Homework 7: Playing with WireGuard

CNS Course Sapienza

Riccardo Salvalaggio 1750157

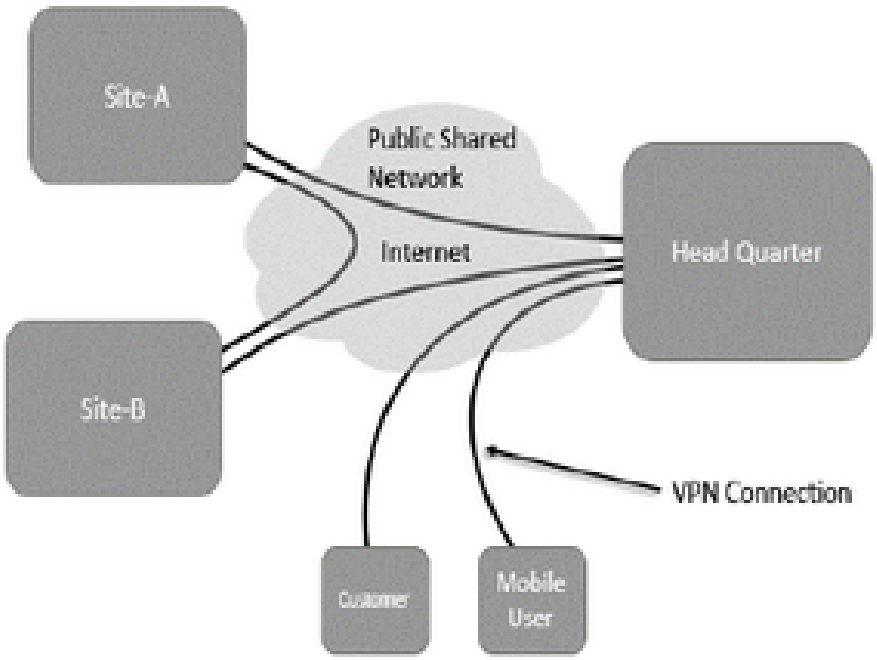
18/12/2020

1 Introduction

In this paper I am going to show how to create,setup and start a VPN with wireguard, a famous VPN manager. The scenario I am assuming is the following one:
I have an ftp server on my machine that I want to access by a remote host in a secure way, the remote host knows the public key of the server (asymmetric encryption) and its address, so it'll connect encrypting the communication. In this case of study we don't need explicit signatures, hashing functions or encryption algorithms (even if Wireguard uses its own one) and so on because we are going to put a secure tunnel between the two hosts where only them can communicate so we are almost sure that no one can manipulate the messages.

2 VPN - Virtual Private Network

VPN (Virtual Private Network) is a networking architecture which is implemented over public network to support privacy in shared public network, it emerged as a cost efficient and reliable solution in networking and telecommunication organizations. VPN are most favorable part of any IT industry because it saves the huge cost of infrastructure by using the public Internet to establish highly secure communication medium from corporate- office to remote sites and remote users.



VPN uses tunneling protocol to support its functionality. Tunneling protocol provides a secure mode of transport for the network services which elemental network does not support directly. The VPN service can be looked from the perspective of different stockholders, presenting the views of the user, customer, network provider and service provider.

2.1 Advantages and disadvantages

	ADVANTAGES	DISADVANTAGES
PPTP	• Less network overhead. • PKI not required. • More connections of PPTP support in VPN server. • In PPTP NAT traversal, NAT compatibility is supported.	• Security and firewall problems. • Support only one tunnel at a time for each user. • No additional authentication. • Access control based upon packet filtering.
L2TP	• Support both IP and Non-IP networks. • Multiple protocol support. • More authentication protocol supported. • Simultaneous multiple support of tunnel. • In IPSec NAT traversal, NAT compatibility is supported.	• Performance issues. • Less connections of L2TP support in VPN server.
IPSec	• Flexible • Configuration is not required to user devices. • More secured data and key exchange. • Supports integrity of the transmitted data. • Compatible with variety of encryption algorithm. • Optimal for gate-to-gate VPN solutions. • Best for always-on connections. • Compatible with NAT.	• Complexity is more. • Only identifies devices. • Routing capabilities not embedded. • Only IP protocol supported. • Reduces performance of the network. • Whole network or subnet will be vulnerable.
SSL/TLS	• VPN client not required. • More secured data and key exchange. • Permits particular resource access in the network.	• Only compatible with web-based applications. • More complex firewall configuration needed. • Increases IT hours in deflecting DoS attacks.

