Asterisk OSP Module
User Guide

June 16, 2006

Table of Contents

1 Introduction

This document provides instructions on how to build and configure Asterisk V1.4 with the OSP Toolkit to enable secure, multi-lateral peering. The OSP Toolkit is an open source implementation of the OSP peering protocol and is freely available from www.sipfoundry.org. The OSP standard defined by the European Telecommunications Standards Institute (ETSI TS 101 321) www.esti.org. If you have questions or need help, building Asterisk with the OSP Toolkit, please post your question on the OSP mailing list at https://list.sipfoundry.org/mailman/listinfo/osp.

2 OSP Toolkit

Please reference the OSP Toolkit document "How to Build and Test the OSP Toolkit" available from www.sipfoundry.org/OSP/OSPclient .

2.1 Build OSP Toolkit

The software listed below is required ti build and use the OSP Toolkit:

- OpenSSL (required for building) - Open Source SSL protocol and Cryptographic Algorithms (version 0.9.7g recommended) from www.openssl.org. Pre-compiled OpenSSL binary packages are not recommended because of the binary compatibility issue.
- Perl (required for building) - A programming language used by OpenSSL for compilation. Any version of Perl should work. One version of Perl is available from www.activestate.com/ActivePerl. If pre-compiled OpenSSL packages are used, Perl package is not required.
- C compiler (required for building) - Any C compiler should work. The GNU Compiler Collection from www.gnu.org is routinely used for building the OSP Toolkit for testing.
- OSP Server (required for testing) - Access to any OSP server should work. Open source OSP servers are available from www.sipfoundry.org/osp, a free commercial OSP server may be downloaded from www.transnexus.com and an OSP server osptestserver.transnexus.com is freely available on the internet for testing for testing. Please contact support@transnexus.com for testing access to osptestserver.transnexus.com.

2.1.1 Unpacking the Toolkit

After downloading the OSP Toolkit (version 3.3.4 or later release) from www.sipfoundry.org, perform the following steps in order:

1) Copy the OSP Toolkit distribution into the directory where it will reside, say /usr/src.
2) Un-package the distribution file by executing the following command:

```
gunzip -c OSPToolkit-###.tar.gz | tar xvf -
```

Where ### is the version number separated by underlines. For example, if the version is 3.3.4, then the above command would be:

```
gunzip -c OSPToolkit-3_3_4.tar.gz | tar xvf -
```

A new directory (TK-3_3_4-20051103) will be created within the same directory as the tar file.

3) Go to the TK-3_3_4-20051103 directory by running this command:

```
cd TK-3_3_4-20051103
```

Within this directory, you will find directories and files similar to what is listed below if the command "ls -F" is executed):

```
ls -F
enroll/
RelNotes.txt                          lib/
README.txt                            license.txt
bin/                                  src/
crypto/                               test/
include/
```

2.1.2 Preparing to build the OSP Toolkit

4) Compile OpenSSL according to the instructions provided with the OpenSSL distribution (You would need to do this only if you don't have openssl already).
5) Copy the OpenSSL header files (the *.h files) into the crypto/openssl directory within the osptoolkit directory. The OpenSSL header files are located under the openssl/include/openssl directory.
6) Copy the OpenSSL library files (libcrypto.a and libssl.a) into the lib directory within the osptoolkit directory. The OpenSSL library files are located under the openssl directory.
   Note: Since the Asterisk requires the OpenSSL package. If the OpenSSL package has been installed, 4~6 are not necessary.

2.1.3 Building the OSP Toolkit

7) Optionally, change the install directory of the OSP Toolkit. Open the Makefile in the /usr/src/TK-3_3_4-20051103/src directory, look for the install path variable – INSTALL_PATH, and edit it to be anywhere you want (defaults /usr/local).
   Note: Please change the install path variable only if you are familiar with both the OSP Toolkit and the Asterisk. Otherwise, it may case that the Asterisk does not support the OSP protocol.
8) From within the OSP Toolkit directory (/usr/src/TK-3_3_4-20051103), start the compilation script by executing the following commands:

```
cd src
make clean; make build
```

2.1.4 Installing the OSP Toolkit

The header files and the library of the OSP Toolkit should be installed. Otherwise, you must specify the OSP Toolkit path for the Asterisk.

9) Use the same script to install the Toolkit.

```
make install
```

The make script is also used to install the OSP Toolkit header files and the library into the INSTALL_PATH specified in the Makefile.

Note: Please make sure you have the rights to access the INSTALL_PATH directory. For example, in order to access /usr/local directory, normally, you should be root.

