[[](https://habr.com/ru/users/Deleted-user/)](https://habr.com/ru/users/Deleted-user/" \o "Deleted-user)

[**Deleted-user**](https://habr.com/ru/users/Deleted-user/)24 ноя 2023 в 09:24

OpenConnect: недетектируемый VPN, который вам понравится

**Средний**

**15 мин**

**219K**

[Настройка Linux\*](https://habr.com/ru/hubs/linux/)[Информационная безопасность\*](https://habr.com/ru/hubs/infosecurity/)[Системное администрирование\*](https://habr.com/ru/hubs/sys_admin/)[Сетевые технологии\*](https://habr.com/ru/hubs/network_technologies/)

Обзор

Тут недавно проскочила новость, что со следущего месяца Роскомнадзор [запрещает писать про VPN и технологии обходов блокировок](https://habr.com/ru/companies/xeovo/articles/776066/). Я лично на запреты Роскомнадзора клал *<вырезано цензурой>*, но Хабр, видимо, будет вынужден ограничить доступ к подобным статьям из РФ, поэтому давайте от души поразвлекаемся в последнюю неделю пока можно. А потом я уйду на покой и наконец-то буду писать статьи про C++.

Я уже написал здесь много статей на тему прокси-протоколов и прокси-клиентов, которые очень сложно детектировать и заблокировать, и которые используют пользователи в Китае, Иране, Ираке, Туркменистане, и теперь вот в России (мы здесь в отличной компании, правда?). Но довольно часто мне в комментариях писали, мол, это все отлично, но нам нужен именно VPN для целей именно VPN - доступа в частные локальные сети, либо для соединения клиентов между собой, и желательно так, чтобы его не заблокировали ~~обезьяны с гранатой~~. Поэтому сегодня мы поговорим именно о VPN.

Классические OpenVPN, Wireguard и IPSec отметаем сразу - их уже давно умеют блокировать и блокировали не раз. [Модифицированный Wireguard от проекта Amnezia под названием AmneziaWG](https://habr.com/ru/companies/amnezia/articles/769992/) — отличная задумка, но есть одно *но*. Некоторое время назад, во время известных событий в Дагестане, РКН пытался заблокировать Telegram в некоторых регионах. Telegram-прокси для DPI выглядят совсем недетектируемо - как набор рандомных байт без каких-то сигнатур. И знаете что сделал РКН? Они просто заблокировали *все* неопознанные протоколы, работащие поверх TCP. HTTP работает, HTTPS работает, SMTP работает, IMAP работает, а все что "неизвестное и ни на что не похожее" - нет. И таким образом у людей еще перестал работать Shadowsocks, который тоже выглядит как набор рандомных байт и не детектируется. Были ещё, например, сообщения о том, что перестали работать подключения к Radmin (протокол которого DPI, видимо, тоже не знает). И есть неиллюзорная вероятность, что когда начнут резать все неопознанное по TCP, народ массово перепрыгнет на UDP, и РКН в любой момент может начать резать все "неопознанное" и с UDP тоже. И AWG такую блокировку уже не переживет.

Поэтому надо искать что-то, что не только скрывает трафик, но и умеет маскироваться под что-нибудь безобидное. На Хабре уже не раз упоминали про [Cloak](https://github.com/cbeuw/Cloak), в который можно спрятать OpenVPN, замаскировав его под какой-нибудь популярный веб-сайт, такой вариант тоже поддерживается в клиенте Amnezia, но они сами пишут, что скорость работы у такой связки не очень.

Классическими VPN-протоколами, которые внешне неотличимы от обычного HTTP являются SoftEther, MS SSTP и AnyConnect/OpenConnect. В своей первой статье "[Интернет-цензура и обход блокировок: не время расслабляться](https://habr.com/ru/articles/710980/)" я обращал внимание на то, что все они на тот момент были уязвимы к детектированию методом active probing, что позволяло их элементарно заблокировать. Однако с недавних пор в новые версии сервера OpenConnect завезли защиту от такого, и теперь им вполне можно пользоваться.

Изначально протокол появился под названием AnyConnect, и его создателем была всем известная компания Cisco. OpenConnect же - полностью опесорсная реализация клиента и сервера для этого протокола, полностью совместимая с ним.

