English • Deutsch • Español • Français • Italiano • Polski • Português • ??????? • Svenska • ???(????)? • ???(??)? •

OpenVPN is a full-featured SSL VPN solution which can accomodate a wide range of configurations, including road warrior access, home/office/campus telecommuting, WiFi security, secure branch office linking, and enterprise-scale remote access solutions with load balancing, failover, and fine-grained access-controls.

Contents

Security

Interesting article about strength and security of PKI today Pro-Linux (german)

Support

There are a great variety of support options for OpenVPN which discuss all of the OpenVPN features in detail. If you have any questions, first check out these resources:

- OpenVPN FAQ
- OpenVPN Related Project Services

In doubt always refer to the OpenVPN documentation.

Getting OpenVPN

Before you begin, you'll need to obtain and install the OpenVPN client (with the GUI (for e.g. Windows)), Mac) for your client computer.

Please note that it consists of one binary for both client and server connections, so don't look for individually labeled client or server packages.

OpenVPN authentication

You will have several possibilities to authenticate to an openvpn session. Some authentication methods are supported by the gui directly (should be preferred), others have to be set up in daemon mode. Which you will use depends on some things: how secure, easy to setup, easy to maintain

Public Key Infrastructure (PKI) [Certificates]

Follow these instructions to create the required certificates and keys. But there is also a Webmin module for linux servers to create your PKI infrastructure (OpenVPN CA). You begin by creating a key and certificate pair for the certificate authority (CA) you are establishing. Then for the server and each client, you create a private key and certificate pair and sign the certificates using the CA's key. Afterwards, you should have the following files:

```
ca.crt
ca.key
dh{n}.pem
server.crt
server.key
```

and a private key and certificate pair for each client. For example, if you have two clients, you should have:

```
client1.crt
client1.key
client2.crt
client2.key
```

Note: The CA private key (ca.key) is used only to sign the certificates. It should be kept securely and not be copied to the server or clients.

Static Key

On Linux you can create a pair of keys as easy as

```
openvpn --genkey --secret /home/"user"/static.key
```

On Windows system you must already have openvpn installed. To create a pair of keys go to

```
programs -> openvpn -> generate a static openvpn key
```

Copy the content of the keys to the ddwrt gui static key field.

Info
create keypair

OpenVPN in DD-WRT

In this section, it is assumed that you already have the DD-WRT installed in your router and that its a current build (newer than 01/2011). OpenVPN is only available on units with at least 8mb flash (except the Broadcom VPN build.)

The DD-WRT GUI Server and Client modes are very straight forward. So when defaults are left as is, it should run out of the box for 90% of the users.

1. Using the Web Interface, go to the "Services" tab and then the "VPN" tab (for older versions of dd-wrt go to the "Administration" tab and then the "Services" sub-tab).
2. Enable *OpenVPN Daemon* or *OpenVPN Client*. If further options do not appear, click **Apply Settings**.
3. Fill in needed parameters (see below). Everything else not mentioned here is taken care of automatically (e.g. firewall)
4. Click **Apply Settings**.

GUI: Client Mode

Fixed Parameters

There are default parameters that will always be written to the config file and which you **cant** change if you use GUI client mode:

```
ca /tmp/openvpncl/ca.crt
cert /tmp/openvpncl/client.crt
key /tmp/openvpncl/client.key
management 127.0.0.1 16
management-log-cache 50
verb 4
```

```
mute 5
syslog
writepid /var/run/openvpncl.pid
client
resolv-retry infinite
nobind
persist-key
persist-tun
script-security 2
tls-client
fast-io
```

Changeable Parameters

DD-WRT default settings in {}
OpenVPN config language in []

Server IP/Name (e.g. 192.168.1.1)
: IP address/hostname of the OpenVPN server you want to connect to. [remote xxx]

Port
: port which OpenVPN server is listening on. {1194} [port xxx]

Tunnel Device (TUN/TAP)
: The mode of tunneling. TUN: routing (layer 3), TAP: bridgeing (layer 2. can be used for routing, too, but its not very common). [dev-type tun/tap]

Tunnel Protocoll (UDP/TCP)
: The subprotocol the connection will use on the connection. {udp} [proto udp/tcp]

