## Contents

1. Contents
   1.1 Getting started .............................................. 4
   1.2 Setup a secure environment with Asterisk ....................... 10
   1.3 Setup a secure environment with Freeswitch ..................... 19

2. Indices and tables .................................................. 25
SFLphone is a robust, standards-compliant enterprise softphone, for desktop and embedded systems. It is designed to handle several hundred calls a day. SFLphone is available under the GNU GPL license, version 3.

Please visit the official website for a complete list of features: http://sflphone.org.
CHAPTER 1

Contents
1.1 Getting started
1.1.1 Install SFLphone

First, you need to add the official SFLphone PPA (Personal Package Archive). This allows us to push new versions for older distributions.

**Note:** This step is not mandatory as Ubuntu provides SFLphone packages in its *universe* repository.

**Solution 1: Software Center**

- Open Software Center (in Unity, press the Windows key and type *software center*)

![Ubuntu Software Center](image)

- In *Edit > Software sources*, select the *Other software* tab
- Click *add* and enter *ppa:savoirfairelinux*
- Click *close*
- Click the little arrow next to *All Software* and select sflphone
- Select *GNOME client for SFLphone* and click *Install*

**Solution 2: Command-line**

Add the repository to your software sources:

```
sudo add-apt-repository ppa:savoirfairelinux
```

Now, update the package list:

```
sudo apt-get update
```

You can now install the latest SFLphone version:
sudo apt-get install sflphone-client-gnome

Solution 3: Building from sources (not recommended)

Please refer to the instructions here to build SFLphone from source.

1.1.2 Configuring an existing account

The simplest way to configure SFLphone is to use the First Run wizard.

- In the Unity lens interface (top left dock icon or Windows key), type \texttt{sflphone}
- After a few seconds, a wizard window should appear on the screen with a welcome message
- Press \texttt{Continue} to start the wizard

\textbf{Note:} You can always return to the installation wizard in SFLphone by clicking the menu Call > Configuration Assistant

\begin{center}
\includegraphics[width=\textwidth]{fig12.png}
\end{center}

\textbf{Figure 1.2:} Account configuration wizard: first step.

\begin{description}
\item [Account] Select \texttt{Register an existing SIP or IAX2 account}
\item [VoIP Protocols] You can select here the communication protocol to use to make calls (if unsure, use SIP).
\item [Account settings] This step allows you to set up your account, by specifying the hostname, username, password, ...\end{description}
Figure 1.3: Account configuration wizard: Account window.

Figure 1.4: Account configuration wizard: VoIP Protocols window.

Figure 1.5: Account configuration wizard: Account settings.
Here are the details of each settings:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>A name you will remember (Example: Workplace)</td>
</tr>
<tr>
<td>Hostname</td>
<td>Usually the server address, (Example: sip.sflphone.org)</td>
</tr>
<tr>
<td>Username</td>
<td>Usually your phone extension (Example: 123), but may also be a name</td>
</tr>
<tr>
<td>Password</td>
<td>Your account password (Warning: will be stored in plain text)</td>
</tr>
<tr>
<td>Voicemail number (optional)</td>
<td>Another username you can call to play your voicemail. Not every account has one</td>
</tr>
<tr>
<td>Secure communications with ZRTP</td>
<td>Do not check (see security section)</td>
</tr>
</tbody>
</table>

Press **Continue** and **Apply** when you are ready to register your account.

---

**Account registration**

The last panel displays an overview of your account settings. You may now click on **Close**, as the installation wizard is finishow now.

If you now select **Edit > Accounts**, your new account should be registered, and appear in green.

---

### 1.1.3 SIP security basics

The first thing to know about SIP security is that there is usually none at all. By default, everything is transmitted unencrypted and readable by any software that can grab traffic between you and your peer. If you are using proxies along the way, each of them may decide to ignore certain security options. The second important detail to retain is that SIP security alone (SIPS, SIP-TLS) does not encrypt your communications. SIPS only encrypts the handshake between both peers. To encrypt the media stream (aka, voice) itself, you also need to enable Secure RTP (aka, SRTP). Likewise, only having SRTP enabled may encrypt your audio but will not obscure information about the call itself (i.e. participants, IP addresses, etc.).

