Ikev2 ipsec/wireguard/3proxy/clam av updater 4 in 1

В примере используется следующая конфигурация:

Внешний IP: **91.149.232.254**

Внешний интерфейс: **eth0**

ikev2 сеть: **10.10.30.0/24**

Подсеть с которой разрешен доступ к 3proxy: **94.229.240.0/20**

Доступ к админке WireGuard разрешен с ip: **94.229.246.136**

Далее по тексту меняем их на свои

1) Обновляем дистр и ставим нужные пакеты

Для Ubuntu

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo   "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Для Debian

```
curl -fsSL https://download.docker.com/linux/debian/gpg |gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
```

```
tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Для Ubuntu\Debian

```
apt update
apt upgrade
reboot
apt install ufw fail2ban ca-certificates gnupg lsb-release build-essential docker-ce docker-ce-cli containerd.io mc
zip strongswan libstrongswan-standard-plugins strongswan-pki nginx python3-pip mc
/etc/init.d/nginx start
```

Устанавливаем и настраиваем ClamAV:

```
pip3 install cvdupdate
cvd config set --dbdir /var/www/html
cvd update
crontab -e
В крон добавляем задачу:
30 */4 * * * /bin/sh -c "/usr/local/bin/cvd update &> /dev/null"
```

2) Ставим 3proxy для обхода блокировок

```
wget https://github.com/z3APA3A/3proxy/archive/0.9.3.tar.gz && tar xzf 0.9.3.tar.gz && cd 3proxy-* && make -f
Makefile.Linux
adduser --system --disabled-login --no-create-home --group proxy3
mkdir -p /var/log/3proxy && mkdir /etc/3proxy
cp bin/3proxy /usr/bin/
chown proxy3:proxy3 -R /etc/3proxy
chown proxy3:proxy3 /usr/bin/3proxy
chown proxy3:proxy3 /var/log/3proxy
id proxy3
nano /etc/3proxy/3proxy.cfg
```

Конфиг /etc/3proxy/3proxy.cfg заменить uid/gid полученные из id proxy3:

```
setgid 115
setuid 110


nserver 1.1.1.1
```

```
nserver 8.8.8.8

nscache 65536
timeouts 1 5 30 60 180 1800 15 60

external 91.149.232.254
internal 91.149.232.254

daemon

log /var/log/3proxy/3proxy.log D
logformat "- +_L%t.%. %N.%p %E %U %C:%c %R:%r %O %I %h %T"
rotate 30
auth none
#auth strong
#users sega:CL:pass

allow * * * 80-88,8080-8088 HTTP
allow * * * 443,8443 HTTPS
socks -p8083
proxy -n
```

Добавляем в автозагрузку:

`nano /etc/systemd/system/3proxy.service`

```
[Unit]
Description=3proxy Proxy Server

[Service]
Type=simple
ExecStart=/usr/bin/3proxy /etc/3proxy/3proxy.cfg
ExecStop=/bin/kill `/usr/bin/pgrep -u proxy3`
RemainAfterExit=yes
Restart=on-failure

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

Применяем юнит:

```
systemctl daemon-reload
systemctl enable 3proxy
```

3) Подготавливаем ufw

3.1) Для работы с докером:

`nano /etc/ufw/after.rules`

Добавить в конец файла

```
# BEGIN UFW AND DOCKER
*filter
:ufw-user-forward - [0:0]
:ufw-docker-logging-deny - [0:0]
:DOCKER-USER - [0:0]
-A DOCKER-USER -j ufw-user-forward

-A DOCKER-USER -j RETURN -s 10.0.0.0/8
-A DOCKER-USER -j RETURN -s 172.16.0.0/12
-A DOCKER-USER -j RETURN -s 192.168.0.0/16

-A DOCKER-USER -p udp -m udp --sport 53 --dport 1024:65535 -j RETURN

-A DOCKER-USER -j ufw-docker-logging-deny -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -d 192.168.0.0/16
-A DOCKER-USER -j ufw-docker-logging-deny -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -d 10.0.0.0/8
-A DOCKER-USER -j ufw-docker-logging-deny -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -d 172.16.0.0/12
-A DOCKER-USER -j ufw-docker-logging-deny -p udp -m udp --dport 0:32767 -d 192.168.0.0/16
-A DOCKER-USER -j ufw-docker-logging-deny -p udp -m udp --dport 0:32767 -d 10.0.0.0/8
-A DOCKER-USER -j ufw-docker-logging-deny -p udp -m udp --dport 0:32767 -d 172.16.0.0/12

-A DOCKER-USER -j RETURN

-A ufw-docker-logging-deny -m limit --limit 3/min --limit-burst 10 -j LOG --log-prefix "[UFW DOCKER BLOCK] "
-A ufw-docker-logging-deny -j DROP

COMMIT
# END UFW AND DOCKER
```