By default, the OSP Toolkit is compiled in the production mode. The following table identifies which default features are activated with each compile option:

Default Feature	Production	Development
Debug Information Displayed	No	Yes

The "Development" option is recommended for a first time build. The "CFLAGS" definition in the Makefile must be modified to build in development mode.

2.1.5 Building the Enrollment Utility

Device enrollment is the process of establishing a trusted cryptographic relationship between the VoIP device and the OSP Server. The Enroll program is a utility application for establishing a trusted relationship between and OSP client and an OSP server. Please see the document "Device Enrollment" at www.sipfoundry.org/OSP/OSPclient for more information about the enroll application.

10) From within the OSP Toolkit directory (/usr/src/TK-3_3_4-20051103), execute the following commands at the command prompt:

```
cd enroll
make clean; make linux
```

Compilation is successful if there are no errors anywhere in the compiler output. The enroll program is now located in the /usr/src/TK-3_3_4-20051103/bin directory. By this point, a fully functioning OSP Toolkit should have been successfully built.

2.2 Obtain Crypto Files

The OSP module in Asterisk requires three crypto files containing local certificate (localcert.pem), private key (pkey.pem), and CA certificate (cacert_0.pem). Asterisk will try to load the files from the Asterisk public/private key directory - /var/lib/asterisk/key. If the files are not present, the OSP module will not start and the Asterisk will not support the OSP protocol. Use the enroll.sh script from the toolkit distribution to enroll the Asterisk OSP module with an OSP server to obtain the crypto files. Documentation explaining how to use the enroll.sh script (Device Enrollment) to enroll with an OSP server is available at www.sipfoundry.org/OSP/ospclient. Copy the files file generated by the enrollment process to the Asterisk configuration directory.

Note: The osptestserver.transnexus.com is configured only for sending and receiving non-SSL messages, and issuing signed tokens. If you need help, post a message on the OSP mailing list of www.sipfoundry.org or send an e-mail to support@transnexus.com.

The enroll.sh script takes the domain name or IP addresses of the OSP servers that the OSP Toolkit needs to enroll with as arguments, and then generates pem files – cacert_#.pem, certreq.pem, localcert.pem, and pkey.pem. The '#' in the cacert file name is used to differentiate the ca certificate file names for the various SP's (OSP servers). If only one address is provided at the command line, cacert_0.pem will be generated. If 2 addresses are provided at the command line, 2 files will be generated – cacert_0.pem and cacert_1.pem, one for each SP. The example below shows the usage when the client is registering with osptestserver.transnexus.com. If all goes well, the following text will be displayed. The gray boxes indicate required input.

```
./enroll.sh osptestserver.transnexus.com
Generating a 512 bit RSA private key
.......................+++++++++++
.........+++++++++++
writing new private key to 'pkey.pem'
-----
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.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

Error Code returned from openssl command : 0

CA certificate received
[SP: osptestserver.transnexus.com]Error Code returned from getcacert
command : 0

output buffer after operation: operation=request
output buffer after nonce: operation=request&nonce=1655976791184458
X509 CertInfo context is null pointer
Unable to get Local Certificate
depth=0 /CN=osptestserver.transnexus.com/O=OSPServer
verify error:num=18:self signed certificate
```

```
verify return:1
depth=0 /CN=osptestserver.transnexus.com/O=OSPServer
verify return:1
The certificate request was successful.
Error Code returned from localcert command : 0
```

The files generated should be copied to the /var/lib/asterisk/key directory.

Note: The script enroll.sh requires AT&T korn shell (ksh) or any of its compatible variants. The /usr/src/TK-3_3_4-20051103/bin directory should be in the PATH variable. Otherwise, enroll.sh cannot find the enroll file.

3 Asterisk

3.1 OSP Support Implementation

In Asterisk, all OSP support is implemented as dial plan functions.

3.1.1 OSPAuth

OSP token validation function.
Input:
- OSPPEERIP: last hop IP address
- OSPINTOKEN: inbound OSP token
- provider: OSP service provider configured in osp.conf. If it is empty, default provider is used.
- priority jump

Output:
- OSPINHANDLE: inbound OSP transaction handle
- OSPINTIMELIMIT: inbound call duration limit
- OSPAUTHSTATUS: OSPAuth return value. SUCCESS/FAILED/ERROR

3.1.2 OSPLookup

OSP lookup function.
Input:
- OSPPEERIP: last hop IP address
- OSPINHANDLE: inbound OSP transaction handle
- OSPINTIMELIMIT: inbound call duration limit
- exten: called number
- provider: OSP service provider configured in osp.conf. If it is empty, default provider is used.
- priority jump

