nginx的配置https方向代理

Nginx+https双向验证
说明：
要想实现nginx的https，nginx必须启用http_ssl模块；在编译时加上--with-http_ssl_module参数就ok。另外系统必须安装了openssl；openssl为开源软件，在Linux（或UNIX/Cygwin）下创建一个简单的CA。我们可以利用这个CA进行PKI、数字证书相关的测试。比如，在测试用Tomcat或Apache构建HTTPS双向认证时，我们可以利用自己建立的测试CA来为服务器端颁发服务器数字证书，为客户端（浏览器）生成文件形式的数字证书（可以同时利用openssl生成客户端私钥）

1. 查看系统是否安装了openssl；
rpm -qa | grepopenssl
如果没有，centos系统使用yum install openssl –y 安装就行
2. 查看nginx编译时是否加上了http_ssl模块；
nginx -V
如果存在--with-http_ssl_module说明nginx支持

3. 服务器-客户端双向验证；创建相关目录，存放相关证书和key，我这里放在/data/genssl下面，路径可以自己随意指定，nginx能够找到就行；
所有用到的配置文件我全部存放在genssl目录下面
使用的时候只要将这个目录copy到当前目录就可以了
cd genssl

openssl.conf配置文件，内容如下：
cat openssl.conf
default_md = sha1 这里一定设置sha1，其他教程都是md5，最后导致提示根证书是弱加密，或者是伪造的
countryName = match #国家;比如cn,代表中国
stateOrProvinceName = match #州或省;guangdong
organizationName = match #组织,可以理解为公司;mbook
organizationalUnitName = match #组织单位,也可以理解为公司;mbook
localityName = optional #城市;guangzhou
commonName = supplied #网站域名;*.mbook.cn
emailAddress = optional #邮件地址;admin@mbook.cn
注：你也可以直接修改openssl的配置文件，这样的话后面制作证书的代码中就不用引用这个配置文件了

创建过程主要分为4个大步骤
1.首先使用new_ca.sh脚本创建新的CA根证书
new_ca.sh
```
#########################################################################
#!/bin/sh
#get the real_dir
real_dir=$(dirname `/usr/sbin/lsof -p $$ | gawk '$4 =="255r"{print $NF}'`)
cd $real_dir
#create the dir
[ ! -d private ] && mkdir private
# Generate the key.
openssl genrsa -out private/ca.key
# Generate a certificate request.
openssl req -new -key private/ca.key -out private/ca.csr
# Self signing key is bad... this could work with a third party signed key... registeryfly has them on for $16 but I'm too cheap lazy to get one on a lark.
# I'm also not 100% sure if any old certificate will work or if you have to buy a special one that you can sign with. I could investigate further but since this
# service will never see the light of an unencrypted Internet see the cheap and lazy remark.
# So self sign our root key.
openssl x509 -req -days 3650 -in private/ca.csr -signkey private/ca.key -out private/ca.crt
# Setup the first serial number for our keys... can be any 4 digit hex string... not sure if there are broader bounds but everything I've seen uses 4 digits.
echo FACE > private/serial
# Create the CA's key database.
touch private/index.txt
# Create a Certificate Revocation list for removing 'user certificates.'
openssl ca -config ./openssl.cnf -gencrl -out $real_dir/private/ca.crl -crldays 1095
#########################################################################
```

在执行执行new_ca.sh脚本生成新的ca证书过程中会有交互过程具体如下:

Country Name (2 letter code) [XX]:CN
所在广州cn
State or Province Name (full name) []:GD
所在省份广东
Locality Name (eg, city) [Default City]:GZ
所在地广州市
Organization Name (eg, company) [Default Company Ltd]:FY
所在组织凡跃
Organizational Unit Name (eg, section) []:YW
所在部门运维
Common Name (eg, your name or your server's hostname) []:zabbix.fanyue65.com
这个很中要,如果是给一个网站用直接填完整的域名,如果给多个用可以使用*.fanyue65.com前缀就用*
常用名字或者你的主机名zabbix.fanyue65.com
Email Address []:zabbix.fanyue65.com
邮箱地址这个最好和网站后缀域名一样
A challenge password []:
提示输入密码直接回去不理会
An optional company name []:
可选的公司注册名字,回车不理会

