VTI Router Implementation

Table of Content

Disclaimer

VTI appliance created in EIS Group and published into public domain under GPL licence.

Software is based on OpenSource Linux software and is provided 'As Is', no guarantees.

This document describe files and objects, used in VTI router, and their layouts.
(Document source - https://docs.google.com/document/d/1JKRTOtmFycApsS8Fxa3RsCPOHr-RHjHau9XAPrkaIPY/edit#head - here is the recent version)
Project on github - https://github.com/eis-group-opensource/vti-router

VTI tunnels and their status.

Primary task for this router is to keep VTI tunnels, where VTI is for Virtual Tunnel Interface, which wraps around standard IPSEC tunnel, so that while network uses normal IPSEC protocol and normal IPSEC packets format, router see it not as abstract 'ipsec access list' (as in Cisco ASA or in policy based IPSEC) but as normal network interface, with IP address, netmask, possible routing over it and so on - when 2 routers are connected by VTI, everything which comes into vti interface on one router is received from vti interface on other router.

Components.

- For IPSEC and VTI support - strongswan. Other ‘swan’ systems do not support it or has serious bugs. It is located in /etc/strongswan
- Linux version - we use CentOS7, but update kernel to kernel 4. There is a good chance that it can work with Ubuntu, just it was not tested (and initial setup script will not work).
- For routing - quagga. It is not so stable and clean as Cisco IOS, but it works pretty well, if you do not try to make something fancy. Quagga is located in /etc/quagga.
- For VTI management - our scripts, mostly written on shell. They are located in /usr/local/scripts, together with template files for strongswan and quagga, which are installed by INITIAL_SETUP.sh and replace some default files in them.
- One extra component is health monitoring script; it is called by cron every 10 minutes. It was required, because strongswan do not provide persistent connections by itself, and all together router-to-AWS connections has a tendency to disappear or freeze, in rare cases. To prevent it, we added script which check enabled tunnels, and if some are down or can not ping, try to reestablish it (after it was created, we did not experience a single outage for a few month, running 2 VTI routers per each AWS network for redundancy).

Directories and files.

- **/usr/local/scripts** (symlink as ~root/scripts) - VTI management scripts and work directory for ‘awailable’ vti interfaces.
  - **WORK** - directory for work; place aws description files here for the parser. System creates WORK/`hostname -s`.d directory for ‘available’ vti interfaces (it creates vtiN.property file when yu run aws parser or add vti manually).
    - **WORK/`hostname -s`.d** - directory with ‘vtiN.properties’ files
  - **FILES.d** - contains files we copy into /etc/ for our scripts; they include fresh quagga config, very important up/down scripts for strongswan and our strongswan configuration files, crontab file for automated health monitoring and self repair, and so on.

  Scripts themselves.
  - **./INITIAL_SETUP.sh** - it configure system itself (very first time) and can reconfigure (remove old configuration and create fresh empty one) IPSEC and routig system(s), and can add necessary rpm’s (except kernel) as well. It is interactive script, just call it ‘cd ~/scripts; ./INITIAL_SETUP.sh’; It store server configuration in /etc/sysconfig/`hostname -s`.properties and can reuse them.
  - **./VTI_OPS.sh** (alias **VTI**) - main script to add / remove / enable / disable / check status. System creates shortcut to it, which can be called ‘**VTI**’ as a root from any place. Typical commands are:

        - VTI -add vti1 vti2 - create files for vti1 and vti2 (they must be ‘available’) and place it in DISABLED mode.
        - VTI -enable vti1 vti2 - enable (activate) previously created interfaces vti1 and vti2
        - VTI -disable vti1 vti2 - disable (deactivate) vti1 and vti2
        - VTI -remove vti1 vti2 - remove all files (except created when done ‘available) related to these interfaces (not just in strongswan, but in quaggam too).
    - **./VTI_ADD.sh** - used to create vti (property file) manually; it is interactive script which ask questions and creates description of interface. By default, property file is created in WORK/`hostname -s`.d directory, you can specify different one by -d option and add directory for other side of tunnel by -r option. Can be used for example to describe interconnection between our own routers.
    - **./VTI_ADDAWS.sh [options] aws_description_file [vti1 [vti2]]** - aws configuration parser; it read VPN description file, provided by AWS, and create property files (or file, if you configure single vti only) for vti interfaces (they became **AVAILABLE**).
    - **./VTI_RM.sh vtiN** - removes properties file (notice - you can remove properties of active VTI, it just wil make it unavailable for future -add operation).
    - Few other scripts are not used in normal life:
        - update_kernel.sh - used only when creatig appliance, no need to run it.
        - refresh_scripts.sh - used to resync scripts and files from this directory (from FILES.d) into the system, and when we maintain sccripts sources outside of router itself.
        - SYNC_AVAIL.sh - used only if you create properties in non standard place.
- **/etc/strongswan/ipsec-vti.sh** - main internal script which is called by strongswan when status of VTI changes; it can be updated by INITIAL_SETUP.sh . Do not change it!
- **/etc/strongswan/restart_vti.sh** - it must be called by cron, and it check enabled tunnels and try to reconnect them if they fail. It must be called every 5 - 10 minutes (and it is called every 10 minutes by default in appliance).
- **/etc/quagga/reset_vti.sh** - this script is called internally to let quagga router know about new vti interface. Do not change / remove.
- **/etc/strongswan/vti.d** - directory, where we create files for vti configuration in swan. For each active VTI interface (for example, vtiN), 3 files will be created:
    - /etc/strongswan/vti.d/**vtiN.secrets** - file with shared key
    - /etc/strongswan/vti.d/**vtiN.init** - vti description created for ipsec-vti.sh script. It contain information such as IP addresses, MTU, start/stop scripts and so on.
    - /etc/strongswan/vti.d/**vtiN.conf** - strongswan configuration for this VTI.
- **/etc/strongswan/vti.d/.DISABLED** - directory for disabled vti interfaces - vti.conf is crated in it first (so you can edit it before making active) and placed back here when yoiu ‘disable’ interface.

- **/etc/sysconfig/iptables -** this file is adjusted by the system so that IPSEC packets can pass thru.
- **/etc/quagga/bgp.conf** - file is adjusted by the system when you -add/-delete interface (by VTI command) Run ‘vtysh , then sh run’ to see configuration.

See APPENDIX for the script(s) arguments.

File formats.

You can edit some files manually, so we show file which are created, when new VTI is created.

- When aws VPN file is parsed or VRI added manually, file /usr/local/scripts/WORK/`hostname -s`.d/vtiN.properties is created. Here is example of this file:

```
### Generated Thu Oct 27 20:29:20 PDT 2016
VTI="vti1"
VTI_MTU="1420"
VTI_OIP_LOCAL="63.2xx.2xx.38"
VTI_OIP_REMOTE="52.25.31.90"
LOCAL_GW="63.2xx.2xx.1"
VTI_IIP_LOCAL="169.254.13.22/30"
VTI_IIP_REMOTE="169.254.13.21/30"
VTI_PSK="xxxxDEJB2wtXzc4jYRGvADIhVL5xxxx"
VTI_MARK="12"
#### OPTIONAL
ID="aws23_dev2lab23gw02_GW1"
AWS_ID="vpn-05e0fe17"
VTI_BGP_LOCAL_AS="65003"
VTI_BGP_REMOTE_AS="7224"
VTI_BGP_LOCAL_IP="169.254.13.22/30"
VTI_BGP_REMOTE_IP="169.254.13.21"
VTI_O1=
VTI_O2=
VTI_O3=
VTI_O4=
VTI_O5=
```

Most options are obvious; some need comment:
LOCAL_GW - when tunnel is created, system adds routing to the VTI_OIP_REMOTE address, via specified LOCAL_GW.
VTI_MARK - uniq number for this VTI, in 8-base (so it can not be 9 for example); auto generated.
ID - added to vtiN connection name. Can be empty or can be some comment you use to recognize VTI purpose. You will see it in ‘VTI -status’ output.
BGP options - you can specify unique local BGP for each bgp connection. If BGP options are not specified, bgp configuration will not be updated.
Extra options, not reflected herem are:

VTI_O1 ... VTI_O5 will be added into connection configuration for strongswan, so you can customize connection using any strongswan configuration options.

When this file exist, VTI -list show this vti in list of available, as here:

```
VTI -list
Active: *
Disabled: vti1 vti2
Available: vti1 vti2
```

- Next, vti interface must be added into the system (it will be added in disabled state). VTI -add vtiN creates vtiN.init and vtiN.secrets files in /etc/strongswan/vti.d, and vtiN.conf file in /etc/strongswan/vti.d/.DISABLED. Let's see these files:

Shared key is in *.secrets file:

```
cat /etc/strongswan/vti.d/vti1.secrets
63.2xx.2xx.38 52.25.xx.90 : PSK xxxxxEJB2wtXzc4jYRGvADIhVLxxxxx
```

Strongswan configuration is in .conf file:

```
cat /etc/strongswan/vti.d/.DISABLED/vti1.conf
#BEGIN:vti1
conn vti1.aws23_dev2lab23gw02_GW1
       left=63.2xx.2xx.38
       right=52.25.1xx.90
       auto=start
       mark=12
       forceencaps=yes
#END:vti1
```

If you specified VTI_O1 - VTI_O5 in properties file, they will be added here, too. As you can see, connection name ALWAYS starts with vti interface name, but then can has a suffix with any mnemonics name.

