Apporter une IPv4 de datacenter sur un pfSense via un VPN

L'objectif est de faire descendre une IP de datacenter sur le pfSense avec un tunnel OpenVPN et du proxyARP.

Il vous sera nécessaire :

- un serveur OpenVPN linux avec:
    - Une IP fixe pour initier la session VPN
    - Une IP supplémentaire (nommé “IP Fail-Over” chez OVH par exemple)

Dans notre exemple, notre IP supplémentaire sera 172.32.0.1

Configuration du serveur OpenVPN

La configuration d'OpenVPN est classique avec quelques exception, par exemple :

proxyarp.conf

```
mode server
tls-server
proto udp
port 1194
dev tap0
cipher AES-256-CBC
keepalive 10 30
persist-key
persist-tun
verb 3
status proxyarp_status.log
log-append /var/log/openvpn-proxyarp.log

ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/server.crt
key /etc/openvpn/easy-rsa/keys/server.key
dh /etc/openvpn/easy-rsa/keys/dh4096.pem
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 0
auth sha256
keysize 256
comp-lzo no

script-security 2
client-connect /etc/openvpn/proxy-arp-up.sh
client-disconnect /etc/openvpn/proxy-arp-down.sh
```

Vous noterez l'utilisation OBLIGATOIRE d'une interface TAP, l'absence de configuration réseau et l'ajout des trois lignes suivantes :

```
script-security 2
client-connect /etc/openvpn/proxyarp_up.sh
client-disconnect /etc/openvpn/proxyarp_down.sh
```

et d'ajouter dans le dossier */etc/openvpn* les deux fichiers suivant (en les adaptant) :

proxyarp_up.sh

```
#!/bin/bash

echo '1' > /proc/sys/net/ipv4/conf/all/proxy_arp
ifconfig tap0 up
ip route add 172.32.0.1 dev tap0
```

et

proxyarp_down.sh

```
#!/bin/bash

ip route del 172.32.0.1 dev tap0
ifconfig tap0 down
```

et pour finir de les rendre exécutable :

```
# chmod +x proxyarp_up.sh proxyarp_down.sh
```

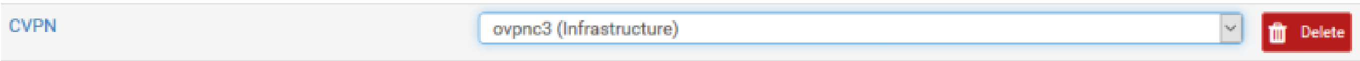
Configuration du client OpenVPN pfSense

On va créer un client OpenVPN sur pfSense, si on suit l'exemple plus haut :