Encryption Cipher (None and Blowfish to AES512)
: The encryption algorithm that will be used for the tunnel. Blowfish "fastest" to AES512 "safest". {AES128} [cipher xxx]

Hash Algorithm (None and MD4 to SHA512)
: The hash algorithm that will be used. MD4: fastest (maybe unsafe) to SHA512. {SHA256} [auth xxx]

Advanced options
: Leave defaults as is if you dont know what you are doing. {disabled} []

TLS Cipher
: What encryption algorithm OpenVPN should use for encrypting its control channel. {disabled} []

LZO Compression
: Enables compression over VPN. This might speedup the connection. Must be the same value as on server. {adaptive} [comp-lzo yes/no/adaptive/disabled]

NAT
: Enable network address translation on the client side of the connection. Enables the NAT-firewall to protect clients. {disabled}

Bridge TAP to br0
: Enable a transparent bridge across the tunnel to the local LAN. Works only in TAP mode with NAT disabled. {disabled}

Local IP Address
: In cases you will not get an ip from the server. Not very common. {empty)

Subnet Mask
: Subnet Mask for the IP Address.

TUN MTU Setting
: set the mtu of the tunnel {1500} [tun-mtu xxx]

UDP Fragment across the tunnel

set mss-fix and fragmentaion accross the tunnel. {empty} [fragment xxx]

UDP MSS-Fix
: = value of Fragment. Only usen with udp. should be set on one side only. [mssfix]

nsCertType verification
: Checks to see if the remote server is using a valid type of certificate meant for OpenVPN connections. As this is a security feature of OpenVPN, it should be left enabled. {checked} []

TLS Auth Key
: The static key OpenVPN should use for generating HMAC send/receive keys. {disabled} []

Additional Config
: Any additional configurations you want to define for the VPN connection. {empty}

Policy based Routing
: allow only special clients to use the tunnel. Add IPs in the form 0.0.0.0/0 to force clients to use the tunnel as default gateway. One line per IP. {} []

Public Server Cert
: Certificate of OpenVPN CA (not the server's public cert) in *pem* form; only part between (and including) `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` is necessary; as it is stored in nvram, everything else from that file should be removed to conserve space.

CA Cert
: CA certificate; also only part between 'BEGIN' and 'END' is required.

Private Client Key
: Key associated with certificate above; should be kept secret because anybody who knows this key can successfully authenticate as this client.

DH PEM
: Diffie Hellmann parameter.

Static Key
: used for p2p links. No pki needed.

GUI: Server Configuration

This section describes how to configure an OpenVPN server that uses SSL certificates for client authentication, which is recommended. The method offers better security than using a static key and allows multiple clients to connect at the same time.

Fixed Parameters

There are default parameters that DD-WRT will always wite to the config file and which you cant change if you use the GUI server mode:

```
dh /tmp/openvpn/dh.pem
ca /tmp/openvpn/ca.crt
cert /tmp/openvpn/cert.pem
key /tmp/openvpn/key.pem
keepalive 10 120
verb 4
```

```
mute 5
log-append /var/log/openvpn
writepid /var/run/openvpnd.pid
management 127.0.0.1 5002
management-log-cache 50
mtu-disc yes
topology subnet
client-config-dir /tmp/openvpn/ccd
script-security 2
ifconfig-pool-persist /tmp/openvpn/ip-pool 86400
fast-io
passtos
```

Customizable Parameters

DD-WRT default settings in {}
OpenVPN config language in []

Start Type
: use "System". "WAN Up" doesnt work. {}

Config via (GUI/config file)
: GUI {}

Server Mode (TUN/TAP)
: The mode of tunneling. TUN: routing (layer 3), TAP: bridging networks (layer 2). {} [dev-type tun/tap]

DHCP-Proxy mode
: Only in bridge mode. Let the clients use the network dhcp server not the openvpn dhcp. {} []

Pool start Ip
: 1st ip of the ip pool used (Only in bridge mode). []

Pool end IP
: Last ip of the ip pool used (Only in bridge mode). []

Gateway
: Default gateway to use (Only in bridge mode). []

Network (e.g. 10.10.10.0)
: Network to use for the tunnel (Only in routing mode). []

Netmask (e.g. 255.255.255.0)
: Netmask of the used network. []

Block DHCP accross the tunnel
: Dont allow DHCP requests across tunnel (Only in bridge mode).

