Splunk® Enterprise Securing Splunk Enterprise 8.0.1

Configure authentication extensions for SAML tokens

Generated: 12/29/2019 7:59 pm
Configure authentication extensions for SAML tokens

When you configure a Splunk platform instance to use a single sign-on scheme that uses the Security Assertion Markup Language (SAML), depending on the identity provider (IdP) that the scheme connects to, you might have to create or configure authentication extensions, or scripts, that interface with the IdP to process user logins and fetch user information.

In general, if the IdP that you use for authentication with SAML does not support Attribute Query Requests (AQRs), then you can configure authentication extensions to ensure that login and user information retrieval works successfully. When you configure authentication extensions, you must provide details about how your IdP performs these actions in the scripts, then configure the authentication system to use the scripts in the authentication.conf configuration file.

- If the SAML IdP that you use supports AQRs, this content does not apply to you. You can configure tokens as you would with any other type of authentication scheme. Proceed to Create authentication tokens for instructions.
- To learn how to configure authentication extensions, see "Configure authentication extensions" later in this topic.
- To learn more about how authentication extensions work, continue reading.

How authentication extensions work with SAML IdPs

When you connect to a SAML IdP that does not support AQRs, there is no way for the auth system to know how to process login and get information about a user natively, as could be done with AQRs.

Splunk has implemented a system where you can configure an authentication extension to perform the login processing and user retrieval functions as a way to interface with an IdP that cannot provide that information natively. In this case, you provide the extension, which is a Python script, configure the Splunk platform to use the extension, supply the required arguments to the extension, and potentially make changes to the extension so that it works with your IdP.

The authentication extensions support two separate script functions:
getUserInfo

This function lets the Splunk platform retrieve user information from the IdP as needed, and is a drop-in replacement for AQRs. You can configure an authentication extension to use this script function to gather user information after a successful login as if it did support AQRs.

This function lets the Splunk platform perform post-processing on information that the IdP returns after a successful SAML login. You can configure this script function in cases where IdPs return information that needs post-processing. An example of such a case is Microsoft Azure, which returns a link to a list of groups when a user that is a member of more than 150 groups logs in. Without this extension, the SAML response to the login request appears to be empty.

As part of configuring the scripts, you can supply the following arguments and process the following returned objects:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Arguments</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>getUserInfo</td>
<td>--username=&lt;username&gt;</td>
<td>--status=success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--userInfo=&lt;username&gt;;&lt;realname&gt;;&lt;roles&gt;</td>
</tr>
</tbody>
</table>

**Notes:**

- `userInfo` must specify a semicolon-delimited list.
- `<username>` is required.
- `<realname>` is optional, but its semicolon is required.
- `<roles>` is required. To return multiple roles, use colons to separate the roles.

For example: `admin:power`

This example returns just the roles for a user named "testUser":

- `--status=success`  
  `--userInfo=testUser;Test;admin:power`

login

--userInfo=<username>;<realname>; --status=success|fail
**Note:** `userInfo` contains role information that the IdP returns as a SAML assertion.

Splunk includes sample authentication extensions that work with various well-known IdPs that do not support AQRs, such as Azure and Okta. You can make changes to these extensions so that they work with your specific IdP.

You do not need to implement both extensions in your environment for successful SAML logins. You can configure the extensions that best address your specific authentication use case.

**Technical information on the authentication extensions**

There are two scripts that comprise authentication extensions, both of which exist in the `$SPLUNK_HOME/share/splunk/authScriptSamples` directory:

- `commonAuth.py`: This script is the base script that retrieves specific arguments from the auth system, as provided by the second, IdP-specific script. It comes with the Splunk Enterprise installation package.
- `IdP-specific script`: This script takes IdP-specific arguments, such as API key, base URL for accessing the IdP, and so on, and initiates contact with the IdP for the purposes of establishing a login session or retrieving user information. The Splunk Enterprise installation package includes several example scripts with support for the Azure and Okta IdPs. You can modify and use these scripts or write a custom script that works with your specific IdP. In any case, you must always include the `commonAuth.py` script for authentication to work.

These scripts exist in the `$SPLUNK_HOME/etc/auth/scripts` directory, and changes to `authentication.conf` activate the scripts when the Splunk platform authentication system initiates contact with the SAML IdP to log in or retrieve user information.

Several settings in `authentication.conf` control how this process works:

<table>
<thead>
<tr>
<th>Setting</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>scriptPath</td>
<td>Performs the connection to the IdP during the login or user retrieval process.</td>
</tr>
<tr>
<td>Setting</td>
<td>What it does</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>scriptFunctions</code></td>
<td>Provides the functions that the extension should use when connecting to the IdP to log in and retrieve user information.</td>
</tr>
<tr>
<td><code>getuserInfoTTL</code></td>
<td>Determines how long the Splunk platform retains user information that it has retrieved from the IdP.</td>
</tr>
<tr>
<td><code>scriptTimeout</code></td>
<td>Determines how long the Splunk platform lets the extension run while logging in or retrieving user information from the IdP.</td>
</tr>
<tr>
<td><code>scriptSecureArguments</code></td>
<td>Provides the credentials, tokens, or other items that the extension needs to initiate a connection to the IdP for login or user information retrieval.</td>
</tr>
</tbody>
</table>

For more information on these settings, see authentication.conf in the Admin Manual.

Upon a login attempt on a Splunk platform instance that is configured for SAML, the auth system attempts to log in to the IdP with the authentication extension, using the script functions that you provide. Upon a successful connection to the IdP, it then retrieves user information from the IdP to verify that the user is valid and complete the login process.

**Configure authentication extensions**

Perform this procedure after you configure the authentication system to use SAML as an authentication scheme. You must edit configuration files to make this change, as the ability to enable extensions is not available in Splunk Web.

1. Open a shell prompt.
2. Change to the $SPLUNK_HOME/etc/auth/scripts directory. You might need to create this directory if it does not exist.
3. Copy the commonAuth.py script from the
   $SPLUNK_HOME/share/splunk/authScriptSamples directory to the
   $SPLUNK_HOME/etc/auth/scripts directory.
4. Depending on the IdP you use for SAML, do one of the following:
   1. If your IdP is Azure or Okta, copy the sample Azure or Okta script from the $SPLUNK_HOME/share/splunk/authScriptSamples directory to the $SPLUNK_HOME/etc/auth/scripts directory.
   2. If your IdP is neither of these, you can either copy one of the above scripts and modify it or write your own script.
5. Use a text editor to modify the IdP-specific script to include specific information about your IdP. The information you provide here depends specifically on the IdP that you use for SAML.

6. Change to the `$SPLUNK_HOME/etc/system/local` directory.

7. Using a text editor, create or edit `authentication.conf`. At a minimum, add the following entries to the file:

   ```plaintext
   [samlsettings-<yourkey>]
   
   scriptPath = <name of script>
   scriptFunctions = login;getUserInfo
   scriptSecureArguments = <credential, key, etc.>
   ```

8. Restart the Splunk platform.

9. Log out of the Splunk platform.

10. Log into the Splunk platform as a known SAML user and confirm that login is successful.