在Ubuntu系统手撸一个自动创建SSL证书的SHELL脚本

原创 Danileaf_Guo 于 2024-07-11 22:46:59 发布 阅读量880 收藏 16 点赞数 16

文章标签： ubuntu ssl linux 运维 服务器

正文共：1555 字 13 图，预估阅读时间：2 分钟

介于 CentOS 7停服的原因**（CentOS 7停服之后该怎么安装软件呢？）**，为了方便日常使用，还是要研究一下替换成 Ubuntu系统 **（如何在Ubun署KVM并创建虚拟机？）**，先尝试一下在Ubuntu系统中部署openVPN服务端。

对于openVPN或者SSL VPN而言，这种以加密协议为基础提供远程的安全连接服务，主要基于数字证书利用数字签名的方法对SSL服务器和SSL客户验证**（VSR自送的的SSL VPN功能，你要不要？）**。所以，使用Ubuntu系统，我们还是要先创建证书**（使用Easy-RSA配置生成SSL证书）**。

关于如何创建一份可以自动创建Easy-RSA证书的脚本，我们之前已经做了详细介绍**（手撸一个自动创建SSL证书的SHELL脚本）**。接下来，我们验证RSA在Ubuntu系统的使用是否存在差异，以及之前的脚本需要做何调整。

首先是软件安装，对应的命令为：

```
apt install -y easy-rsa
```

```
root@tietou-vm:~# apt list easy-rsa
Listing... Done
easy-rsa/jammy,jammy 3.0.8-1ubuntu1 all
root@tietou-vm:~# apt install -y easy-rsa
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libccid opensc opensc-pkcs11 pcscd
The following NEW packages will be installed:
  easy-rsa libccid opensc opensc-pkcs11 pcscd
0 upgraded, 5 newly installed, 0 to remove and 206 not upgraded.
Need to get 1,464 kB of archives.
After this operation, 4,960 kB of additional disk space will be used.
Get:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/universe amd64 libccid amd64 1.5.0-2 [83.1 kB]
Get:2 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-updates/universe amd64 pcscd amd64 1.9.5-3ubuntu1 [58
.1 kB]
Get:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/universe amd64 opensc-pkcs11 amd64 0.22.0-1ubuntu2 [9
33 kB]
Get:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/universe amd64 opensc amd64 0.22.0-1ubuntu2 [346 kB]
Get:5 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/universe amd64 easy-rsa all 3.0.8-1ubuntu1 [44.1 kB]
Fetched 1,464 kB in 6s (244 kB/s)
Selecting previously unselected package libccid.
(Reading database ... 213150 files and directories currently installed.)
Preparing to unpack .../libccid_1.5.0-2_amd64.deb ...
Unpacking libccid (1.5.0-2) ...
Selecting previously unselected package pcscd.
Preparing to unpack .../pcscd_1.9.5-3ubuntu1_amd64.deb ...
Unpacking pcscd (1.9.5-3ubuntu1) ...
Selecting previously unselected package opensc-pkcs11:amd64.
Preparing to unpack .../opensc-pkcs11_0.22.0-1ubuntu2_amd64.deb ...
Unpacking opensc-pkcs11:amd64 (0.22.0-1ubuntu2) ...
Selecting previously unselected package opensc.
Preparing to unpack .../opensc_0.22.0-1ubuntu2_amd64.deb ...
Unpacking opensc (0.22.0-1ubuntu2) ...
Selecting previously unselected package easy-rsa.
Preparing to unpack .../easy-rsa_3.0.8-1ubuntu1_all.deb ...
Unpacking easy-rsa (3.0.8-1ubuntu1) ...
Setting up libccid (1.5.0-2) ...
Setting up pcscd (1.9.5-3ubuntu1) ...
Created symlink /etc/systemd/system/sockets.target.wants/pcscd.socket → /lib/systemd/system/pcscd.socket.
pcscd.service is a disabled or a static unit, not starting it.
Setting up opensc-pkcs11:amd64 (0.22.0-1ubuntu2) ...
Setting up easy-rsa (3.0.8-1ubuntu1) ...
Setting up opensc (0.22.0-1ubuntu2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for mailcap (3.70+nmu1ubuntu1) ...
Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
root@tietou-vm:~#
```