Port
: port which OpenVPN server listens on. {1194} [port xxx]

Tunnel Protocoll (UDP/TCP)
: The subprotocol the connection will use on the real used tcp connection. {udp} [proto udp/tcp]

Encryption Cipher (None and Blowfish to AES512)
: The encryption algorithm that will be used for the tunnel. Blowfish: fastest to AES512 safest. {AES128} [cipher xxx]

Hash Algorithm (None and MD4 to SHA512)
: The hash algorithm that will be used. MD4: fastest (maybe unsafe) to SHA512. {SHA256} [auth xxx]

Advanced options
: Leave defaults as is if you dont know what you are doing. {disabled}

LZO Compression

Enables compression over VPN. This might speedup the connection. Must be the same value as on server. {yes} [comp-lzo yes/no/adaptive/disabled]

Redirect default Gateway
: Force the clients to use the tunnel as default gateway. {disabled} []

Allow Client to Client
: Allow clients to see each other. {disabled} [client-to-client]

Allow duplicate cn
: allow to use 1 client cert to use on multiple clients (security risc)

TUN MTU Setting
: set the mtu of the tunnel {1500} [tun-mtu xxx]

MSS-Fix/Fragment across the tunnel
: set mss-fix and fragmentaion accross the tunnel. {empty} [fragment xxx] [mssfix]

TLS Cipher
: What encryption algorithm OpenVPN should use for encrypting its control channel. {disabled} []

TLS Auth Key
: The static key OpenVPN should use for generating HMAC send/receive keys.

Client connect script
: . {empty} []

Additional Config
: Any additional configurations you want to define for the VPN connection. {empty}

Public Server Cert
: Certificate of OpenVPN CA (not the server's public cert) in *pem* form; only part between (and including) `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` is necessary; as it is stored in nvram, everything else from that file should be removed to conserve space.

Public Client Cert
: Client certificate issued by CA for this particular router; also only part between 'BEGIN' and 'END' is required

Private Client Key
: Key associated with certificate above; should be kept secret because anybody who knows this key can successfully authenticate as this client

Auth with username and password

I wanted to make the access to my network easy but secure. Certificates are nice, but I still like username and password even more. You can use the following simple script to check for usernames/passwords.

Add this to the config on the server and set the right path for the script:

```
auth-user-pass-verify /path/to/your/script/verify.sh via-file
```

This line tells openvpn server to check for passed username and password by calling the script verify.sh and passing the username and password in a tmp file

Every client have to have this in its config file:

```
auth-user-pass
```

This just tells openvpn client to ask the user for username and password or s/he will not be able to log in. You can also use "--auth-user-pass" (instead of the line in the config file) on the command line.

This is a simple shell script called "verify.sh" which contains usernames and passwords in clear text. I wanted to encrypt passwords with "passwd", but command "passwd" is not available in dd-wrt (and I have no idea how dd-wrt encrypts root password for /etc/passwd). If someone knows how to encrypt passwords in dd-wrt, plese add it **here**.

[Edit by mrwizeman Dec 18 2011] Hmm ok, I pasted a script here like 2 years ago which grtz was kind enough to modify to support multiple users but after trying it, I couldnt get it to work and I couldnt really understand all of the bracketbracket if statements either so here is a modified version from me as well, which now has multiple users support as well as hash capabillity and it lowercases the username properly as well: users file should contain colon separated entries like this: <username>:<hash>

```
#!/bin/sh
USERS=`cat ./users`
genhash() {

        HASHPASS=`echo -n "$1$2" | md5sum | sed s'/\  -//'`
        i=0
        while [ $i -lt 10 ]; do
                HASHPASS=`echo -n $HASHPASS$HASHPASS | md5sum | sed s'/\  -//'`
                i=`expr $i + 1`
        done
        echo -n $HASHPASS
}
verify() {
        Login=`echo $1 | awk '{print tolower($0)}'`
        echo Logging in as: $Login
        #echo Password is: $2
        [[ $# -eq 2 ]] || exit 1
        for i in $USERS; do
                Name=${i%:*}
                PassHash=${i#*:}
                Logincmp=`echo $Name | awk '{print tolower($0)}'`
                #echo Logincmp is: $Logincmp
                if [ "$Logincmp" == "$Login" ]
                then
                        #echo "Login('$Login') is equal to Logincmp('$Logincmp')"
                        GENHASH=`genhash "$Login" "$2"`
                        #echo genhash returned $GENHASH
                        #echo and PassHash is: $PassHash
                        if [ "$GENHASH" == "$PassHash" ]
                        then
                                echo Password hashes match
                                exit 0
                        fi
                fi
        done
}
if [ "$1" == "--genhash" ]
then
        Login=`echo $2 | awk '{print tolower($0)}'`
        echo `genhash "$Login" "$3"`
        exit 1
fi
verify `cat $*`
exit 1
```