3.2) Для работы с ikev2:

`nano /etc/ufw/before.rules`

В начало файла

```
*nat
-A POSTROUTING -s 10.10.30.0/24 -o eth0 -m policy --pol ipsec --dir out -j ACCEPT
-A POSTROUTING -s 10.10.30.0/24 -o eth0 -j MASQUERADE
COMMIT

*mangle
-A FORWARD --match policy --pol ipsec --dir in -s 10.10.30.0/24 -o eth0 -p tcp -m tcp --tcp-flags SYN,RST SYN -m tcpmss --mss 1361:1536 -j TCPMSS --set-mss 1360
COMMIT
```

3.3) Разрешающие правила ufw:

```
ufw allow 22
#HTTP
ufw allow 80
#IPSEC
ufw allow 500,4500/udp
#Доступ к 3proxy и нужных ip/подсетей
ufw allow from 94.229.240.0/20 to any port 8083
ufw allow from 94.229.240.0/20 to any port 3128
#Админка WireGuard
ufw route allow from 94.229.246.136 to any port 51821
#WireGuard порт
ufw route allow from any to any port 51820
#доступ из сетей docker во внешний мир
ufw route allow from 172.16.0.0/12 to any
#Доступ из ipsec сети
ufw route allow from 10.10.30.0/24 to any
#Включаем фаервол
ufw enable
```

4) Настройка Ikev2

`cd /etc/ipsec.d`

#Генерируем ключи:
#CA

```
ipsec pki --gen --type rsa --size 4096 --outform pem > private/ca.pem
ipsec pki --self --ca --lifetime 3650 --in private/ca.pem --type rsa --digest sha256 --dn "CN=91.149.232.254" --outform pem > cacerts/ca.pem
```

Серверный

```
ipsec pki --gen --type rsa --size 4096 --outform pem > private/debian.pem
ipsec pki --pub --in private/debian.pem --type rsa | ipsec pki --issue --lifetime 3650 --digest sha256 --cacert cacerts/ca.pem --cakey private/ca.pem --dn "CN=91.149.232.254" --san 91.149.232.254 --flag  serverAuth --outform pem > certs/debian.pem
```

Клиенты(client1,client2,client3 и т.д.):

```
ipsec pki --gen --type rsa --size 4096 --outform pem > private/client1.pem
ipsec pki --pub --in private/client1.pem --type rsa | ipsec pki --issue --lifetime 3650 --digest sha256 --cacert cacerts/ca.pem --cakey private/ca.pem --dn "CN=client1" --san client1 --flag clientAuth --outform pem > certs/client1.pem
```

Генерируем pfx сертификат

```
openssl pkcs12 -export -out client1.pfx -inkey private/client1.pem -in certs/client1.pem -certfile cacerts/ca.pem
zip client1.zip client1.pfx && mv client1.zip /var/www/html
```

Правим конфиги:

`nano /etc/ipsec.conf`

```
config setup
        uniqueids=never
        charondebug="ike 2, knl 2, cfg 2, net 2, esp 2, dmn 2,  mgr 2"

conn %default
        keyexchange=ikev2
        ike=aes256-aes128-sha256-sha1-modp3072-modp2048-modp1024
        esp=aes256-aes128-sha256-sha1-modp3072-modp2048-modp1024
        #ike=aes128gcm16-sha2_256-prfsha256-ecp256!
        #esp=aes128gcm16-sha2_256-ecp256!
        fragmentation=yes
        rekey=no
        compress=yes
        dpdaction=clear
```

```
        left=%any
        leftauth=pubkey
        leftsourceip=91.149.232.254
        leftid=91.149.232.254
        leftcert=debian.pem
        leftsendcert=always
        leftsubnet=0.0.0.0/0
        right=%any
        rightauth=pubkey
        rightsourceip=10.10.30.0/24
        rightdns=8.8.8.8,8.8.4.4

conn ikev2-pubkey
        auto=add
```