如果转换成SHELL脚本，则可以是：

```
# 安装Easy-RSA
install_easyrsa() {
  echo "正在安装Easy-RSA..."
  apt install -y easy-rsa
  echo "Easy-RSA安装完成。"
```

```
}
```

然后是初始化Easy-RSA环境，在Ubuntu系统中，Easy-RSA的安装路径是/usr/share/easy-rsa/，需要依赖easyrsa文件来生成证书和密钥等文件。

```
root@tietou-vm:~# ll /usr/share/easy-rsa
total 116
drwxr-xr-x   3 root root  4096  7月 10 22:16 ./
drwxr-xr-x 277 root root 12288  7月 10 22:16 ../
-rwxr-xr-x   1 root root 76923 11月 19  2021 easyrsa*
-rw-r--r--   1 root root  4616  9月 10  2020 openssl-easyrsa.cnf
-rw-r--r--   1 root root  8925  9月 10  2020 vars.example
drwxr-xr-x   2 root root  4096  7月 10 22:16 x509-types/
root@tietou-vm:~#
```


可以看到，机构配置范例文件vars.example也位于这个目录，复制一份。

```
cp /usr/share/easy-rsa/vars.example /usr/share/easy-rsa/vars
```

```
root@tietou-vm:~# cp /usr/share/easy-rsa/vars.example /usr/share/easy-rsa/vars
root@tietou-vm:~# ll /usr/share/easy-rsa
total 128
drwxr-xr-x   3 root root  4096  7月 10 22:25 ./
drwxr-xr-x 277 root root 12288  7月 10 22:16 ../
-rwxr-xr-x   1 root root 76923 11月 19  2021 easyrsa*
-rw-r--r--   1 root root  4616  9月 10  2020 openssl-easyrsa.cnf
-rw-r--r--   1 root root  8925  7月 10 22:25 vars
-rw-r--r--   1 root root  8925  9月 10  2020 vars.example
drwxr-xr-x   2 root root  4096  7月 10 22:16 x509-types/
root@tietou-vm:~#
```


如果转换成SHELL脚本，则可以是：

```
# 初始化Easy-RSA环境
init_easyrsa() {
  echo "初始化Easy-RSA环境..."
  cd /usr/share/easy-rsa/
  # 复制vars.example为vars，避免覆盖
  if [ -f vars ]; then
    mv vars vars.backup
  fi
  cp vars.example vars
  echo "Easy-RSA环境初始化完成。"
}
```

然后我想修改机构信息中的部分字段为以下信息。

```
set_var EASYRSA_REQ_COUNTRY     "CN"
set_var EASYRSA_REQ_PROVINCE    "Beijing"
set_var EASYRSA_REQ_CITY        "Haidian"
set_var EASYRSA_REQ_ORG         "TIETOU_TECH"
set_var EASYRSA_REQ_EMAIL       "tietou@h3cadmin.cn"
set_var EASYRSA_REQ_OU          "Tietou_openVPN"
```

正常需要使用vi命令进行编辑，如果转换成SHELL脚本，则可以直接进行插入：

```
# 更新vars文件中的机构信息字段
update_vars() {
  echo "更新vars文件中的机构信息字段..."
  cd /usr/share/easy-rsa/
  # 向vars文件中插入机构信息字段
  echo 'set_var EASYRSA_REQ_COUNTRY     "CN"' >> vars
  echo 'set_var EASYRSA_REQ_PROVINCE    "Beijing"' >> vars
  echo 'set_var EASYRSA_REQ_CITY        "Haidian"' >> vars
  echo 'set_var EASYRSA_REQ_ORG         "TIETOU TECH"' >> vars
  echo 'set_var EASYRSA_REQ_EMAIL       "tietou@h3cadmin.cn"' >> vars
  echo 'set_var EASYRSA_REQ_OU          "Tietou openVPN"' >> vars
  echo "vars文件中的机构信息字段已更新。"
}
```