Output:
- OSPOUTHANDLE: outbound transaction handle
- OSPTECH: outbound protocol
- OSPDEST: outbound destination
- OSPCALLING: outbound calling number
- OSPOUTTOKEN: outbound OSP token

- OSPRESULTS: number of remain destinations
- OSPOUTTIMELIMIT: outbound call duration limit
- OSPLOOKUPSTATUS: OSPLookup return value. SUCCESS/FAILED/ERROR

3.1.3 OSPNex

OSP lookup next function.
Input:
- OSPINHANDLE: inbound transaction handle
- OSPOUTHANDLE: outbound transaction handle
- OSPINTIMELIMIT: inbound call duration limit
- OSPRESULTS: number of remain destinations
- cause: last destination disconnect cause
- priority jump

Output:
- OSPTECH: outbound protocol
- OSPDEST: outbound destination
- OSPCALLING: outbound calling number
- OSPOUTTOKEN: outbound OSP token
- OSPRESULTS: number of remain destinations
- OSPOUTTIMELIMIT: outbound call duration limit
- OSPNEXTSTATUS: OSPLookup return value. SUCCESS/FAILED/ERROR

3.1.4 OSPFinish

OSP report usage function.
Input:
- OSPINHANDLE: inbound transaction handle
- OSPOUTHANDLE: outbound transaction handle
- OSPAUTHSTATUS: OSPAuth return value
- OSPLOOKUPSTATUS: OSPLookup return value
- OSPNEXTSTATUS: OSPNext return value
- cause: last destination disconnect cause
- priority jump

Output:
- OSPFINISHSTATUS: OSPLookup return value. SUCCESS/FAILED/ERROR

3.2 Build with OSP Support

If the OSP Toolkit is installed in the default install directory, /usr/local, no additional configuration is required. If the OSP Toolkit is installed in another directory, say /myosp, Asterisk must be configured with the location of the OSP Toolkit.

```
--with-osptk=/myosp
```

Note: Please change the install path only if you familiar with both the OSP Toolkit and the Asterisk. Otherwise, the change may results Asterisk not supporting the OSP protocol.

Now, you can compile Asterisk according to the instructions provided with the Asterisk distribution.

3.3 Configure with OSP Support

3.3.1 osp.conf

```
;
; Open Settlement Protocol Sample Configuration File
;
; This file contains configuration of providers that
; are used by the OSP subsystem of Asterisk.  The section
; "general" is reserved for global options.  Each other
; section declares an OSP Provider.  The provider "default"
; is used when no provider is otherwise specified.
;
[general]
;
; Should hardware accelleration be enabled?  May not be changed
; on a reload.
;
accelerate=no
;
; Defines the token format that Asterisk can validate.
; 0 - signed tokens only
; 1 - unsigned tokens only
; 2 - both signed and unsigned
; The defaults to 0, i.e. the Asterisk can validate signed tokens
only.
;
tokenformat=0
;
[default]
;
; All paths are presumed to be under /var/lib/asterisk/keys unless
; the path begins with '/'
;
; Specify the private keyfile.  If unspecified, defaults to the name
; of the section followed by "-privatekey.pem" (e.g. default-
privatekey.pem)
;
privatekey=pkey.pem
;
; Specify the local certificate file.  If unspecified, defaults to
; the name of the section followed by "-localcert.pem"
;
localcert=localcert.pem
;
; Specify one or more Certificate Authority keys.  If none are
listed,
; a single one is added with the name "-cacert.pem"
;
cacert=cacert_0.pem
;
; Specific parameters can be tuned as well:
```

```
;
; maxconnections: Max number of simultaneous connections to the
provider (default=20)
; retrydelay:     Extra delay between retries (default=0)
; retrylimit:     Max number of retries before giving up (default=2)
; timeout:        Timeout for response in milliseconds (default=500)
;
maxconnections=20
retrydelay=0
retrylimit=2
timeout=500
;
; List all service points for this provider
;
;servicepoint=http://osptestserver.transnexus.com:1080/osp
servicepoint=http://OSP server IP:1080/osp
;
; Set the "source" for requesting authorization
;
;source=foo
source=[host IP]
;
; Set the authentication policy.
; 0 - NO
; 1 - YES
; 2 - EXCLUSIVE
; Default is 1, validate token but allow no token.
;
authpolicy=1
```

3.3.2 zapata/sip/iax.conf

There is no configuration required for OSP.

3.3.3 extensions.conf

An Asterisk box can be configured as OSP source/destination gateway or OSP proxy.