Чем же хорош OpenConnect по сравнению с альтернативами?

В отличие от **OpenVPN**, **IPSec** и **WG** он внешне выглядит как самое обычное HTTPS-подключение.

В отличие от **SoftEther VPN**, для него существуют клиенты под все популярные платформы: Windows, Linux, macOS, Android, iOS, да и не одни (об этом мы поговорим чуть позже). Ну и есть защита от active probing.

В сравнении с **MS SSTP** у него более производительная серверная часть под Linux, а еще, когда блокировок трафика не осущевляется, он может в дополнение к HTTPS TLS-подключению поднимать DTLS поверх UDP для еще более лучшей производительности (Softether тоже умеет добавлять UDP, но у них свой выглядящий "неизвестным" протокол, а здесь же популярный и всем известный DTLS, который, например, используется в WebRTC). Ну и есть защита от active probing.

OpenConnect очень легко настраивается на клиентах - не нужно кучи параметров и конфигов, требуется только URL (адрес сервера), имя пользователя и пароль.  
Плюс OpenConnect пришел из энтерпрайз-мира, и в нем есть всякие энтепрайзный штучки, например, LDAP- и Radius- и ActiveDirectory-авторизация пользователей.

Установка и настройка

Опенсорсный сервер OpenConnect называется OCserv и живет по адресу <https://gitlab.com/openconnect/ocserv>. Кроме того, его можно найти в репозиториях почти всех популярных Linux-дистрибутивов. Изменения в версиях можно посмотреть в [чейнджлоге](https://gitlab.com/openconnect/ocserv/-/blob/master/NEWS), отмечу самое важное для нас:  
в версии 1.2.0 добавили режим "camouflage" для защиты от active probing;  
в версии 1.2.1 добавили поддержку клиента OneConnect (об этом чуть позже);  
в версии 1.2.3 (она еще официально не зарезилизась!) поправили баг, который мешал подключению некоторых клиентов от Cisco при включенной маскировке (по факту мобильные версии работают нормально и с 1.2.2, а вот десктопная с включенной маскировкой подключаться отказывается).

Поэтому смотрите, что из этого вам надо и проверяйте, какая версия в репах вашего дистрибутива. Если она там сильно старая, то всегда можно собрать вручную, инструкция по сборке есть в README в git-репе и там нет ничего особо сложного.

Если вы хотите ставить сервер через Docker-образ, то там все не очень. Из тех существующих контейнеров что я видел, самый приличный это <https://github.com/aminvakil/docker-ocserv/tree/master> - он на базе последней релизной версии 1.2.2, и если ему подсунуть сертификаты по правильному пути, то он будет использовать их вместо самоподписанных. Но у него в конфиге по-прежнему не хватает опций, связанных с camouflage (и возможности их настроить переменными при разворачивании контейнера), и еще было бы неплохо сразу интегрировать letsencrypt. Если кто-то хочет заняться - автор довольно дружелюбный и открыт к pull request'ам.

Во второй части статьи я приведу пример разворачивания через Docker. Есть также способ собрать Docker-контейнер с 1.2.3 (точнее, с последним master из гита) вручную, я его тоже опишу.

Для тех, кто собирает ручками, systemd-файл в Debian для OCserv выглядит вот так

[Unit]

Description=OpenConnect SSL VPN server

Documentation=man:ocserv(8)

After=network-online.target

[Service]

PrivateTmp=true

PIDFile=/run/ocserv.pid

Type=simple

ExecStart=/usr/sbin/ocserv --foreground --pid-file /run/ocserv.pid --config /etc/ocserv/ocserv.conf

ExecReload=/bin/kill -HUP $MAINPID

[Install]

WantedBy=multi-user.target

Разработкой и поддержкой кода занимаются товарищи из компании Redhat.  
Официальный сайт проекта - <https://ocserv.gitlab.io/www/>  
Так же на официальном сайте есть очень интересный раздел "How-to", где можно подчерпнуть всякую полезную всячину: <https://ocserv.gitlab.io/www/recipes.html> - там, например, есть примеры по настройке TOTP, интеграции с Prometheus, и т.д., короче говоря, там очень много интересного.