I had to wait for the time to update from 1970 until I create my users file so here is what my start script looks like in case someone wants to use it:

```
YEAR="`date +%Y`"
COUNT=0
while [ "$YEAR" = "1970" ] && [ $COUNT -lt 60 ]; do
  # System has not yet synchronized the date with ntp.
  sleep 1
  YEAR="`date +%Y`"
  COUNT=$(($COUNT + 1))
done
# A 60 second wait is long enough. If there is a problem still then
# likely the router booted faster than the ISP CPE. Force a time sync.
if [ "$YEAR" = "1970" ] && [ $COUNT -ge 60 ]; then
  # Initialize the date/time.
  NTP_POOL_SERVER="`nvram get ntp_server`"
  /usr/bin/killall ntpclient
  ntpclient -l -h $NTP_POOL_SERVER
#   stopservice process_monitor
#   startservice process_monitor
fi


openvpn --mktun --dev tap0
brctl addif br0 tap0
ifconfig tap0 0.0.0.0 promisc up
echo "user1:abababababababababababababababa
user2:abababababababababababababababa" > /tmp/users
```

You may disregard whatever is written below this point , but im going to leave it here for history's sake...

[Edit by mrwizeman Aug 31 2009] Ok so I put it **here**... here is a way I came up with that make this work with an encrypted password, now, let me explain how it works, I hash the username and password and then add the hash to itself, and hash that 10 times everytime I add the hashes together, I got that idea from the author of the passwd we use Poul-Henning Kamp but he does it 1000 times...

You can easily change this script to hash it 1000 times, but I think 10 is enough it will take a bruteforce program forever to first hash the user and pass and then hash the hashes 10 times, just to find out if it matches, and besides the weak ass processor of the routers we use will take forever to check our credentials if we do it that way... so anyway here it is: you have to run it in telnet the first time to figure out what your hash is, then change that in your script, the hash in the script I paste here is user: test and pass: test

First the script to generate a hash for you:

```
#!/bin/sh

genhash() {
        echo You are generating a HASH for user: $1
        echo with the password                          : $2
        HASHPASS=`echo -n $1$2 | md5sum | sed s'/\  -//'`
        i=0
        while [ $i != 10 ]; do
            HASHPASS=`echo -n $HASHPASS$HASHPASS | md5sum | sed s'/\  -//'`
            #echo [$i] HASHPASS=$HASHPASS
            i=`expr $i + 1`
```

```
        done
        echo HASHPASS=$HASHPASS
}
genhash $1 $2
```

Then the actual script for your check:

```
#!/bin/sh

HASH='1bbd7254581aaab10868ccfdc0860d68'
#echo HASH = $HASH
#echo param 1 = $1
#echo param 2 = $2

vpn_verify() {
        if [[ ! $1 ]] || [[ ! $2 ]]; then
            #echo "No username or password: $*"
            exit 1
        fi
        HASHPASS=`echo -n $1$2 | md5sum | sed s'/\  -//'`
        #echo HASHPASS = $HASHPASS
        #if [ $HASH == $HASHPASS ]; then
        #    echo MATCH!!
        #else
        #    echo NO MATCH!!!
        #fi
        i=0
        while [ $i != 10 ]; do
            HASHPASS=`echo -n $HASHPASS$HASHPASS | md5sum | sed s'/\  -//'`
            #echo [$i] HASHPASS=$HASHPASS
            i=`expr $i + 1`
        done
        #echo HASHPASS=$HASHPASS
        if [ $HASH == $HASHPASS ]; then
            #echo MATCH!!
            exit 0
        else
            #echo NO MATCH!!!
            exit 1
        fi
}
if [[ ! $1 ]] || [[ ! -e $1 ]]; then
     #echo "No file"
     exit 1
fi
vpn_verify `cat $1`
#echo "No user with this password found"
exit 1
```

I suck at shellscripting, so if somone spots any errors or want to add support for multiple users that would be cool...