然后就是生成证书文件了，还是在/usr/share/easy-rsa，通过以下命令初始化**PKI（Public Key Infrastructure，公钥基础设施）**目录结构：

```
./easyrsa init-pki
```

```
root@tietou-vm:/usr/share/easy-rsa# ./easyrsa init-pki

Note: using Easy-RSA configuration from: /usr/share/easy-rsa/vars

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /usr/share/easy-rsa/pki


root@tietou-vm:/usr/share/easy-rsa# ll
total 132
drwxr-xr-x   4 root root  4096  7月 10 22:30 ./
drwxr-xr-x 277 root root 12288  7月 10 22:16 ../
-rwxr-xr-x   1 root root 76923 11月 19  2021 easyrsa*
-rw-r--r--   1 root root  4616  9月 10  2020 openssl-easyrsa.cnf
drwx------   4 root root  4096  7月 10 22:30 pki/
-rw-r--r--   1 root root  9194  7月 10 22:27 vars
-rw-r--r--   1 root root  8925  9月 10  2020 vars.example
drwxr-xr-x   2 root root  4096  7月 10 22:16 x509-types/
root@tietou-vm:/usr/share/easy-rsa#
```


执行命令之后，Easy-RSA会自动根据vars文件中的变量，在PKI目录下生成一份新的openssl -easyrsa.cnf文件。

```
root@tietou-vm:/usr/share/easy-rsa# ll pki/
total 32
drwx------ 4 root root 4096  7月 10 22:30 ./
drwxr-xr-x 4 root root 4096  7月 10 22:30 ../
-rw------- 1 root root 4616  7月 10 22:30 openssl-easyrsa.cnf
drwx------ 2 root root 4096  7月 10 22:30 private/
drwx------ 2 root root 4096  7月 10 22:30 reqs/
-rw------- 1 root root 4595  7月 10 22:30 safessl-easyrsa.cnf
root@tietou-vm:/usr/share/easy-rsa#
```

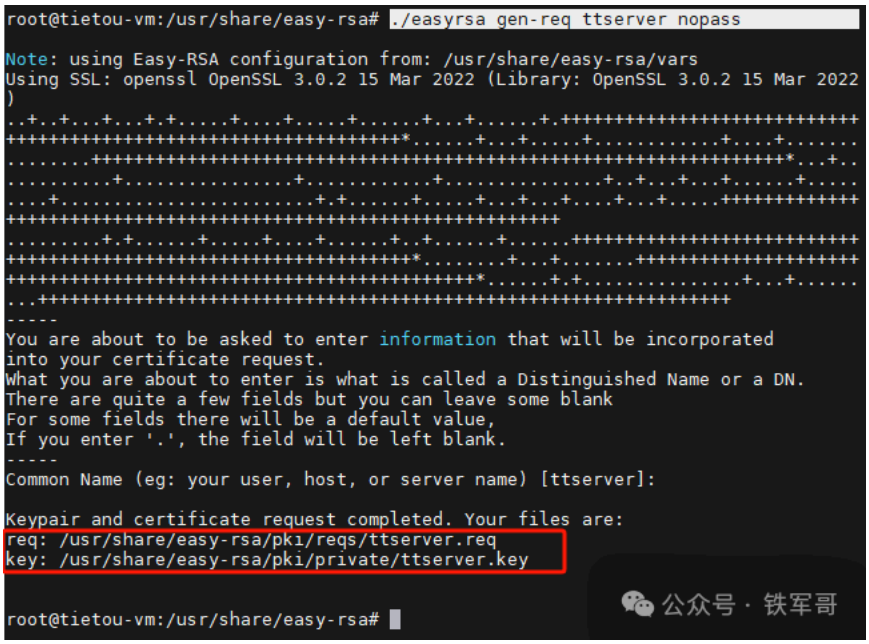

接下来创建根证书，用于**CA（Certificate Authority，证书颁发机构）**对之后生成的server 和client证书进行签名时使用。为了方便，我们带nopass证书，nopass表示不对CA密钥进行加密，这样在签署证书时就可以跳过密码验证。

```
./easyrsa build-ca nopass
```

```
root@tietou-vm:/usr/share/easy-rsa# ./easyrsa build-ca nopass

Note: using Easy-RSA configuration from: /usr/share/easy-rsa/vars
Using SSL: openssl OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022
)
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:
/usr/share/easy-rsa/pki/ca.crt


root@tietou-vm:/usr/share/easy-rsa#
```

