

[КАК СТАТЬ АВТОРОМ](#)[Тестировщики, вам сюда](#)[Пристегни ремни и держись...](#) Deleted-user

2 дек 2023 в 14:59

GOST: швейцарский нож для туннелирования и обхода блокировок

 Средний 13 мин 88K

Информационная безопасность*, Системное администрирование*, Сетевые технологии*

[Обзор](#)

Начиная со вчера Роскмониторинг запрещает писать про инструменты для обхода блокировок, и это событие мы отметим очередной статьей про инструменты для обхода блокировок.

Сегодня я расскажу о замечательном инструменте под названием GOST. Не пугайтесь, он не имеет никакого отношения к ГОСТ-шифрованию или чему-то подобному, на самом деле это **Go Simple Tunnel**. Он действительно *simple* (простой) в использовании и настройке, но при этом невероятно мощный, поскольку поддерживает огромное количество протоколов и транспортов, из которых вы при желании сможете построить самые упоротые и бронебойные комбинации.

Поддерживаемые протоколы (в скобках указывается версия, начиная с которой поддерживается):

- http - HTTP
- http2 - HTTP2
- socks4 - SOCKS4 (2.4+)
- socks4a - SOCKS4A (2.4+)
- socks5 - SOCKS5
- ss - Shadowsocks
- ss2 - Shadowsocks с AEAD (2.8+)
- sni - SNI (2.5+)
- forward - специальный "протокол" для переадресации коннектов на определенный порт или хост
- relay - собственный протокол GOST, поддерживает проксирование TCP и UDP,

 +174  749 113

Поддерживаемые транспорты:-

- tcp - просто TCP
- tls - TLS
- mtls - мультиплексированный TLS (несколько потоков в рамках одного TLS-коннекта) (2.5+)
- ws - Websocket
- mws - мультиплексированный Websocket (по аналогии с mtls) (2.5+)
- wss - Websocket с TLS-шифрованием
- mwss - мультиплексированный Websocket с TLS-шифрованием :) (2.5+)
- kcp - протокол KCP, о котором я рассказывал в [одной из статей](#) (2.3+)
- quic - QUIC (2.4+)
- ssh - SSH (2.4+)
- h2 - HTTP2 (2.4+)
- h2c - HTTP2 без шифрования (2.4+)
- obfs4 - OBFS4, используемый, например, Тор-бриджами (2.4+)
- ohhttp - HTTP Obfuscation (2.7+)
- otls - TLS Obfuscation (2.11+)

Ну и как я сказал, протоколы и транспорты можно комбинировать, например relay+kcp relay+tls relay+mtls http+kcp http+tls

Благодаря всему этому GOST является отличным инструментом как для повседневного использования, так и для экспериментов - когда цензоры вдруг начали активно банить то, чем вы пользуетесь, можно быстро и просто экспериментировать с разными комбинациями протоколов чтобы найти те, что не заблокированы.

Итак, где можно скачать и почитать про это чудо? Существует две версии GOST: v3 и v2. v3 - активно развивающаяся в настоящий момент, представляет собой глубокий рефакторинг версии v2. v2 - уже вроде как объявлена deprecated. Между собой они совместимы не полностью, поэтому мажорные версии на клиенте и сервере должны совпадать.

Github v3: <https://github.com/go-gost/gost/>

Сайт с документацией v3: <https://gost.run/>

Релизы для скачивания v3: <https://github.com/go-gost/gost/releases>

▸ ну и если вам зачем-то понадобится v2

Со ссылками закончили, теперь попробуем сделать что-нибудь интересное (**спойлер**: самое вкусное будет ближе к концу статьи!).

Конфигурацию в GOST можно задавать как аргументами командной строки, так и в виде YAML-файла. С командной строкой очень удобно, не надо изучать синтаксис конфига, и можно легко экспериментировать, просто меняя параметры при перезапусках. Конфиг - если вы хотите солидно и надолго. Я буду приводить примеры с командной строкой, если интересно, как сделать то же самое с конфигом - на официальном сайте с документацией для каждой фичи есть примеры и того и того, плюс объяснена основная логика конфига (там есть всякие Listener'ы, Handler'ы, и т.д.).

Будьте внимательны, довольно часто параметры командной строки GOST содержат символ `&` (когда вы передаете сразу несколько параметров), как в URL'ах, а в линуксовых шеллах этот символ имеет специальное сакральное значение - в итоге у вас в GOST передается только то что слева от этого символа. Я об этом совершенно забыл, когда экспериментировал, что привело к возгоранию стула в попытках осознать, почему ничего не работает как надо. Решение простое - берите такие параметры в кавычки. В разных примерах вы можете увидеть разные типы написания аргументов, `-L "something"` и `-L="something"` - по факту это одно и то же, работает и так и так.

