UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea triennale in *Ingegneria Informatica e dell'Automazione*

Implementazione di OpenVPN su router 4G per site-to-site vpn in ambiente CG-NAT

#TODO Study and configuration of a site-to-site VPN in CG-NAT environment

Relatore:
Prof. Ennio Gambi

Correlatore:
Ing. Adelmo De Santis

Tesi di Laurea di:
Alessandro Illuminati
matricola 1078466

Anno accademico 2020-2021

Prefazione

#TODO da riscrivere
Nell'ambito del mio percorso universitario ho avuto modo di approfondire le tematiche relative al mondo delle reti e del networking, a tal proposito grazie alla possibilità offerta dal Dipartimento di Ingegneria dell'Informazione, dal Prof. Ennio Gambi e dall'Ing. Adelmo De Santis ho conseguito con successo la certificazione "*HUAWEI HCIA Routing and Switching*". Successivamente, grazie alle competenze acquisite, ho collaborato con alcuni miei colleghi per progettare e realizzare una implementazione di una VPN site-to-site attraverso una connessione radiomobile per conto dell'azienda Esse-ti S.r.l.
In questo elaborato verranno esposte le principali fasi del progetto realizzato, ponendo un particolare focus sulle problematiche iniziali affrontate e all'architettura di rete nel cui ambito è stata realizzata la comunicazione tramite un canale sicuro.

Indice

1 Introduzione **1**
1.1 Intro su ip/tcp 1
1.2 openvpn 1
1.3 openwrt 1

2 Overview dell'architettura e delle componenti utilizzate **2**
2.1 Obbiettivo da ottenere 2
2.2 Specifiche dei componenti 3

3 Configurazione del server **7**
3.1 Overview della configurazione e prerequisiti 7
3.2 Creazione della Public key infrastructure Certificate Authority (PKI CA) 7
3.3 Configurazione della PKI di OpenVPN 10
3.4 Firma del certificato opnevpn dalla CA 11
3.5 generazione tls-crypt pre-shared key 13
3.6 Generazione delle chiavi per i clients 13

Elenco delle figure

2.1 Schema concettuale dell'obbiettivo da raggiungere 2
2.2 Schema concettuale dell'architettura che si dovrà implementare 3
2.3 Topologia virtuale 3
2.4 4G.Router 4

Nella didascalia di ogni immagine vi è il link della pagina web da cui è stata presa, inoltre, sono citate anche accanto ai link nella sitografia.

Capitolo 1

Introduzione

#TODO da scrivere da 0

1.1 Intro su ip/tcp

1.2 openvpn

1.3 openwrt

Capitolo 2

Overview dell'architettura e delle componenti utilizzate

2.1 Obbiettivo da ottenere

In una collaborazione tra il Dipartimento di Ingegneria dell'Informazione e l'azienda **Esse-ti S.R.L.** ci è stato esposto un progetto che consiste nel:

- fornire a dei clienti un router 4G, su cui possono essere connessi vari dispositivi, ad es. di tipo domotico.
- rendere questi dispositivi accessibili ai clienti attraverso internet

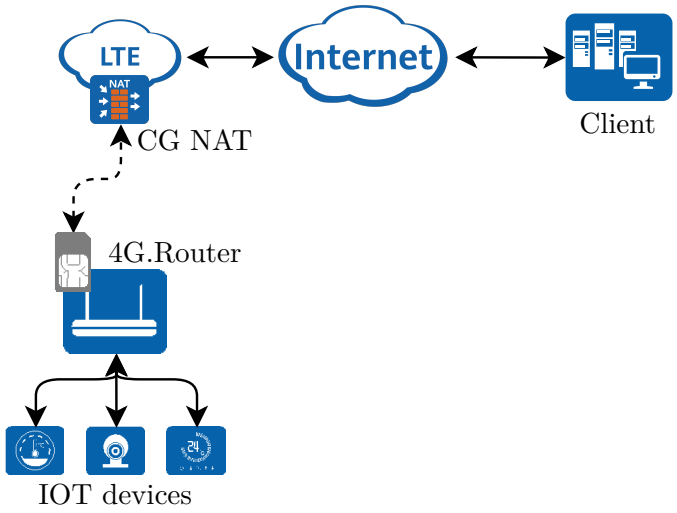


Figura 2.1: Schema concettuale dell'obbiettivo da raggiungere

Data la presenza del CG-NAT si vede subito che non è realizzabile a meno che il cliente non abbia un'IP pubblico e la sua macchina venga configurata opportunamente. Questo però non è possibile

nel caso generale, quindi per risolvere efficacemente questa topologia si deve necessariamente introdurre una terza macchina provvista di IP pubblico e che funga da ponte tra il 4G.Router e il cliente.