3 Wireguard

WireGuard is a free and open-source software application and communication protocol that implements virtual private network (VPN) techniques to create secure point-to-point connections in routed or bridged configurations. It is run as a module inside the Linux kernel (or the BSD kernel), and aims for better performance and more power saving than the IPsec and OpenVPN tunneling protocols.
Wireguard support ChaCha20 to encrypt the tunnel, UDP as communication protocol and can be used for various topologies: point-to-point, start and mesh.
Of course, topologies can be made, but not on the same tunnel.

4 Implementation

The WireGuard configuration is as simple as setting up SSH. A connection is established by an exchange of public keys between server and client. Only a client that has its public key in its corresponding server configuration file is allowed to connect (Asymmetric encryption). WireGuard sets up standard network interfaces (such as wg0 and wg1), which behave much like the commonly found eth0 interface. This makes it possible to configure and manage WireGuard interfaces using standard tools such as ifconfig and ip.
After having installed Wireguard, we start with the setup:

4.1 Server side

1) First we generate the private an public key in order to encrypt the tunnel:

```
umask 077
wg genkey | tee privatekey | wg pubkey > publickey
```

Last command save keys to be used in next steps.

2) Then we create the file wg0.conf in /etc/wireguard/ folder and configure it with some parameters:

```
[Interface]
PrivateKey = <Private Key generated before>
```

```
#address of the server
Address = 10.0.0.1/24, fd86:ea04:1115::1/64
ListenPort = 51820  #port for incoming requests.

#instruction to configure firewall for which connection to accept.
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT;
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE;
ip6tables -A FORWARD -i wg0 -j ACCEPT;
ip6tables -t nat -A POSTROUTING -o eth0 -j MASQUERADE;

PostDown = iptables -D FORWARD -i wg0 -j ACCEPT;
iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE;
ip6tables -D FORWARD -i wg0 -j ACCEPT;
ip6tables -t nat -D POSTROUTING -o eth0 -j MASQUERADE

SaveConfig = true     #remember of new peers.
```

3) Finally we setup firewall rules by adding ssh security.
SSH or Secure Shell is a cryptographic network protocol for operating network services securely over an unsecured network.

```
sudo ufw allow 22/tcp
sudo ufw allow 51820/udp
sudo ufw enable
```

These commands permit connections on port 22 if protocol tcp, and on port 51820 if protocol udp.
4) As last step we start WireGuard.

```
wg-quick up wg0
sudo systemctl enable wg-quick@wg0  #allows to restart on boot
```

4.2 Client side

1) As before we start generating the keys needed for the encryption with:

```
umask 077
wg genkey | tee privatekey | wg pubkey > publickey
```

2) We continue with the configuration in wg0.conf:

```
[Interface]
PrivateKey = <Output of private key file that contains
your private key>
```

```
Address = 10.0.0.2/24, fd86:ea04:1115::5/64
```

4.3 Connection

After configuring client and server, we have to put them in communication, and in order to do it we can edit che configuration files in this way:
Edit client's wg0.conf and add these lines:

```
[Peer]
PublicKey = <Server Public key> # in pure asymmetric approach
Endpoint = <Server Public IP>:51820
AllowedIPs = 10.0.0.2/24, fd86:ea04:1115::5/64
```

Finally we enable wireguard on both the hosts.

```
wg-quick up wg0
systemctl enable wg-quick@wg0
```

4.4 Test

In the end we can check that all is fine, pinging the server from the client:

```
ping 10.0.0.1
sudo wg
```

5 Real implementation

After the theoretical guide I simulated this situation in my environment. Of course, in order to have a client-server interaction you need two machines, so for instance two laptop or, as in my case, a pc and a smartphone. I started setupping an ftp server on my machine using classic procedure on Ubuntu and opened it from Filezilla (most famous ftp manager), then I connected to this server from my smartphone (Android) with an ftp application.
As explained at the beginning, the goal of this homework is to make secure the communication of a critical application, so the next step has been to setup the VPN following the guide explained at section 4 in order to have an encryption tunnel between my smartphone and my pc. As for the ftp, again I installed a specific application for VPN on my smartphone, in this case it is possible to get the mobile version of Wireguard, configure it and start it.
The ensurance of encryption is given testing the VPN, if request/respond works it means you are connected protected by the VPN and your communication has become private.