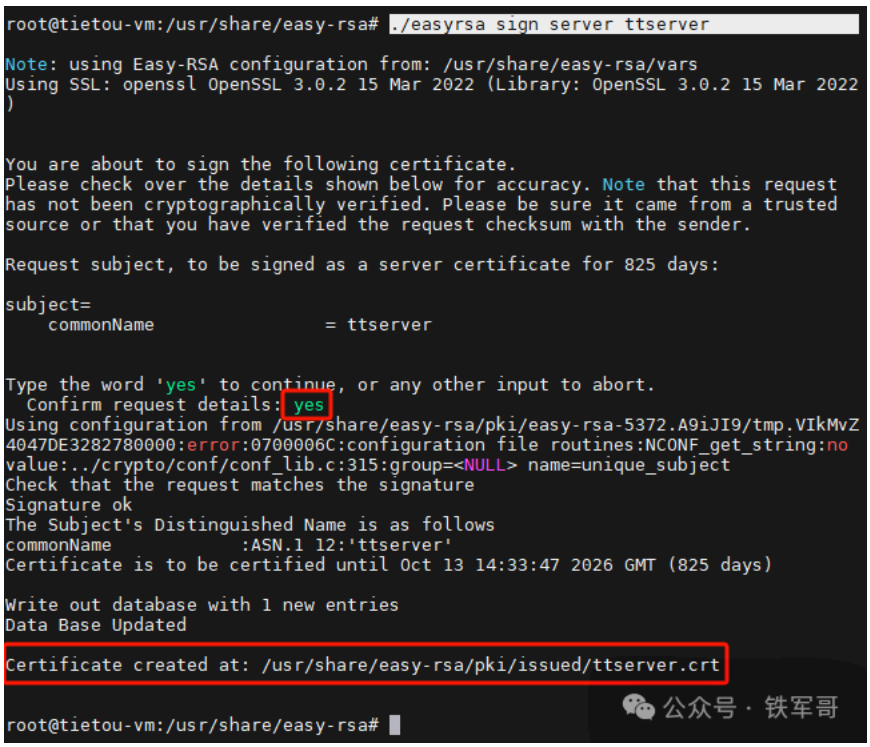

创建服务器端证书，同时指定nopass参数表示不对私钥进行加密。

```
./easyrsa gen-req ttserver nopass
```



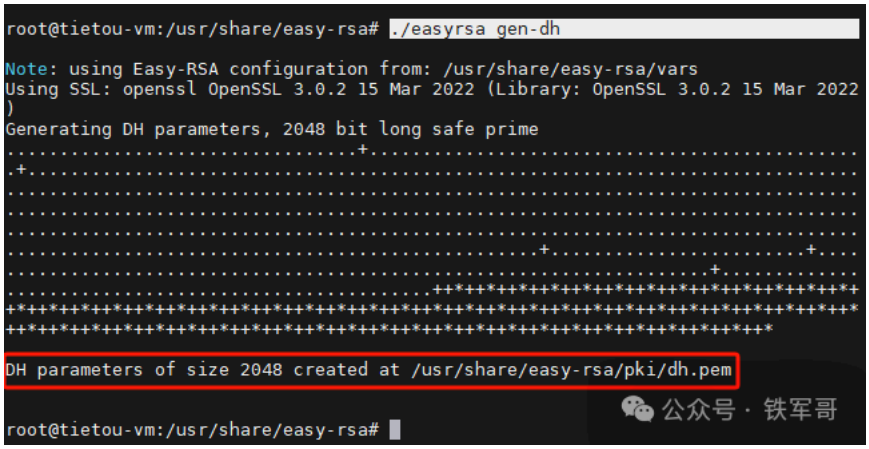
给服务器端证书ttserver进行签名。

```
./easyrsa sign server ttserver
```