`nano /etc/ipsec.secrets`

```
: RSA debian.pem
```

`ipsec restart`

WireGuard

`mkdir /opt/wireguard/ && cd /opt/wireguard/ && nano docker-compose.yml`

```
version: "3.8"
services:
  wg-easy:
    environment:
      # ⚠ Required:
      # Change this to your host's public address
      - WG_HOST=91.149.232.254

      # Optional:
      - PASSWORD=PASS
      # - WG_PORT=51820
      #- WG_DEFAULT_ADDRESS=10.99.99.1
      - WG_DEFAULT_DNS=1.1.1.1
      # - WG_MTU=1420
```

```
  # - WG_ALLOWED_IPS=192.168.15.0/24, 10.0.1.0/24
  # - WG_PRE_UP=echo "Pre Up" > /etc/wireguard/pre-up.txt
  # - WG_POST_UP=echo "Post Up" > /etc/wireguard/post-up.txt
  # - WG_PRE_DOWN=echo "Pre Down" > /etc/wireguard/pre-down.txt
  # - WG_POST_DOWN=echo "Post Down" > /etc/wireguard/post-down.txt

image: weejewel/wg-easy
container_name: wg-easy
volumes:
  - .:/etc/wireguard
ports:
  - "51820:51820/udp"
  - "51821:51821/tcp"
restart: unless-stopped
cap_add:
  - NET_ADMIN
  - SYS_MODULE
sysctls:
  - net.ipv4.ip_forward=1
  - net.ipv4.conf.all.src_valid_mark=1
```

