# Best Practices: Securing your OpenTok App

TokBox recognizes that security is an essential consideration for any business interested in integrating real time communications into its website, app or service. The OpenTok platform is a reliable and secure platform on which you can build applications that meet your company, industry or client security needs.

Whether you're new to the OpenTok platform, or have years of experience, here is a useful set of best practices you can employ when developing with OpenTok to help you build a secure application.

| Best Practice | Details |
|---|---|
| **Personal Information** | |
| • **Keep the API key and secret private and secure** | The API key and secret are used to create tokens that grant access to sessions, retrieve archive metadata and change archive storage credentials, as well as other administrative operations on your account.<br><br>To avoid compromising your credentials, you should **always** keep your API secret private. Some key measures you can take:<br><br>- Never save the API secret in any public source code repositories<br>- Never save the API secret in any client side libraries, or even compiled mobile SDKs<br>- Use only https URLs to make REST calls to the OpenTok servers |
| • **Generate a unique Session ID per call and token per participant** | You need to generate a Session ID to initiate a call. The tokens that enable the participants to join are unique to a Session ID. The tokens have an expiry but it may be longer than the duration of your call. Therefore, if you have consecutive meetings using the same session ID, earlier users may still be able to connect to the new meeting.<br><br>To avoid this:<br><br>- Generate a unique Session ID for each new meeting<br>- Generate a unique token for each participant of that meeting.<br><br>See here on how to generate tokens and sessions: http://bit.ly/1HAeVZF |

| | |
|---|---|
| • **Ensure server generating token is behind authenticated endpoint** | It is important to place the server generating the token behind an authenticated endpoint because anyone with access to that server could end up generating new tokens and could abuse the app to generate usage. |
| • **Don't use personal information in token data** | The token data is a string containing metadata describing the connection. However, this data is passed to all users in the session and is also readable through the OpenTok client logs.<br><br>This means you should **never** use unencrypted sensitive or personal information in the token data.<br><br>See here on how to add data to your tokens: http://bit.ly/1HAeVZF |
| **Relayed vs Routed Mode** | |
| • **End to End Media Encryption** | During a routed session, media streams are decrypted while in the OpenTok Platform and then encrypted once again prior to being sent to the subscribing client. This decryption is necessary for managing group sessions, intelligent quality control, and archiving of sessions (if used).<br><br>However, if your application requires full end-to-end encryption of all media, you may choose to use relayed sessions. Be aware that you would not be able to use archiving, and performance will not be managed as well in low bandwidth / high packet loss networks or with groups. |
| **Archiving** | |
| • **Manage archive deletion** | An archive successfully uploaded to your storage will be automatically deleted from the OpenTok archiving server at the time of upload.<br><br>In case of failure to upload, OpenTok storage is provided as a default fallback option. This means the archive will be stored for 72 hours on the OpenTok server.<br><br>You will be alerted via email for every archive that fails to reach your storage. You can then use the REST API to download the archive from the archive URL.<br><br>Following the download, you can choose to immediately delete the archive to avoid it being on the OpenTok storage for the remainder of the fallback period.<br><br>See here for more information on how to delete an archive: http://bit.ly/1LurEMa |

| | |
|---|---|
| • **Encrypt archives** | Using the REST API it's possible to encrypt archives using partner certificates (the password is stored as part of the metadata on the archive).<br><br>Upon accessing the archive, you can then use the password to decrypt the archives and access the media.<br><br>Please note this is currently a beta API. |
| • **Control who is able to begin archiving** | You can only archive a session using the REST API. To control who can initiate the archive, you can programmatically decide which view of the application includes the option to start archiving. Those unauthorized would have a limited view with no option to archive.<br><br>In addition to authenticating your own users, you should also consider a strategy for authorization. Once you know who a user is (authentication), and that they are in fact allowed to start an archive (authorization), you've reduced the risk of any unintended user from causing archives to be recorded. |
| • **Ensure minimum set of privileges** | The minimum set of permissions required to upload archives to our storage is mentioned here: http://bit.ly/1Eozkt7<br><br>Partners should not provide any additional permissions for the credentials that they store with OpenTok. |
| **Alerts and Controls** | |
| • **Limit maximum number of users in a session** | In order to have more control over the number of people in a session, your application should limit the maximum number of users in a session. This could be useful if you're trying to limit your usage. |
| • **Display subscriber count** | You may also wish to set up a subscriber count to display in your application. This is a useful way of knowing when a connection is subscribing but not publishing. |
| • **Set up Moderator permissions to force disconnect** | OpenTok platform provides the capability to remove a user from a session. For example, in case of violation of terms of services, you can enable the moderator in the session to remove the violating participant from the session via force disconnect or force unpublish. Find out how here: http://bit.ly/1FJ0dzo |
| • **Allocate Subscriber-only permissions for those that do not need to publish** | In some use cases you may want to limit the number of people who can publish in a session. This is easily implemented by not extending publish permissions to every participant.<br><br>See here for generating tokens with the right roles: http://bit.ly/1Fry7Sm |