Два основных типа аргументов у GOST: `-L` и `-F`. `-L (listen)` - это когда процесс слушает входящие подключения на каком-то порту по какому-то протоколу. Аргументов `-L` может быть несколько, в таком случае он будет слушать их одновременно. А `-F (forward)` - это когда процесс будет перенаправлять полученные запросы на какой-то сервер по какому-то протоколу. Аргументов `-F` может быть тоже несколько - тогда у вас получится цепочка серверов (клиент подключится сначала к первому серверу, потом через него ко второму, и т.д.). Есть также варианты балансировки и автоматического выбора серверов из списка, но об этом будет чуть позже.

Итак, что же мы можем сделать?

Начнем, например, с банальнейшего HTTP-прокси.

```
// На сервере:  
gost -L http://user:pass@:8080
```

```
// На клиенте:  
gost -L http://:8080 -L socks5://:1080 -F http://user:pass@SERVER_IP:8080
```

Мы можем подключаться или сразу к серверу на 8080 порт как к HTTP-прокси, или же подключиться к клиенту на 8080 порт как HTTP-прокси, и на 1080 порт как с SOCKS-прокси, а клиент передаст наше подключение на сервер.

Но так не интересно. Просто нешифрованный HTTP — ну кто так делает в наше время? Добавим TLS!

Если вы просто активируете TLS в Gost, то он при запуске сгенерирует самоподписанный сертификат с дефолтным именем хоста `gost.run` - для тестов локально это okay, но в реальной жизни использовать такое я не советую. Можно опцией конфигурации генерировать сертификат с другим именем, но мы, как приличные люди, будем использовать нормальные полноценные сертификаты для своего домена, чтобы вызывать меньше подозрений. Домен можно [получить даже на DynDNS-сервисах бесплатно](#), а TLS-сертификат можно получить с помощью Let's Encrypt, инструкций в интернете очень много. Сервер GOST автоматически подхватит ваши сертификаты, если в его рабочей директории обнаружит файлы `cert.pem`, `key.pem` и `ca.pem`, либо можно указать пути до них вручную в аргументах (обратите внимание на кавычки, про которые я говорил ранее):

```
// На сервере:
gost -L "http+tls://:443?certFile=cert.pem&keyFile=key.pem&caFile=ca.pem"

// На клиенте:
gost -L "http://:8080 -F tls://SERVER_IP_OR_DOMAIN:443?secure=true"
```

Так, теперь у нас есть HTTP-прокси, работающий через TLS - как будто вы обращаетесь к какому-то обычному веб-сайту. В качестве `SNI` при TLS-соединении будет использоваться адрес сервера, который вы указали в строке подключения, если вам надо его переопределить, например, для [domain fronting](#), можно добавить опцию `&serverName=yourdomain.com`

Не хватает еще, чтобы сервер вел себя как обычный веб-сайт при попытке зайти к нему браузером. GOST умеет и такое тоже:

```
// На сервере, одно из:
gost -L "http+tls://user:pass@:443?probeResistance=host:www.example.com:80"
gost -L "http+tls://user:pass@:443?probeResistance=code:403"
gost -L "http+tls://user:pass@:443?probeResistance=file:/var/www/index.html"
```

В первом случае все не-прокси запросы будут переданы на сайт www.example.com (лучше всего использовать для маскировки какой-нибудь реальный, но непопулярный сайт), во втором случае сервер будет отвечать на все запросы 403-м HTTP-кодом, в третьем случае - выдавать содержимое страницы из файла по указанному пути.

Помните замечательную статью "[Безопасный HTTP-прокси менее чем за 10 минут](#)" от @YourChief? Мы сейчас сделали ровно то же самое. У нас благодаря GOST есть прокси, который работает по стандартному HTTP протоколу поверх стандартного TLS. А это значит, что вы можете легко использовать его безо всяких клиентов, прямо из браузера. Можно задать его как системный прокси в Windows, можно задать его как основной прокси в Firefox и Chromium, можно задать его как прокси в расширениях типа SwitchyOmega с возможностью его удобно включать/выключать, задавать правила для доменов, которые вы хотите посещать через прокси, или без него напрямую, и так далее. Можно даже использовать его на Android через AdGuard - обо всем этом можно прочитать в статье по ссылке выше. Инициатором подключения к прокси будет являться сам браузер, а значит TLS fingerprint у такого подключения будет ровно такой же, как у вашего браузера - это даже лучше чем XRay с uTLS. А для сторонних наблюдателей, желающих проверить сервер методом active probing, он будет отдавать фейковый веб-сайт.

