Wowza Streaming Engine™ media server software extends SecureToken playback protection to all supported streaming protocols and includes hashing options for generating the security token that’s exchanged between Wowza Streaming Engine and its clients. SecureToken is a challenge/response system that helps to protect content against spoofing threats. Each connection is protected by a random key and a password (shared secret).

**Note:** Wowza Streaming Engine 4.1 or later is required to use the functionality described in this article.

### Overview

Wowza Streaming Engine has a **SecureToken** module that provides playback protection using a security token that’s exchanged between the server and clients. You can enable Wowza Streaming Engine to use a secure hash algorithm (SHA) for generating a security token for playback protection over all supported streaming protocols. The supported secure hash algorithms are:

- SHA-256
- SHA-384
- SHA-512

When a player requests a stream from Wowza Streaming Engine, the request URL has a set of public SecureToken query parameters, one of which is a hash generated by the client webpage. The hash is generated from the content path, the SecureToken query parameters, and a private shared secret. After receiving the request, the **SecureToken** module uses an algorithm to generate a hash based on the content path, the received SecureToken query parameters, and the private shared secret. Wowza Streaming Engine compares this hash to the hash it gets from the client. If the values don’t match, it rejects the connection.

**Note:** To use SecureToken with Wowza Player, additional configuration is required. For details, see [Configuring players for SecureToken](#).

You can enable Wowza Streaming Engine to use the Tiny Encryption Algorithm (TEA) instead of a hash algorithm for playback protection over RTMP. When an Adobe Flash player requests an RTMP stream from Wowza Streaming Engine, the **SecureToken** module uses the TEA to generate a unique encrypted key based on the private shared secret for the pending connection and returns the key as part of the `NetConnection.onStatus info` object. The client decrypts the unique key using the same shared secret and sends the result back to Wowza Streaming Engine, which compares this key to the originally generated key. If the values don’t match, Wowza Streaming Engine rejects the connection.

**Note:** Client-side code must be integrated into the Flash-based player so that it can respond correctly to the SecureToken challenge. For details, see [Configuring players for SecureToken](#).

### Setting up SecureToken protection in Wowza Streaming Engine

This section shows you how to set up SecureToken playback protection in Wowza Streaming Engine Manager for an example application named **vod**. A **vod** application is included in the default Wowza Streaming Engine installation.

1. Make sure Wowza Streaming Engine Manager is running. For more information, see [Start and stop Wowza Streaming Engine](#).
2. In the **Applications** contents panel, click **vod**, and then click **Playback Security**.
3. Click **Edit**.
4. Under **SecureToken**, select the type of secure token playback protection that you want to use. We recommend **Protect all protocols using hash (SecureToken version 2)**, which protects all streaming protocols using the latest hash algorithms.

To protect only the RTMP protocol using the Tiny Encryption Algorithm, select **Protect RTMP protocol using TEA (SecureToken version 1)**. This option only protects the RTMP protocol and does not protect any other streaming protocol.

5. Under **Shared Secret**, enter a string of alphanumeric (a - z, A - Z, 0 - 9) characters in the box, or click **Generate SecureToken**.
**Shared Secret** to enter a random string of alphanumeric characters.

**Important:** Your web server administrator must use this same shared secret value to generate the client-side hash when the client webpage is generated. JavaScript code shouldn’t be used in the client webpage to generate the hash as the code is visible in the webpage source and would pose a potential security risk.

6. If you opted to protect all streaming protocols using a hash algorithm, specify the hashing options:

   a. Select a hash algorithm in the **Hash Algorithm** list.

   For compatibility with Flash players that you’ve already configured to use SecureToken with previous versions of Wowza Streaming Engine, you can select **Use TEA for RTMP token security** to protect RTMP streams using the Tiny Encryption Algorithm. All other streaming protocols will continue to be protected using the hash algorithm that you selected.

   You can enable additional security by selecting **Include client IP address in hash generation**, which causes the hash that’s generated to be unique for each connecting client.

   **Note:** Due to network complexities, the client webpage that generates the client-side hash may use a client IP address that’s different from the IP address detected by Wowza Streaming Engine. This will cause Wowza Streaming Engine to generate a hash that’s different from the hash it gets from the client and the connection will be rejected.

   b. Enter a custom query parameter prefix in the **Hash Query Parameter Prefix** box or use the default value (wowzatoken).

   The prefix value can only have the following characters that are safe to use in URLs: alphanumeric characters (a - z, A - Z, 0 - 9), percent sign (%), period (.), underscore (_), tilde (~), and hyphen (-).

7. Click **Save**.

**SecureToken query parameters**

SecureToken public query parameters are included in request URLs sent from clients to Wowza Streaming Engine. Wowza Streaming Engine must be able to identify query parameters that are used with SecureToken in request URLs so that it can include them when generating its hash; therefore, each query parameter to be used with SecureToken must begin with a SecureToken query parameter prefix. Other query parameters in request URLs should not use a SecureToken query parameter prefix as this will cause Wowza Streaming Engine to generate an improper hash. The examples in this section use the SecureToken query parameter prefix wowzatoken.

- **hash** – *(Required)* The hash generated at the client as a URL-safe Base64-encoded string. URL-safe Base64 encoding replaces the '+' character with the '-' character and the '/' character with the '_' character. For example:
  
  wowzatokenhash=VSNlS5S2Na5KxwwiVLX1cHwC90CF21HdmCm9v_8Bh0o=

- **starttime** – *(Optional)* The time (in UTC seconds) when SecureToken playback security starts. If this parameter isn’t included in the URL or if the parameter is included with a value of 0, SecureToken playback security is enabled immediately. Player connections that occur before the start time will be rejected. For example:
  
  wowzatokenstarttime=1405036800 *(11 July 2014 00:00:00 GMT)*

- **endtime** – *(Optional)* The time (in UTC seconds) when SecureToken playback security stops. If this parameter isn’t included in the URL or if the parameter is included with a value of 0, SecureToken playback security is always enabled, the SecureToken doesn’t expire, and security may be more vulnerable. Player connections that occur after the end time will be rejected. For example:
  
  wowzatokenendtime=1405123200 *(12 July 2014 00:00:00 GMT, 24 hours from the **starttime**.)*

  **Note:** We recommend that you specify an **endtime**, as it’s required in most use cases; however, a SecureToken that doesn’t expire may be needed in some situations, so you can omit this parameter if necessary.

- **Custom parameters** – You can add custom query parameters to be used in the generation of the hash. These parameters can be key/value pairs or single key values. Pairs look like this:
A single key value looks like this:

A unique customer-specific hash parameter to ensure that a unique SecureToken hash is generated per customer, like this:

You can add a unique customer-specific hash parameter to ensure that a unique SecureToken hash is generated per customer, like this:

Hash generation

Wowza Streaming Engine and its clients must follow certain rules to ensure the string that's created and hashed is identical and that the correct hash is generated in both places.

**Important:** The client web server should generate the hash when it generates the client webpage. You shouldn't use JavaScript code in the client webpage to generate the hash as the code is visible in the webpage source and would pose a potential security risk.

1. Start the string with the content path to the streaming asset (live stream name or VOD file name). The content path is the part of the URL that starts with the application name (excluding the '/' that precedes the application name) and continues through to the end of the stream name or file name. Be sure to exclude all HTTP request keywords after the stream name or file name (for example, /manifest.m3u8, /media.ts, /Manifest, /manifest.f4v, and so on).
2. Append the '?' character to the path that you created in the previous step. This character separates the content path from the public SecureToken query parameters that follow.
3. Append the public SecureToken query parameters, shared secret, and client IP address (if applicable) to the '?' character that you created in the previous step. These items must be in alphabetical order and separated by the '&' character.

The following examples use SHA-256, in binary, to calculate the hash.

**HLS example**

This example is based on an Apple HLS VOD request where no application instance is specified in the URL. The prefix that's used to identify the public query parameters is set to a custom value. A custom public query parameter is included in the hash generation, as well as the client IP address, and the SecureToken start and end times are specified.

**Content URL:** http://192.168.1.1:1935/vod/sample.mp4/playlist.m3u8
**Content path:** vod/sample.mp4
**Custom SecureToken prefix:** myTokenPrefix
**Custom SecureToken public query parameter:** myTokenPrefixCustomParameter=abcdef
**Token start time:** myTokenPrefixstarttime=1395230400
**Token end time:** myTokenPrefixendtime=1500000000

The parameters used to create the string used for hashing (not in alphabetical order):

- myTokenPrefixstarttime=1395230400
- myTokenPrefixendtime=1500000000
- myTokenPrefixCustomParameter=abcdef
- mySharedSecret
- 192.168.1.2

String used for hashing (in required alphabetical order):

vod/sample.mp4?192.168.1.2&mySharedSecret&myTokenPrefixCustomParameter=abcdef&myTokenPrefixendtime=1500000000&myTokenPrefixstarttime=1395230400

HLS manifest request URL sent to Wowza Streaming Engine:

http://192.168.1.1:1935/vod/sample.mp4/playlist.m3u8?myTokenPrefixstarttime=1395230400&myTokenPrefixendtime=1500000000&myTokenPrefixCustomParameter=abcdef&myTokenPrefixhash=TgJft5hsjKyC5Rem_EoUNP7xZrxbqVPhhd0GxIcA2oo=

**Smooth Streaming example**
This example is based on a Smooth Streaming VOD request where the application instance is specified in the URL. The default query parameter prefix (wowzatoken) is used, the client IP address is included in the hash generation, and the SecureToken start and end times are specified.

**Content URL:** http://192.168.1.1:1935/vod/sample.mp4/Manifest  
**Content path:** vod/_myInstance_/sample.mp4  
**Token start time:** wowzatokenstarttime=1395230400  
**Token end time:** wowzatokenendtime=1500000000

The parameters used to create the string used for hashing (not in alphabetical order):

- wowzatokenstarttime=1395230400  
- wowzatokenendtime=1500000000  
- xyzSharedSecret  
- 192.168.1.2

String used for hashing (in required alphabetical order):

vod/_myInstance_/sample.mp4?192.168.1.2&wowzatokenendtime=1500000000&wowzatokenstarttime=1395230400&xyzSharedSecret

Smooth Streaming request URL sent to Wowza Streaming Engine:

http://10.0.2.31:1935/vod/_myInstance_/sample.mp4/Manifest?wowzatokenstarttime=1395230400&wowzatokenendtime=1500000000&wowzatokenhash=acr_wmhSVBarMD1cBIOuBGSO6_owxSyljzdaCCB2t:

**RTSP example**

This example is based on an RTSP VOD request where the application instance is specified in the URL. The default query parameter prefix (wowzatoken) is used, a custom public query parameter is included in the hash generation, and the SecureToken end time is specified. The client IP address isn’t included in the hash generation and the SecureToken start time isn’t specified (SecureToken playback security is enabled immediately).

**Content URL:** rtsp://192.168.1.1:1935/vod/sample.mp4  
**Content path:** vod/_myInstance_/sample.mp4  
**Custom SecureToken public query parameter:** wowzatokenCustomParameter=abcdef  
**Token end time:** wowzatokenendtime=1500000000

The parameters used to create the string used for hashing (not in alphabetical order):

- wowzatokenCustomParameter=abcdef  
- wowzatokenendtime=1500000000  
- xyzSharedSecret

String used for hashing (in required alphabetical order):

vod/_myInstance_/sample.mp4?wowzatokenCustomParameter=abcdef&wowzatokenendtime=1500000000&xyzSharedSecret

RTSP URL sent to Wowza Streaming Engine:

rtsp://10.0.2.31:1935/vod/_myInstance_/sample.mp4?wowzatokenCustomParameter=abcdef&wowzatokenendtime=1500000000&wowzatokenhash=kJ591xB2LT-X0QA9UdoRx61uw6A IoSc_3Cy_9h118=

**Configuring players for SecureToken**

To use SecureToken, the following players require additional configuration:

- Wowza Player  
- Players that support Flash

**Wowza Player**

To use the SecureToken module with Wowza Player, do the following:

1. Create a custom Wowza Player with Wowza Player Builder.
2. In the player embed code, add the `withCredentials` property and set it to `true`.

For more information, see Creating a custom Wowza Player with Wowza Player Builder and Customize Wowza Player with configuration properties.

Players that support Flash

If you opt to protect all streaming protocols using a hash algorithm, you must pass the SecureToken query parameters to players that support Flash so that the parameters are then sent to Wowza Streaming Engine. For example, using JW Player 6.8, parameters can be passed in as follows:

Where is:

```plaintext
rtmp://[wowza-ip-address]:1935/vod/sample.mp4?
wowzatokenendtime=1500000000&wowzatokenCustomParameter=abcdef&wowzatokenhash=kJ591xB2LT-X0QA9DofRx61wp6A_IoSc_jCx_9h118=
```

If you opt to protect the RTMP streaming protocol using the Tiny Encryption Algorithm, you must configure your client-side ActionScript player code so that it can respond to the SecureToken challenge.

SecureToken property reference

This section describes the application properties that you can use to set up SecureToken playback security. You can set these properties by adding them to the container at the end of the `[install-dir]/conf/[applicationName]/Application.xml` file. Be sure to get the correct container—there are several in `Application.xml`.

Notes:
- You must restart the Wowza Streaming Engine after configuring the properties.
- If you set up SecureToken playback security in Wowza Streaming Engine Manager, you don't need to configure them in the `Application.xml` file.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>securitySecureTokenVersion</code></td>
<td>Selects the SecureToken version to use. Valid values are 1, which protects playback over RTMP using the Tiny Encryption Algorithm (TEA), or 2, which protects playback over all supported streaming protocols using a hash algorithm.</td>
</tr>
<tr>
<td><code>securitySecureTokenSharedSecret</code></td>
<td>Specifies the string that's used as the shared secret by Wowza Streaming Engine and client for all versions of SecureToken. The string must contain only alphanumeric characters (a - z, A - Z, 0 - 9).</td>
</tr>
<tr>
<td><code>securitySecureTokenUseTEAforRTMP</code></td>
<td>When <code>securitySecureTokenVersion</code> is set to 2, this property, when set to <code>true</code>, overrides SecureToken playback protection using the hash algorithm for RTMP streams and uses TEA instead.</td>
</tr>
<tr>
<td><code>securitySecureTokenHashAlgorithm</code></td>
<td>When <code>securitySecureTokenVersion</code> is set to 2, this property specifies the hash algorithm that's used to generate the hash. Valid values are SHA-256, SHA-384, and SHA-512.</td>
</tr>
<tr>
<td><code>securitySecureTokenQueryParametersPrefix</code></td>
<td>When <code>securitySecureTokenVersion</code> is set to 2, this property specifies the prefix that's used on all SecureToken query parameters. Valid values must use only the following characters: alphanumeric (a - z, A - Z, 0 - 9), percent (%), period (.), underscore (_), tilda (~), and hyphen (-)/B</td>
</tr>
<tr>
<td><code>securityDebugLogRejections</code></td>
<td>Used for all <code>securitySecureTokenVersion</code> values. Set to <code>true</code> to log rejected connection attempts.</td>
</tr>
<tr>
<td><code>securityDebugLogDetails</code></td>
<td>Used for all <code>securitySecureTokenVersion</code> values. Set this value to <code>true</code> to enable SecureToken debug logging.</td>
</tr>
</tbody>
</table>

Note: To add this property in Wowza Streaming Engine Manager, see Additional logging.

<table>
<thead>
<tr>
<th>Notes:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>To add this property in Wowza Streaming Engine Manager, see Additional logging.</td>
<td></td>
</tr>
<tr>
<td>Enabling debug logging can produce a lot of log entries, which can create very large log files.</td>
<td></td>
</tr>
</tbody>
</table>
SecureToken parameters encoded into manifest and media responses

The Apple HLS, Adobe HDS, and MPEG-DASH protocols include the SecureToken query parameters in the submanifest and media names returned in request responses.

- These query parameters are the parameters that Wowza Streaming Engine receives with the first manifest request.
- The parameters are Base64-encoded, where the `_' character is replaced by the `!' character.
- The parameters are identified in the submanifest or media name in the response by the `_tk` token followed by the modified Base64-encoded string.

**Apple HLS example**

**Request:**
http://192.168.1.1:1935/vod/sample.mp4/playlist.m3u8?
myTokenPrefixstarttime=1395230400&myTokenPrefixendtime=1500000000&myTokenPrefixCustomParameter=abcdef&myTokenPrefixhash=o4ICgenL8-KU2uktN6XrN8kCF1Xh17k65PorS2odRk=

**Submanifest name in response:**
chunklist_w1955037621_1tkd293emF0b2t1bnN0XJ0dG1t2T0xMzk1MjMwNDAwJndvd3phd09r2W51bmR0aW1lPTE1MDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
securitySecureTokenVersion property is missing, using SecureToken Version 1, play security enabled for RTMP only

INFO server comment - ModuleCoreSecurity.onAppStart[security-vod/_definst_]: Play: using SecureToken Version 1, play security enabled for RTMP only

Troubleshooting

Additional logging
You can turn on the **securityDebugLogRejections** and **securityDebugLogDetails** application properties to log additional information, which can help to identify configuration problems. To add these properties in Wowza Streaming Engine Manager, you must set them as custom properties in the application. See [Configure properties](#) for details.

The following are the values that must be used in the **Add Custom Property** dialog box for each custom logging property:

**securityDebugLogRejections**

<table>
<thead>
<tr>
<th>Path</th>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root/Application</td>
<td>securityDebugLogRejections</td>
<td>Boolean</td>
<td>true</td>
</tr>
</tbody>
</table>

**securityDebugLogDetails**

<table>
<thead>
<tr>
<th>Path</th>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root/Application</td>
<td>securityDebugLogDetails</td>
<td>Boolean</td>
<td>true</td>
</tr>
</tbody>
</table>

Session rejected due to non-matching hash values

The following is logged if an HTTP session is rejected because the server and client hashes don’t match.

**Note:** The **securityDebugLogDetails** property must be enabled.

INFO application app-start _definst_ security-vod/_definst_
INFO server comment - ModuleCoreSecurity.onStreamCreate[security-vod/_definst_]: Client is null. No Secure Token check.
INFO cupertino connect 1549778894 -
INFO stream create sample.mp4 -
INFO server comment - ModuleCoreSecurity.onHTTPSessionCreate[security-vod/_definst_/sample.mp4]: All security checks passed.
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:current time stamp: 1403636858
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:sessionId: 1549778894
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:hashReceived: 1Rw3Kgn0-peLuu39ygEOOo1jg_xWRJ1U3a2emyLBU=
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:creating httpToken for SessionId:1549778894 uri:/security-vod/sample.mp4/playlist.m3u8?wowzatokentime=1395230400&wowzatokentimeend=1500000000&wowzatokenhsh=1Rw3Kgn0-peLuu39ygEOOo1jg_xWRJ1U3a2emyLBU=
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:client IP: 10.0.2.31
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:hashCalculated: IF0Ai8BCqLPf-fEpIbJ5HT6xUDw6a601WRJ1U3a2emyLBU=
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:string hashed: security-vod/sample.mp4?sharedsecret&wowzatokenhsh=1Rw3Kgn0-peLuu39ygEOOo1jg_xWRJ1U3a2emyLBU=
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:token start time stamp: 1395230400
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:token end time stamp: 1500000000
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:SecureTokenDef:Hash:1Rw3Kgn0-peLuu39ygEOOo1jg_xWRJ1U3a2emyLBU=, doesn't match hash calculated, IF0Ai8BCqLPf-fEpIbJ5HT6xUDw6a601WRJ1U3a2emyLBU=
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:string hashed: security-vod/sample.mp4?sharedsecret&wowzatokentime=1500000000&wowzatokentimeend=1500000000&wowzatokenhsh=1Rw3Kgn0-peLuu39ygEOOo1jg_xWRJ1U3a2emyLBU=
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:token start time stamp: 1395230400
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:token end time stamp: 1500000000
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:SecureTokenDef:Hash:1Rw3Kgn0-peLuu39ygEOOo1jg_xWRJ1U3a2emyLBU=, doesn't match hash calculated, IF0Ai8BCqLPf-fEpIbJ5HT6xUDw6a601WRJ1U3a2emyLBU=
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:string hashed: 1FGT5uuUvQPRB-NIaxw4110pAtjQbrezyMG1X03aR=
INFO server comment - [security-vod/_definst_]ModuleCoreSecurity:string hashed: security-
The first log entry highlighted in red is the string used by the server to generate its hash. This string is in the required alphabetical order and must be identical to the string used to generate the client-side hash that was sent to the server.

The second log entry highlighted in red is the server hash and the received hash. The received hash is the exact string received via the query parameters.