarding[-1].family='ipv4'
        uci commit firewall
    fi

    if uci show firewall | grep -q "@ipset.*name='
       vpn_domains_internal'"; then
        printf "\033[32;1mSet already exist\033[0m\n"
    else
        printf "\033[32;1mCreate set\033[0m\n"
        uci add firewall ipset
        uci set firewall.@ipset[-1].name='vpn_domains_internal'
        uci set firewall.@ipset[-1].match='dst_net'
        uci commit firewall
    fi

    if uci show firewall | grep -q "@rule.*name='
       mark_domains_intenal'"; then
        printf "\033[32;1mRule for set already exist\033[0m\n"
    else
        printf "\033[32;1mCreate rule set\033[0m\n"
        uci add firewall rule
        uci set firewall.@rule[-1]=rule
        uci set firewall.@rule[-1].name='mark_domains_intenal'
        uci set firewall.@rule[-1].src='lan'
        uci set firewall.@rule[-1].dest='*'
        uci set firewall.@rule[-1].proto='all'
        uci set firewall.@rule[-1].ipset='vpn_domains_internal'
        uci set firewall.@rule[-1].set_mark='0x2'
        uci set firewall.@rule[-1].target='MARK'
        uci set firewall.@rule[-1].family='ipv4'
        uci commit firewall
    fi

    if uci show dhcp | grep -q "@ipset.*name='vpn_domains_internal'
       "; then
```

```
        printf "\033[32;1mDomain on vpn_domains_internal already
            exist\033[0m\n"
    else
        printf "\033[32;1mCreate domain for vpn_domains_internal
            \033[0m\n"
        uci add dhcp ipset
        uci add_list dhcp.@ipset[-1].name='vpn_domains_internal'
        uci add_list dhcp.@ipset[-1].domain='youtube.com'
        uci add_list dhcp.@ipset[-1].domain='googlevideo.com'
        uci add_list dhcp.@ipset[-1].domain='youtubekids.com'
        uci add_list dhcp.@ipset[-1].domain='googleapis.com'
        uci add_list dhcp.@ipset[-1].domain='ytimg.com'
        uci add_list dhcp.@ipset[-1].domain='ggpht.com'
        uci commit dhcp
    fi

    sed -i "/done/a sed -i '/youtube.com\\\|ytimg.com\\\|ggpht.com
        \\\|googlevideo.com\\\|googleapis.com\\\|youtubekids.com/d'
        /tmp/dnsmasq.d/domains.lst" "/etc/init.d/getdomains"

    service dnsmasq restart
    service network restart

    exit 0
}

install_awg_packages() {
    PKGARCH=$(opkg print-architecture | awk 'BEGIN {max=0} {if ($3
        > max) {max = $3; arch = $2}} END {print arch}')

    TARGET=$(ubus call system board | jsonfilter -e '@.release.
        target' | cut -d '/' -f 1)
    SUBTARGET=$(ubus call system board | jsonfilter -e '@.release.
        target' | cut -d '/' -f 2)
    VERSION=$(ubus call system board | jsonfilter -e '@.release.
        version')
    PKGPOSTFIX="_v${VERSION}_${PKGARCH}_${TARGET}_${SUBTARGET}.ipk"
    BASE_URL="https://github.com/Slava-Shchipunov/awg-openwrt/
        releases/download/"