Также имейте в виду, что OpenConnect, как и другие подобные VPN, работает поверх HTTPS. То есть да, вам будет нужен домен и TLS-сертификат. Домен может быть любым, хоть DynDNS (см. [статью про прокси](https://habr.com/ru/articles/687512/), там есть соображения и советы на этот счет), а сертификаты для него можно сгенерировать бесплатно с LetsEncrypt и Certbot - инструкций об этом в интернете очень много.

По умолчанию конфиги сервера лежат в /etc/ocserv, а именно:  
*ocserv.conf* - основной конфиг;  
*ocpasswd* - файл со списком пользователей для подключения к VPN-серверу;  
Возможно также создать директорию "config-per-user" и кидать в нее файлы, соответствующие имени пользователя из ocpasswd, чтобы переопределять какие-либо параметры конфигурации для конкретного юзера (например, если нужно постоянно выдавать какому-нибудь юзеру определенный IP-адрес).

Важно: если вы собираете сервер руками из исходников, имейте в виду, что там в репе пример конфига (sample в дире docs) сильно отличается то дефолтного конфига, который поставляется в пакетах дистрибутивов. Проверяйте внимательно все указанные в конфиге пути (например, к файлу ocpasswd).

Добавляются новые пользователи в "ocpasswd" очень просто, с комплекте с OCserv идет утилита с таким же названием "ocpasswd", то есть перейдя в /etc/ocserv и запустив ocpasswd, передав ей в качестве аргумента имя пользователя, которое вы хотите добавить в список, она спросит у вас пароль для него и добавит новый аккаунт куда надо. Естественно, никто не запрещает открывать этот файл в текстовом редакторе и удалять оттуда строки, которые уже не нужны.

А теперь пройдемся по основным конфигурационным параметра с ocserv.conf. Сам этот файл и в гитлаб-репе (под названием sample в папке docs), и при установке пакетов из репозиториев, очень хорошо и подробно документирован комментариями. Я здесь выделю лишь самые важные и прокомментирую от себя:

# Директива заставляет сервер слушать только на определенном IP-адресе,

# а не на всех интерфейсах сразу

# listen-host = [IP|HOSTNAME]

# То же самое, но для UDP. Если закомментировано,

# то будет использовать значение из предыдущего параметра

udp-listen-host = localhost

# Номер порта для входящих подключений.

# По умолчанию 443, и должен быть таким, чтобы быть похожим на HTTPS

tcp-port = 443

# То же самое для UDP. Если закомментировано, то UDP использоваться не будет.

# udp-port = 443

# Пользователь и группа под которыми будут работать воркеры (рабочие процессы)

# сервера

run-as-user = ocserv

run-as-group = ocserv

# TLS-сертификат вашего сервера. Нужно подсунуть сюда пути для сертификатов

# для вашего домена - можете сгененировать их с помощью Letsencrypt/Certbot.

server-cert = /etc/letsencrypt/live/yourdomain.com/fullchain.pem

server-key = /etc/letsencrypt/live/yourdomain.com/privkey.pem

# Можно выводить веселое сообщение для всех подключающихся клиентов.

# Но лучше не надо. Бесит.

# banner = "Hello Habr"

# Если у сервер у вас стоит за чем-то типа HAProxy, то эта опция может пригодится

# listen-proxy-proto = false

# Можно ограничить количество одновременно подключенных клиентов

max-clients = 32

# И одновременно подключенных одинаковых клиентов (с одинаковым логином), 0 - безлимитно

max-same-clients = 0

# ваш домен

default-domain = hellohabr.com

# диапазон IP-адресов, которые вы будете выдавать подключенным клиентам

ipv4-network = 192.168.0.1

ipv4-netmask = 255.255.255.0

# можно также задать в альтернативной форме

#ipv4-network = 192.168.1.0/24

# То же самое для IPv6, если он вам нужен

ipv6-network = fec0::c0ca:c01a:cafe::0/48

# предотвращает утечку DNS, должно быть true

tunnel-all-dns = true

# DNS-сервер, которые будут использовать ваши клиенты.

# Их можем быть несколько

dns = 1.1.1.1

dns = 8.8.8.8

# маршруты, которые будут переданы клиентам: какие диапазоны IP

# нужно будет отправлять через VPN

# route = 10.10.10.0/255.255.255.0

# route = 192.168.0.0/255.255.0.0

# route = fef4:db8:1000:1001::/64

# default = все

route = default

# либо можно пойти "от противного"

# no-route = 192.168.5.0/255.255.255.0

# про это я уже упоминал чуть выше, можно указать путь к дире

# с файлами для переопределения параметров конфигурации для отдельных юзеров

# config-per-user = /etc/ocserv/config-per-user/

# должно быть true если мы подключаемся клиентами от Cisco

cisco-client-compat = true

А теперь про самое интересное: про маскировку. Суть маскировки в том, что когда клиент подключается к серверу, то сервер ждет от клиента специальное "волшебное слово" в URL после знака вопроса, например, обычный URL подключения выглядит как "https://myserver.com", а с секретом - "https://myserver.com/?mysecretword". Если сервер видит в адресе секретное слово - он пропускает клиента в VPN, если не видит, то выдает сообщение об ошибке, как обычный веб-сервер. За этот функционал отвечают следущие настройки в конфиге:

# Включаем маскировку

camouflage = true

# Задаем свое секретное слово, которое клиенты должны иметь в URL'е после знака вопроса

camouflage\_secret = "mysecretkey"

# А вот тут интересное. Если этот параметр не задан (закомментирован),

# то при отсутствии или несовпадении кодовоего слова сервер вернет

# ошибку HTTP 404, очень похожую на ту, что возвращает веб-сервер Apache.

# А если этот параметр \_задан\_, то сервер вернет ошибку HTTP 401,

# означающую "необходимо авторизоваться", как это делают многие

# веб-сервисы, админ-панели, морды разных устройств, и т.д.

# - в этом случае браузеры обычно показывают окошко с предложением

# ввести логин и пароль и текстовым сообщением,

# которое можно задать ниже, а OCserv будет "отвергать" все предложенные пароли.

# Если вы решили использовать этот вариант,

то желательно поменять это сообщение с дефолтного на какое-то свое.

camouflage\_realm = "My admin panel"

Содержимое HTML-страниц, выдаваемых при 404 и 401 ошибках, к сожалению, захардкожено в исходниках, но если уж будете пересобирать сервер, то хорошо бы и поменять - оно там в виде строковых констант, и код очень простой, можете заглянуть в гит: <https://gitlab.com/openconnect/ocserv/-/blob/master/src/worker-http-handlers.c?ref_type=heads#L41>

Из еще прикольных штук - в конфиге можно задать скрипт, который будет запускаться когда к серверу каждый раз подключился/отключился клиент (в скрипт через переменные окружения передаются имя пользователя, его внешний и внутренний IP-адрес, версия клиента, и т.д.), и вписав в скрипт какой-нибудь telegram-send, можно получать сообщения в телеграм-бот о том, кто из пользователей подключился или отключился :)

Еще есть утилитка occtl (для ее работы надо разрешить соответствующую опцию в конфиге), которая умеет показывать текущее состояние сервера и управлять им (например, кикать подключенных пользователей). Данные она выдает как в простом текстовом виде, так и в JSON, поэтому при желании можно вывести статистику сервера (количество подключенных пользователей, объем данных за сессию, и т.д.) в систему мониторинга.

Ну и классический пункт при настройке всевозможных VPN'ов - сделать так, чтобы юзеры могли выходить через VPN в интернет. С помощью iptables (можно засунуть эти команды в /etc/rc.local) это можно сделать, например, так (проверьте, какой диапазон IP для клиентов вы задали в конфиге):

iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ens3 -j SNAT --to-source <ваш внешний IPv4>

ну либо использовать MASQUERADE (инструкций в интернете тоже полным-полно).

