SIP TLS

Implementing TLS greatly enhances the security. It's also rather confusing to get it working.
I put together a really simple set of procedures to configure Asterisk 1.8.x to accept TLS connections from its clients.

There are several basic steps we need to do:
1 - Create or add a certificate on the asterisk server
2 - Add some configuration settings into the sip.conf file
3 - Configure the clients to use TLS

Let's look at each step, one at a time.

First, your asterisk server needs a certificate. For this exercise, we're going to use a self-signed cert, which is enough to get started. You'll need openssl installed on your sever, so if you don't have it, load it up. If in doubt, simply type "openssl" at a command prompt - if openssl is on your system you should see a "OpenSSL" prompt. If so, you're ready for the next step.

Creating a server key - We need to create a digital key for our server. This is not the actual "certificate", but is needed to create it. Assuming you're asterisk program was loaded with defaults, your configuration files should be under /etc/asterisk. Let's go to that directory and create a new directory called "certificates" (mkdir certificates). Change to the new directory (cd certificates) and make sure the path to the directory is /etc/asterisk/certificates. You can do this by running the "pwd" command in *nix.

From /etc/asterisk/certificates, we're going to create a server key by typing the following:
```
openssl genrsa -out key.pem 1024
```

You should see something like "Generating RSA private key, 1024 bit long modulus". If so, things are going well. If you do a listing of the directory, you should see something like:
```
-rw-r--r-- 1 root root 887 2010-08-30 21:39 key.pem
```

The key.pem is your server key. Make a backup of this file onto a CD or USB drive or whatever - you may need it in the future.

OK, so we cut a server key - the next step is to create a certificate request. Type this:
```
openssl req -new -key key.pem -out request.pem
```

You'll be prompted for the following:
Country Name - Enter a TWO character country code like US, UK, DE, etc.
State or Province Name - If you're in the US, this would be the state, typed out. Do no use abbreviations!
Locality Name - enter the city name where you are located at (i.e. Dallas, Memphis, whatever)
Organization Name - Enter your company name or even your personal name if this is a home server
Organizational Unit Name - This can be the same as the org name, or a division name if you wish
Common name - This *NEEDS* to be the FQDN name of your server, for example, asterisk.something.com
Email Address - leave this blank by simply hitting return
A challenge password - leave this blank by simply hitting a return
An optional company name - This could be your company initials (like IBM) or simply left blank

You should be back at your command prompt at this point. If you do a directory listing of /etc/asterisk/certificates, you should now see:
```
-rw-r--r-- 1 root root 887 2010-08-30 21:39 key.pem
-rw-r--r-- 1 root root 639 2010-08-30 21:49 request.pem
```

The key.pem file is your server key and the request.pem is your certificate request.
If you're going to get a certificate from a real CA, the request.pem file is what you would send over to have the CA sign. For this exercise, we're going to simply sign our own certificate by running the following command:
```
openssl x509 -req -days 3650 -in request.pem -signkey key.pem -out certificate.pem
```

This will produce a new file called certificate.pem. This *is* your new certificate. Note that the -days 3650 essentially made it good for 10 years. You may wish to edit that somewhat, but this is probably a good setting since you'll end up replacing the certs long before this one expires.

You have no further use for the certificate request file (request.pem), but I'd suggest leaving it in the directory. It's small and won't bother anybody.

The hard part is done - it's all downhill from here. Now that we have a server key (key.pem) and a certificate (certificate.pem) we're going to make a new file by adding the two files together.
We want to create a file with the name of your system, so using the example above, we'll name this new file "asterisk.something.com.pem". We'll create this by first copying the certificate file, then appending the server key to the end of it. Assuming you're on a *nix box, you should type the following commands from /etc/asterisk/certificates:
```
cp certificate.pem asterisk.something.com.pem
cat key.pem >> asterisk.something.com.pem
```

Note that we used a double ">>". if you used a single ">", you overwrote the certificate portion.
If you did, just do the two steps again.

We now have a server key, a certificate, and a certificate "chain" file (asterisk.something.com.pem). Now we go to /etc/asterisk, or wherever your sip.conf resides. so we can edit sip.conf and add the following items:

```
tlsenable=yes
tlsbindaddr=192.168.0.1 (put your actual ip address of your box here)
tlscertfile=/etc/asterisk/certificates/asterisk.something.com.pem
tlsdontverifyserver=no
tlscipher=DES-CBC3-SHA
tlsclientmethod=tlsv1
```

One note here - if your box is nat'ed, the tlsbindaddress needs to be your internal address, in other words, the ip address that shows up when you type "ifconfig". Don't put your public facing address here. For clarity, be sure you show the actual name of the certificate chain file under tlscertfile.

Finally, go to one of your sip client contexts (still in sip.conf) and add:
```
transport=tls
```

In fact, you should do this for each client that will be accessing your asterisk server through TLS. Hopefully you're using templates for your client configs - if so, simply create a new template that includes sip tls and configure your clients that way. If you don't know what templates are, don't worry about it - the simple way works, just add the transport=tls statement to each users context.

That's it. At this point you should have an asterisk box that speaks TLS. The final step is to have each of your clients configure their devices/softphones to use TLS. If you're using a Snom 3xx phone, you would do this by adding ";transport=tls" after the host name or ip address in "Outbound Proxy" settings. Each client will have to figure out how to configure their end to work.

One parting note - TLS greatly enhances the sip portion of your clients communications. Using real certificates (as opposed to self-signed certs) you can prevent MITM (man in the middle) compromises. But even using self-signed certs prevents someone from capturing your login credentials. However, TLS by itself *DOES NOT PROTECT THE VOICE TRAFFIC*.

Sip is a signaling protocol, but your actual voice data goes across using RTP. To protect that portion of your communications, you need to do either tunnel the traffic in a VPN, or add SRTP (secure RTP). SRTP encrypts the actual voice portion of the calls. The good news is that Asterisk 1.8.x supposedly has full support for this. Once the 1.8.x comes out of beta, I'll add a second tutorial on how to implement SRTP.

Good luck!

Andrew
afried@deteque.com

Created by: asilx, Last modification: Tue 31 of Aug, 2010 (03:28) by afried