3.3.3.1 OSP Source Gateway

```
[PhoneSrcGW]
; Set calling number if necessary
exten => _XXXX.,1,Set(CALLERID(numner)=CallingNumber)
; OSP lookup using default provider, if fail/error jump to 2+101
exten => _XXXX.,2,OSPLookup(${EXTEN}||j)
; Set calling number which may be translated
exten => _XXXX.,3,Set(CALLERID(number)=${OSPCALLING})
; Dial to destination, 60 timeout, with call duration limit
exten =>
_XXXX.,4,Dial(${OSPTECH}/${OSPDEST},60,oL($[${OSPOUTTIMELIMIT}*1000])
)
; Wait 3 seconds
exten => _XXXX.,5,Wait,3
; Hangup
```

```
exten => _XXXX.,6,Hangup
; Deal with OSPLookup fail/error
exten => _XXXX.,2+101,Hangup
; OSP report usage
exten => h,1,OSPFinish(${HANGUPCAUSE})
```

3.3.3.2 OSP Destination Gateway

```
[PhoneDstGW]
; Get peer IP
exten => _XXXX.,1,Set(OSPPEERIP=${SIPCHANINFO(peerip)})
; Get OSP token
exten => _XXXX.,2,Set(OSPINTOKEN=${SIP_HEADER(P-OSP-Auth-Token)})
; Validate token using default provider, if fail/error jump to 3+101
exten => _XXXX.,3,OSPAuth(|j)
; Ringing
exten => _XXXX.,4,Ringing
; Wait 1 second
exten => _XXXX.,5,Wait,1
; Dial phone, timeout 15 seconds, with call duration limit
exten =>
_XXXX.,6,Dial(${DIALOUTANALOG}/${EXTEN:1},15,oL($[${OSPINTIMELIMIT}*1
000]))
; Wait 3 seconds
exten => _XXXX.,7,Wait,3
; Hangup
exten => _XXXX.,8,Hangup
; Deal with OSPAuth fail/error
exten => _XXXX.,3+101,Hangup
; OSP report usage
exten => h,1,OSPFinish(${HANGUPCAUSE})
```

3.3.3.3 Proxy

```
[GeneralProxy]
; Get peer IP
exten => _XXXX.,1,Set(OSPPEERIP=${SIPCHANINFO(peerip)})
; Get OSP token
exten => _XXXX.,2,Set(OSPINTOKEN=${SIP_HEADER(P-OSP-Auth-Token)})
; Validate token using default provider, if fail/error jump to 3+101
exten => _XXXX.,3,OSPAuth(|j)
; OSP lookup using default provider, if fail/error jump to 4+101
exten => _XXXX.,4,OSPLookup(${EXTEN}||j)
; Set calling number which may be translated
exten => _XXXX.,5,Set(CALLERID(number)=${OSPCALLING})
; Dial to 1st destination, 60 timeout, with call duration limit
exten =>
_XXXX.,6,Dial(${OSPTECH}/${OSPDEST},24,oL($[${OSPOUTTIMELIMIT}*1000])
)
; OSP lookup next, if fail/error jump to 7+101
exten => _XXXX.,7,OSPNext(${HANGUPCAUSE}||j)
```

```
; Set calling number which may be translated
exten => _XXXX.,8,Set(CALLERID(number)=${OSPCALLING})
; Dial to 2nd destination, 60 timeout, with call duration limit
exten =>
_XXXX.,9,Dial(${OSPTECH}/${OSPDEST},25,oL($[${OSPOUTTIMELIMIT}*1000])
)
; OSP lookup next, if fail/error jump to 10+101
exten => _XXXX.,10,OSPNext(${HANGUPCAUSE}||j)
; Set calling number which may be translated
exten => _XXXX.,11,Set(CALLERID(number)=${OSPCALLING})
; Dial to 3rd destination, 60 timeout, with call duration limit
exten =>
_XXXX.,12,Dial(${OSPTECH}/${OSPDEST},26,oL($[${OSPOUTTIMELIMIT}*1000]
))
; Hangup
exten => _XXXX.,13,Hangup
; Deal with OSPAuth fail/error
exten => _XXXX.,3+101,Hangup
; Deal with OSPLookup fail/error
exten => _XXXX.,4+101,Hangup
; Deal with 1st OSPNext fail/error
exten => _XXXX.,7+101,Hangup
; Deal with 2nd OSPNext fail/error
exten => _XXXX.,10+101,Hangup
; OSP report usage
exten => h,1,OSPFinish(${HANGUPCAUSE})
```