2生成服务器的证书脚本如下:

```
sh  new_server.sh  zabbix.fanyue65.com
############################################################
#!/bin/bash
#get the real_dir
if [ $# -lt 1 ];then
        printf "usage:$0 servername\n"
        exit 1
fi
servername=$1
real_dir=$(dirname `/usr/sbin/lsof -p $$ | gawk '$4 =="255r"{print $NF}'`)
cd $real_dir
#create the dir
[ -d private ] && mkdir private
# Create us a key. Don't bother putting a password on it since you will need it to start apache. If you have a better work
around I'd love to hear it.
openssl genrsa -out private/${servername}.key
# Take our key and create a Certificate Signing Request for it.
openssl req -new -key private/${servername}.key -out private/${servername}.csr
# Sign this bastard key with our bastard CA key.
openssl ca -config ./openssl.cnf -in private/${servername}.csr -cert private/ca.crt -keyfile private/ca.key -out
private/${servername}.crt
#####################################################################
```

执行new_server.sh脚本时特别注意交互过程中的参数必须和前面的new_ca.sh脚本一直否则会失败。

执行new_server.sh
和上面的提示差不多
Sign the certificate? [y/n]:y
这里提示你是否注册证书选择y
1 out of 1 certificate requests certified, commit? [y/n]y
证书请求签证选择y

3. 在nginx部署ssl

```
        listen       443;  https默认的端口是443
        server_name  zabbix.fanyue65.com;   域名
        ssl on;
        ssl_certificate      /usr/local/nginx/conf/genssl/private/users/zabbix.fanyue65.com.crt; #new_server.sh执行时候生成的
        ssl_certificate_key  /usr/local/nginx/conf/genssl/private/users/zabbix.fanyue65.com.key; #new_server.sh执行时候生成的
        ssl_client_certificate /usr/local/nginx/conf/genssl/private/ca.crt;  #new_ca.sh执行时生成的
        ssl_session_timeout 5m;
    ssl_verify_client  on;
        ssl_verify_depth 1;
        ssl_protocols SSLv2 SSLv3 TLSv1;
        ssl_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
        ssl_prefer_server_ciphers on;
```

为了能够自动跳转到https，在虚拟主机里面添加如下代码
server {
listen 80;
server_name zabbix.fanyue65.com;
rewrite ^(.*) https://$server_name$1 permanent;
}
这样能够实现自动跳到https
启动nginx ,等待客户连接，如果此时连接服务器，将提示400 Bad request certification的错误，故还需要生成客户端证书

4. 生成client证书
按照提示一步一步来，这里要注意的是客户证书的几个项目要和根证书匹配
sh new_user.sh zabbix.fanyue65.com
```
#####################################################
#!/bin/sh
#if not input certify_username,print usage
[ -z "$1" ] && { echo usage:$0 certify_username;exit 1; }
#get the real_dir
real_dir=$(dirname `/usr/sbin/lsof -p $$ | gawk '$4 =="255r"{print $NF}'`)
cd $real_dir
#create the dir
[ -d private ] && mkdir private
# The base of where our SSL stuff lives.
base="$real_dir/private"
# Were we would like to store keys... in this case we take the username given to us and store everything there.
[ ! -d $base/users ] && mkdir -p $base/users

# Let's create us a key for this user... yeah not sure why people want to use DES3 but at least let's make us a nice big key.
openssl genrsa -des3 -out $base/users/$1.key 1024
# Create a Certificate Signing Request for said key.
openssl req -new -key $base/users/$1.key -out $base/users/$1.csr
# Sign the key with our CA's key and cert and create the user's certificate out of it.
openssl ca -config ./openssl.cnf -in $base/users/$1.csr -cert $base/ca.crt -keyfile $base/ca.key -out $base/users/$1.crt

# This is the tricky bit... convert the certificate into a form that most browsers will understand PKCS12 to be specific.
# The export password is the password used for the browser to extract the bits it needs and insert the key into the user's
keychain.
# Take the same precaution with the export password that would take with any other password based authentication scheme.
openssl pkcs12 -export -clcerts -in $base/users/$1.crt -inkey $base/users/$1.key -out $base/users/$1.p12
#####################################################################
```