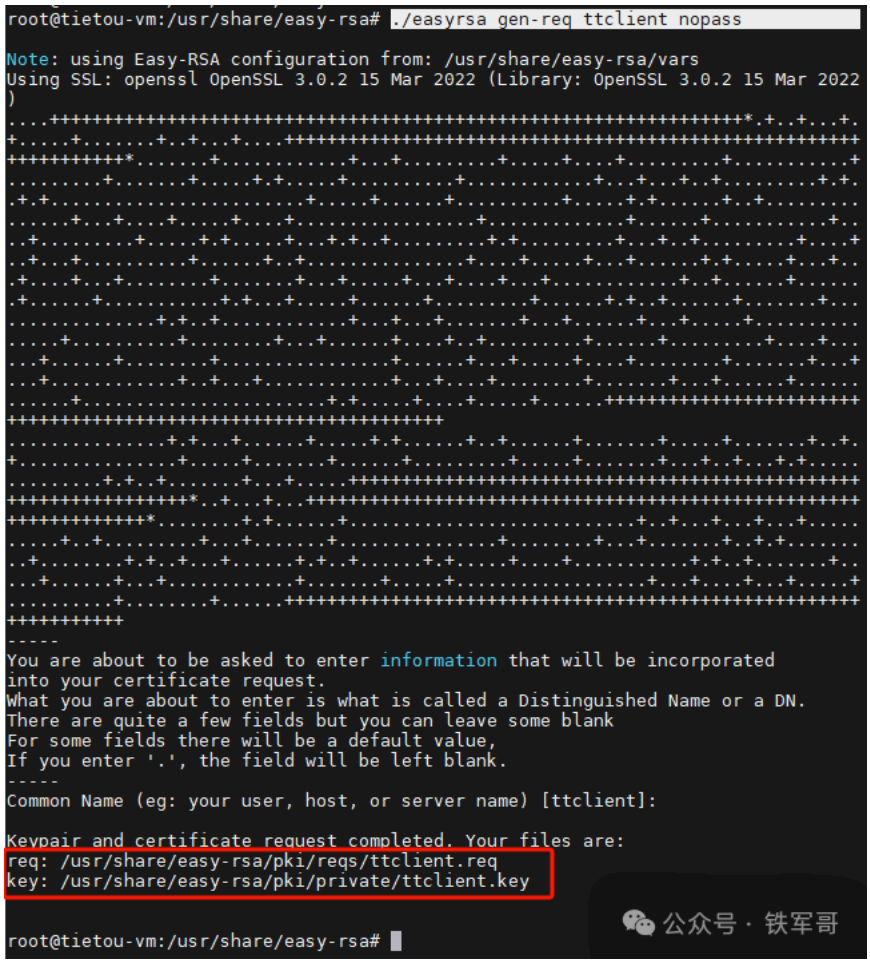
然后创建Diffie-Hellman文件，也就是秘钥交换时的DH算法，确保密钥可以穿越不安全网络。

```
./easyrsa gen-dh
```



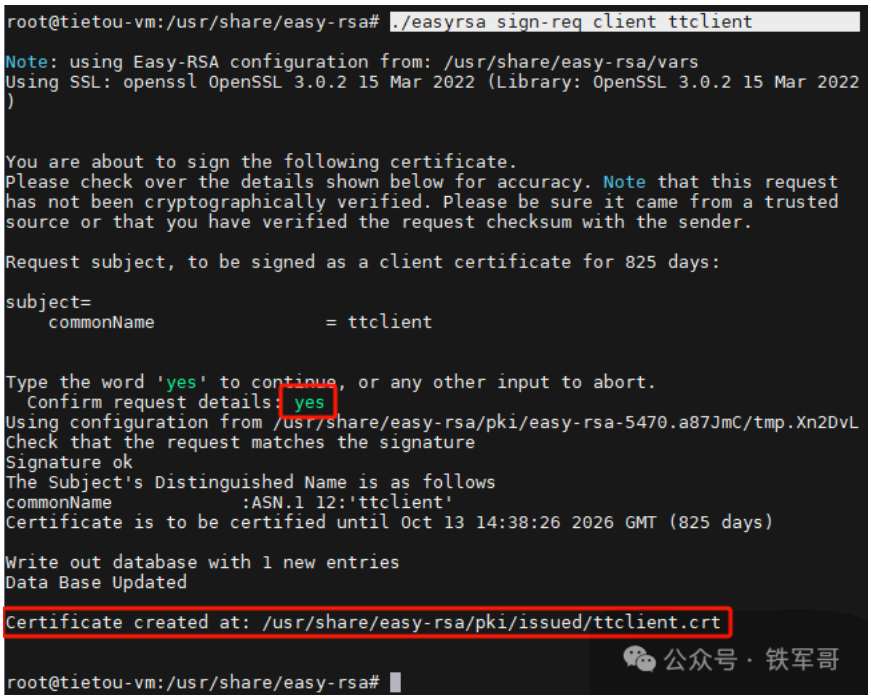
接下来创建客户端的证书，同时指定nopass参数表示不对私钥进行加密。

```
./easyrsa gen-req ttclient nopass
```