И не забудьте включить forwarding:

# в /etc/sysctl.conf (и потом сделать sysctl -p)

net.ipv4.ip\_forward = 1

Если вам нужен IPv6, то тут все чуть сложнее - конечно, если у вас есть сразу /64 или даже /48 подсеть IPv6, то хочется выдавать клиентам сразу белые IPv6 адреса... Но тут встает проблема того, что практически у всех хостеров IPv6-подсети для VPS "не-routed" - то есть не будет такого, чтобы все пакеты на все адреса вашей подсетки оборудование хостера слало на порт вашего сервера. Там все гораздо замудреннее и основано на сложной ICMP-логике, для подобного часто советуют использовать [radvd](https://en.wikipedia.org/wiki/Radvd) или что-то подобное, но короче говоря, у меня ни один из вариантов не заработал. Поэтому для простоты можно раздать клиентам фейковые локальные IPv6 и точно так же заNAT'ить на один внешний IPv6, да простят меня сетевые боги за такое кощунство:

ip6tables -t nat -A POSTROUTING -s fec0::c0ca:c01a:cafe::0/127 -o ens3 -j SNAT --to-source <your\_ipv6>

Всё. В принципе, настройка завершена, и после запуска демона сервера он должен начать принимать подключения.

Использовать ли UDP?

Решать вам. С одной стороны, он в теории должен ускорять работу. С другой стороны, UDP бегающие вместе с висящим TCP-подключением на тот же сервер может показаться подозрительным для цензоров, когда запахнет жареным. У меня без UDP (с чистым TCP) и

настроенным в системе BBR

в/etc/sysctl.conf, не забудьте потом сделать "sysctl -p"

net.core.default\_qdisc=fq

net.ipv4.tcp\_congestion\_control=bbr

между клиентом в Европе и OpenConnect-сервером в России без проблем бегало 150 мегабит/сек (возможно было бы и больше, все уперлось в тариф домашнего интернета).

Если надо хостить на том же сервере сайт

То используйте SNI-прокси. Сайт будет отзываться на одном домене (поддомене), а VPN-сервер на другом. В качестве SNI-прокси может работать Nginx с модулем ssl\_preread (я об этом рассказывал в статьях про проксирование через CDN) или HAProxy - инструкций в интернете по запросу "sni proxy" более чем достаточно.

На 443 порту на публичном адресе в таком случае будет слушать Nginx или HAProxy, а OCserv должен будет слушать на localhost'е на каком-нибудь другом порту - это для TCP, а UDP (если вы его используете) должен по-прежнему быть на публичном адресе. И если вы используете UDP, то очень полезным будет настроить в конфиге OCserv опцию "listen-proxy-proto" в "true", и активировать "Proxy Protocol" в Nginx/HAProxy - тогда они будут сообщать OCserv'у реальные внешние IP-адреса подключающихся клиентов, и OCserv сможет заматчить их с соответствующими UDP-пакетами.

См. также: <https://ocserv.gitlab.io/www/recipes-ocserv-multihost.html>

Установка через Docker

# Получаем сертификат от LetsEncrypt

sudo certbot certonly --standalone --preferred-challenges http -d example.com

# Скачиваем образ Docker

docker pull quay.io/aminvakil/ocserv

# Создаем и запускаем контейнер

# Обратите внимание на пути к сертификатам вашего домена

docker run --name ocserv --sysctl net.ipv4.ip\_forward=1 --cap-add NET\_ADMIN --security-opt no-new-privileges -p 443:443 -p 443:443/udp -v /etc/letsencrypt/live/example.com/privkey.pem:/etc/ocserv/certs/server-key.pem -v /etc/letsencrypt/live/example.com/cert.pem:/etc/ocserv/certs/server-cert.pem -d quay.io/aminvakil/ocserv

# Включаем камуфляж и меняем секретное слово

docker exec ocserv sed -i '/^camouflage = /{s/false/true/}' /etc/ocserv/ocserv.conf

docker exec ocserv sed -i '/^camouflage\_secret = /{s/mysecretkey/yournewecretkey/}' /etc/ocserv/ocserv.conf

# Добавляем юзера для подключения

docker exec -ti ocserv ocpasswd -c /etc/ocserv/ocpasswd yourusername

# Удаляем дефолтного юзера test

docker exec -ti ocserv ocpasswd -c /etc/ocserv/ocpasswd -d test

# Перезапускаем контейнер

docker restart ocserv

И имейте в виду, что если вы разворачиваете сервер в Docker, то клиенты смогут выходить во внешний интернет и в локальную сеть за сервером, смогут коммуницировать между собой, но вот из локалки вы до них достучаться скорее всего не сможете (не помогает даже host network - в комментариях есть вопрос на эту тему, помогите человеку, если кто-нибудь что-нибудь знает).