[Edit by grtz Sep 28 2010] Here is the optimized version with support for multiple users. Everything is in one file and smaller, little bit faster and more secure:

```
#!/bin/sh
```

```
USERS="test:556a2224e307429d714ed8a8134e68e0"
#USERS=`cat ./users`

genhash() {
        HASHPASS=`echo -n "$1$2" | md5sum | sed s'/\  -//'`
        i=0
        while [ $i -lt 10 ]; do
                HASHPASS=`echo -n $HASHPASS$HASHPASS | md5sum | sed s'/\  -//'`
                i=`expr $i + 1`
        done
        echo -n $1:$HASHPASS
}

verify() {
        [[ $# -eq 2 ]] || exit 1
        for i in $USERS; do
                [[ "$i" == `genhash "$1" "$2"` ]] && exit 0
        done
}

[[ $1 == "--genhash" ]] && echo `genhash "$2" "$3"`
[[ -e "$*" ]] || exit 1
verify `cat "$*"`
exit 1
```

You can as before put users in an external file in format "username1:encrypted_password1" (one per line):

```
username1:encrypted_password1
username2:encrypted_password2
```

or put everything in USERS directly in verify.sh and split entries with space (like in the original script below):

```
USERS="username1:encrypted_password1 username2:encrypted_password2"
```

Run "verify.sh --genhash username password" to create encrypted string. Copy the result in your file with users or directly in the script as shown above:

```
root@router:$ ./verify.sh --genhash test 12345
test:556a2224e307429d714ed8a8134e68e0
```

or even easier if you use external file for users/password:

```
root@router:$ ./verify.sh --genhash test 12345 >> ./users
```

--Grtz 22:42, 28 September 2010 (CEST)

Original authors script below, the one without encrypted pass:

```
#!/bin/sh

 ## format: username:password username:password ...
 ## you can even have same usernames with different passwords
 USERS='user1:pass1 user2:pass2 user3:pass3'
```

```
## you could put username:password in
## a separate file and read it like this
#USERS=`cat file_with_users`

vpn_verify() {
    if [[ ! $1 ]] || [[ ! $2 ]]; then
        #echo "No username or password: $*"
        exit 1
    fi

    ## it can also be done with grep or sed
    for i in $USERS; do
        if [[ "$i" == "$1:$2" ]]; then

            ## you can add here logging of users
            ## if you have enough space for log file
            #echo `date` $1:$2 >> your_log_file

            exit 0
        fi
    done
}

if [[ ! $1 ]] || [[ ! -e $1 ]]; then
    #echo "No file"
    exit 1
fi

## $1 is file name which contains
## passed username and password
vpn_verify `cat $1`

#echo "No user with this password found"
exit 1
```

You can delete all lines which begin with #. "echo" commands are here just for you to know what happens at that and they can be used for debugging.

This verification with the script does not work if openvpn is running in "daemon" mode. I have no idea what is the reason for that, probably some wrong interpretation of the output ("exit 0" means user/pass OK, everything else means user/pass NOT OK). Thats why I am running openvpn server as a background process and all output is going to /dev/null:

```
openvpn --config openvpn.conf >/dev/null 2>&1 &
```

--Comma 23:53, 11 August 2008 (CEST)

I was getting the error

```
openvpn_execve: external program may not be called due to setting of --script-security level
```

I had to add the following lines to my config file work in v24-SP2

```
tmp-dir /tmp/openvpn
script-security 3
```

Additionally, the ">/dev/null 2>&1 &" hack wouldn't work but running from the command line did. I resorted to using the /tmp/myvpn symlink as well as --daemon in the command line. All works now with out an interactive shell. --JoeM 23:28, 22 October 2008 (CEST)

Client Configuration

1) Copy the CA certificate and a private key and certificate pair to the client.