Most registrar and SIP servers have some level of support for security, however, all implementations are not created equal. As of Asterisk 1.8 (the default in many server operating systems), some security options are not as well supported as they should be. For a safer system, Freeswitch is the best free software option. Asterisk also needs to be recompiled to enable SRTP (see https://projects.savoirfairelinux.com/projects/sflphone/wiki/Security ). This, in turn, will force you to manually update your SIP server everytime a security patch is available, introducing security vulnerabilities of its own.
**Important:** TODO show security options here
1.2 Setup a secure environment with Asterisk
1.2.1 Set up a basic Asterisk server

**Important:** **Prerequisites:** an Ubuntu server or virtual machine  
**Outcome:** a basic SIP server with 2 accounts

**Install Asterisk**

Install Asterisk with `apt-get`:

```bash
sudo su
apt-get install asterisk
```

It will ask you for a country code, you can check [http://countrycode.org/](http://countrycode.org/) to get yours.

You can check if Asterisk is operational using:

```bash
service asterisk status
```

**Basic configuration**

**Add SIP accounts**

Now, using your favourite text editor, make a backup of `/etc/asterisk/sip.conf` and replace it with:

```ini
[general]
context=internal
allowguest=no
allowoverlap=no
bindport=5060
bindaddr=0.0.0.0
srvlookup=no
disallow=all
allow=ulaw
allow=g722
allow=alaw
allow=gsm
alwaysauthreject=yes
canreinvite=no
session-timers=refuse
localnet=192.168.1.0/255.255.255.0

[666]
```

1.2. Setup a secure environment with Asterisk
callerid=anonymous
type=friend
host=dynamic
secret=123
context=internal

[777]
type=friend
host=dynamic
secret=456
context=internal

Be sure to update \textit{localnet} to match your network settings. Run \texttt{ifconfig} command to check your public IP address:

To apply the settings, execute \texttt{rasterisk} and type:

\begin{verbatim}
sip set debug on
sip reload
\end{verbatim}

To confirm the new dialplan, run:

\begin{verbatim}
sip show users
\end{verbatim}

\section*{Dialplan}

Now, setup a new dialplan to be able to call other users. Edit \texttt{/etc/asterisk/extensions.conf} and add the following lines:

\begin{verbatim}
[users]
include => default
include => trunklocal
include => iaxtel700
include => trunktollfree
include => iaxprovider
exten => 777,n,Dial(SIP/777,777,Tt)
\end{verbatim}
1.2.2 Configuring Asterisk encryption

In this part, we will use TLS to encrypt the data streams.

# Operate as root to install openssl
sudo su -
apt-get install openssl

# Create the directory and get a certificate
mkdir /etc/certs
cd /etc/certs/
wget https://raw.github.com/rillian/asterisk-opus/master/contrib/scripts/ast_tls_cert
sh ast_tls_cert -C pbx.MyHostName -O "NSA proof (?) server" -d /etc/certs/

# Now fill out the form.
sh ast_tls_cert -m client -c /etc/certs/ca.crt -k /etc/certs/ca.key -C mynsauser.MyHostName -O "NSA proof (?) server" -d /etc/certs
chown asterisk:asterisk ./ -R

Note: This How-To has assumed the same computer as server and client for simplicity’s sake. In a real-world context, this will rarely be the case. So you will have to safely upload the certificates to the client computer. To do this, the scp command allows you to upload the key over an encrypted connection. In our case, you can simply copy them using cp as it doesn’t change anything.

scp mynsauser.pem username@hostname:/tmp/
scp ca.crt username@hostname:/tmp/

Now, Asterisk need to be configured to use this certificate for encrypted calls. Backup your current /etc/asterisk/sip.conf, then open sip.conf and replace its contents with:

```
[general]
context=internal
allowguest=no
allowsubscribe=yes
allowoverlap=no
bindport=5060
bindaddr=0.0.0.0 ;192.168.48.213
tlsbindaddr=0.0.0.0 ;192.168.48.213	srlookup=no
disallow=all
allow=ulaw
allow=g722
allow=alaw
allow=gsm
alwaysauthreject=yes
canreinvite=no
;nat=yes
session-timers=refuse
localnet=192.168.48.0/255.255.252.0
tlsenable=yes
tlservfile=/etc/certs/asterisk.pem
tlscafile=/etc/certs/ca.crt
tlscipher=TLSv1
```

1.2. Setup a secure environment with Asterisk
;tlsclientmethod=tlsv1
tlsdontverifyserver=no
tlsbindaddr=0.0.0.0

[777]
callerid=NSA
type=friend
secret=nsa
host=dynamic
transport=tls
port=5061
context=internal
dtmfmode=rfc2833
insecure = invite,port
nat = yes

[888]
callerid=NSA
type=friend
secret=nsa
host=dynamic
transport=tls
port=5061
context=internal
dtmfmode=rfc2833
insecure = invite,port
nat = yes

[999]
callerid=PLAY
type=friend
secret=play
host=dynamic
transport=tls
port=5061
context=local
dtmfmode=rfc2833
;insecure = invite,port
nat = yes

And in extensions.conf, in the [local] section, add:

exten => 999,1,Answer
exten => 999,2,Playback(tt-weasels)
exten => 999,3,Wait(10)
exten => 999,4,Hangup

Now, run the rasterisk command and type:

sip reload
dialplan reload

Asterisk should now be using TLS for message passing.

