Administrator's Guide for Helix Authentication Extension

Overview

Helix Authentication Service support for Helix Core server and Helix Core clients, such as P4V, requires a Helix Core Server Extension.

This extension, like the the Node.js authentication service, runs on Linux systems with Security-Enhanced Linux (SELinux) enabled and in **enforcing** mode.

If you chose to enable Security-Enhanced Linux (SELinux), the extension runs in **enforcing** mode.

For information about Helix Core Server Extensions, see Helix Core Extensions Developer Guide.

Prerequisites

- This document assumes that you have read "Administrator's Guide for Helix Authentication Service", which is available at https://github.com/perforce/helix-authentication-service.
- Helix Core Server, version 2019.1 or later.

Support

The configuration of the Helix Authentication Service to work with both the Identity Provider (IdP) and the Perforce server product requires an experienced security administrator.
This effort might require assistance from Perforce Support.

Certificates

Included with the authentication service files are sample self-signed certificates. For production systems, these files should be replaced with certificates signed by a valid certificate authority (CA).

The Helix Server extension reads the CA certificate from `ca.crt` and the client certificates from `client.crt` and `client.key` within the directory named by the `arch-dir` attribute of the installed extension.

To update these files, replace them with new content, keeping the names the same. To find the `arch-dir` directory, use the `p4 extension` command:

```
$ p4 extension --list --type=extensions
... extension Auth::loginhook
... [snip]
... arch-dir server.extensions.dir/117E9283-732B-45A6-9993-AE64C354F1C5/1-arch
... data-dir server.extensions.dir/117E9283-732B-45A6-9993-AE64C354F1C5/1-data
```

where `[snip]` means some information has been omitted.

Build

If you already have the `loginhook.p4-extension` file, go to the **Install** section.

If you want to build the loginhook extension from the source code of the authentication server, open a terminal window and issue the following command:

```
$ p4 extension --package loginhook
```

The result will be a zip file named `loginhook.p4-extension`

Install `loginhook.p4-extension`

To install the extension, run the following command in a terminal window:

```
$ p4 extension --install loginhook.p4-extension -y
Extension 'Auth::loginhook#1.0' installed successfully.
$ p4 admin restart
```

The `restart` is necessary `because p4d` prepares the authentication mechanisms during startup. This is true when adding or removing `auth-` related triggers, as well as when installing or removing the loginhook extension.

If this is not the first time you are installing the extension, remove the existing extension before reinstalling it. See "Removing the Extension".

Configure

Configure the Extension at both the "global" and "instance" level. To learn about these levels, see the "Server extension configuration (global and instance specs)" topic in Helix Core Extensions Developer Guide .

Global

Start by getting the global configuration of the extension:

```
$ p4 extension --configure Auth::loginhook
[snip]

ExtP4USER:     sampleExtensionsUser

ExtConfig:
    Auth-Protocol:
        ... Authentication protocol, saml or oidc.
    Service-URL:
        ... The authentication service base URL.
```

where `[snip]` means some information has been omitted.

The first field to change is `ExtP4USER` which should be the Perforce user that will own this extension, typically a "super" or administrative user.

The `Service-URL` field must be changed to the address of the authentication service by which the Helix Server can make a connection.

The `Auth-Protocol` can be any value supported by the authentication service. This determines the authentication protocol for SSO users to authenticate. This setting is optional because the authentication service will use its own settings to determine the protocol.

Instance

To configure a single "instance" of the extension, include the `--name` option along with the `--configure` option. This example uses `loginhook-all` to be descriptive.

```
p4 extension --configure Auth::loginhook --name loginhook-all -o
[snip]
ExtConfig:
    enable-logging:
        ... Extension will write debug messages to a log if 'true'.
    name-identifier:
        ... Field within IdP response containing unique user identifer.
    non-sso-groups:
        ... Those groups whose members will not be using SSO.
    non-sso-users:
        ... Those users who will not be using SSO.
    user-identifier:
        ... Trigger variable used as unique user identifier.
```

where `[snip]` means some information has been omitted.

All of these settings have sensible defaults. However, for the extension to be enabled, we must configure it. You might want to change either the `non-sso-groups` or `non-sso-users` fields to a list of Perforce groups and users that are not participating in the SSO authentication integration.