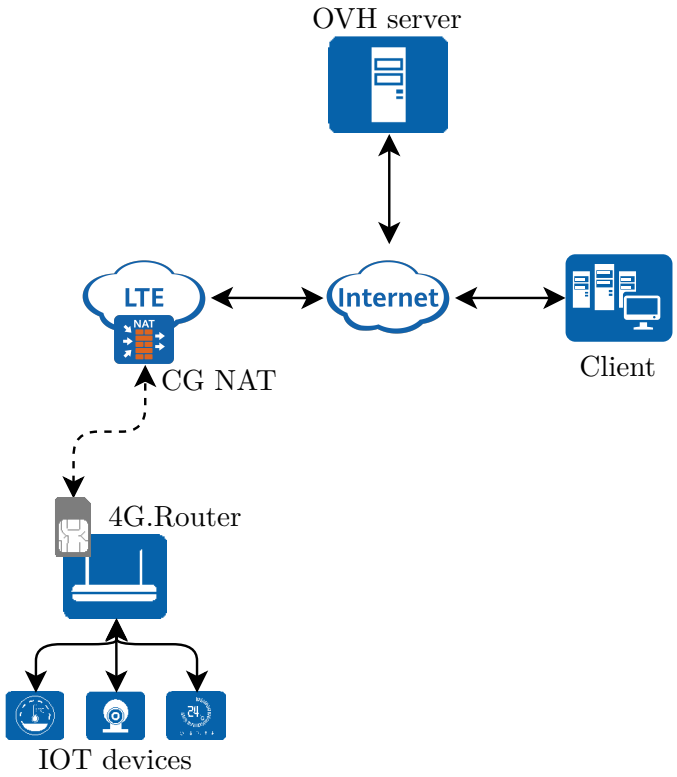


Figura 2.2: Schema concettuale dell'architettura che si dovrà implementare

In questo modo si può configurare una VPN sul server OVH e connettervi sia il 4G.Router che la macchina del cliente. In questo modo l'unica configurazione che il cliente dovrà fare è l'installazione di un cliente VPN, ciò è il minimo possibile di configurazione.

La configurazione virtuale vista dal 4G.Router e dai clienti sarà quindi:

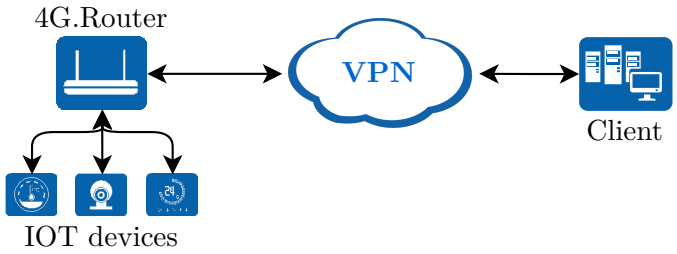


Figura 2.3: Topologia virtuale

2.2 Specifiche dei componenti

i componenti necessari sono:

- Esse-ti 4G.Router
- Server
- Host domotico
- Macchina del cliente

vediamo le caratteristiche minime che i componenti dovranno avere:

Esse-ti 4G.Router

Ci è stato fornito dall'azienda Esse-ti, consiste in un gateway 4G con funzionalità di router. Le specifiche complete possono essere trovate sul sito del produttore (link)

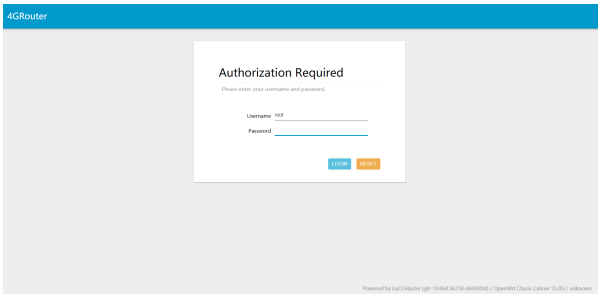
Figura 2.4: 4G.Router

Per l'implementazione di questa architettura sono necessarie solo un sub-set delle specifiche:

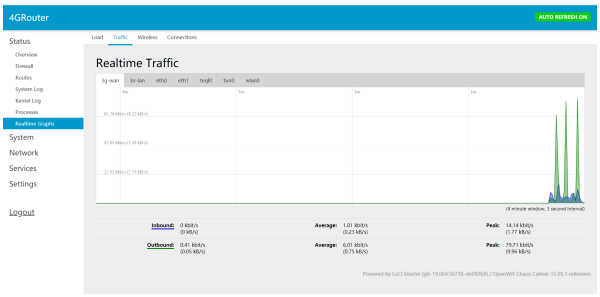
- Access Point wireless per offrire connettività Internet Wi-Fi a dispositivi wireless
- Client Dynamic DNS per consentire all'utente di raggiungere da remoto, tramite Internet, il router stesso e tutti i dispositivi connessi via Wi-Fi o porta LAN
- Gateway telefonico per consentire l'invio e la ricezione di chiamate attraverso la rete 4G LTE/UMTS/GSM a telefoni fissi, combinatori o altri dispositivi telefonici collegati all'ingresso FXS

Presenta inoltre come sistema operativo una versione personalizzata di OpenWrt.

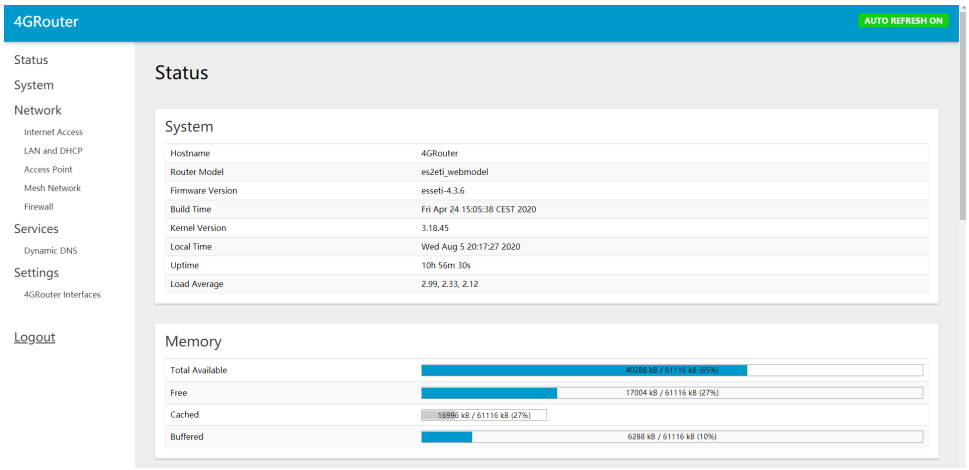
La configurazione del dispositivo puo' essere fatta sia da terminale, entrando in ssh, sia da interfaccia web:



(a) Schermata di autenticazione



(b) Grafico del traffico



(c) Schermata con stato riassuntivo

L'interfaccia web e' una versione personalizzata di Luci.

Per semplicita' si fara' riferimento all'*Esse-ti 4G.Router* chiamandolo semplicemente router.

VPS OVHCloud

La VPS ha il solo vincolo di dover avere un'ip pubblico e una connessione a internet abbastanza veloce. Dovra' infatti sopportare un traffico simmetrico in upload / download.

Per la realizzazione della topologia e' stata selezionata una macchina una VPS del provider OVHCloud, con le seguenti caratteristiche:

- 2 core virtuali
- 4Gb di memoria ram
- 80Gb di storage NVMe
- 500Mbps simmetrici di banda
- ipv4 pubblico
- Ubuntu 16.04

Per semplicita' si fara' riferimento alla *VPS OVHCloud* come server.

Host domotico

Per effettuare le varie operazioni di testing e' stato aggiunta raspberry pi che ha svolto il ruolo di "host domotico". Sono state fatti test con ping e iperf per testare che tutta la topologia sia stata configurata correttamente.

Macchina del cliente

Deve poter essere una qualunque macchina, non ha vincoli di sistema operativo

Necessita di avere il client openvpn installato:

- con sistema operativo Windows si deve scaricare l'eseguibile dal sito ufficiale
- su linux e' sufficiente cercare nei repository ufficiali della distribuzione che si sta usando.