给客户端证书ttclient进行签名。

```
./easyrsa sign-req client ttclient
```



至此，证书生成就结束了，涉及到的主要命令为：

```
./easyrsa init-pki
./easyrsa build-ca nopass
./easyrsa gen-req ttserver nopass
./easyrsa sign server ttserver
./easyrsa gen-dh
./easyrsa gen-req ttclient nopass
./easyrsa sign-req client ttclient
```

如果转换成SHELL脚本，则可以是：

```
# 生成证书和密钥
generate_certs() {
  echo "正在生成证书和密钥..."
  cd /usr/share/easy-rsa/
  # 初始化PKI目录结构
  ./easyrsa init-pki
  # 清空证书目录
  echo "yes" | ./easyrsa clean-all
  # 生成证书和密钥文件
  echo | ./easyrsa build-ca nopass
  echo | ./easyrsa gen-req ttserver nopass
  echo "yes" | ./easyrsa sign server ttserver
  ./easyrsa gen-dh
  echo | ./easyrsa gen-req ttclient nopass
  echo "yes" | ./easyrsa sign-req client ttclient
  echo "证书和密钥生成完成。"
}
```

为了方便管理，我们在/目录下创建一个SSL-cert文件夹，然后根据当前时间创建一个新目录，形如202308022132，同时将生成的证书和密钥文件复制到新目录下。如果写成SHELL脚本，则可以是：

```
# 创建SSL-cert目录
generate_dir() {
  # 获取当前时间
  current_time=$(date +"%Y%m%d%H%M")
  # 创建新的目录
  new_dir="/SSL-cert/${current_time}"
  mkdir -p "${new_dir}"
  # 将证书和密钥复制到新目录下
  cp /usr/share/easy-rsa/pki/ca.crt "${new_dir}/ca.crt"
  cp /usr/share/easy-rsa/pki/issued/ttserver.crt "${new_dir}/ttserver.crt"
```

```
  cp /usr/share/easy-rsa/pki/private/ttserver.key "${new_dir}/ttserver.key"
cp /usr/share/easy-rsa/pki/issued/ttclient.crt "${new_dir}/ttclient.crt"
cp /usr/share/easy-rsa/pki/private/ttclient.key "${new_dir}/ttclient.key"
  echo "证书和密钥已复制到目录：${new_dir}"
}
```

最后，我们再加一个主函数。

```
# 主函数
main() {
  install_easyrsa
  init_easyrsa
  update_vars
  generate_certs
  generate_dir
}
```