`docker compose up -d`

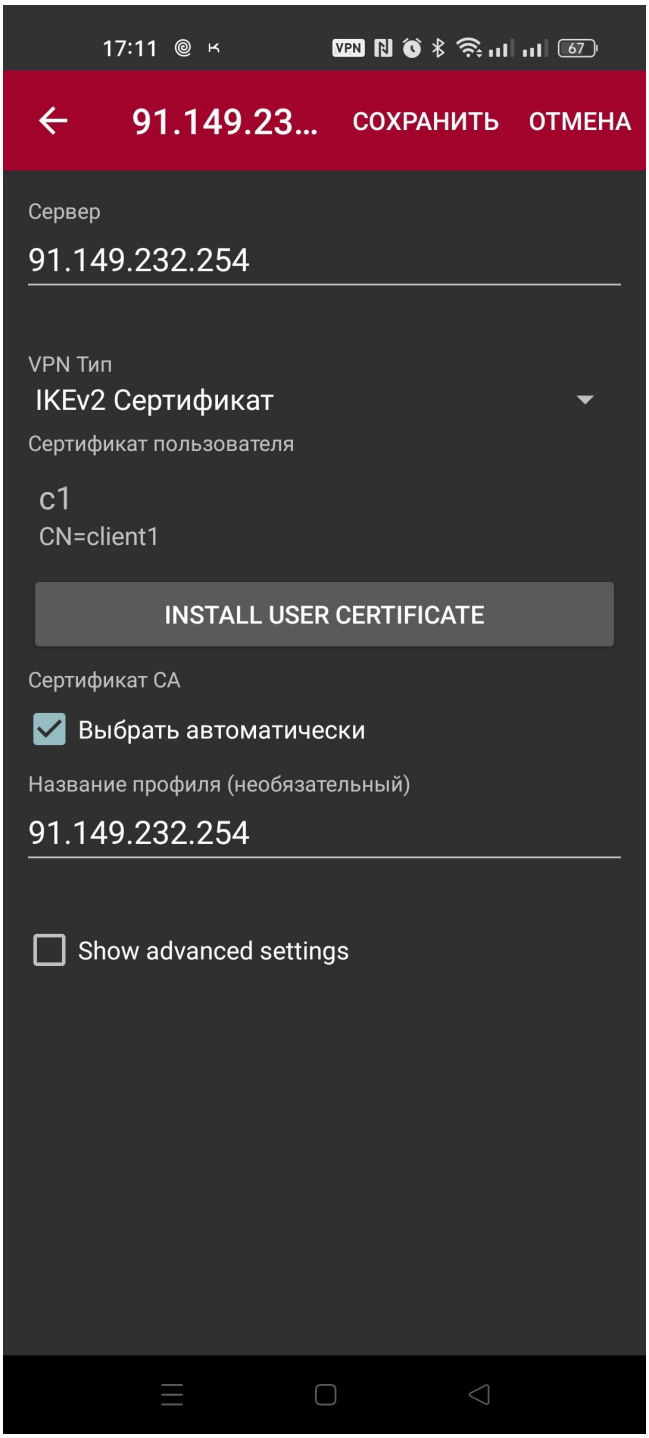
Ну и на последок

`reboot`

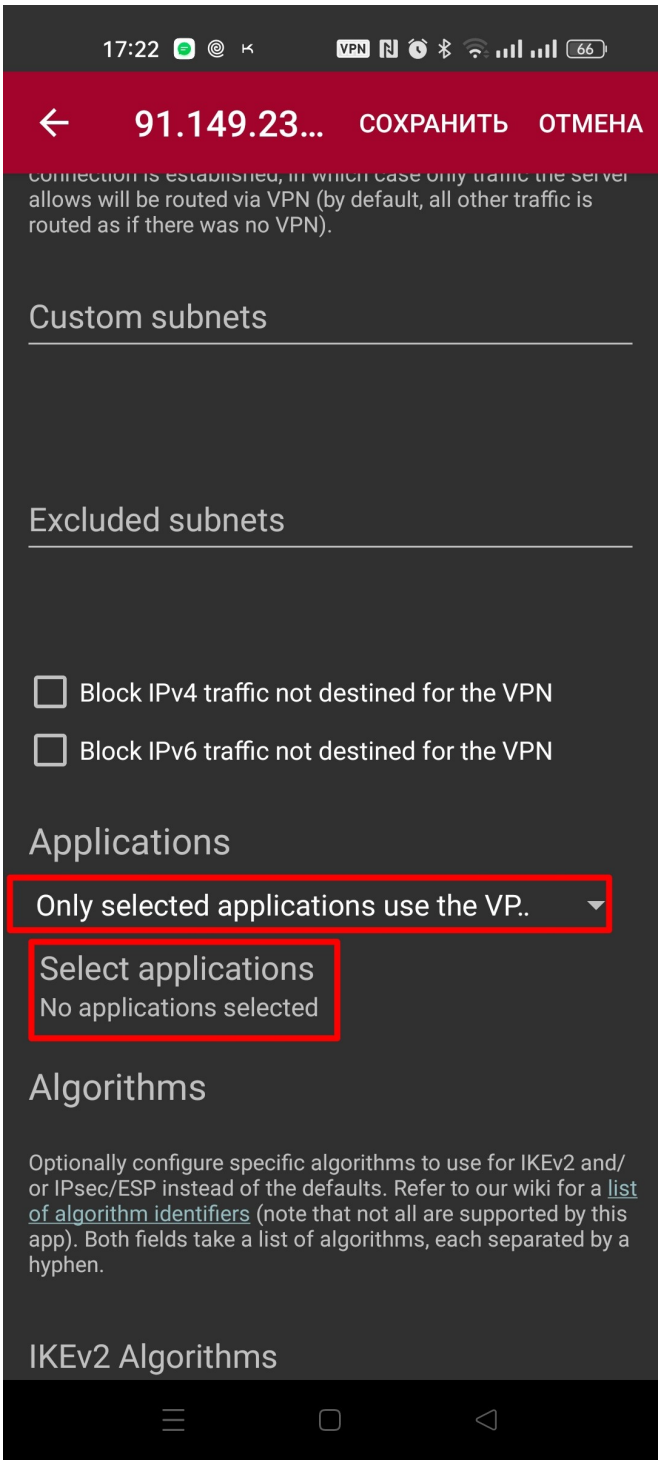
Для ikev2 на андройде используем strongSwan

1) скачиваем созданный сертификат на телефон http://91.149.232.254/client1.zip и распаковываем его.

2) Запускаем приложение и создаем новый профиль, указываем сервер и тип vpn: IKEv2 Сертификат и жмем install user sertificate и указываем на сертификат который мы скачали и распаковали в 1 пункте.



