

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Ubuntu部署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客户端验证（VSR白送的的SSL VPN功能，你要不要？）。所以，使用Ubuntu系统，我们还是先创建证书（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。

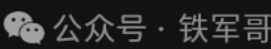
首先是软件安装，对应的命令为：

```
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 openssl openssl-pkcs11 pcscl
The following NEW packages will be installed:
  easy-rsa libccid openssl openssl-pkcs11 pcscl
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 pcscl amd64 1.9.5-3ubuntu1 [58.1 kB]
Get:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/universe amd64 openssl-pkcs11 amd64 0.22.0-1ubuntu2 [93.3 kB]
Get:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/universe amd64 openssl 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 pcscl.
Preparing to unpack .../pcscl_1.9.5-3ubuntu1_amd64.deb ...
Unpacking pcscl (1.9.5-3ubuntu1) ...
Selecting previously unselected package openssl-pkcs11:amd64.
Preparing to unpack .../openssl-pkcs11_0.22.0-1ubuntu2_amd64.deb ...
Unpacking openssl-pkcs11:amd64 (0.22.0-1ubuntu2) ...
Selecting previously unselected package openssl.
Preparing to unpack .../openssl_0.22.0-1ubuntu2_amd64.deb ...
Unpacking openssl (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 pcscl (1.9.5-3ubuntu1) ...
Created symlink /etc/systemd/system/sockets.target.wants/pcscl.socket → /lib/systemd/system/pcscl.socket.
pcscl.service is a disabled or a static unit, not starting it.
Setting up openssl-pkcs11:amd64 (0.22.0-1ubuntu2) ...
Setting up easy-rsa (3.0.8-1ubuntu1) ...
Setting up openssl (0.22.0-1ubuntu2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for mailcap (3.70+nmulubuntu1) ...
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：

```
1 # 安装Easy-RSA
2 install_easyrsa() {
3     echo "正在安装Easy-RSA..."
4     apt install -y easy-rsa
5     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：

```
1 # 初始化Easy-RSA环境
2 init_easyrsa() {
3     echo "初始化Easy-RSA环境..."
4     cd /usr/share/easy-rsa/
5     # 复制vars.example为vars，避免覆盖
6     if [ -f vars ]; then
7         mv vars vars.backup
8     fi
9     cp vars.example vars
10    echo "Easy-RSA环境初始化完成。"
11 }
```

然后我想修改机构信息中的部分字段为以下信息。

```
1 set_var EASYRSA_REQ_COUNTRY "CN"
2 set_var EASYRSA_REQ_PROVINCE "Beijing"
3 set_var EASYRSA_REQ_CITY "Haidian"
4 set_var EASYRSA_REQ_ORG "TIETOU TECH"
5 set_var EASYRSA_REQ_EMAIL "tietou@h3cadmin.cn"
6 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：

```
1 # 更新vars文件中的机构信息字段
2 update_vars() {
3     echo "更新vars文件中的机构信息字段..."
4     cd /usr/share/easy-rsa/
5     # 向vars文件中插入机构信息字段
6     echo 'set_var EASYRSA_REQ_COUNTRY "CN"' >> vars
7     echo 'set_var EASYRSA_REQ_PROVINCE "Beijing"' >> vars
8     echo 'set_var EASYRSA_REQ_CITY "Haidian"' >> vars
9     echo 'set_var EASYRSA_REQ_ORG "TIETOU TECH"' >> vars
10    echo 'set_var EASYRSA_REQ_EMAIL "tietou@h3cadmin.cn"' >> vars
11    echo 'set_var EASYRSA_REQ_OU "Tietou openVPN"' >> vars
12    echo "vars文件中的机构信息字段已更新。"
13 }
```

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
```

[illegible]

给服务器端证书ttserver进行签名。

```

root@tietou-vm:/usr/share/easy-rsa# ./easyrsa sign server ttserver

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 sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 825 days:

subject=
commonName              = ttserver

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
Using configuration from /usr/share/easy-rsa/pki/easy-rsa-5372.A9iJI9/tmp.VikMvZ
4047DE3282780000:error:0700006C:configuration file routines:NCONF_get_string:no
value:../crypto/conf/conf_lib.c:315:group=<NULL> name=unique_subject
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'ttserver'
Certificate is to be certified until Oct 13 14:33:47 2026 GMT (825 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /usr/share/easy-rsa/pki/issued/ttserver.crt

```

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


```

root@tietou-vm:/usr/share/easy-rsa# ./easyrsa sign-req client ttclient

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 sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a client certificate for 825 days:

subject=
  commonName                = ttclient

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
Using configuration from /usr/share/easy-rsa/pki/easy-rsa-5470.a87JmC/tmp.Xn2DvL
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName      :ASN.1 12:'ttclient'
Certificate is to be certified until Oct 13 14:38:26 2026 GMT (825 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /usr/share/easy-rsa/pki/issued/ttclient.crt

```

至此，证书生成就结束了，涉及到的主要命令为：

```

1  ./easyrsa init-pki
2  ./easyrsa build-ca nopass
3  ./easyrsa gen-req ttserver nopass
4  ./easyrsa sign server ttserver
5  ./easyrsa gen-dh
6  ./easyrsa gen-req ttclient nopass
7  ./easyrsa sign-req client ttclient