Чего-то не хватает? Ах да, если какие-нибудь пронырливые личности сделают к нашему серверу HTTP CONNECT-запрос, то он демаскирует себя ответом "407 Proxy authentication required". Добавим опцию "knock", которая говорит о том, что GOST не будет реагировать на CONNECT-метод без предоставления логина/пароля до тех пор, пока запрос с этим методом не поступит для какого-то только нам известного секретного адреса:

```
// на сервере:  
gost -L "http+tls://user:pass@:443?probeResistance=code:403&knock=www.secretaddr.com"
```

В этом случае при задании прокси в браузере ничего сразу работать не будет (точно так же как для внешних наблюдателей), зато когда вы попытаетесь открыть через прокси этот секретный адрес, сервер наконец-то ответит "HTTP 407 Proxy authentication required", ваш браузер покажет окошко с предложением ввести логин и пароль, зашифрует их, и в дальнейшем уже сам будет подставлять эти реквизиты при каждом запросе на прокси - все будет работать как часы. В итоге у нас получился простой, но тем не менее устойчивый к блокировкам прокси, почти не хуже чем XRay с VLESS (а до тех мест, где он хуже, наш родной Роскомнадзор дорастет еще очень нескоро).

Ну а мы идем дальше. То что будет дальше, уже, к сожалению, не получится использовать с одним только браузером, обязательно нужен GOST-клиент.

Можно использовать mTLS вместо TLS, тогда запросы будут мультиплексироваться в рамках одного подключения - просто замените "tls" на "mTLS" в параметрах.

Можно вместо TCP и TLS туннелироваться по UDP с пакетами типа "непонятное нечто" - мы будем использовать протокол KCP, который, кстати, создан в том числе и для эффективной работы через нестабильные каналы связи с большими потерями.

```
// на сервере:  
gost -L "relay+kcp://:5555?kcp.keepalive=10&kcp.mode=fast3&kcp.key=abababab"  
  
// на клиенте:  
gost -L http://:8080 -L socks5://:1080 -F "relay+kcp://SERVER_ADDR:5555?kcp.keepalive=10&kcp.mode=fast3&kcp.key=abababab"
```

"kcp.key" поменяйте на свой, порт - тоже.

По умолчанию используется aes-шифрование, можно задать его параметром kcp.crypt с возможными значениями aes, aes-128, aes-192, salsa20, blowfish, twofish, cast5, 3des, tea, xtea, xor, sm4, none.

Если у вас нет необходимости работать через плохие каналы связи, можно добавить параметры kcp.rcvwnd и kcp.sndwnd со значением 2048 - будет работать быстрее.

Если вдруг цензоры режут все неопознанные UDP-протоколы, можно использовать в качестве транспорта DTLS, который тоже работает по UDP и используется, например, в WebRTC:

```
// на сервере:  
gost -L=relay+dtls://:5555  
  
// на клиенте:  
gost -L http://:8080 -L socks5://:1080 -F relay+dtls://SERVER_IP:5555
```

Можно проксироваться через вебсокеты (да, в том числе через CDN, как я уже описывал в [одной из предыдущей статей](#))

```
// на сервере:  
gost -L "relay+wss://:443?path=/secretpath"
```

```
// на клиенте:  
gost -L http://:8080 -L socks5://:1080 -F "relay+wss://SERVERHOST:443?path=/secretpath"
```

Поскольку мы используем "wss" (вебсокеты с шифрованием, в отличие от просто ws, которые без TLS), не забудьте подsunуть TLS сертификаты как я описывал в начале статьи, либо указать в дополнительных опциях на сервере пути к ним. Если вам нужно переопределить поле "Host" запроса (например, для [domain fronting](#)), можно добавить опцию `&host=www.someotherdomain.com`

К сожалению, в случае с вебсокетами заставить GOST отдавать фейковый веб-сайт не получится, но никто не запрещает поставить его за Nginx или Caddy и проксировать вебсокеты на GOST с помощью них, так же как я описывал [в уже упомянутой статье](#).

Можно использовать мультиплексированные вебсокеты - замените "ws" на "mws" или "wss" на "mwss" - тогда разные потоки будут объединяться в рамках одного websocket-подключения, что экономит время при установлении соединений с сервером через прокси.