2) Create an OpenVPN configuration file on your client computer:

```
client
dev tap
proto udp
remote router-address 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client1.crt
key client1.key
ns-cert-type server
comp-lzo
verb 3
```

Replace *router-address* in the fourth line with your router's IP address or host name and, obviously, use the appropriate file names in the script for the client's certificate and key. On Windows, save this script using notepad as a file with extension .ovpn and copy it into \Program Files\OpenVPN\config (along with the files ca.crt, clientX.crt and clientX.key created earlier.)

Notes:

- If you need to use TCP instead of UDP, set the protocol to *tcp-client*. Clients must use the same protocol that the server does.
- Mac OS X users need to add the line

```
up "./vpn-up.sh"
```

to the configuration file and create a script file *vpn-up.sh* in the same directory as the configuration file with the following contents:

```
#!/bin/bash
sleep 2
ipconfig set $1 DHCP
```

This script configures the TAP interface to request an IP address using DHCP. Make sure to make the script executable with 'chmod 755 vpn-up.sh'

3) Use the configuration file you just created with whatever OpenVPN client software you're using to establish a connection to your router.

If everything works, you will be connected to your internal LAN. If your router is running a DHCP server, the TAP interface will be assigned an IP address on your internal LAN. At this point, you just connect to your

home PC the same way you would do it from inside your LAN.

Daemon Mode/Config File

Bridge 2 Networks
Route 2 Networks

Troubleshooting

FAQ

MTU

One verycommon problem is the wrong setting of the mtu/fragment/mss-fix values. Ethernet's max frame size is 1500. In most cases where the Router is the gateway and the vpn connects to the outside world the values must be changed. WAN connections in general will have a lower frame size so the default setting will result in a dropping connection. mtu/fragment must be set the same on both sides!

On tcp connections only mtu can be set. Try setting the mtu to 1300 and see if it works.

On udp connections enable mss-fix on **one** side of the link. Lower the fragemt size to 1300 and try if its working.

In both cases when the link is working u can try to raise the values again to the maximum which will be working.

Logs

logs are written to syslog and can be seen in the GUI: Status -> OpenVPN (not in daemon and script mode)

Correct time

OpenVPN requires client and server to have more or less synchronized time. Therefore make sure that router has correct time. To check it use command `date` and if you get info about year 1970, you should enable NTP client. Also, ntp server has to be outside of your VPN as time should be corrected before VPN is established. If you have syslog enabled (to local server, or the server outside your VPN), errors like `TLS Error: Unroutable control packet received from` *`server_ip`*`:1194` may indicate this problem. (This error message may appear if your certificates are not valid or have expired, too.) If you get an error message saying that your certificate is not yet valid, set the dd-wrt clock to UTC time (in the first configuration page).

To set the time manually connect via telnet or ssh and issue the following command at the prompt:
date MMDDHHMMYYYY

Connecting to DD-WRT OpenVPN Server via Mac Client

Currently, there are two OpenVPN clients for OS X

- Tunnelblick
- Viscosity

Viscosity has a more user friendly gui but is a commercial product. It can also import settings created for the Tunnelblick application.

PROBLEM: When I used the sample client.conf file above, nothing happened when I clicked connect on Tunnelblick.

Tunnelblick is a GUI OpenVPN application for the Mac. The last official release is 2.0.1, but I am going to explain how to connect using version 3.0rc3. This version has everything in one app and requires no extra components to be installed.

NOTE: I used certificates to connect my MacBook Pro to the OpenVPN server. You can find documentation on how to do this further up in this page. Remember to place your certificate files in the same directory as the client.conf file. Tunnelblick looks for the files in ~/Library/openvpn.

Download Tunnelblick and drag it to the Applications folder. It will say that you don't have a client.conf file and will make one for you. You can examine the sample client.conf file that Tunnelblick makes and find that it does a few things differently than the example provided above. Set up your client.conf file using the sample I have provided below, and Tunnelblick should connect properly now.

The file I have provided is the same as the sample, except with a few things disabled and a few things added to make it do what we want. The reason I disabled a few things using the # line is that I didn't want to delete anything, so I just disabled them.