```

如果转换成SHELL脚本，则可以是：

```

1  # 生成证书和密钥
2  generate_certs() {
3      echo "正在生成证书和密钥..."
4      cd /usr/share/easy-rsa/
5      # 初始化PKI目录结构
6      ./easyrsa init-pki
7      # 清空证书目录
8      echo "yes" | ./easyrsa clean-all
9      # 生成证书和密钥文件
10     echo | ./easyrsa build-ca nopass
11     echo | ./easyrsa gen-req ttserver nopass
12     echo "yes" | ./easyrsa sign server ttserver
13     ./easyrsa gen-dh
14     echo | ./easyrsa gen-req ttclient nopass
15     echo "yes" | ./easyrsa sign-req client ttclient
16     echo "证书和密钥生成完成。"
17 }

```

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下。如果写成SHELL脚本，则可以是：

```

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

```

```

11 | cp /usr/share/easy-rsa/pki/private/ttserver.key "${new_dir}/ttserver.key" 12 |
    cp /usr/share/easy-rsa/pki/issued/ttclient.crt "${new_dir}/ttclient.crt"13 |
    cp /usr/share/easy-rsa/pki/private/ttclient.key "${new_dir}/ttclient.key"14 | echo "证书和密钥已复制到目录: ${new_dir}"
15 | }

```

最后，我们再加一个主函数。

```

1 | # 主函数
2 | main() {
3 |     install_easyrsa
4 |     init_easyrsa
5 |     update_vars
6 |     generate_certs
7 |     generate_dir
8 | }

```

然后将这些片段攒成一个文件autossllcert.sh，如下所示：

```

1 | #!/bin/bash
2 | # 安装Easy-RSA
3 | install_easyrsa() {
4 |     echo "正在安装Easy-RSA..."
5 |     apt install -y easy-rsa
6 |     echo "Easy-RSA安装完成。"
7 | }
8 | # 初始化Easy-RSA环境
9 | init_easyrsa() {
10 |     echo "初始化Easy-RSA环境..."
11 |     cd /usr/share/easy-rsa/
12 |     # 备份vars.example为vars，避免覆盖
13 |     if [ -f vars ]; then
14 |         mv vars vars.backup
15 |     fi
16 |     cp vars.example vars
17 |     echo "Easy-RSA环境初始化完成。"
18 | }
19 | # 更新vars文件中的机构信息字段
20 | update_vars() {
21 |     echo "更新vars文件中的机构信息字段..."
22 |     cd /usr/share/easy-rsa/
23 |     # 向vars文件中插入机构信息字段
24 |     echo 'set_var EASYRSA_REQ_COUNTRY "CN"' >> vars
25 |     echo 'set_var EASYRSA_REQ_PROVINCE "Beijing"' >> vars
26 |     echo 'set_var EASYRSA_REQ_CITY "Haidian"' >> vars
27 |     echo 'set_var EASYRSA_REQ_ORG "TIETOU_TECH"' >> vars
28 |     echo 'set_var EASYRSA_REQ_EMAIL "tietou@h3cadmin.cn"' >> vars
29 |     echo 'set_var EASYRSA_REQ_OU "Tietou_openVPN"' >> vars
30 |     echo "vars文件中的机构信息字段已更新。"
31 | }
32 | # 生成证书和密钥
33 | generate_certs() {
34 |     echo "正在生成证书和密钥..."
35 |     cd /usr/share/easy-rsa/
36 |     # 初始化PKI目录结构
37 |     ./easyrsa init-pki
38 |     # 清空证书目录
39 |     echo "yes" | ./easyrsa clean-all
40 |     # 生成证书和密钥文件
41 |     echo | ./easyrsa build-ca nopass
42 |     echo | ./easyrsa gen-req ttserver nopass
43 |     echo "yes" | ./easyrsa sign server ttserver
44 |     ./easyrsa gen-dh
45 |     echo | ./easyrsa gen-req ttclient nopass
46 |     echo "yes" | ./easyrsa sign-req client ttclient
47 |     echo "证书和密钥生成完成。"
48 | }

```



```

49 # 创建SSL-cert目录
50 | generate_dir() {
51     # 获取当前时间
52     current_time=$(date +%Y%m%d%H%M")
53     # 创建新的目录
54     new_dir="/SSL-cert/${current_time}"
55     mkdir -p "${new_dir}"
56     # 将证书和密钥复制到新目录下
57     cp /usr/share/easy-rsa/pki/ca.crt "${new_dir}/ca.crt"
58     cp /usr/share/easy-rsa/pki/issued/ttserver.crt "${new_dir}/ttserver.crt"
59     cp /usr/share/easy-rsa/pki/private/ttserver.key "${new_dir}/ttserver.key"
60     cp /usr/share/easy-rsa/pki/issued/ttclient.crt "${new_dir}/ttclient.crt"
61     cp /usr/share/easy-rsa/pki/private/ttclient.key "${new_dir}/ttclient.key"
62     echo "证书和密钥已复制到目录: ${new_dir}"
63 }
64 # 主函数
65 main() {
66     install_easyrsa
67     init_easyrsa
68     update_vars
69     generate_certs
70     generate_dir
71 }
72 main;
73 exit;

```

并为此文件赋予可执行权限。

```

root@tietou-vm:~/ssl# nano autossllcert.sh
root@tietou-vm:~/ssl# chmod +x autossllcert.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 autossllcert.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