TLS Profile - CLI

In this section:

Related articles:

- TLS for Signaling
- Configuring SBC and LYNC in TLS Environment

The Datagram Transport Layer Security (DTLS) protocol provides authentication, data integrity, and confidentiality for communications between two applications over an Unreliable Datagram Protocol (UDP). The Secure Real-time Transport Protocol (SRTP) provides encryption, message authentication and integrity, and replay protection to the RTP data in both unicast and multicast applications. DTLS-SRTP is an extension to the DTLS protocol, where DTLS acts as the key management protocol. DTLS protocol is also extended to negotiate the SRTP crypto suites and parameters for use with those keys.

WebRTC is a signaling protocol defined for real-time communication between Web browsers. WebRTC has assigned DTLS-SRTP protocol for the media exchange between the browsers. The SBC includes the following DTLS functionality:

- Real-time communication between the web browsers by using DTLS-SRTP while inter-working with SIP networks.
- DTLS on the media path for key management for the SRTP-based media.
- The self-signed certificates to secure and authenticate DTLS associations. DTLS connections are secured by the two browsers sharing self-signed certificates as part of the media connection during a DTLS handshake between the browsers. The certificates are authenticated by checking a fingerprint, which is passed in the signaling path as part of the Session Description Protocol (SDP).

The SBC includes DTLS crypto suites that define a set of ciphers (algorithms used for encrypting data) which allow the selection of an appropriate level of security. When a TLS connection is established, the client and server exchange information about which cipher suites they have in common.

Use the TLS Profile to configure a profile for implementing the TLS protocol for SIP over TLS. The TLS profile is associated with a SIP Signaling Port.

Note
The settings within the default TLS Profile may be modified. Also, the supported transport protocols must be set to allow SIP over TLS.

Note
If a zone's `sipSigPort` is configured for `transportProtocolsAllowed` = `sip-tls-tcp`, the SBC increments the configured `portNumber` by 1 and uses it as the new port number for SIP over TLS signaling. The SBC then opens a TCP socket for SIP over TLS for the new TCP port number.

Example: When `sipSigPort` is configured with a `portNumber` of 5060 and `transportProtocolsAllowed` = `sip-tls-tcp`, the SBC listens on TCP port 5061 for SIP over TLS.

Command Syntax

```
% set profiles security tlsProfile <profile name>
 acceptableCertValidationErrors <invalidPurpose | none>
 allowedRoles <clientandserver | server>
 appAuthTimer <1-60 seconds>
 authClient <false | true>
 cipherSuite1 <cipher suite>
 cipherSuite2 <cipher suite>
 cipherSuite3 <cipher suite>
 clientCertName <name>
    handshakeTimer <1-60 seconds>
 ocspProfileName <name>
 peerNameVerify <disabled | enabled>
 serverCertName <name>
 sessionResumpTimer <0-86400 seconds>
 suppressEmptyFragments <disabled | enabled>
 v1_0 <disabled | enabled>
    v1_1 <disabled | enabled>
    v1_2 <disabled | enabled>
```

Command Parameters

The TLS Profile Parameters are as shown below:

Table 1: TLS Profile Parameters

Parameter	Length/Range	Description
`tlsProfile`	1-23	`<name>` - Name assigned to this Transport Layer Security (TLS) profile. You can add a maximum of 16 TLS Profiles.
`acceptableCertValidationErrors`	N/A	Use this parameter to specify if certificate chain validation errors are acceptable while validating the peer certificate. • `invalidPurpose` • `none` (default)
`allowedRoles`	N/A	Allowed TLS roles for this TLS profile. • `clientandserver` – (default) Choose to select both a TLS client and server role, depending on the request direction. This is primarily for peering applications. • `server` – The SBC will only be a TLS server. This is primarily for access applications.
`appAuthTimer`	1-60	The higher layer authentication timer in seconds. (default = 5).

`authClient`	N/A	Indicates whether or not a TLS client is forced to authenticate itself within TLS. If set to false, the client is not required to authenticate itself at the TLS layer, but must complete authentication within a higher-level protocol after the TLS connection is established (that is, SIP registration). • `false` • `true` (default)
`cipherSuite1`	N/A	Use this parameter to specify the first TLS Cipher Suite choice for this profile. See Supported Cipher Suites table below fo r the list of cipher suites.
`cipherSuite2`	N/A	Use this optional parameter to specify the second TLS Cipher Suite choice for this profile. See Supported Cipher Suites table below fo r the list of cipher suites.
`cipherSuite3`	N/A	Use this optional parameter to specify the third TLS Cipher Suite choice for this profile. See Supported Cipher Suites table below fo r the list of cipher suites.
`clientCertName`	1-23	The name of the default Client Certificate to be used by this TLS profile, created using the `SECURITY PKI` configuration object.
`handshakeTimer`	1-60	The time (in seconds) in which the TLS handshake must be completed. The timer starts when the TCP connection is established. (default = 5)
`ocspProfileName`	1-23	Name of OCSP profile object referenced by TLS profile.