Если вы хотите собрать свой Docker-образ с более свежей версией (последнюю ревизию из master-ветки в git, сейчас это 1.2.3):

Сборка свежайшего ocserv из git в Docker



Клиенты и их настройка

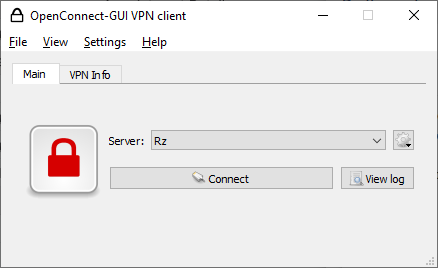
OpenConnect и OpenConnect-GUI

Начнем с родного клиента. **OpenConnect** - опенсорнсный клиент для OpenConnect/AnyConnect (совместим и с тем, и с тем). Можно использовать его напрямую из консольки (читайте manpages), также его использует, например, OpenConnect-плагин для **NetworkManager** в разных дистрибутивах Linux.

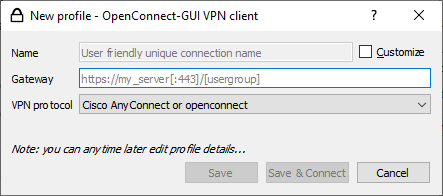
Поведение клиента при подключении к серверу задаётся скриптом (по умолчанию это /etc/vpnc/vpnc-script, можно указать другой), и там можно творить все что угодно, поэтому если вас интересуют всякие сложные настройки маршрутов и split-tunnel, то см. <https://www.infradead.org/openconnect/vpnc-script.html> и <https://github.com/dlenski/vpn-slice>

**OpenConnect-GUI** - графический клиент на той же базе. Написан на Qt, работает под Windows, Linux и macOS. Скачать можно из реп вашего дистрибутива, или с гитлаба разработчиков: <https://gitlab.com/openconnect/openconnect-gui/-/releases>, но имейте в виду, что последний билд там был 5 лет назад. Он, в принципе, работает, но есть небольшие багушки. Более свежие версии можно скачать тут: <https://drive.google.com/drive/folders/1KdzHlYODE-QSYL-JSQoM5vubEw55hhRi>, где выкладываются более свежие snapshot'ы.  
Не пугайтесь непонятной ссылки на Google Drive, это билды [делает один из разработчиков](https://gitlab.com/openconnect/openconnect-gui/-/blob/develop/docs/snapshots.md).

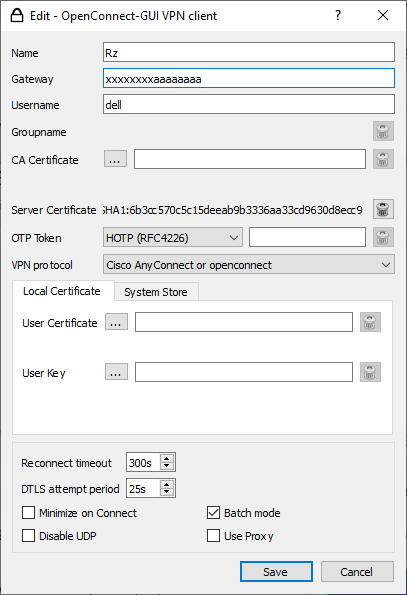
Интерфейс выглядит вот так:



Основное окно



Добавление нового подключения



Редактирование подключение. Обратите внимание на опции Batch mode (сохранение пароля чтобы не вводить его каждый раз) и Disable UDP

Клиент простой, понятный, рабочий, но есть одно НО.