**Warning:** The stream itself is not yet encrypted, only the SIP messages are.
1.2.3 Configuring SFLphone with Asterisk

Once this is done, execute `sflphone-client-gnome`. A configuration wizard will launch if you started it for the first time.

Select the following values:

- **Account**: Register an Existing SIP or IAX2 account, then **Next**
- **VoIP Protocols**: SIP, then **Next**

As for the SIP account settings, input these values:

<table>
<thead>
<tr>
<th>Settings</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>My First Account</td>
</tr>
<tr>
<td>Hostname</td>
<td>&lt;your asterisk server IP&gt;</td>
</tr>
<tr>
<td>Username</td>
<td>666</td>
</tr>
<tr>
<td>Password</td>
<td>123</td>
</tr>
</tbody>
</table>

Note: You may run `ifconfig` to check your IP address.

Then **Next** and **Apply**.

If you go in **Edit > Account**, you should now see *MyFirstAccount* as **Registered** (in green). If you see a **Trying...** status and run **Asterisk** on the same computer as SFLphone, you have to change the default port in your account advanced settings (**Edit > Accounts > Select your account > Edit > Advanced**).

1.2.4 Configuring SFLphone security

**Important**: Prerequisites: You have both ca.crt and the .pem certificates

For **Asterisk** to encrypt your stream, select:

- In **Edit > Account > Your account > Security**, select **ZRTP** in **SRTP key exchanges** (**Asterisk** need to be compiled with SRTP support). Also select **Use TLS transports**.
- In the **Advanced** tab, set your ca.crt as authority and .pem as user certificate.
- Uncheck **Verify incoming certificate (as a server)**
- Click on **Apply**

You are now done!

**Warning**: Please note that this setup is still vulnerable to **Man-in-the-middle attack**, but not to packet sniffers.
Figure 1.7: Account list window, showing the accounts registration status.
Figure 1.8: The account’s advanced settings allows to configure the network interface, port range and registration expiration time.
1.3 Setup a secure environment with Freeswitch
1.3.1 Installing Freeswitch

Please see the Ubuntu Quick Start Guide for an official guide to installing Freeswitch.

Otherwise, to install the latest FreeSwitch from git, run the following commands:

```bash
# Install all dependencies
sudo su -
apt-get install git-core build-essential autoconf automake libtool libncurses5 libncurses5-dev make libjpeg-dev ... libogg-dev libvorbis-dev libperl-dev libgdbm-dev libdb-dev python-dev uuid-dev bison autoconf g++ libncurses-dev

# Download FreeSwitch via git
mkdir -p /usr/local/src
cd /usr/local/src
git clone git://git.freeswitch.org/freeswitch.git
cd freeswitch

# Build
./bootstrap.sh
./configure
make
# You may have to call "make" a few times before it works.

# Install in /usr/local
make all install cd-sounds-install cd-moh-install
```


Now, it is time to see if FreeSwitch is properly loaded. If you already have SFLphone running on the same computer, make sure to close it first.

```
/usr/local/freeswitch/bin/freeswitch
```

After a few seconds, a shell will appear. To test if SIP is ready, enter:

```
sofia status profile internal
```

This should display something like:

```
Name internal
Domain Name N/A
Auto-NAT false
DBName sofia_reg_internal
Pres Hosts 192.168.48.185,192.168.48.185
Dialplan XML
Context public
Challenge Realm auto_from
RTP-IP 192.168.48.185
SIP-IP 192.168.48.185
URL sip:mod_sofia@192.168.48.185:5060
BIND-URL sip:mod_sofia@192.168.48.185:5060;transport=udp,tcp
HOLD-MUSIC local_stream://moh
OUTBOUND-PROXY N/A
CODECS IN G722,PCMU,PCMA,GSM
CODECS OUT G722,PCMU,PCMA,GSM
TEL-EVENT 101
DTMF-MODE rfc2833
CNG 13
SESSION-TO 0
NOMEDIA false
```
1.3. Setup a secure environment with Freeswitch
1.3.2 Configuring Freeswitch security

To have an overview of SIP security, please read the section *SIP security basics*.

```bash
cd /usr/local/freeswitch/bin
sudo ./gentls_cert setup -cn 127.0.0.1 -alt DNS:localhost -org 127.0.0.1
sudo ./gentls_cert create_server -cn 127.0.0.1 -alt DNS:localhost -org 127.0.0.1
```

This will generate a self signed certificate in `/usr/local/freeswitch/conf/ssl/CA/cakey.pem`.