Name	Description	Default

`enable-logging`	Extension will write debug messages to a log if `true`	`false`
`non-sso-users`	Those users who will not be using SSO.	*none*
`non-sso-groups`	Those groups whose members will not be using SSO.	*none*
`user-identifier`	Trigger variable used as unique user identifier, one of: `fullname`, `email`, or `user`.	`email`
`name-identifier`	Field within identity provider user profile containing unique user identifer.	`email`

Debug logging

When enabled, the extension writes debugging logs using the `Perforce.log()` API. The JSON formatted file will appear in the directory identified by the `data-dir` extension attribute. You can find the value for `data-dir` by searching the installed extensions using `p4 extension` as a privileged user.

```
$ p4 extension --list --type=extensions
... extension Auth::loginhook
... [snip]
... data-dir server.extensions.dir/117E9283-732B-45A6-9993-AE64C354F1C5/1-data
```

where `[snip]` means some information has been omitted.

Mapping User Profiles to Perforce Users

Helix user specs have several fields that can be used for matching with the profile information returned from the identity provider. The extension uses the trigger variables exposed by the server, namely `fullname`, `user`, and `email`, and the choice is configured in the extension by setting the `user-identifier` value (default is `email`) in the *instance* configuration.

On the other side of the mapping is the user profile returned by the identity provider. Different protocols and providers return different fields, and there is no one field that works for all. Also, administrators are often free to adjust the output to suit their needs. As such, the extension has another *instance* configuration setting named `name-identifier`, which specifies the name of the field in the user profile that is to be used in matching with the Helix user. This defaults to `email` because that field is likely to be available and unique on both the IdP and Helix.

Generally, with SAML, the `name-identifier` extension setting should be given the value `nameID` because that field is always present in the user profile returned from the SAML IdP. Depending on the format of the name identifier, you will need to select an appropriate value for the `user-identifier`. If the IdP returns a "user name", and it matches the `User` field in the Perforce user spec, set `user-identifier` to `user` in the extension configuration. If the name identifier is an email address, use `email` instead of `user`. The value of `fullname` might also be appropriate.

For OIDC, the user profile often includes an `email` field. The server extension looks for this by default because `name-identifier` defaults to `email`. Hopefully this value matches the `Email` field of the Perforce user spec because the server extension uses `email` for the `user-identifier` by default.

If you are unsure of the contents of the user profile returned from the identity provider, enable the debug logging in either the authentication service or the server extension, and then examine the logs after attempting a login. With the server extension, set the `enable-logging` *instance* configuration setting to `true`, attempt a login, and look for the `log.json` file under the `server.extensions.dir` directory of the Helix depot. For the authentication service, set the `DEBUG` environment variable to `auth:*`, restart the service, attempt the login, and look at the output from the service (either in the console or in a pm2 log file, if you are using pm2).

(Optional) Allowing for non-SSO Users

The process for enabling non-SSO users consists of three steps:

1. Enable database passwords in addition to supporting SSO
2. Set passwords for all users
3. Assign users to the non-SSO group

Enable Database Passwords

To allow for database passwords, and to allow for the super user to set the passwords of users, configure the server using the `p4 configure` command:

```
$ p4 configure set auth.sso.allow.passwd=1
```

Set User Passwords

When the server is running at security level 3, all users must have a password, including SSO users, even though they do not need to know their password. We recommend setting this password to a random value:

```
$ yes $(uuidgen) | p4 -u super passwd username
```

Assigning non-SSO Users

In the server extension, indicate which users and/or groups are *excluded from* SSO authentication. See the `non-sso-groups` and `non-sso-users` settings described above.

Removing the Extension

To remove the login extension, perform the following three steps:

Step 1: Find the installed extension by using the `--list` option.

```
$ p4 extension --list --type=extensions
... extension Auth::loginhook
... rev 1
... developer Perforce
... description-snippet SSO auth integration
... UUID 117E9283-732B-45A6-9993-AE64C354F1C5
... version 1.0
... enabled true
... arch-dir server.extensions.dir/117E9283-732B-45A6-9993-AE64C354F1C5/1-arch
... data-dir server.extensions.dir/117E9283-732B-45A6-9993-AE64C354F1C5/1-data
```

Step 2: Remove that extension by using the `--delete` option as an administrative user.

```
$ p4 extension --delete --yes Auth::loginhook
Extension 'Auth::loginhook and its configurations' successfully deleted.
```

Step 2: Restart the server

```
$ p4 admin restart
```

Without the restart, the server will report an error about a missing hook:

```
Command unavailable: external authentication 'auth-check-sso' trigger not found.
```