Помните в статье "[Интернет-цензура и обход блокировок: не время расслабляться](https://habr.com/ru/articles/710980/)" я рассказывал про детектирование клиентов по TLS-fingerprint'ам? Так вот, клиент openconnect использует довольно редкую библиотеку GnuTLS. И это прям характерный такой признак, что в какой-то стране цензоры вроде как даже просто блочили все исходящие подключения с такими fingerprint'ами, именно чтобы заблокировать OpenConnect. Решения может быть два - можно упороться и пересобрать клиент с поддержкой OpenSSL вместо GnuTLS. Судя по документации, такое возможно, но я не пробовал. Если кто-то сделает - выложите в публичный доступ, люди вам спасибо скажут. Решение два - использовать какой-нибудь другой клиент :)

Ну и да, OpenConnect есть так же под **Android**, и он живет в Google Play: <https://play.google.com/store/apps/details?id=com.github.digitalsoftwaresolutions.openconnect&hl=en_US> и в F-Droid: <https://f-droid.org/ru/packages/app.openconnect/>

Интерфейс спартанский, но работает хорошо. Недостаток тот же, что и десктопной версии, правда, пересобрать будет гораздо сложнее, хотя исходники тоже доступны: <https://github.com/cernekee/ics-openconnect>  
Есть полезная фича: AppFilter - позволяет отправлять через VPN трафик только выбранных приложений.

Клиент на базе openconnect существует так же в **OpenWRT**: это непосредственно пакет "openconnect", так и "luci-proto-openconnect" (веб-морда для него). Очень компактный, работает без проблем, а [вот здесь есть несколько полезных штучек](https://openwrt.org/docs/guide-user/services/vpn/openconnect/extras).

Cisco AnyConnect Client

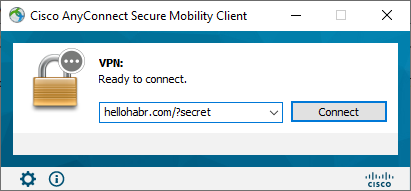
Собственно, оригинальный клиент от **Cisco** - его можно использовать с опенсорсным сервером.

Десктопную версию на сайте Cisco для незарегистрированных клиентов скачать нельзя, но ее можно найти в интернете:

1. <https://its.gmu.edu/knowledge-base/how-to-install-cisco-anyconnect-on-a-windows-computer/>
2. <https://vc.vscht.cz/navody/sit/vpn/windows>

Также он есть в Microsoft Store, и вроде даже говорят, что та версия лучше интегрируется в Windows (можно добавлять ее подключения в список сетевый подключений Windows как другие VPN), но я ее не проверял.

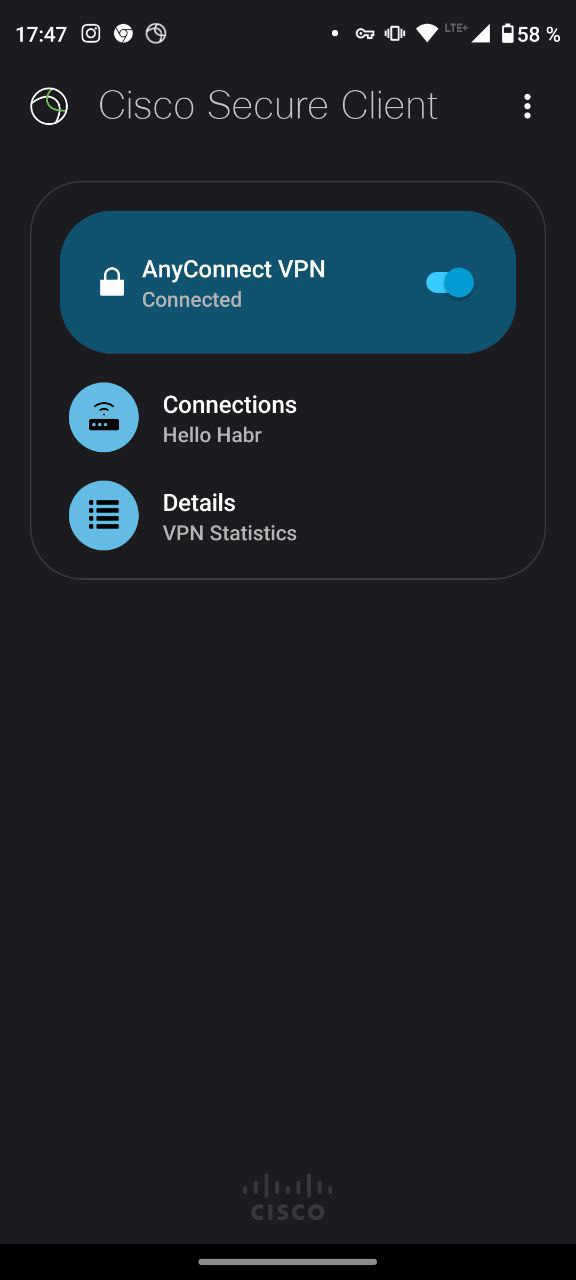
Выглядит тоже очень просто:



