Apache HTTPD

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

Contents

1. Overview

Apache HTTPD is generally compatible with Moonshot through the use of an Apache GSS-API module.

2. Compatibility

2.1. Key

In the tables below, the following icons have the following meanings:

- ✅ - This version of the software has been tested and verified as supporting Moonshot.
- ❌ - This version of the software has been tested and verified as **not** supporting Moonshot.
- ❓ - This version of the software has not yet been tested thoroughly and its status is not known. Let us know if you have tried it and whether it worked or not!

2.2. Compatibility List

Note that accessing supported versions of this software requires a Moonshot compatible client - see the next section for details on which clients are supported.

⚠ Any versions not listed below have not yet been tested. If you do so, please let us know!

Version	Compatible?	Notes
Apache 2.4	✅	Using both the Moonshot and the RedHat GSSAPI mod_auth_gssapi modules
Apache 2.2	✅	Using the Moonshot mod_auth_gssapi module
Apache 2.0	❓	
Apache 1.3	❓	

3. Installation & Configuration

How you set up a Moonshot-enabled version of the Apache HTTP server will differ depending on your OS. See the relevant pages for your particular distribution:

- CentOS 6
- Debian 8

- RHEL 6
- Scientific Linux 6

Instructions for building the module manually for RHEL-based platforms can be found here.

4. Client Compatibility

The following clients are known to work with this server software using Moonshot authentication (click on the link to see further information about enabling Moonshot in that client):

- Microsoft Internet Explorer
- Mozilla Firefox
- Debian's Iceweasel
- Google Chrome
- This small python script
- curl

5. Next Steps

Once you have installed the software, what happens next?

5.1. Account Mapping

To Come

5.2. Testing

The simplest way to test the Apache Moonshot integration is to create a simple script, protected by Moonshot. To do this, do the following:

1. Create a simple script to protect by creating a directory for cgi scripts.

```
$ mkdir /var/www/moonshot
```

2. Create a script in that directory called "hello.cgi" with the following content:

```
#!/bin/sh
# disable filename globbing
set -f
echo Content-type: text/plain
echo
echo "Hello $REMOTE_USER, you have successfully authenticated."
echo "If an identifier is not displayed after Hello, but you can see this page, then your IdP is not
releasing the AAA User-Name attribute to this service and you are anonymous."
echo
echo "Some information about your server:"
echo SERVER_SOFTWARE = $SERVER_SOFTWARE
echo SERVER_NAME = $SERVER_NAME
echo GATEWAY_INTERFACE = $GATEWAY_INTERFACE
echo SERVER_PROTOCOL = $SERVER_PROTOCOL
echo SERVER_PORT = $SERVER_PORT
echo REQUEST_METHOD = $REQUEST_METHOD
echo HTTP_ACCEPT = "$HTTP_ACCEPT"
echo PATH_INFO = "$PATH_INFO"
echo PATH_TRANSLATED = "$PATH_TRANSLATED"
echo SCRIPT_NAME = "$SCRIPT_NAME"
echo QUERY_STRING = "$QUERY_STRING"
echo REMOTE_HOST = $REMOTE_HOST
echo REMOTE_ADDR = $REMOTE_ADDR
echo REMOTE_USER = $REMOTE_USER
echo AUTH_TYPE = $AUTH_TYPE
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH
```

3. Set the permissions appropriately:

```
$ chown -R www-data /var/www/moonshot && chmod 755 /var/www/moonshot/hello.cgi
```

4. Configure Apache to protect this location, by creating a file at `/etc/apache2/conf.d/moonshot.conf` with the following content:

```
<Directory "/var/www/moonshot/">
    AuthType Negotiate
    Require valid-user
    AddHandler cgi-script .cgi
    Options +ExecCGI
</Directory>
```

5. Restart Apache2

```
$ service apache2 restart
```

6. Test by browsing to http://$yourservername/moonshot/hello.cgi with a compatible browser.

6. Troubleshooting

6.1. RP Proxy configuration

6.1.1. 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.

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

```
if (&request:GSS-Acceptor-Service-Name != 'trustidentity') {
    update reply {
        User-Name := "content"
    }
}
```

6.1.2. Setting the GSS-Acceptor-Host-Name and GSS-Acceptor-Service-Name attributes

You may find during testing that you get failures when attempting to authenticate. Run the RP Proxy in debug mode and check the incoming access requests from your webserver. When an access request looks similar to the below, you will notice that the `GSS-Acceptor-Host-Name` and/or `GSS-Acceptor-Service-Name` attributes are not set correctly, or are missing altogether, and you must set them manually.

```
(0) Sent Access-Request Id 1 from 192.168.1.15:48875 to 123.234.232.210:2083 length 128
(0)   User-Name = '@camford.ac.uk'
(0)   EAP-Message = 0x02...
(0)   Message-Authenticator = 0x...
(0)   Trust-Router-COI := 'ov-apc.moonshot.ja.net'
(0)   GSS-Acceptor-Realm-Name := 'moonshotrealm.camford.ac.uk'
(0)   Event-Timestamp = 'Jan 20 2017 10:20:28 GMT'
(0)   NAS-IP-Address = 192.168.1.15
(0)   Proxy-State = 0x30
```

To set these two vital attributes, you must do the following in your RP Proxy configuration:

1. Open the `sites-available/abfab-tls` file in the FreeRADIUS configuration directory (on RHEL platforms, it's `/etc/raddb`, on Debian platforms it's `/etc/freeradius`).
2. Locate the line '`clients radsec-abfab`'.
3. Duplicate the '`client default`' block below it and modify the duplicate to suit the requirements as below, then save the file:

```
    client moonshot-webserver-name {
        ipaddr = ipaddress of the webserver
        proto = tls
        gss_acceptor_host_name = "the actual hostname of the webserver, i.e. webserver.camford.ac.uk"
        gss_acceptor_service_name = "HTTP"
        gss_acceptor_realm_name        = "the same realm name as in the 'client default' block"
        trust_router_coi = "the same realm name as in the 'client default' block"
    }
```


You can name the `client` in the block above anything from just a simple entry to its full hostname. To make things easier, you may use the `shortname` configuration option in the list of options to set a short name that you can use elsewhere in your FreeRADIUS configuration.