2.1) Если нужно что бы через впн работали только определенные приложения то ставим галочку на Show advanced settings и листаем до Applications, Жмем по All applications use the vpn и выбираем only selected applications use the vpn. Ниже появится пункт Select applications, нажимаем его и выбираем нужные нам приложения



3) Сохраняем настройки и подключаемся к vpn

3proxy используется совместно с плагином для браузера

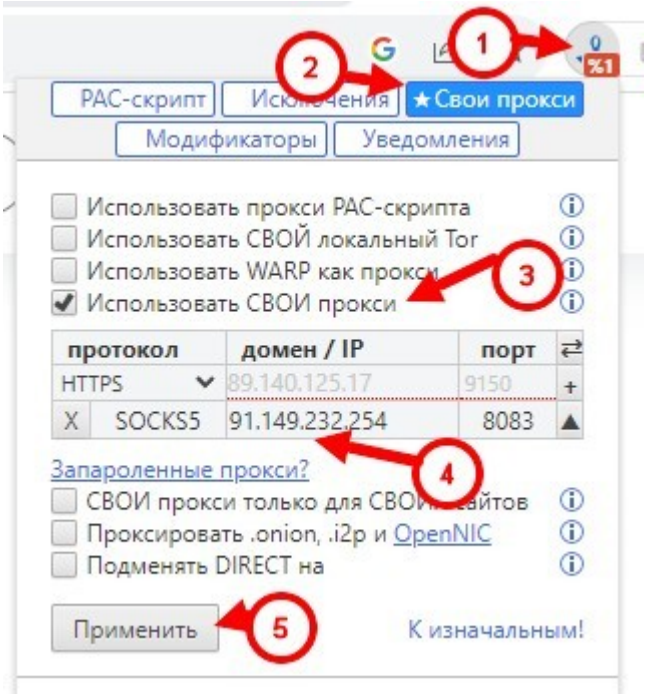
1) После установки плагина открываем его настройки

2) Выбираем пункт "Сво прокси"

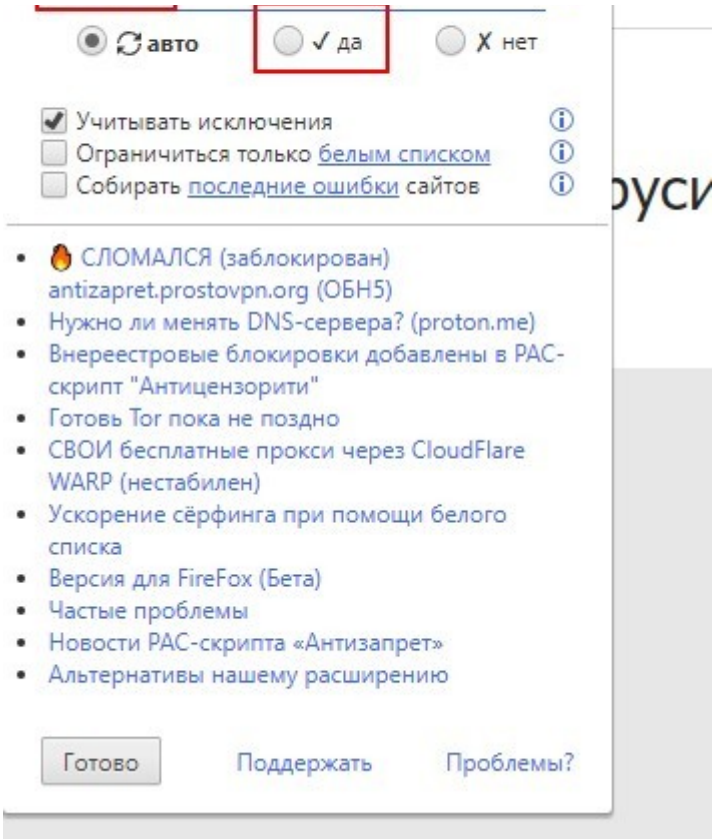
3) Ставим галочку использовать СВОИ прокси

4) Жмем плюсик и добавляем наш прокси (протокол: SOCKS5, ip: ip_servera, port:8083)

5) жмем применить



После этого идем на сайт который не доступен в россии, например https://ark.intel.com и открываем настройки, переходим в исключения и добавляем *.intel.com (обычно плагин подставляет домен сам) Ставим переключатель на ДА и жмем готово. Снова открываем https://ark.intel.com и у нас открывается сайт без проблем(если он не заблокирован в стране где установлен наш сервер)



Revision #7
Created 23 October 2023 05:58:12 by sega
Updated 23 October 2023 09:40:13 by sega