VORKBAARD UIT DE TOEKOMST
Kapitein Vorkbaard to the rescue!

Ahoy, Maatjes! Taal Scifi Tech Twitter

Search for:
Search

Donate
Did one of my articles help you? Help me pay for hosting this site! Any amount is greatly appreciated :)
Donate

Recent Posts
Scratch Monkey, door Charles Stross
IP-based shares in Samba
Het Microsoft-moeras
Blindsight, door Peter Watts
Little Fuzzy, door H. Beam Piper

Recent Comments
Kapitein Vorkbaard on Installing ASSP spamfilter on Ubuntu Server 14.04 LTS
Ellsworth on Installing ASSP spamfilter on Ubuntu Server 14.04 LTS
Saulo on Adding ownCloud 8 to Active Directory 2012 R2 – Part 4: connecting to Active Directory
Kapitein Vorkbaard on Get current user's SID from the command line in Windows 7
Marco on Get current user's SID from the command line in Windows 7

Archives
January 2016
December 2015
November 2015
October 2015
September 2015
August 2015
July 2015
June 2015
May 2015
April 2015
March 2015
February 2015
January 2015
December 2014
October 2014
August 2014
April 2014
February 2014
January 2014
November 2013
September 2013
August 2013
July 2013
June 2013
May 2013
June 2012
May 2012

Categories
Scifi
Taal
Tech
Twitter
Uncategorized

Meta
Log in
Entries RSS
Comments RSS
WordPress.org

« Een Debian-systeem backuppen en restoren
Using PuTTY and keyfiles to SSH into your Ubuntu 12.04 server »

Setup OpenVPN with Google Authenticator on Ubuntu 12.04 LTS server

Written on June 7, 2013 at 13:26, by Kapitein Vorkbaard

OpenVPN is nice. It works on all kinds of servers and nowadays there are clients for **all kinds of devices** as well. I use it to connect to my home network from my laptop when I'm elsewhere and from my Android phone if I'm on a public hotspot because it encrypts all my data.

Recently I added **one time passwords** to my OpenVPN setup. You install Google Authenticator on your phone and it generates a fresh code every thirty seconds. You use that code to add to authentication on your OpenVPN server and this gives you pretty good security because you need:

- **something you have** (certificates);
- **something you know** (your username and password);
- **something unique** (the one time password from Google Authenticator).

These three things – something you have, something you know and something unique – are considered essential for decent security. Of course this is no guarantee for good security as **other components** are equally essential, like keeping your passwords for yourself and having your phone automatically lock after a certain amount of time so your secrets aren't easily available for other people.

That said, it can't hurt to add an extra layer of security to your OpenVPN setup and it's not very difficult so there's really no reason not to do it.

[Edit 2013-12-18: I found one. The SSL/TLS renegotiation handshake occurs once per hour per client. By default OpenVPN caches your credentials (you can turn it off by using the auth-nocache option) but the cached credentials contain *a random number* in the password! Effectively this means your connection will be dropped after exactly one hour. I haven't found a workaround yet and I'm afraid any workaround will compromise other parts of the security philosophy. The solution must lie in implementing Google Authenticator in a separate credential field from the password. If you do not intend to use your connection for more than one hour at a time and then perhaps reconnecting this is fine but for all-day use this can get quite annoying.

Edit 2014-04-04: And here is the solution. Add this line to both the server and client configuration:

```
reneg-sec 0
```

If you set reneg-sec to 36000 on the server and to 0 on the client, the server will ask for a new one-time password every ten hours and the client won't initiate dropping the connection by itself.]

In this article I'll describe how I built **two OpenVPN servers** on a Ubuntu 12.04 LTS server. I chose Ubuntu 12.04 LTS because it's easy to find documentation on it.

(Note: the term *OpenVPN server* refers to an OpenVPN profile on my server. So one Ubuntu server with two OpenVPN servers means one machine serving two different tastes of OpenVPN.)

The first OpenVPN server allows connecting to the LAN from outside but with the **internet breakout at the client** side: traffic from the client to your LAN goes through the vpn but traffic from the client to the internet doesn't and goes outside directly. The advantage of this setup is that it **spares bandwith** on your server's internet connection. You would use this if you are in a trusted network and your vpn server doesn't have a whole lot of bandwith.

The second OpenVPN service has its **internet breakout at the OpenVPN server**. Here, all traffic from the client including internet traffic are routed through the vpn. The advantage here is that the client must adhere your OpenVPN server's **firewall** rules and that all **internet traffic is encrypted** because it's also vpn traffic. This is ideal for public hotspots and other untrusted networks.

A heads up: if you're running more than one OpenVPN server on a server you MUST **use**

unique virtual ip ranges and port numbers. If you're using the same virtual ip ranges for multiple OpenVPN servers on the same machine then only one of those servers will work.