And IP configuration file, used by our script:

```
cat /etc/strongswan/vti.d/vti1.init
# Created Thu Oct 27 20:37:06 PDT 2016
# aws23_dev2lab23gw02_GW1
# vpn-05e0fe17
VTI_INTERFACE=vti1
VTI_LOCALADDR=169.254.13.22/30
VTI_REMOTEADDR=169.254.13.21/30
VTI_MTU=1420
VTI_UP="/etc/quagga/reset_vti.sh vti1"
VTI_DOWN="/etc/quagga/reset_vti.sh vti1"
```

Few extra lines will be added into quagga configuration, too:

```
router bgp 65003
 ...
 ...
 neighbor 169.254.13.21 remote-as 7224
 neighbor 169.254.13.21 route-map from_aws in
...

ip route 52.25.xx.90/32 63.2xx.2xx.1
```

So, system added bgp peer, added inbound filter (if route-map exists, else this line will not be added), and most important, static route to the remote IP (remember, even if vti router is connected to OUTSIDE network, it do not have default routing here and access is restricted in iptables.

Once we verified configuration (and adjusted it) we can activate VTI interface.

```
[root@dev2-lab23-gw02 scripts]# VTI -list
Active: *
Disabled: vti1 vti2
Available: vti1 vti2
[root@dev2-lab23-gw02 scripts]# VTI -enable vti1
vti1 enabled
Reloading strongSwan IPsec configuration...
generating QUICK_MODE request 410661295 [ HASH SA No KE ID ID ]
sending packet: from 63.2xx1.2xx.38[4500] to 52.25.xx.90[4500] (460 bytes)
received packet: from 52.25.xx.90[4500] to 63.2xx.2xx.38[4500] (444 bytes)
parsed QUICK_MODE response 410661295 [ HASH SA No KE ID ID ]
CHILD_SA vti1.aws23_dev2lab23gw02_GW1{10} established with SPIs cf6f7955_i 7a71f5ec_o and TS
0.0.0.0/0 === 0.0.0.0/0
connection 'vti1.aws23_dev2lab23gw02_GW1' established successfully
Reloading strongSwan IPsec configuration...
[root@dev2-lab23-gw02 scripts]# VTI -list
Active: vti1
Disabled: vti2
Available: vti1 vti2
```

This command moved vti.conf from .DISABLED directory and re-read strongswan configuration; in turn, strongswan establish connection and call /etc/strongswan/vti-ipsec.sh script which establish VTI interface (zcript adjust some system parameters, too).

See interface status:

```
 VTI -status
vti1.aws23_dev2lab23gw02_GW1: #3, ESTABLISHED, IKEv1, a36f0a2326887e3d:ad17b945a3fc85bd
  local  '63.251.228.38' @ 63.251.228.38[4500]
  remote '52.25.31.90' @ 52.25.31.90[4500]
  AES_CBC-256/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/MODP_2048_256
  established 203s ago, rekeying in 27829s
  vti1.aws23_dev2lab23gw02_GW1: #9, reqid 3, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
        installed 203s ago, rekeying in 2432s, expires in 3397s
        in  c9e781f5,         0 bytes,       0 packets,     7s ago
        out fcd13ebc,  60 bytes,      1 packets,     86s ago
```

```
       local  0.0.0.0/0
       remote 0.0.0.0/0
  vti1.aws23_dev2lab23gw02_GW1: #10, reqid 3, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
       installed 198s ago, rekeying in 2388s, expires in 3402s
       in  cf6f7955,         0 bytes,       0 packets,     7s ago
       out 7a71f5ec,         0 bytes,       0 packets
       local  0.0.0.0/0
       remote 0.0.0.0/0
  vti1.aws23_dev2lab23gw02_GW1: #11, reqid 3, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
       installed 194s ago, rekeying in 2451s, expires in 3406s
       in  c0f95cce,   2261 bytes,   33 packets,    7s ago
       out 61eb688d,   2842 bytes,   44 packets,    7s ago
       local  0.0.0.0/0
       remote 0.0.0.0/0
vti1: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
10.20.23.0     169.254.13.21   255.255.255.0   UG           0 0            0 vti1
169.254.13.20   0.0.0.0       255.255.255.252 U             0 0            0 vti1
```

As you can see, we have now 10.20.23.0/24 routing, it comes from AWS by BGP.