Можно проксироваться через gRPC, как мы это делали с XRay:

```
// на сервере  
gost -L "relay+grpc://:443?path=/Haha/MySecretTun"  
  
// на клиенте  
gost -L http://:8080 -L socks5://:1080 -F "relay+grpc://SERVER_ADDR:443?path=/Haha/MySe
```

Все сказанное выше в контексте вебсокетов про TLS-сертификаты и про возможность установки GOST за Nginx справедливо и тут.

Кстати, еще одна возможность GOST про которую я бегло упомянул выше - это балансировка между разными прокси, например

```
// на клиенте  
gost -L http://:8080 -L socks5://:1080 -F socks5://192.168.1.1:1080,192.168.1.2:1080?s
```

Само собой, прокси могут быть не только socks5, а любых протоколов, которые поддерживает GOST.

Возможные стратегии (strategy):

- round - "по кругу", чтобы никто не ушел обиженным
- rand - выбирать сервер случайным образом
- fifo - использовать второй сервер, если через первый уже выполняется запрос, использовать третий если через второй уже выполняется запрос, и т.д.
- hash - на основе хеша IP-адреса или домена назначения (запросы к одному и тоже же IP/домену будут всегда идти через один и тот же прокси).

Плюс GOST учитывает, если через какой-то из прокси не удалось подключиться более maxFails раз, то такой прокси отмечается как нерабочий и больше не используется.

И еще одна фишка, о которой не грех будет напомнить еще раз - цепочки прокси:

```
// на клиенте:  
gost -L http://:8080 -L socks5://:1080 -F http://FIRSTPROXYADDR:8080 -F "relay+wss://S
```

Когда вы задаете несколько -F-опций, то GOST подключится сначала к первому серверу, потом через него ко второму, и т.д., и только потом к серверу назначения. Это может пригодиться если вам нужно достучаться до вашего прокси на VPS через местный корпоративный прокси-сервер, либо если вы хотите строить цепочки серверов (например, сначала подключиться к серверу в России, а уже потом через него до прокси-сервера за рубежом).

Впрочем, что мы все о прокси да о прокси... GOST поддерживает TUN-интерфейсы! Немного магии....

```
// на сервере:  
gost -L tun://:5555?net=192.168.123.1/24  
  
// на клиенте  
gost -L tun://:0/SERVER_IP:5555?net=192.168.123.2/24
```

...и вот уже у вас полноценный VPN между двумя машинами - в системе появляется tun-интерфейс, через который мы можете коммуницировать с другой стороной тоннеля, маршрутизировать туда и оттуда трафик, и т.д. В GOST для tun кроме "net" есть еще много разных опций, можно добавлять маршруты, и т.д., короче говоря, смотрите документацию.

Да, важная вещь, при использовании TUN под Windows нужно положить рядом с бинарником GOST библиотеку `wintun.dll`

Трафик будет бегать по UDP, но без шифрования, имейте в виду. Что делать, если хочется по TCP, или с шифрованием? Сама реализация TUN в GOST работает только по UDP, но никто не запрещает завернуть ее в relay-протокол и туннелировать как угодно. Например, по TCP:

```
// на сервере:
gost -L tun://:5555?net=192.168.123.1/24 -L relay+tcp://:4444?bind=true

// на клиенте:
gost -L tun://:0/:5555?net=192.168.123.2/24 -F relay+tcp://SERVER_ADDR:4444
```

(в данном случае порт 5555 используется для внутренних нужд, а TCP-подключение идет по 4444 порту).

Можно сделать то же самое поверх TCP и TLS, тогда у вас будет шифрование и прилично выглядящий снаружи трафик:

```
// на сервере:
gost -L tun://:5555?net=192.168.123.1/24 -L relay+tls://:443?bind=true

// на клиенте:
gost -L tun://:0/:5555?net=192.168.1.2/24 -F relay+tls://SERVER_ADDR:443
```

Можно также использовать для TUN транспорт поверх KCP или DTLS, как уже было описано выше.

А можно даже сделать так:

```
// на сервере:
gost -L tun://:8421?net=192.168.123.1/24 -L "relay+wss://:443?bind=true&?path=/secretpath"
// на клиенте:
gost -L tun://:0/:8421?net=192.168.123.2/24 -F "relay+wss://SERVER_IP:443?path=/secretpath"
```

Догадались, что это получилось? Полноценный VPN поверх вебсокетов, в том числе с возможностью работать через CDN, да. Прелесть, правда?

Окей, а что если у нас заблокировано вообще все? Например, суровый корпоративный корпоративный прокси или обезумевшие цензоры производят man-in-the-middle перешифровку всего трафика и не пропускают вообще ничего подозрительного?