Capitolo 3

Configurazione del server

3.1 Overview della configurazione e prerequisiti

In questa sezione andremo a installare e configurare OpenVPN server sulla VPS di OVHCloud.

I pacchetti necessari sono `openvpn` e `easy-rsa`, che possono essere installati con:

```
$ sudo apt-get update
$ sudo apt-get install -y openvpn easy-rsa
```

E' inoltre necessario avere un editor di testo, ad es. `nano` o `vim`

3.2 Creazione della Public key infrastructure Certificate Authority (PKI CA)

easy-rsa is a CLI utility to build and manage a PKI CA. In laymen's terms, this means to create a root certificate authority, and request and sign certificates, including intermediate CAs and certificate revocation lists (CRL).

La CA puo' essere configurata sulla stessa macchina dove e' stato installato opnevpn, ma cio' e' sconsigliato per motivi di sicurezza, supponiamo quindi di usare un secondo server chiamato *server CA*

La utility `easy-rsa` mette a disposizione il comando `make-cadir`, che permette di creare una cartella pronta ad ospitare la Certificate Authority.

Andiamo quindi a crearla, nella home ad esempio:

```
$ mkdir ~/openvpn-ca
$ ln -s /usr/share/easy-rsa/* ~/openvpn-ca/
$ chmod 700 /home/ubuntu/openvpn-ca/
$ cd openvpn-ca/
$ ./easyrsa init-pki

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /home/ubuntu/openvpn-ca/pki

$ la
easyrsa  openssl-easyrsa.cnf  pki  vars.example  x509-types
```

Ora si devono personalizzare le variabili `vars`, si puo' sia partire da un file vuoto oppure modificare `vars.example` per poi rinominarlo `vars`. Andiamo quindi a creare un nuovo file vars:

```
$ vim vars
set_var EASYRSA_REQ_COUNTRY  "IT"
set_var EASYRSA_REQ_PROVINCE "MC"
set_var EASYRSA_REQ_CITY     "Recanati"
set_var EASYRSA_REQ_ORG      "Esse-ti"
set_var EASYRSA_REQ_EMAIL    "s.gasparrini@esse-ti.it"
set_var EASYRSA_REQ_OU       "Esse-ti"

set_var EASYRSA_ALGO         "ec"
set_var EASYRSA_DIGEST       "sha512"
```

Le variabili nel primo blocco determinano i dati che poi verranno registrati nei certificati.

Le ultime 2 sono opzioni di sicurezza, in particolare si setta l'algoritmo di cifratura

A questo punti si deve laciare il comando `build-ca` per costruire la CA:

```
$ ./easyrsa build-ca

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020

Enter New CA Key Passphrase:
Re-Enter New CA Key Passphrase:
read EC key
writing EC key

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
 ↪  DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:

CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/home/ubuntu/openvpn-ca/pki/ca.crt

```

Eseguendo il comando verra' chiesto di inserire una passshare, che verra' usata per criptare la chiave privata appena generata. Il secondo propt e' relativo al nome da dare alla certificazione, in questo caso e' stato lasciato il valore di default `Easy-RSA CA`.

Se si legge il file `/openvpn-ca/pki/ca.crt` si vedra' il classico formato da certificato:

```
$ cat ca.crt
-----BEGIN CERTIFICATE-----
MIIB/TCCAYKgAwIBAgIUCYshxSm8eH1mf504HqQqULfVrakwCgYIKoZIzj0EAwQw
[...]
a3Qts307lAO5q49hK8hn5h43w+2MHvaiC0jXtCJp+hvR
-----END CERTIFICATE-----
```

3.3 Configurazione della PKI di OpenVPN

Il procedimento e' simile al precedente, ma questa volta va eseguito sul server.

Creiamo quindi una cartella per ospitare la PKI, es `/openvpn-pki`, e linkiamo `easy-rsa`. Inoltre limitiamo i permessi all'utente non root che stimao usando, in questo caso "ubuntu".

```
$ mkdir ~/openvpn-pki
$ ln -s /usr/share/easy-rsa/* ~/openvpn-pki/
$ sudo chown ubuntu ~/openvpn-pki/
$ chmod 700 ~/openvpn-pki/
$ cd ~/openvpn-pki/
```

Andiamo a creare un file `vars`:

```
$ vim vars
set_var EASYRSA_ALGO    "ec"
set_var EASYRSA_DIGEST  "sha512"
```

e concludiamo la creazione della PKI con il comando:

```
$ ./easyrsa init-pki

Note: using Easy-RSA configuration from: ./vars

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /home/ubuntu/openvpn-pki/pki

```

Now that your OpenVPN server has all the prerequisites installed, the next step is to generate a private key and Certificate Signing Request (CSR) on your OpenVPN server. After that you'll transfer the request over to your CA to be signed, creating the required certificate. Once you have a signed certificate, you'll transfer it back to the OpenVPN server and install it for the server to use.