执行过程中交互提示
Enter pass phrase for.....:要求输入密码
Verifying - Enter pass phrase for .....:要求输入密码
Enter pass phrase for .....:要求输入密码，全部输入一致
.....
.....
.....
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Enter pass phrase for .....key:这里输入的密码是当导入证书需要的密码
Enter Export Password:
Verifying - Enter Export Password:同上

最后会在当前目录的users下面生成客户端的证书,还会设置一个p12的证书,这个证书就是浏览器用来登录的时候的证书,需要导入浏览器证书列表.才可以访问https的地址,导入证书中的提示输入密码需要使用刚才创建客户端最后设置的那个密码

访问是提示网站证书不受信任，点击下面的继续访问
注明：
各位童鞋，自己的证书自己保管好，在公司上网站就需要导入证书，回家需要访问公司某个内部网站，也需要导入证书！！！
最后附上自动添加客户端证书的脚本，其中省去了交互功能，因为如果给100个人生成证书，多大好几百个交互过程，这个过程会让人崩溃的，所以使用了expect自动传入参数。脚本内容如下

```
#!/bin/bash
#write by yayun 2013-05-29
#batch add client certificate for nginx https

base="/data/nginx/ca"
```

```
fori in $(awk -F "@" '{print $1}' name.txt)
do
mkdir -p $base/$i/
opensslgenrsa -out $base/$i/$i.key 2048

Country=cn
State=guangdong
Locality=guangzhou
Organization=mbook
Common=*.mbook.cn
Passwd=P@ssw0rdmbook.cn888
Email=$i@mbook.cn

expect -c "
spawnopensslreq -new -key $base/$i/$i.key -out $base/$i/$i.csr
expect {
        \"*XX*\" {send \"$Country\r\"; exp_continue}
        \"*full name*\" {send \"$State\r\"; exp_continue}
        \"*Default City*\" {send \"$Locality\r\"; exp_continue}
        \"Organization*\" {send \"$Organization\r\"; exp_continue}
        \"Organizational*\" {send \"$Organization\r\"; exp_continue}
        \"*hostname*\" {send \"$Common\r\"; exp_continue}
        \"Email*\" {send \"$Email\r\"; exp_continue}
        \"*password*\" {send \"\r\"; exp_continue}
        \"*company name*\" {send \"\r\"; exp_continue}
    }
"
expect -c "
spawnopensslca -in $base/$i/$i.csr -cert $base/private/ca.crt -keyfile $base/private/ca.key -out $base/$i/$i.crt -config
"/data/nginx/ca/conf/openssl.conf"
expect {
        \"*certificate?*\" {send \"y\r\"; exp_continue}
        \"*commit?*\" {send \"y\r\"; exp_continue}
    }
"
expect -c "
spawnopenssl pkcs12 -export -clcerts -in $base/$i/$i.crt -inkey $base/$i/$i.key -out $base/$i/$i.p12
expect {
        \"Enter*Password:\" {send \"$Passwd\r\"; exp_continue}
        \"Verifying*Password:\" {send \"$Passwd\r\"; exp_continue}
    }
"
done
其中name.txt记录了用户名，邮箱，脚本自动循环读取。整个过程结束！
#####################################################################
执行客户端脚本中报错
Sign the certificate? [y/n]:y
failed to update database
TXT_DB error number 2

vim index.txt.attr
unique_subject = yes 将yes改为no
#####################################################################
```