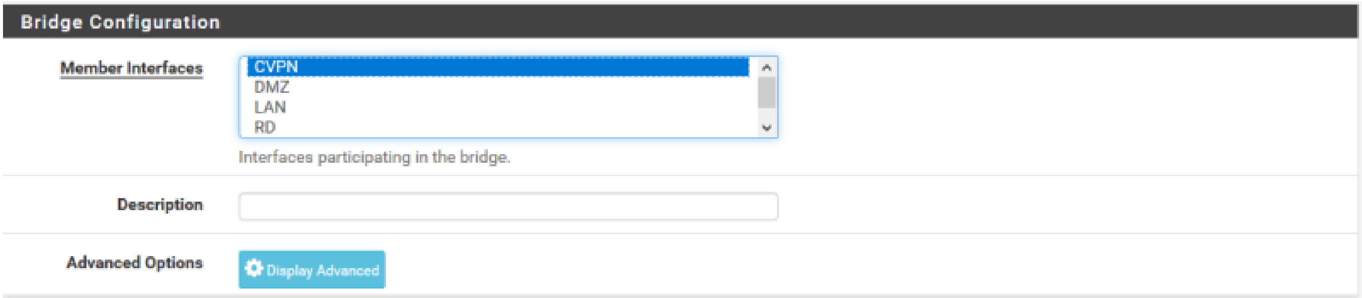

General Information
Disabled
Disable this client
Set this option to disable this client without removing it from the list.
Server mode
Peer to Peer (SSL/TLS)
Protocol
UDP on IPv4 only
Device mode
tap - Layer 2 Tap Mode
"tun" mode carries IPv4 and IPv6 (OSI layer 3) and is the most common and compatible mode across all platforms.
"tap" mode is capable of carrying 802.3 (OSI Layer 2.)
Interface
WAN
The interface used by the firewall to originate this OpenVPN client connection
Local port
Set this option to bind to a specific port. Leave this blank or enter 0 for a random dynamic port.
Server host or address
The IP address or hostname of the OpenVPN server.
Server port
1194
The port used by the server to receive client connections.
Proxy host or address
The address for an HTTP Proxy this client can use to connect to a remote server.
TCP must be used for the client and server protocol.
Proxy port
Proxy Authentication
none
The type of authentication used by the proxy server.
Description
Infrastructure
A description may be entered here for administrative reference (not parsed).
User Authentication Settings
Username
Leave empty when no user name is needed
Password
Password
Leave empty when no password is needed
Password
Confirm
Authentication Retry
Do not retry connection when authentication fails
When enabled, the OpenVPN process will exit if it receives an authentication failure message. The default behavior is to retry.
Cryptographic Settings
TLS Configuration
Use a TLS Key
A TLS key enhances security of an OpenVPN connection by requiring both parties to have a common key before a peer can perform a TLS handshake. This layer of HMAC authentication allows control channel packets without the proper key to be dropped, protecting the peers from attack or unauthorized connections.The TLS Key does not have any effect on tunnel data.
TLS Key
Paste the TLS key here.
This key is used to sign control channel packets with an HMAC signature for authentication when establishing the tunnel.
TLS Key Usage Mode
TLS Authentication
In Authentication mode the TLS key is used only as HMAC authentication for the control channel, protecting the peers from unauthorized connections.
Encryption and Authentication mode also encrypts control channel communication, providing more privacy and traffic control channel obfuscation.
Peer Certificate Authority
Peer Certificate Revocation list
None
Client Certificate
Encryption Algorithm
AES-256-CBC (256 bit key, 128 bit block)
The Encryption Algorithm used for data channel packets when Negotiable Cryptographic Parameter (NCP) support is not available.
Enable NCP
Enable Negotiable Cryptographic Parameters
Check this option to allow OpenVPN clients and servers to negotiate a compatible set of acceptable cryptographic Encryption Algorithms from those selected in the NCP Algorithms list below.
NCP Algorithms
AES-128-CBC (128 bit key, 128 bit block)
AES-128-CFB (128 bit key, 128 bit block)
AES-128-CFB1 (128 bit key, 128 bit block)
AES-128-CFB8 (128 bit key, 128 bit block)
AES-128-GCM (128 bit key, 128 bit block)
AES-128-OFB (128 bit key, 128 bit block)
AES-192-CBC (192 bit key, 128 bit block)
AES-192-CFB (192 bit key, 128 bit block)
AES-192-CFB1 (192 bit key, 128 bit block)
AES-192-CFB8 (192 bit key, 128 bit block)
AES-256-GCM
AES-128-GCM
Available NCP Encryption Algorithms
Click to add or remove an algorithm from the list
Allowed NCP Encryption Algorithms. Click an algorithm name to remove it from the list
The order of the selected NCP Encryption Algorithms is respected by OpenVPN.
Auth digest algorithm
SHA256 (256-bit)
The algorithm used to authenticate data channel packets, and control channel packets if a TLS Key is present.
When an AEAD Encryption Algorithm mode is used, such as AES-GCM, this digest is used for the control channel only, not the data channel.
Set this to the same value as the server. While SHA1 is the default for OpenVPN, this algorithm is insecure.
Hardware Crypto
No Hardware Crypto Acceleration
Tunnel Settings
IPv4 Tunnel Network
This is the IPv4 virtual network used for private communications between this client and the server expressed using CIDR notation (e.g. 10.0.8.0/24). The second usable address in the network will be assigned to the client virtual interface. Leave blank if the server is capable of providing addresses to clients.