peerNameVerify	N/A	This flag is used to verify the authenticity of a client certificate. When enabled, the SBC validates the TLS peer name of the SIP peer against either a Common Name (CN) present in the subject or the DNS names or IP addresses in the Subject Alternate Name Extension (SAN) field in the certificate presented by the peer. For example, if the SBC establishes a TLS session with "sonus.customer.net", this feature verifies the CN in the subject (or a DNS entry in the SAN field) of the certificate contains "sbc.customer.net" as the value. • disabled (default) • enabled **NOTE**: This flag is used in conjunction with the trunk group parameter, tlsPeerName, which associates the TLS peer name with a trunk group. Refer to SIP TG - Signaling - TLS Peer Name - CLI for details.
serverCertName	1-23	Specifies the name of the Server Certificate to be used by this TLS profile, created using the SECURITY PKI configuration object.
sessionResumpTimer	0-86400	The TLS session resumption period (in seconds) for which cached sessions are retained. TLS allows successive connections to be created within one TLS session (and the resumption of a session after a TLS connection is closed or after a server card failover) without repeating the entire authentication and other setup steps for each connection, except when the space must be reclaimed for a new session. (default = 3600)
suppressEmptyFragments	N/A	Enable flag to prevent the SBC from inserting empty fragments when sending packets on TLS over TCP connection in support of older versions of TLS implementation. • disabled (default) • enabled
v1_0	N/A	TLS protocol version 1.0 (see note below) • disabled • enabled (default)
v1_1	N/A	TLS protocol version 1.1 (see note below) • disabled (default) • enabled

`v1_2`	N/A	TLS protocol version 1.2 (see note below) • `disabled` (default) • `enabled`

Note
Sonus recommends using the highest TLS version supported by both the SBC and the peer equipment.

Table 2: Supported DTLS Crypto Suites

Authentication Mechanism	Public/Private Key Pair	Confidentiality Cipher and Mode	Integrity Cipher
RSA-WITH-NULL-SHA The integrity cipher used for the TLS Record protocol.	RSA	NULL	SHA-1
RSA-WITH-AES-128-CBC-SHA (default) Confidentiality cipher and mode for the TLS Record protocol.	RSA	AES-128-CBC	SHA-1
RSA-WITH-AES-128-CBC-SHA-256 Confidentiality cipher and mode for the TLS Record protocol with SHA-256 as the hash function.	RSA	AES-128-CBC	SHA-256
RSA-WITH-AES-256-CBC-SHA Confidentiality cipher and mode for the TLS Record protocol with AES 256 encryption.	RSA	AES-256-CBC	SHA-1
RSA-WITH-AES-256-CBC-SHA-256* Confidentiality cipher and mode for the TLS Record protocol with AES 256 encryption and SHA-256 as the hash function.	RSA	AES-256-CBC	SHA-256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384** Confidentiality cipher and mode for TLS Recod with AES256 CBC and SHA384 as hash function.	ECDH-ECDSA	AES-256-CBC	SHA-384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384** Confidentiality cipher and mode for TLS Recod with AES256 GCM and SHA384 as hash function.	ECDH-ECDSA	AES-256-GCM	SHA-384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA Confidentiality cipher and mode for the TLS Record protocol using ECDHE (Elliptic Curve Diffie-Hellman key Exchange) with AES128 CBC and SHA as hash function.	ECDHE-RSA	AES-128-CBC	SHA-1
TLS-ECDHE-RSA-WITH-AES-256-CBC-SHA-384* Confidentiality cipher and mode for the TLS Record protocol using ECDHE (Elliptic Curve Diffie-Hellman key Exchange) with AES256 CBC and SHA384 as hash function.	ECDHE-RSA	AES-256-CBC	SHA-384

* To use this cipher, TLS version 1.2 must be enabled in the TLS Profile.

** To use this cipher, TLS version 1.2 must be enabled in the TLS Profile and SSL certificates must be created using ECC keys.

Terms used in this table:

RSA – Authentication based on X.509 certificates using RSA public/private key pairs
AES-128 – Advanced Encryption Standard (128-bit key length)
CBC – Cipher Block Chaining
SHA – Secure Hash algorithm

Note
When FIPS-140-2 mode is enabled, do not use the `rsa-with-null-sha` option.

Command Examples

```
show profiles security tlsProfile defaultTlsProfile
appAuthTimer       5;
handshakeTimer     5;
sessionResumpTimer 3600;
cipherSuite1       rsa-with-aes-128-cbc-sha;
allowedRoles       clientandserver;
v1_0               enabled;
v1_1               enabled;
v1_2               enabled;

set profiles security tlsProfile defaultTlsProfile ocspProfileName myOcspProfile
commit

show profiles security tlsProfile defaultTlsProfile
appAuthTimer       5;
handshakeTimer     5;
sessionResumpTimer 3600;
cipherSuite1       rsa-with-aes-128-cbc-sha;
allowedRoles       clientandserver;
ocspProfileName    myOcspProfile;
v1_0               enabled;
v1_1               enabled;
v1_2               enabled;
```

```
set profiles security tlsProfile TLS-1 v1_2 enabled
set profiles security tlsProfile TLS-1 cipherSuite1 tls_ecdh_ecdsa_with_aes_256_gcm_sha384
commit

show profiles security tlsProfile TLS-1
cipherSuite1 tls_ecdh_ecdsa_with_aes_256_gcm_sha384;
v1_2         enabled;
```