Also try and use a **non-standard private ip** range on your server's lan. Most home routers use 192.168.0.0/24 or 192.168.0.1/24 and it's virtually impossible to use a vpn from an ip range connecting to a lan with that same ip range. Use something like 10.84.12.0/24 or 192.168.173.0/24 so you'll have a good chance you won't be connecting from that same ip range in someone else's network.

Before we begin:

- My server's wan network interface is called 'wan'. Yours may be called eth1 or something else. Replace 'wan' with your interface's name.
- I'm doing this as root so I don't have to type 'sudo' before every command.

Enabling your server's routing capability

Enable packet forwarding by opening up **/etc/sysctl.conf** and uncommenting the line where it says

```
net.ipv4.ip_forward=1
```

Now set up ip masquerading so your clients can break out:

```
# iptables -t nat -A POSTROUTING -o wan -j MASQUERADE
```

Add the above line to **/etc/rc.local** to have it enabled after a reboot.

Installing OpenVPN

This is the easy part.

```
# apt-get install openvpn
```

Creating certificates

The OpenVPN package now contains a convenient certificate authority which we'll use. Feel free to use any other ssl facility you like.

Create a certificate autothority (ca):

```
# cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0
/etc/openvpn/easy-rsa
# cd /etc/openvpn/easy-rsa

Edit the /etc/openvpn/easy-rsa/vars file to your needs:

export EASY_RSA="/etc/openvpn/easy-rsa"
export KEY_COUNTRY="NL" (or whichever country you're in)
export KEY_PROVINCE="ZH") (or whichever province you're in)
export KEY_CITY="Rotterdam" (idem)
export KEY_ORG="Vorkbaard, Inc." (your organisation's name. Make
one up if you need.)
export KEY_EMAIL="your@email.address"
```

Make sure easy-rsa can find openssl-cnf to prevent an error:

```
# ln -s openssl-1.0.0.cnf openssl.cnf
```

Commit your changes:

```
# source ./vars
```

Create a ca certificate

```
# ./build-ca openvpn
```

Enter the correct values. Noone's checking them and you may use the same values as before. It's handy to fill these out with true values in case you need to troubleshoot and you have more than a couple of users. For the common name, enter your server's fqdn, e.g. myserver.example.com.

Create a key server:

```
# ./build-key-server myserver
```

('Myserver' would be my server's hostname)

Enter the correct values; these will be used to sign user certificates.

You can leave **A challange password** and **An optional company name** empty.

```
Sign the certificate? y
Commit? y
```

Set up Diffie-Helleman parameters

```
# ./build-dh
```

Create the OpenVPN config files

We'll first setup the OpenVPN servers and clients and get them working. Then we'll add the Google Authenticator bits. It's easier to troubleshoot that way. We'll call the local breakout one **'general'** and the vpn breakout one **'routeall'.** You can use any name. Choose one that describes your vpn server so its easy to recognize while going through logfiles for troubleshooting. I'll use non-standard ports; feel free to use any ports you like.

Create a file **/etc/openvpn/general.ovpn** and put this in it:

```
dev tun
proto udp

# Here comes the port name. Remember this must be unique for
every OpenVPN server on your system!
port 1095
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/myserver.crt
key /etc/openvpn/easy-rsa/keys/myserver.key
dh /etc/openvpn/easy-rsa/keys/dh1024.pem

# Here comes the virtual ip range. Remember this must be unique
for every OpenVPN server on your system!
server 10.8.0.0 255.255.255.0

# Here comes your server's lan subnet. Substitute with your own!
push "route 192.168.193.0 255.255.255.0"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"

# Float: if your client switches from one network to another
(which you might if you're testing) this tells OpenVPN to ignore
errors on packages that got lost because of that.
float

# Use a per-server logfile for easy troublesooting.
log-append /var/log/openvpn-1095-general.log
```

Create an OpenVPN config with server side breakout

This is essentially the same as the local breakout, but call this file **/etc/openvpn/routeall.ovpn** and add this line for server side breakout:

```
push "redirect-gateway def1"
```

change the port number and the virtual ip:

```
port 1094
server 10.9.0.0 255.255.255.0
```

and change the logfile to reflect this profile's name:

```
log-append /var/log/openvpn-1094-routeall.log
```

For troubleshooting, look in the logfiles. I like to use this method:

```
# tail -f /var/log/openvpn-1094-routeall.log
```

This keeps the logfile on the screen and updates it in realtime.

Restart the OpenVPN service:

```
# service openvpn restart
```

Create user profiles

We'll create a user profile for user Lucas. We need to do this for every user separately.

```
# cd /etc/openvpn/easy-rsa
# ./clean-all
# source ./vars
# ./build-key lucas
```

Enter your preferred values. You can leave **A challange password** and **An optional company** name empty.

```
Sign the certificate? y
Commit? y
```

Now send these three files to the client:
/etc/openvpn/easy-rsa/keys/**ca.crt**
/etc/openvpn/easy-rsa/keys/**lucas.key**
/etc/openvpn/easy-rsa/keys/**lucas.crt**

On the client

On the client, make a directory, put the three files in it (ca.crt, lucas.key and lucas.crt) and create a file called myserver-general.ovpn (or whatever name you would like to give the profile) and put this in it:

```
dev tun
client
proto udp
remote your.servers.fqdn.or.public.ip.address 1095
resolv-retry infinite
ca ca.crt
cert lucas.crt
key lucas.key
verb 3
```

'Verb 3' means the level of verbosity. If you need to troubleshoot the client, increase the level of verbosity for more status messages. Decrease for less.