**Important**: Self-signed certificates are not entirely secure and void the chain of trust. If you want care about security, please generate a certificate signed by an authority.

Now, in `/usr/local/freeswitch/conf/vars.xml`, enable **TLS**:

**Original**:

```xml
<X-PRE-PROCESS cmd="set" data="sip_tls_version=tlsv1"/>

<!-- Internal SIP Profile -->
<X-PRE-PROCESS cmd="set" data="internal_auth_calls=true"/>
<X-PRE-PROCESS cmd="set" data="internal_sip_port=5060"/>
<X-PRE-PROCESS cmd="set" data="internal_tls_port=5061"/>
<X-PRE-PROCESS cmd="set" data="internal_ssl_enable=false"/>

<!-- External SIP Profile -->
<X-PRE-PROCESS cmd="set" data="external_auth_calls=false"/>
<X-PRE-PROCESS cmd="set" data="external_sip_port=5080"/>
<X-PRE-PROCESS cmd="set" data="external_tls_port=5081"/>
<X-PRE-PROCESS cmd="set" data="external_ssl_enable=false"/>
```

**New**:

```xml
<X-PRE-PROCESS cmd="set" data="sip_tls_version=tlsv1"/>

<!-- Internal SIP Profile -->
<X-PRE-PROCESS cmd="set" data="internal_auth_calls=true"/>
<X-PRE-PROCESS cmd="set" data="internal_sip_port=5060"/>
<X-PRE-PROCESS cmd="set" data="internal_tls_port=5061"/>
<X-PRE-PROCESS cmd="set" data="internal_ssl_enable=true"/>
<X-PRE-PROCESS cmd="set" data="internal_ssl_dir=${base_dir}/conf/ssl"/>

<!-- External SIP Profile -->
<X-PRE-PROCESS cmd="set" data="external_auth_calls=false"/>
<X-PRE-PROCESS cmd="set" data="external_sip_port=5080"/>
<X-PRE-PROCESS cmd="set" data="external_tls_port=5081"/>
<X-PRE-PROCESS cmd="set" data="external_ssl_enable=true"/>
<X-PRE-PROCESS cmd="set" data="external_ssl_dir=${base_dir}/conf/ssl"/>
```
Now, in the Freeswitch shell, execute:

```
reloadxml
```

Back in `/usr/local/freeswitch/bin`, it is now time to create certificate for users:

```
sudo ./gentls_cert create_client -cn 1002 -out 1002.pem
```

Be sure to `cp/scp` the following certificates to all relevant users.

**Note:** This *How-to* is always using the same computer as server and client for simplicity purpose. In a real context, this will rarely be the case. So you will have to safely upload the certificates to the client computer. To do this, the `scp` command allow you to create an encrypted upload of the key, thus, invulnerable to man in the middle attack. In this case, you can copy them using `cp` as it doesn’t change anything.

```
scp /usr/local/freeswitch/conf/ssl/CA/cacert.pem username@hostname:/home/username/
scp /usr/local/freeswitch/conf/ssl/agent.pem username@hostname:/home/username/
```

If you use an alternate upload method, please double check if the target file owned by the same used as the sflphone process (usually your current username) and have **660** permissions. The `scp` lines will automatically do that for you.

Optional:

In this how-to, we run Freeswitch as root. This, of course, as Freeswitch is a network facing application, is a potential attack vector. If you change Freeswitch user, do not forget to use:

```
cd /usr/local/freeswitch/
fnd -iname conf/ssl/ | xargs chown myfreeswitchuser:myfreeswitchuser
```


### 1.3.3 Configuring SFLphone with Freeswitch

Freeswitch already provides a few usable accounts you can use right away. They are numbered from **1000** to **1015** and the *default password* is **1234**.

The configuration files for every accounts are stored in `/usr/local/freeswitch/conf/directory/default/`. You can edit accounts individually.

### 1.3.4 Configuring SFLphone security

Configuring a secure Freeswitch account is trivial.

First, make sure you uploaded the `cacert.pem` and `agent.pem` as described in the earlier steps. Once this is done, create your account using **Edit > Account > New** and in the **Security** tab, **check Use TLS transports** and select **ZRTP** in **SRTP key exchange**.

Now, in the **Edit** dialog, add both your `cacert.pem` and `agent.pem` and press **OK**.
Figure 1.9: Account settings for a Freeswitch account.
Indices and tables

• genindex
• modindex
• search