You might notice that the command

```
up "./vpn-up.sh"
```

This is a fix that I found here.

You will need to make a script called vpn-up.sh and place it in the same directory as the client.conf file. Make the files contents:

```
#!/bin/bash
sleep 2
ipconfig set tap0 DHCP
```

client.conf:

```
#############################################
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server.     #
#                                            #
# This configuration can be used by multiple #
# clients, however each client should have   #
# its own cert and key files.                #
#                                            #
# On Windows, you might want to rename this  #
# file so it has a .ovpn extension           #
#############################################

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
dev tap

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel
# if you have more than one.  On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
#dev-node MyTap

# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
proto tcp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote xxxxxxxxx.dyndns.org 1194

# Choose a random host from the remote
# list for load-balancing.  Otherwise
# try hosts in the order specified.
#remote-random

# Keep trying indefinitely to resolve the
# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
user nobody
group nobody

# Try to preserve some state across restarts.
persist-key
```

```
persist-tun

# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here.  See the man page
# if your proxy server requires
# authentication.
#http-proxy-retry # retry on connection failures
#http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca ca.crt
cert client1.crt
key client1.key

up "./vpn-up.sh"

# Verify server certificate by checking
# that the certicate has the nsCertType
# field set to "server".  This is an
# important precaution to protect against
# a potential attack discussed here:
# http://openvpn.net/howto.html#mitm

# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server".  The build-key-server
# script in the easy-rsa folder will do this.
ns-cert-type server

# If a tls-auth key is used on the server
# then every client must also have the key.
#tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
#cipher x

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
comp-lzo

# Set log file verbosity.
verb 3

# Silence repeating messages
;mute 20
```

UPDATE Sept 1, 2009
The most recent release of Tunnelblick is 3.0rc14. It seems as though the previously mention bash script does not work anymore. The following python script does work, however:

```
#!/usr/bin/python
import os, sys
try:
    tun_dev, tun_mtu, link_mtu = sys.argv[1:4]
except:
    sys.exit(0)
if tun_dev[0:3] == 'tap'
    os.system('/usr/sbin/ipconfig set "%s" DHCP' % os.environ['dev'])
```

Can't Load DH Parameters

If you get the following:

```
Tue Jan 23 03:03:05 2007 OpenVPN 2.0.7 mipsel-unknown-linux [SSL] [LZO] [EPOLL] built on Jan 15 2
Tue Jan 23 03:03:05 2007 Cannot load DH parameters from dh1024.pem: error:0906D064:lib(9):func(10
Tue Jan 23 03:03:05 2007 Exiting
```

you should check your certificates. Did you paste them correctly?

Note: you *must* include new lines.

Connection reset by peer (WSAECONNRESET) (code=10054)

If you received error when trying to connect to your VPN server verify the following:

- Make sure the tap-device was successfully "up'ed". It is, if you find it in the list when running "ifconfig" without further arguments. If it is not displayed in the list of available interfaces have a look at "Yet another evolution" above.
- Make sure the protocol is the same on the server and client, ie UDP/TCP
- Verify the port number is the same on the client and server; Default port is UDP 1194
- Check to see if the port is open on the server by using nMap or another utility

Incoming Packet Rejected

If you get incoming packets being rejected when trying to connect with the VPN client (OpenVPN) make sure that you are forwarding the 1194 port (if configured to use that port) to the proper server/router (usually 192.168.1.1).

Example TCP/UDP Rejection:

- [Date] TCP/UDP: Incoming packet rejected from 192.168.1.1:1194[2], expected peer address: w.x.y.z:1194 (allow this incoming source address/port by removing --remote or adding --float)

Port Forwarding is configured on v24 SP 2 under 'NAT / Qos'.

Authenticate/Decrypt failure

```
Error: Authenticate/Decrypt packet error: cipher final failed
```

If this happens to you, you probably have a mismatch in the configuration between server and client on the "cipher"-option. (E.g.: cipher AES-128-CB / cipher AES-256-CB) This is not a problem of DD-WRT or OpenVPN but just a config issue which can happen if you follow some of those guidelines strictly without knowing what the config options mean.