Тадааам!

```
// на сервере:
gost -L relay+pht://:80?authorizePath=/authorize&pushPath=/push&pullPath=/pull

// на клиенте:
gost -L http://:8000 -L socks5://:1080 -F relay+pht://SERVERADDR:80?authorizePath=/aut
```

PHT - это **plain HTTP tunnel**. То есть туннель, работающий через простой HTTP. Никакого метода CONNECT, никаких вебсокетов - только обычные GET-запросы. Максимально просто, максимально тупо, максимально надежно - пролезет через что угодно :) Да, скорость будет низкая (по сути дела у нас полудуплексная коммуникация), но это лучше, чем ничего. И да, таким образом можно, например, туннелировать через CDN, которые не поддерживают вебсокет (или просят за них денег, например Fastly) - желательно при этом выключить в настройках CDN кэширование, мы же не звери какие-то. Плюс PHT можно спрятать за Nginx с фейковым сайтом.

А что если даже HTTP не работает? Помните недавнюю статью "Пакуем весь трафик в Ping message, чтобы не платить за интернет" про **ICMP-туннель**, когда информация передается внутри обычных пингов? GOST тоже так может (в v3)!

Сначала отключим ответы на пинги на уровне операционной системы на сервере, иначе мы будем получать в два раза больше ответов чем нужно:

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

а потом (клиент и сервер должны быть запущены от root-пользователя, чтобы иметь возможность низкоуровнево работать с сокетами):

```
// на сервере
gost -L relay+icmp://:0
```

```
// на клиенте  
gost -L http://8080 -L socks5://:1080 -F "relay+icmp://SERVERADDR:12345?keepalive=true"
```

Да, работать будет еще медленнее (ICMP обычно идут с минимальным приоритетом, могут теряться по пути, и т.д.). Но это тоже лучше чем ничего. И как подметили в упомянутой статье, иногда таким образом можно вылезти в интернет, даже когда доступ в интернет вообще запрещен - например, отключен за неуплату или требует авторизации.

На этом остановимся, но нужно отметить что то, что я рассказал выше - это едва ли половина возможностей GOST. Еще он может:

- слушать на разных интерфейсах
- применять разные маршруты для разных адресов назначения, типа, это через этот прокси, это напрямую, а это вообще не пускать
- строить туннели через SSH (да, тот самый SSH); при использовании вместе с протоколом Relay, через SSH можно проксировать не только TCP как обычно, но и UDP
- строить туннели через Shadowsocks (правда, самый свежий поддерживаемый протокол там Shadowsocks-AEAD, но тоже неплохо)
- проксировать не только через HTTP, но и HTTP/2 и HTTP/3
- аутентифицировать клиентов по TLS-сертификатам
- отдавать статический контент по HTTP
- ограничивать скорость передаваемых данных (Limiter)
- пробрасывать порты - передавать подключения, приходящие на локальный порт, на какой-то определенный хост/порт - можно сделать свой аналог stunnel, например
- работать в режиме Transparent proxy (когда вы заворачиваете весь трафик сети на прокси без нужды настраивать что-то на клиентах)
- резолвить домены через DNS с разными параметрами
- проксировать DNS-запросы, в том числе с подменой адресов
- пробрасывать порты в обратном направлении (reverse proxy), когда со стороны сервера или через сервер нужно подключиться к чему-то на клиенте
- пробрасывать порты как reverse proxy с резервированием и горизонтальным масштабированием

- создавать разные аккаунты для клиентов (для протокола relay) с разными логинами и паролями, если вам надо ограничивать доступ по желанию
- работать с unix sockets
- работать с последовательными портами (COM-портами) - можно, например, сделать "удлинитель COM-порта через TCP" как ser2net
- отдавать статистику в Prometheus
- предоставлять Rest API для изменения настроек "на лету"

И еще кучу кучу всего. Еще раз напоминаю сайт с документацией: <https://gost.run/en/>

И да, если вам интересно, как использовать вебсокет- или plain-HTTP-туннели для работы через CDN, маскируясь под чужие домены, смотрите мою следующую статью "**Domain fronting для чайников, и как его использовать для обхода блокировок**".