Интересная особенность - в клиенте от Cisco невозможно сохранить пароль подключения к серверу, он будет спрашивать его каждый раз. Но если вы используете camouflage-режим с секретным словом, то пароли для пользователей можно ставить очень простые, чтобы не было проблем с запоминанием и вводом.

Существует так же версия для Android под названием Cisco Secure Client: <https://play.google.com/store/apps/details?id=com.cisco.anyconnect.vpn.android.avf&hl=en>  
и под iOS под тем же названием:  
<https://apps.apple.com/us/app/cisco-secure-client/id1135064690>

скриншот



Работает в целом нормально.

Одно но - в багтрекере в гитлабе OCserv люди жаловались, что были проблемы при подключении к OCserv с активированным camouflage с использованием клиентов Cisco. Когда я пробовал и пользовался клиентами по ссылкам выше, у меня все работало, но может в новых версиях что-то сломали, или я какой-то сказочный рукожоп-наоборот (у которого работает то, что не работает у других). Если у вас не заработает - в мастер-бранче репы гитлаба есть версия 1.2.3 с фиксом.

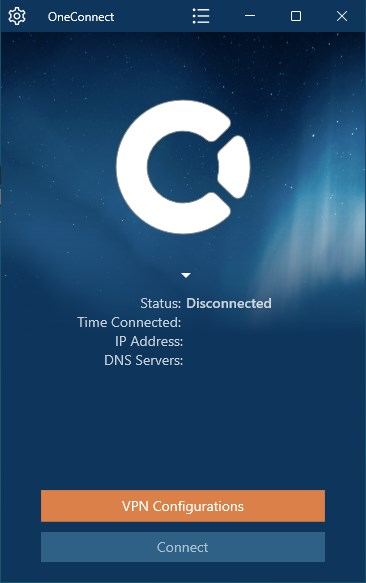
OneConnect

Клиент от компании Clavister - они делают какие-то свои корпоративные решения, используя протокол, аналогичный AnyConnect/OpenConnect. Начиная с версии 1.2.1 OCServ может принимать подключения от OneConnect.

Правда, он со странностями. Версия на Android не подключается к серверу с валидным сертификатом LetsEncrypt, ругаясь на невалидный сертификат. Версия для iOS подключается к тому же серверу без проблем, но только при выключенном camouflage (с включенным camouflage не работает - не может распарсить URL).

Сайт разработчика: <https://www.clavister.com/products/oneconnect/>  
Скачать с **MS Store**: <https://apps.microsoft.com/detail/clavister-oneconnect/9P2L1BWS7BB6?hl=en-US&gl=US>  
Скачать на **Android**: <https://play.google.com/store/apps/details?id=com.clavister.oneconnect&hl=en_US>  
Скачать для **macOS и iOS**: <https://apps.apple.com/us/app/clavister-oneconnect/id1565970099>

скриншот



Интерфейс красивый. Реализация клиента протокола, как я понял, у них своя.  
Еще интересная фишка, как и во всяких китайский прокси, можно добавлять реквизиты сервера на мобильные устройства с помощью прямых ссылок и QR-кода.  
Прямые ссылки генерируются типа

oneconnect://configuration?desc=Hello%20Habr&server=myserver.com&port=443

и, видимо, если ее перевести в QR-код, то она отсканируется (я не проверял, это только догадка).

А теперь главная проблема - оно не умеет работать с camouflage-урлами. Увы.

На этом всё.

Удачи, и да прибудет с вами сила