We can disable interface, it just move vti1.conf back to DISABLED and re-read configuration so strongswan disconnect tunnels.

```
 VTI -disable vti1
Reloading strongSwan IPsec configuration...
closing CHILD_SA vti1.aws23_dev2lab23gw02_GW1{9} with SPIs c9e781f5_i (0 bytes) fcd13ebc_o
(60 bytes) and TS 0.0.0.0/0 === 0.0.0.0/0
sending DELETE for ESP CHILD_SA with SPI c9e781f5
generating INFORMATIONAL_V1 request 908048759 [ HASH D ]
sending packet: from 63.251.228.38[4500] to 52.25.31.90[4500] (92 bytes)
closing CHILD_SA vti1.aws23_dev2lab23gw02_GW1{10} with SPIs cf6f7955_i (0 bytes) 7a71f5ec_o
(0 bytes) and TS 0.0.0.0/0 === 0.0.0.0/0
sending DELETE for ESP CHILD_SA with SPI cf6f7955
generating INFORMATIONAL_V1 request 2232773923 [ HASH D ]
sending packet: from 63.251.228.38[4500] to 52.25.31.90[4500] (92 bytes)
closing CHILD_SA vti1.aws23_dev2lab23gw02_GW1{11} with SPIs c0f95cce_i (4423 bytes)
61eb688d_o (5056 bytes) and TS 0.0.0.0/0 === 0.0.0.0/0
IKE_SA [3] closed successfully
vti1 disabled
Reloading strongSwan IPsec configuration...
[root@dev2-lab23-gw02 scripts]# VTI -status
[root@dev2-lab23-gw02 scripts]#
```

And finally we can delete interface from the system, so it’s IP and number can be reused:

```
VTI -delete vti1

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

dev2-lab23-gw02.sjclab.exigengroup.com#      conf t
dev2-lab23-gw02.sjclab.exigengroup.com(config)#      no ip route 52.25.xx.xx/32 63.2xx.2xx.1
dev2-lab23-gw02.sjclab.exigengroup.com(config)#      exit
```

```
dev2-lab23-gw02.sjclab.exigengroup.com#      write mem
Building Configuration...
Configuration saved to /etc/quagga/zebra.conf
Configuration saved to /etc/quagga/ospfd.conf
Configuration saved to /etc/quagga/bgpd.conf
[OK]
dev2-lab23-gw02.sjclab.exigengroup.com#

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

dev2-lab23-gw02.sjclab.exigengroup.com# conf t
dev2-lab23-gw02.sjclab.exigengroup.com(config)# router bgp 65003
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# no neighbor 169.254.13.21 remote-as
7224
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# no neighbor 169.254.13.21 local-as
65003
% Specify remote-as or peer-group commands first
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# exit
dev2-lab23-gw02.sjclab.exigengroup.com(config)#                    exit
dev2-lab23-gw02.sjclab.exigengroup.com# write mem
Building Configuration...
Configuration saved to /etc/quagga/zebra.conf
Configuration saved to /etc/quagga/ospfd.conf
Configuration saved to /etc/quagga/bgpd.conf
[OK]
dev2-lab23-gw02.sjclab.exigengroup.com#
Reloading strongSwan IPsec configuration...
```

And remove property file cd ~/scripts; ./VTI_RM.sh vti1

That's it. THis is full life circle of VTI interface and all commands involved in it's management.

AZURE specifics.

AZURE mode added into VTI_ADD.sh script. Separate document created with an example of AZURE configuration.

APPENDIX. Scripts options.

VTI or scripts/VTI_OPS.sh:

```
VTI
Usage: /usr/local/scripts/VTI_OPS.sh [-e </etc>] [-d <DIR>] [-noreload] [-nobgp]
{-add|-delete|-enable|-disable} vti1 ...
   /usr/local/scripts/VTI_OPS.sh [-e </etc>] -d <dir>] -status|-list [vtiN]
```

VTI_ADDAWS.sh

```
./VTI_ADDAWS.sh
Using directory WORK/dev2-lab23-gw02.d - you can change it by -d <dir> option
Usage: ./VTI_ADDAWS.sh [-d directory] [-f] <generic-config-file-from-amazon.txt> [vti1
[vti2]]
```

VTI_ADD.sh

```
-d <work directory> - default WORK/`hostname -s`.d
-r <work directory for reverse tunnel> - if specified, it creates properties for 2 ends of
tunnel.
```