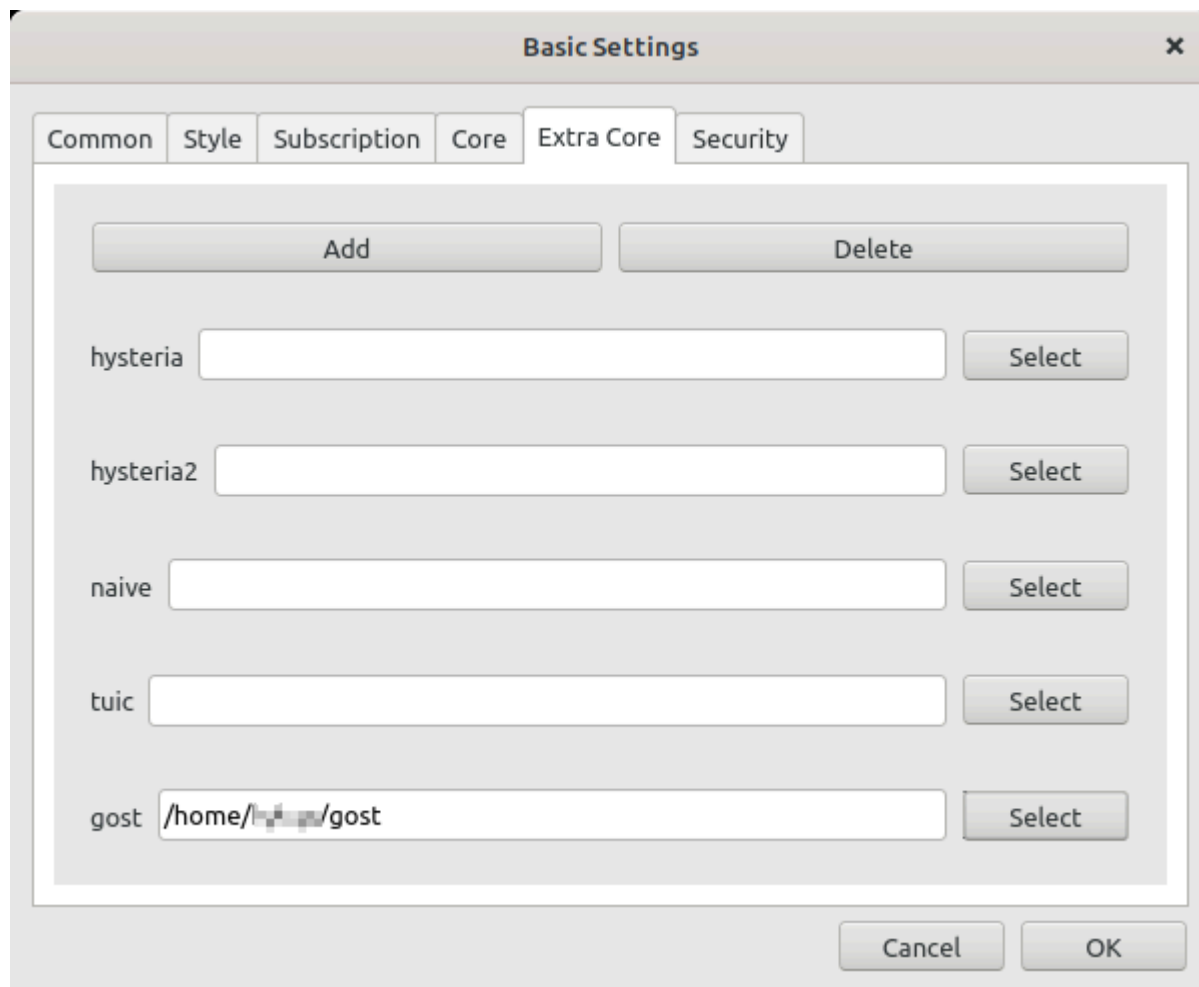
А теперь немного советов, как это запустить.

Можно запускать клиент или сервер вручную командой, как показано выше. Можно написать YAML-файл с конфигов и запускать клиент/сервер указывая этот конфиг.

- [Можно создать systemd-юнит в Linux](#)
- [Можно установить его как сервис в Windows](#)

А можно добавить его даже в наш любимый **Nekoray**:

Идем в Basic Settings -> Extra core. Нажимаем Add, вводим имя ("gost"). Появляется новая строка, там кликаем Select и выбираем наш бинарник GOST:



Чтобы добавить новое подключение, идем в Server -> New profile, выбираем там тип Custom (extra core), выбираем наш core "gost", вводим адрес и порт сервера, а в "Command" можно вставить что-то типа

```
-L socks://:%socks_port% -F relay+tls://%server_addr%:%server_port%
```

Естественно, используйте протокол и параметры, которые нужны вам. Обратите внимание, здесь уже никаких кавычек.