Come nome e' stato scelto "server":

```
$ ./easyrsa gen-req server nopass

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020
Generating an EC private key
writing new private key to
↪ '/home/ubuntu/openvpn-pki/pki/private/server.key.438W2xM0g9'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
↪ DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [server]:

Keypair and certificate request completed. Your files are:
req: /home/ubuntu/openvpn-pki/pki/reqs/server.req
key: /home/ubuntu/openvpn-pki/pki/private/server.key

```

This will create a private key for the server and a certificate request file called server.req. Copy the server key to the /etc/openvpn/server directory:

```
$ sudo cp /home/ubuntu/openvpn-pki/pki/private/server.key
↪ /etc/openvpn/server/
```

Il secondo file creato, `server.req`, corrisponde ad una *Certificate Signing Request (CSR)* che va firmata e validata dalla CA. In questo modo ogni client che si fida della CA si fidera' di conseguenza del server OpenVPN

3.4 Firma del certificato opnevpn dalla CA

Dobbiamo quindi copiare il file `server.req` nel *server CA*, possiamo qualunque metodo purche' sia sicuro, ad esempio con `scp`:

```
$ scp -3
↪  ubuntu@openvpn_server:/home/ubuntu/openvpn-pki/pki/reqs/server.req
↪  ubuntu@ca_server:/tmp
```

Dobbiamo qundi spostarci sul server CA e importare la *certificate request* e firmarlo:

```
$ cd ~/openvpn-ca
$ ./easyrsa import-req /tmp/server.req server
$ ./easyrsa sign-req server server
Using configuration from /home/ubuntu/openvpn-ca/pki/safessl-easyrsa.cnf
Check that the request matches the signature
Signature ok
The Subject\'s Distinguished Name is as follows
commonName            :ASN.1 12:'ChangeMe'
Certificate is to be certified until Mar 11 15:50:45 2025 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated
```

Verra' creato un file in `/openvpn-ca/pki/issued` chiamato `server.crt` che conterra' la chiave pubblica che verra' usata dal server openvpn e inoltre la firma della CA.

Ora si devono copiare i file `ca.crt` e `server.crt` dal *server CA* al *server OpnenVPN*:

```
$ scp -3 ubuntu@ca_server:/home/ubuntu/openvpn-ca/pki/issued/server.crt
↪  ubuntu@openvpn_server:/tmp
$ scp -3 ubuntu@ca_server:/home/ubuntu/openvpn-ca/pki/ca.crt
↪  ubuntu@openvpn_server:/tmp
```

Possiamo quindi tornare sul *server OpenVPN* e copiare i 2 file da `/tmp` a `/etc/openvpn/server` :

```
$ sudo cp /tmp/server.crt /etc/openvpn/server
$ sudo cp /tmp/ca.crt /etc/openvpn/server
```

3.5 generazione tls-crypt pre-shared key

Per aumentare ulteriormente la sicurezza del nostro *server OpenVPN* possiamo creare un'ulteriore chiave, che consiste un una chiave preshared che verra' inserita in tutte le configurazioni e serve ad offuscare il certificato in fase di validazione. Quindi in caso di attacco si dovra' conoscere anche questa chiave.

La creazione va fatta sul *server OpenVPN*:

```
$ cd ~/openvpn-pki/
$ openvpn --genkey --secret ta.key
```

il file generato `ta.key` dovra' essere copiato nella directory del server openvpn:

```
$ sudo cp ta.key /etc/openvpn/server
```

3.6 Generazione delle chiavi per i clients

Creiamo una cartella nella home che ospitera' le chiavi dei client e le configurazioni openvpn:

```
$ mkdir -p ~/client-configs/keys
$ chmod -R 700 ~/client-configs
```

Bibliografia

[1] https://info.support.huawei.com/network/imagelib/getImagePartList?product_family=Router&product_type=Access%20Router%7CIOT%20Gateway&domain=&lang=en