    AWG_DIR="/tmp/amneziawg"
    mkdir -p "$AWG_DIR"
```

```
if opkg list-installed | grep -q amneziawg-tools; then
    echo "amneziawg-tools already installed"
else
    AMNEZIAWG_TOOLS_FILENAME="amneziawg-tools${PKGPOSTFIX}"
    DOWNLOAD_URL="${BASE_URL}v${VERSION}/${
       AMNEZIAWG_TOOLS_FILENAME}"
    curl -L -o "$AWG_DIR/$AMNEZIAWG_TOOLS_FILENAME" "
       $DOWNLOAD_URL"

    if [ $? -eq 0 ]; then
        echo "amneziawg-tools file downloaded successfully"
    else
        echo "Error downloading amneziawg-tools. Please,
           install amneziawg-tools manually and run the script
           again"
        exit 1
    fi

    opkg install "$AWG_DIR/$AMNEZIAWG_TOOLS_FILENAME"

    if [ $? -eq 0 ]; then
        echo "amneziawg-tools file downloaded successfully"
    else
        echo "Error installing amneziawg-tools. Please, install
            amneziawg-tools manually and run the script again"
        exit 1
    fi
fi

if opkg list-installed | grep -q kmod-amneziawg; then
    echo "kmod-amneziawg already installed"
else
    KMOD_AMNEZIAWG_FILENAME="kmod-amneziawg${PKGPOSTFIX}"
    DOWNLOAD_URL="${BASE_URL}v${VERSION}/${
       KMOD_AMNEZIAWG_FILENAME}"
    curl -L -o "$AWG_DIR/$KMOD_AMNEZIAWG_FILENAME" "
       $DOWNLOAD_URL"

    if [ $? -eq 0 ]; then
        echo "kmod-amneziawg file downloaded successfully"
    else
```

```
            echo "Error downloading kmod-amneziawg. Please, install
                kmod-amneziawg manually and run the script again"
            exit 1
        fi

        opkg install "$AWG_DIR/$KMOD_AMNEZIAWG_FILENAME"

        if [ $? -eq 0 ]; then
            echo "kmod-amneziawg file downloaded successfully"
        else
            echo "Error installing kmod-amneziawg. Please, install
               kmod-amneziawg manually and run the script again"
            exit 1
        fi
    fi

    if opkg list-installed | grep -q luci-app-amneziawg; then
        echo "luci-app-amneziawg already installed"
    else
        LUCI_APP_AMNEZIAWG_FILENAME="luci-app-amneziawg${PKGPOSTFIX
           }"
        DOWNLOAD_URL="${BASE_URL}v${VERSION}/${
           LUCI_APP_AMNEZIAWG_FILENAME}"
        curl -L -o "$AWG_DIR/$LUCI_APP_AMNEZIAWG_FILENAME" "
           $DOWNLOAD_URL"

        if [ $? -eq 0 ]; then
            echo "luci-app-amneziawg file downloaded successfully"
        else
            echo "Error downloading luci-app-amneziawg. Please,
               install luci-app-amneziawg manually and run the
               script again"
            exit 1
        fi

        opkg install "$AWG_DIR/$LUCI_APP_AMNEZIAWG_FILENAME"

        if [ $? -eq 0 ]; then
            echo "luci-app-amneziawg file downloaded successfully"
        else
```

```
            echo "Error installing luci-app-amneziawg. Please,
               install luci-app-amneziawg manually and run the
               script again"
            exit 1
        fi
    fi

    rm -rf "$AWG_DIR"
}

# System Details
MODEL=$(cat /tmp/sysinfo/model)
source /etc/os-release
printf "\033[34;1mModel: $MODEL\033[0m\n"
printf "\033[34;1mVersion: $OPENWRT_RELEASE\033[0m\n"

VERSION_ID=$(echo $VERSION | awk -F. '{print $1}')

if [ "$VERSION_ID" -ne 23 ]; then
    printf "\033[31;1mScript only support OpenWrt 23.05\033[0m\n"
    exit 1
fi

printf "\033[31;1mAll actions performed here cannot be rolled back
   automatically.\033[0m\n"

check_repo

add_packages

add_tunnel

add_mark

add_zone

show_manual

add_set

dnsmasqfull
```

```
add_dns_resolver

add_getdomains

printf "\033[32;1mRestart network\033[0m\n"
/etc/init.d/network restart

printf "\033[32;1mDone\033[0m\n"
```