Edit

Common

Type

Custom (Extra Core)

Name

My GOST tunnel

Address

123.23.45.99

Port

443

Core

gost

Json Editor

Command

port% -F relay+tls://%server_addr%:%server_port%

Config Suffix

Mapping Port*

0

Socks Port*

0

Preview

example:

server-address: "127.0.0.1:%mapping_port%"
listen-address: "127.0.0.1"
listen-port: %socks_port%
host: your-domain.com
sni: your-domain.com

Custom Outbound Settings

Not set

Custom Config Settings

Not set

Cancel

OK

Нажав на Preview, можно посмотреть, какую команду для запуска GOST сгенерировал Nekoraу, чтобы убедиться, что все правильно.

Другой вариант - вставить в большое поле в окне ваш YAML-конфиг, выбрать "Config suffix" "yaml", а в команде запуска упомянуть %config% - тогда Nekoraу создаст файл с нужным содержимым и передаст путь к нему в этот аргумент.

А что с мобильными клиентами... с ними пока не очень. Есть плагин GOST для клиента Shadowsocks-Android, но он позволяет только заворачивать в него Shadowsocks, плюс он

основан на старой версии GOST и не поддерживает самое вкусное (PHT- и ICMP-туннели).

На iOS GOST поддерживается в великолепном клиенте Shadowrocket, правда, судя по всему, тоже на базе старой версии и без вкусок. Я не тестировал.

И существуют пакеты для OpenWRT, но они, кажется, сделаны для v2:

<https://github.com/kenzok8/openwrt-packages/tree/master/gost>

<https://github.com/kenzok8/openwrt-packages/tree/master/luci-app-gost>

На этом всё.

Удачи, и да прибудет с вами сила.

Если вы хотите сказать спасибо автору — сделайте пожертвование в один из благотворительных фондов: "Подари жизнь", "Дом с маяком", "Антон тут рядом".

Теги: gost, go simple tunnel, tls, proxy, туннель, websocket, vpn

Хабы: Информационная безопасность, Системное администрирование, Сетевые технологии

Редакторский дайджест

Присылаем лучшие статьи раз в месяц

**11****0**

Карма

Рейтинг

Deleted user @Deleted-user

Так вышло

Комментарии 113

Публикации

ЛУЧШИЕ ЗА СУТКИ

ПОХОЖИЕ

**UranusExplorer**

12 часов назад

Надежный обход блокировок в 2024: протоколы, клиенты и настройка сервера от простого к сложному

**Простой**

36 мин



27K

Тutorial

**+198**

444



73

**Telnov_WIKI**

19 часов назад

Исследование деградации Li-ion аккумуляторов в результате “быстрой” зарядки

**Простой**

4 мин



19K

Аналитика

**+43**

48



39

**Lunathecacat**

вчера в 12:00

115 лет прогресса: от механического осциллографа до самодельного цифрового

**Простой**

9 мин



6.7K

Ретроспектива

**+37**

42



13

**Exosphere**

вчера в 12:27

Инженеры, мы в ваших руках

**Простой**

6 мин



4.4K

**+29**

25



13

**DAN_SEA**

20 часов назад

Ещё один шаг в сторону оптических наушников

**Средний**

10 мин



8K

[Обзор](#) +27 23 50**madyouth**

23 часа назад

Как мы создаём редакторы документов. Ядро и его роль в кроссплатформенной разработке

 10 мин  1.1K +27 19 1**iMonin**

19 часов назад

Энергетика большой страны. Почему мы все не можем отапливаться электричеством?

 28 мин  10K +25 36 67**sacredtree**

22 часа назад

Почему рациональный выбор невозможен

 8 мин  3.2K +18 38 7**it_union**

3 часа назад

Снижение зарплат в ИТ

 2 мин  10K +16 8 10**WizAlx**

19 часов назад

Интеграция нативных SDK во Flutter-приложение

 9 мин  851[Тutorial](#)[Перевод](#)