IPv6 Tunnel Network
This is the IPv6 virtual network used for private communications between this client and the server expressed using CIDR notation (e.g. fe80::/64). When set static using this field, the ::2 address in the network will be assigned to the client virtual interface. Leave blank if the server is capable of providing addresses to clients.
IPv4 Remote network(s)
IPv4 networks that will be routed through the tunnel, so that a site-to-site VPN can be established without manually changing the routing tables. Expressed as a comma-separated list of one or more CIDR ranges. If this is a site-to-site VPN, enter the remote LAN/s here. May be left blank for non site-to-site VPN.
IPv6 Remote network(s)
These are the IPv6 networks that will be routed through the tunnel, so that a site-to-site VPN can be established without manually changing the routing tables. Expressed as a comma-separated list of one or more IP/PREFIX. If this is a site-to-site VPN, enter the remote LAN/s here. May be left blank for non site-to-site VPN.
Limit outgoing bandwidth
Between 100 and 100,000,000 bytes/sec
Maximum outgoing bandwidth for this tunnel. Leave empty for no limit. The input value has to be something between 100 bytes/sec and 100 Mbytes/sec (entered as bytes per second). Not compatible with UDP Fast I/O.
Compression
No LZO Compression [Legacy style, comp-lzo no]
Compress tunnel packets using the LZO algorithm.
Compression can potentially increase throughput but may allow an attacker to extract secrets if they can control compressed plaintext traversing the VPN (e.g. HTTP). Before enabling compression, consult information about the VORACLE, CRIME, TIME, and BREACH attacks against TLS to decide if the use case for this specific VPN is vulnerable to attack.
Adaptive compression will dynamically disable compression for a period of time if OpenVPN detects that the data in the packets is not being compressed efficiently.
Type-of-Service
Set the TOS IP header value of tunnel packets to match the encapsulated packet value.
Don't pull routes
Bars the server from adding routes to the client's routing table
This option still allows the server to set the TCP/IP properties of the client's TUN/TAP interface.
Don't add/remove routes
Don't add or remove routes automatically
Do not execute operating system commands to install routes. Instead, pass routes to –route-up script using environmental variables.
Advanced Configuration
Custom options
Enter any additional options to add to the OpenVPN client configuration here, separated by semicolon.
UDP Fast I/O
Use fast I/O operations with UDP writes to tun/tap. Experimental.
Optimizes the packet write event loop, improving CPU efficiency by 5% to 10%. Not compatible with all platforms, and not compatible with OpenVPN bandwidth limiting.
Send/Receive Buffer
Default
Configure a Send and Receive Buffer size for OpenVPN. The default buffer size can be too small in many cases, depending on hardware and network uplink speeds. Finding the best buffer size can take some experimentation. To test the best value for a site, start at 512KiB and test higher and lower values.
Gateway creation
Both
IPv4 only
IPv6 only
If you assign a virtual interface to this OpenVPN client, this setting controls which gateway types will be created. The default setting is 'both'.
Verbosity level
default
Each level shows all info from the previous levels. Level 3 is recommended for a good summary of what's happening without being swamped by output.
None: Only fatal errors
Default through 4: Normal usage range
5: Output R and W characters to the console for each packet read and write. Uppercase is used for TCP/UDP packets and lowercase is used for TUN/TAP packets.
6-11: Debug info range

Avec pour même spécificité : l'interface TAP et pas de configuration de réseau.

Il faut ensuite assigner l'interface ovpncX comme interface du pfSense sans IP :



puis de créer un bridge avec UNIQUEMENT l'interface créer précedement :



puis assigner ce bridge a une interface, et lui assigner l'IP Fail-Over de votre hébergeur ainsi que sa passerelle (souvent identique a celle de votre serveur).

Si celle-ci n'est pas dans le même réseau, il vous faudra cocher la case **Use non-local gateway** dans la gateway.