然后将这些片段攒成一个文件autosslcert.sh，如下所示：

```
#!/bin/bash
# 安装Easy-RSA
install_easyrsa() {
  echo "正在安装Easy-RSA..."
  apt install -y easy-rsa
  echo "Easy-RSA安装完成。"
}
# 初始化Easy-RSA环境
init_easyrsa() {
  echo "初始化Easy-RSA环境..."
  cd /usr/share/easy-rsa/
  # 备份vars.example为vars，避免覆盖
  if [ -f vars ]; then
    mv vars vars.backup
  fi
  cp vars.example vars
  echo "Easy-RSA环境初始化完成。"
}
# 更新vars文件中的机构信息字段
update_vars() {
  echo "更新vars文件中的机构信息字段..."
  cd /usr/share/easy-rsa/
  # 向vars文件中插入机构信息字段
  echo 'set_var EASYRSA_REQ_COUNTRY     "CN"' >> vars
  echo 'set_var EASYRSA_REQ_PROVINCE    "Beijing"' >> vars
  echo 'set_var EASYRSA_REQ_CITY        "Haidian"' >> vars
  echo 'set_var EASYRSA_REQ_ORG         "TIETOU_TECH"' >> vars
  echo 'set_var EASYRSA_REQ_EMAIL       "tietou@h3cadmin.cn"' >> vars
  echo 'set_var EASYRSA_REQ_OU          "Tietou_openVPN"' >> vars
  echo "vars文件中的机构信息字段已更新。"
}
# 生成证书和密钥
generate_certs() {
  echo "正在生成证书和密钥..."
  cd /usr/share/easy-rsa/
  # 初始化PKI目录结构
  ./easyrsa init-pki
  # 清空证书目录
  echo "yes" | ./easyrsa clean-all
  # 生成证书和密钥文件
  echo | ./easyrsa build-ca nopass
  echo | ./easyrsa gen-req ttserver nopass
  echo "yes" | ./easyrsa sign server ttserver
  ./easyrsa gen-dh
  echo | ./easyrsa gen-req ttclient nopass
  echo "yes" | ./easyrsa sign-req client ttclient
  echo "证书和密钥生成完成。"
}
```

```
# 创建SSL-cert目录
generate_dir() {
  # 获取当前时间
  current_time=$(date +"%Y%m%d%H%M")
  # 创建新的目录
  new_dir="/SSL-cert/${current_time}"
  mkdir -p "${new_dir}"
  # 将证书和密钥复制到新目录下
  cp /usr/share/easy-rsa/pki/ca.crt "${new_dir}/ca.crt"
  cp /usr/share/easy-rsa/pki/issued/ttserver.crt "${new_dir}/ttserver.crt"
  cp /usr/share/easy-rsa/pki/private/ttserver.key "${new_dir}/ttserver.key"
  cp /usr/share/easy-rsa/pki/issued/ttclient.crt "${new_dir}/ttclient.crt"
  cp /usr/share/easy-rsa/pki/private/ttclient.key "${new_dir}/ttclient.key"
  echo "证书和密钥已复制到目录：${new_dir}"
}
# 主函数
main() {
  install_easyrsa
  init_easyrsa
  update_vars
  generate_certs
  generate_dir
}
main;
exit;
```

并为此文件赋予可执行权限。

```
root@tietou-vm:~/ssl# nano autosslcert.sh
root@tietou-vm:~/ssl# chmod +x autosslcert.sh
root@tietou-vm:~/ssl# ll
total 12
drwxr-xr-x 2 root root 4096  7月 10 22:46 ./
drwx------ 6 root root 4096  7月 10 22:44 ../
-rwxr-xr-x 1 root root 2342  7月 10 22:46 autosslcert.sh*
root@tietou-vm:~/ssl#
```

找台干净的主机跑一下试试。

```
证书和密钥生成完成。
证书和密钥已复制到目录：/SSL-cert/202407102248
root@tietou-vm:~/ssl# ll /SSL-cert/202407102248
total 36
drwxr-xr-x 2 root root 4096  7月 10 22:48 ./
drwxr-xr-x 3 root root 4096  7月 10 22:48 ../
-rw------- 1 root root 1204  7月 10 22:48 ca.crt
-rw------- 1 root root 4499  7月 10 22:48 ttclient.crt
-rw------- 1 root root 1708  7月 10 22:48 ttclient.key
-rw------- 1 root root 4617  7月 10 22:48 ttserver.crt
-rw------- 1 root root 1704  7月 10 22:48 ttserver.key
root@tietou-vm:~/ssl#
```

Nice，成功了！

长按二维码
关注我们吧