hh.ru проверил, к кому российские айтишники пойдут работать и почему

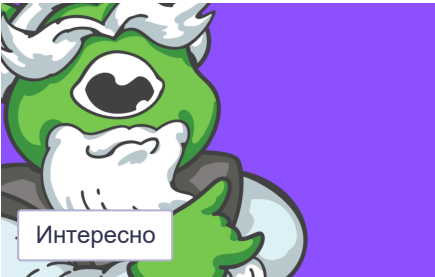
Турбо

Показать еще

МИНУТОЧКУ ВНИМАНИЯ



Из горнила конкуренции:
лучшие технобренды России



Глупым вопросам и ошибкам —
быть! IT-менторство на ХК

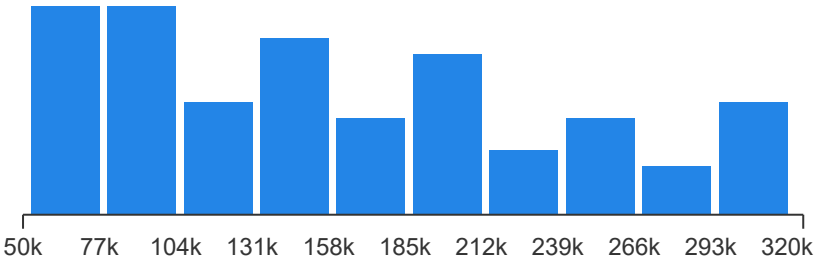


Топим снег скидками:
промокодус в деле

СРЕДНЯЯ ЗАРПЛАТА В IT

168 040 **₽/мес.**

— средняя зарплата во всех IT-специализациях по данным из 22 229 анкет, за 1-ое пол. 2024 года. Проверьте «в рынке» ли ваша зарплата или нет!



Проверить свою зарплату

ЧИТАЮТ СЕЙЧАС

Надежный обход блокировок в 2024: протоколы, клиенты и настройка сервера от простого к сложному

27K

73

Снижение зарплат в ИТ

10K

10

Голодные игры начались. Развитие ИИ приведёт к естественному отбору населения

43K

196

Исследование деградации Li-ion аккумуляторов в результате “быстрой” зарядки

19K

39

Сколько мы заработали за год на 1 товаре из Китая. Продаем коврики для ноутбука на маркетплейсах

39K

65

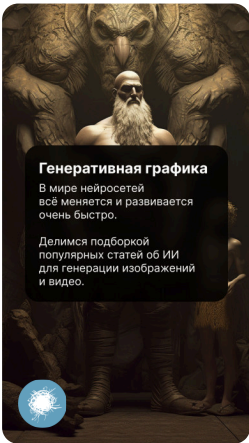
hh.ru проверил, к кому российские айтишники пойдут работать и почему

Турбо

ИСТОРИИ



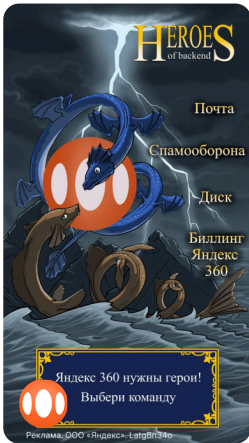
GitVerse: открой вселенную кода



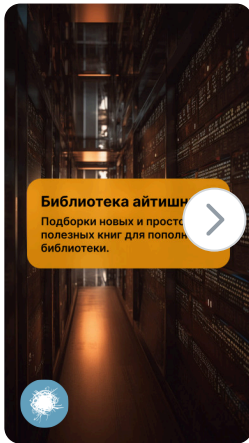
Нейросети: интересное



Что умеет калькулятор зарплат в ИТ



Яндекс 360 призывает героев бэкэнда



Полезные книги для библиотеки айтишника

РАБОТА

- Системный администратор


92 вакансии
- Специалист по информационной безопасности


140 вакансий
- DevOps инженер

28 вакансий


Все вакансии

БЛИЖАЙШИЕ СОБЫТИЯ


Как решать алгоритмический блок при найме в IT?
Разбираемся на Тренировках по алгоритмам 5.0 от Яндекса



Хабр Карьера



ВАЙБ ЧЕК


Онлайн-курс «Delivery Manager»


Почему становиться Delivery Manager'ом — плохая идея?

Открытый урок
18 марта в 20:00

Серия занятий «Тренировки по алгоритмам 5.0» от Яндекса


 1 марта – 19 апреля


 19:00


 Онлайн

Подробнее в календаре

Тестировщики, выбирайте себе команду по вайбам на Хабр Карьере


 18 – 24 марта


 09:00 – 23:00


 Онлайн

Подробнее в календаре

Открытый урок: почему становиться Delivery Manager'ом — плохая идея?»

 18 марта



 Онлайн

Подробнее в календаре

Ваш аккаунт	Разделы	Информация	Услуги
Войти	Статьи	Устройство сайта	Корпоративный блог
Регистрация	Новости	Для авторов	Медийная реклама
	Хабы	Для компаний	Нативные проекты
	Компании	Документы	

- Авторы

Песочница
- Соглашение

Конфиденциальность
- Образовательные

программы

Стартапам



Настройка языка

Техническая поддержка

© 2006–2024, Habr