The `ipaddr` configuration option also accepts CIDR-formatted IP address blocks if you have multiple servers you want to connect as the same host.

4. Open the `sites-available/abfab-tr-idp` file in the FreeRADIUS configuration directory (on RHEL platforms, it's `/etc/raddb`, on Debian platforms it's `/etc/freeradius`).
5. Locate the line '`psk_authorize`'. Insert the following block before that line, then save the file:

```
if ("%{client:gss_acceptor_host_name}") {
    update request {
        GSS-Acceptor-Host-Name = "%{client:gss_acceptor_host_name}"
    }
}
if ("%{client:gss_acceptor_service_name}") {
    update request {
        GSS-Acceptor-Service-Name = "%{client:gss_acceptor_service_name}"
    }
}
```


This issue should be rectified and no longer necessary from FreeRADIUS 3.0.13 onwards.

6. Restart your RP Proxy, and re-test. Your entry should now look something more like this:

```
(0) Sent Access-Request Id 10 from 192.168.1.15:47941 to 123.234.232.210:2083 length 159
(0)   User-Name = '@camford.ac.uk'
(0)   GSS-Acceptor-Service-Name = 'HTTP'
(0)   GSS-Acceptor-Host-Name = 'webserver.camford.ac.uk'
(0)   EAP-Message = 0x02...
(0)   Message-Authenticator = 0x2a8cf41456b3b2e94473a1eb2849e561
(0)   Trust-Router-COI := 'ov-apc.moonshot.ja.net'
(0)   GSS-Acceptor-Realm-Name := 'moonshotrealm.camford.ac.uk'
(0)   Event-Timestamp = 'Jan 20 2017 10:34:18 GMT'
(0)   NAS-IP-Address = 192.168.1.15
(0)   Proxy-State = 0x30
```

6.2. Incompatibility with other Apache modules

Currently there is a limited problem with incompatibility between the Moonshot mechanism and other Apache modules, such as `mod_shib` or `mod_php`. The reason seems to be that Apache modules are dynamically loaded using the `RTLD_GLOBAL` flag. This then causes duplicated symbol names to be overwritten in memory, resulting in calls to the wrong symbol and thus segmentation faults.

One of the symptoms of this incompatibility is text similar to the below in the Apache error log (`error_log`):

error_log entry

```
[date time] [error] [client 1.2.3.4] Failed to establish authentication: Invalid token was supplied (Unknown
error)
[date time] [notice] child pid 12345 exit signal Segmentation fault (11)
```

Apache developers acknowledge this bug in the documentation of `apr_dso_load` (https://apr.apache.org/docs/apr/1.6/group__apr__dso.html#gaedc8609c2bb76e5c43f2df2281a9d8b6):

> *We aught to provide an alternative to RTLD_GLOBAL, which is the only supported method of loading DSOs today. (sic)*

An alternative (and preferred) way of work-around this issues is to manually patch `libapr` to force the usage of the RTDL_DEEPBIND flag (as suggested by the developers in https://bz.apache.org/bugzilla/show_bug.cgi?id=62083#c1). In this case, you need to ensure that the locally installed library is chosen before the system's one. The APR library can be downloaded from https://apr.apache.org/download.cgi.

6.2.1. mod_shib Shibboleth module

Shibboleth

In order to co-exist with the Shibboleth module, you should use the following option:

```
ShibCompatValidUser On
```

The Shibboleth documentation says that:

```
#
# Turn this on to support "require valid-user" rules from other
# mod_authn_* modules, and use "require shib-session" for anonymous
# session-based authorization in mod_shib.
```

This module is currently not compatible with the Shibboleth2 service provider Apache module. The reason is that `mod_shib` makes use of `libshibsp-lite.so`, whereas Moonshot makes use of `libshibsp.so`. As both libraries provide symbols with the same name but different implementations the system gets confused and fails.

To resolve this error, disable the Shibboleth module.

As alternative, consider the **mod_auth_mellon** SAML2 module. It has been tested with Moonshot and been found to be compatible. More documentation on how to configure **mod_auth_mellon** can be found here: https://jdennis.fedorapeople.org/doc/mellon-install/mellon-install-guide.html.

6.2.2. mod_php5 JSON submodule

This module is not compatible with the PHP5 JSON submodule. The reason is that the module is built with `libjson-c.so` whereas Moonshot is built with `libjansson.so`. Both libraries share symbol names that fail when Apache dynamically loads them.

To resolve this error, disable the PHP5 JSON submodule. In Debian, just delete the `/etc/php5/apache2/conf.d/20-json.ini` file. Note that no JSON support will then be available from PHP.

Consider upgrading your PHP installation to at least version 7.1, where this incompatibility does not exist. On Debian and Ubuntu, the standard repositories will contain PHP 7.2. On RedHat, CentOS or Scientific Linux, follow the instructions for the REMI PHP 7.2 repositories.

6.3. AppArmor and SELinux

Ubuntu's AppArmor and the RedHat SELinux systems are also known to interfere with Apache's loading of the Moonshot module. Follow the appropriate operating system's instructions on how to allow Apache to access files outside its assigned directories.