Make a second file called myserver-routeall.ovpn based on myserver-general.ovpn but change the port number and virtual subnet to reflect your routeall OpenVPN service. You now have two OpenVPN profiles on your client.

Download and install an OpenVPN client you like and initiate the connection. Verify that both profiles work. If they don't, check the logfiles to find out why they don't and fix it. Check for typos, check for port forwarding if you need that. If they don't work, don't go through to the Google Authenticator part.

And now for the fun part.

Adding Google Authenticator to the mix

We use Linux's Pluggable Authentication Modules (PAM) system to require Google Authenticator when connecting to the OpenVPN server. At the moment of writing Google Authenticator is missing a piece that's necessary for us to use it so we'll download the source code, edit in the missing piece and compile it.

Install the pam developer tools:

```
# apt-get install libpam-dev
```

Prepare the Google Authenticator code

Download the Google Authenticator source code from Google, extract it and save it somewhere.

```
# mkdir ~/gauth
# cd ~/gauth
# wget https://google-authenticator.googlecode.com/files/libpam-
google-authenticator-1.0-source.tar.bz2
# tar xvf *.bz
# cd lib*
```

Open the file **Makefile** and between the license part and where it says 'VERSION := 1.0' add this line:

```
LDFLAGS="-lpam"
```

Save the file and compile it:

```
# make
# make install
```

Implementing Google Authenticator in OpenVPN

On your server open **/etc/openvpn/general.ovpn** (the other one as well if you like) and add this line:

```
plugin /usr/lib/openvpn/openvpn-auth-pam.so openvpn
```

Restart the OpenVPN servers:

```
# service openvpn restart
```

Make a PAM file that works with OpenVPN:

```
# cp /etc/pam.d/common-account /etc/pam.d/openvpn
```

Open **/etc/pam.d/openvpn** and add these lines:

```
auth requisite pam_google_authenticator.so forward_pass
auth required pam_unix.so use_first_pass
```

Now **as the user who will be using the OpenVPN connections**, execute

```
$ google-authenticator
```

…and follow the instructions. Meanwhile on your phone install Google Authenticator and create a profile with the information presented by google-authenticator on your server.

Executing google-authenticator adds a file **.google_authenticator** in the user’s home directory. This file must have no rights except read for the user:

```
$ chmod 400 /home/lucas/.google_authenticator
```

On the client, edit the .ovpn files and add:

```
auth-user-pass
```

Using OpenVPN with Google Authenticator

If all is well you now have a six digit number generator on your phone. Fire up the OpenVPN connection on your client and log in with these credentials:

```
username: yourusername
password: yourpassword573984
```

That’s right: the six digit Google Authenticator code is added directly to your password. So every time you log in, you have a unique password-six digit code combination.

If your username is lucas, your password is p@55w0rd and the current six digit code on your phone is 822546, you would log in with

```
username: lucas
password: p@55w0rd822546
```

If things don’t work, check OpenVPN’s log file on the client and on the server check **/var/log/auth.log** and **/var/log/openvpn-1095-general.log** (or whatever you’re having your OpenVPN server log to).

Tech Authenticator, Google, one time password, openvpn, otp 3 Comments

Recieve new post updates: Entries (RSS)
Recieve follow up comments updates: RSS 2.0

Written by Kapitein Vorkbaard

View all posts by: Kapitein Vorkbaard

« Een Debian-systeem backuppen en restoren

Using PuTTY and keyfiles to SSH into your Ubuntu 12.04 server »

Leave a comment

3 Comments.

Gabe

Hi Kapitein,
Excellent write-up. Having primarily been using CentOS and no experience with gauth outside of google accounts, your guide was most helpful.
Couple of items:
1 - In the Create User Profiles section, if you do the ./clean-all it will wipe what you did previously that being creating

2013-11-04

the CA and key server.
2 – The hidden file to change to read-only for me was actually:
“google_authenticator” not “google-authenticator”
Thanks again for taking time to share this, especially the google integration piece.
—
Gabe

Reply

Kapitein Vorkbaard
2013-11-04

Hi Gabe, thanks for you comments.
1 – Are you certain? What you say makes sense and I may have botched my process by trying it so many times on the same installation… I guess creating the user certs in a separate directory would fix it.
2 – Typo, fixed it. Thanks for the notice!

Reply

Frankie
2015-03-13

Con lo que comprendo, se debería darr de alta como autónomo en el momento en quee haya accedido a lla situación que lo fuerza a ello.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment