Apache HTTPD on RHEL/CentOS/SL 7

The Apache HTTP server is the Apache Software Foundation's web server. See the project's website for more details.

Contents

- 1. System Preparation
  - 1.1. Turn off SELinux
    - 1.1.1. Temporarily
    - 1.1.2. Permanently
  - 1.2. Add the Moonshot libraries
- 2. Installation Instructions
  - 2.1. Using the standard RedHat mod_auth_gssapi module
  - 2.2. Using the Moonshot mod_auth_gssapi module
- 3. Configuration Instructions
  - 3.1. Protecting a location with Moonshot
    - 3.1.1. Using the standard RedHat mod_auth_gssapi module
    - 3.1.2. Using the Moonshot mod_auth_gssapi module
  - 3.2. Populating REMOTE_USER
    - 3.2.1. Using the RedHat mod_auth_gssapi module
    - 3.2.2. Using the Moonshot mod_auth_gssapi module
  - 3.3. Accessing Moonshot attributes
    - 3.3.1. Using the RedHat mod_auth_gssapi module
    - 3.3.2. Using the Moonshot mod_auth_gssapi module
- 4. Testing Moonshot
- 5. HTTPS Internet Explorer compatibility

Apache Moonshot module information

These instructions relate to both the Apache mod_auth_gssapi module package shipped in the base distribution ("the RedHat module") and the older Moonshot version ("the Moonshot module") thereof.

All of the instructions below assume that you have root access, and will work as the root user (either directly or using sudo).

1. System Preparation

1.1. Turn off SELinux

Currently, Moonshot will not work while SELinux is in enforcing mode. Until we resolve this, simply turn SELinux to permissive mode. This can be done temporarily (i.e., on reboot it will be turned back on), or permanently (the change will persist).

1.1.1. Temporarily

The following command will turn Enforcing mode off:

```
$ echo 0 > /selinux/enforce
```

1.1.2. Permanently

Edit `/etc/sysconfig/selinux` and change "SELINUX=enforcing" to "SELINUX=permissive". Reboot the system.

1.2. Add the Moonshot libraries

If you have not already done so, you first need to follow the instructions on how to install the Moonshot Libraries on RHEL/CentOS/SL 6.

2. Installation Instructions

2.1. Using the standard RedHat mod_auth_gssapi module

1. To use the Apache module, install it from the base repository:

```
$ yum install mod_auth_gssapi
```

2. Ensure that the certificates referenced in `/etc/radsec.conf` can be read by the Apache user:

```
$ su - --shell=/bin/bash apache
$ cat path_to_ca.pem
$ cat path_to_client.pem
$ cat path_to_client.key
```

3. If they cannot be read, add the Apache user to the group that has read access to the certificates:

```
$ usermod -a -G <group> apache
```

4. Verify that the `KeepAlive` option is enabled in the Apache configuration file `/etc/httpd/conf/httpd.conf`:

```
KeepAlive On
```

5. Restart Apache:

```
$ service httpd restart
```

2.2. Using the Moonshot mod_auth_gssapi module

1. To use the Apache module, install it and the Kerberos workstation package from the base repository:

```
$ yum --disablerepo=base install mod_auth_gssapi
$ yum install krb5-workstation
```

2. Add a dummy Kerberos key to make the module happy:

```
$ ktutil
ktutil:  addent -password -p HTTP/localhost@YOUR-WEBSERVER-HOSTNAME -k 1 -e aes256-cts
<enter any password>
ktutil:  wkt /etc/httpd/krb5.keytab
ktutil:  quit
```

3. Export the location of the keytab file into Apache's config:

```
$ echo export KRB5_KTNAME=/etc/httpd/krb5.keytab >> /etc/httpd/envvars
```

> **Alternative**
>
> Alternatively, you can use the `GSSKrb5Keytab` configuration directive in the `Location` directive in Section 3.1 to specify the keytab.

4. Assign the correct permissions to the keytab file:

```
$ chown apache.apache /etc/httpd/krb5.keytab
```

5. Ensure that the certificates referenced in `/etc/radsec.conf` can be read by the Apache user:

```
$ su - --shell=/bin/bash apache
$ cat path_to_ca.pem
$ cat path_to_client.pem
$ cat path_to_client.key
```

6. If they cannot be read, add the Apache user to the group that has read access to the certificates:

```
$ usermod -a -G <group> apache
```

7. Verify that the `KeepAlive` option is enabled in the Apache configuration file `/etc/httpd/conf/httpd.conf`:

```
KeepAlive On
```

8. Restart Apache:

```
$ service httpd restart
```

3. Configuration Instructions

Shibboleth2 Apache module incompatibility

Please read Section 6.2 in Apache HTTPD on module incompatibilities.

3.1. Protecting a location with Moonshot

3.1.1. Using the standard RedHat mod_auth_gssapi module

To protect a particular location on your Apache server, you must configure it with an AuthType of "GSSAPI".

Here's a sample configuration that can get you started.

Example

To allow anyone with a valid Moonshot account to access `/wherever`, you would do the following:

```
<Location "/wherever">
    AuthType GSSAPI
    AuthName "Moonshot Login"
    Require valid-user
    AddHandler cgi-script .cgi
    Options +ExecCGI
    GssapiConnectionBound On
    GssapiNameAttributes json
</Location>
```

Configuration Directives

For more information on the configuration directives supported by the RedHat mod_auth_gssapi module, see its homepage at https://github.com/modauthgssapi/mod_auth_gssapi

3.1.2. Using the Moonshot mod_auth_gssapi module

To protect a particular location on your Apache server, you must configure it with an AuthType of "Negotiate".

The module-shipped `/etc/httpd/conf.d/auth_gssapi.conf` file contains a sample configuration that can get you started.

Example

To allow anyone with a valid Moonshot account to access `/wherever`, you would do the following:

```
<Location "/wherever">
    AuthType Negotiate
    AddHandler cgi-script .cgi
    Options +ExecCGI
    Require valid-user
</Location>
```

Compatibility

Additionally, in an effort to provide cross-compatibility, the Moonshot mod_auth_gssapi module broadly supports the configuration directives that the RedHat mod_auth_gssapi module supports.

For more information on the configuration directives supported by the RedHat mod_auth_gssapi module, see its homepage at https://github.com/modauthgssapi/mod_auth_gssapi

3.2. Populating REMOTE_USER

Web services often rely on the `REMOTE_USER` Apache environment variable for user information, such as a local user account or a pseudonymous identifier.

3.2.1. Using the RedHat mod_auth_gssapi module

To populate `REMOTE_USER`, enable the `GssapiImpersonate` configuration directive:

```
<Location "/wherever">
    :
    GssapiImpersonate On
    :
</Location>
```

3.2.2. Using the Moonshot mod_auth_gssapi module

To populate `REMOTE_USER`, update the FreeRADIUS reply from the RP Proxy with the `User-Name` RADIUS attribute in the RP Proxy's post-auth section:

```
update reply {
        User-Name := "content"
}
```

3.3. Accessing Moonshot attributes

3.3.1. Using the RedHat mod_auth_gssapi module

The RedHat module has the ability to access all the attributes in the GSSAPI response, including the raw RADIUS attributes and any attributes that were transformed by the Shibboleth attribute map in the Moonshot mechanism. To access these attributes, use the `GssapiNameAttributes` directive:

```
<Location "/wherever">
    :
    GssapiNameAttributes <environment variablename> <GSSAPI attribute name>
    :
</Location>
```

Example accessing the User-Name attribute

This example accesses the RADIUS `User-Name` attribute and stores it in the `RADIUS_USER_NAME` environment variable where a script can read it.

```
<Location "/wherever">
    :
    :
    GssapiNameAttributes RADIUS_USER_NAME urn:ietf:params:gss:radius-attribute 1
</Location>
```

3.3.2. Using the Moonshot mod_auth_gssapi module

The Moonshot module can use either the Shibboleth attribute resolver library to map RADIUS and SAML attributes to internal Shibboleth attributes, and then to environment variables, or use its own internal JSON attribute resolver to map either RADIUS attributes or SAML attributes to environment variables. Read more at Configure a Linux Server's Attribute Resolution about how to configure Shibboleth or the internal JSON attribute resolvers.

We are working on enhancements that allow the Moonshot module to expose attributes in the same way as the RedHat module.

4. Testing Moonshot

Either use a browser to access your protected location, or use cURL to retrieve it:

```
curl --negotiate -u ":" http://your-url/wherever
```

5. HTTPS Internet Explorer compatibility

For updated best practice with Internet Explorer connections, you should also read Microsoft's HTTPS and Keep-Alive Connections article.