NativeSPTroubleshootingCommonErrors

The following errors are commonly encountered by users, usually when initially setting up their SP.

opensaml::SecurityPolicyException: Message expired, was issued too long ago.

Barring an actual replay attack, your SP's clock isn't synchronized with the clock of the IdP that issued the message. All servers using SAML **MUST** maintain accurate time. Refer to your OS documentation for information on how to synchronize with a reliable time source.

Message was signed, but signature could not be verified.

The certificate that was used to sign the message didn't match the one the SP expected based on metadata. That can be caused by, in order of likelihood:

1. The certificate in the metadata is different from the one configured in `relying-party.xml`, and hence, the one in the message. You should change them so they match.
2. If PKIX(CN matching with a signed root) is being used, the CN of the certificate used to sign the message is not the same as the CN expected by the KeyName of that provider's metadata.
3. The IdP is using the wrong entityID and mistakenly trying to spoof another IdP.

Unable to establish security of incoming assertion

This error will be presented in the browser for a variety of different underlying reasons. Check `shibd.log` for more useful debugging information.

Unable to locate metadata for identity provider (`https://identities.supervillain.edu/idp/shibboleth`).

This message indicates the SP tried to initiate a session with an entityID it doesn't recognize as belonging to an identity provider. That's probably because the entityID in the SessionInitiator has no corresponding metadata loaded, but it could also happen with a discovery service that knows more than you do.

1. Check that the entityID is correct, and names the identity provider you intend to speak to.
2. Make sure you're loading the metadata of that identity provider.
3. Make sure that the identity provider's metadata contains everything good IdP metadata does.
4. If the problem occurs with a IdP which previously worked with this SP, then the issue may be that the metadata has expired.

HTTP POST form data is lost when Shibboleth session expired or does not exist yet

Sometimes a user submits HTTP POST data to a page that requires a valid Shibboleth session. An example could be a user that completes a web form. If the user does not have yet a valid Shibboleth session or if his session expired, he is redirected to his Identity Provider and forced to re-authenticate. Coming back to the protected page his HTTP POST data is lost due to the redirects. The following steps explain how the SP (2.2 or above required) can be configured to preserve HTTP POST data:

1. In the <Sessions> element of the shibboleth2.xml configuration file add the attributes 'postData="ss:<id>"'
   where <id> is the id value of the <Storage> element. The default will be to add 'postData="ss:mem"'.
2. Add 'postTemplate="postTemplate.html"' that points to the default HTML file to use when resubmitting the HTTP POST request.
3. Save shibboleth2.xml and restart the shibd daemon

Using the above-mentioned settings, the SP will preserve and re-post the HTTP POST data after the user got a valid Shibboleth session. Note that this can have unintended consequences if the user clicks on the back button of his web browser. In this case the HTTP POST data might be resent again. Therefore, the application that processes the POST data should take this into account when accepting the data.
For further information, have a look at the configuration options postData, postTemplate and postExpire on NativeSPSessions.

SAML message delivered with POST to incorrect server URL.

When a SAML message is addressed to a location inconsistent with where the SP believes it's running, this error will be thrown. The SP pulls much of this information from the web environment.

1. Verify that the server name and port are properly set in accordance with the SP's metadata.
2. Rewriting rules in effect for the `Shibboleth.sso` handler path must be consistent with the SP's metadata.
3. The IdP needs to properly address the SAML response.

opensaml::saml2md::MetadataException: Security of SAML 1.x SSO POST response not established.

The usual cause for this is an incoming SAML assertion/response from an issuer for which the SP has no metadata loaded. This means either the metadata is wrong, or the IdP in question is using the wrong `entityID` in its configuration, so the URI passed to the SP doesn't match what it expects.

More specific information is usually available from the `shibd.log` file.

opensaml::FatalProfileException: A valid authentication statement was not found in the incoming message.

Non-specific. You need to check the log for specific information about why the incoming assertion was invalid.

supplied TrustEngine failed to validate SSL/TLS server certificate

The IdP's metadata provides the rules for determining whether a certificate used for a signature or found at a SOAP endpoint is acceptable. This error means it wasn't acceptable. Either the metadata is wrong, or the certificate is, but they don't "match".

Unable to resolve any key decryption keys

The SP received encrypted XML (usually an EncryptedAssertion) and couldn't decrypt it. The SP's metadata probably doesn't contain the same public key (s) the SP is configured to use (or the credentials didn't load).

error:1408F06B:SSL routines:SSL3_GET_RECORD:bad decompression

Appears in `shibd.log` during back-channel communications. Seems to be very unpredictable, but is caused by some kind of OpenSSL bug when zlib is used to compress the SSL traffic. Particularly common on the second (and subsequent) attempts to use a cached SSL session. There doesn't seem to be a real solution to this, but a couple of bad work-arounds include:

- Recompiling OpenSSL on the server to exclude zlib. See these HowTo guides for recompiling OpenSSL on Debian Etch and Red Hat.
- Turning off SSL session reuse at either end. The SP can do this by adding a `<TransportOption>` element to the configuration with a value of 0 for option number 150.

At this time, it is not known if running with Tomcat's SSL implementation is subject to the bug, but running without Apache may be a solution for some sites.

ERROR Shibboleth.AttributeResolver []: exception during SAML query to <url>: CURLSOAPTransport failed while contacting SOAP responder: SSL certificate problem, verify that the CA cert is OK. Details: error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed

Appears in `shibd.log` during back-channel communications. This indicates that one of the peers rejected the certificate of the other.

If the log includes errors mentioning a "TrustEngine" failing to verify the SSL certificate, the error indicates that the SP rejected the IdP's server certificate and the metadata is wrong or invalid in some respect.

Otherwise, this usually indicates that the IdP rejected the certificate the SP presented, but did so using a layer of code inside the Apache mod_ssl module. This happens when Apache is misconfigured by allowing mod_ssl to validate the certificate. You need to make sure the SSLVerifyClient option is set to "optional no_ca" on the IdP.

If you think it is set, you're either mistaken or the certificate violated a built-in requirement that mod_ssl refuses to disable. Sometimes this can be corrected by upgrading to a newer Apache version, but it may not be something you can fix.

You may also consider running the IdP without Apache, which is now supported and documented. Note that you can do this for back-channel communication on port 8443 while still running Apache in front of the container for front-channel support on port 443, if your authentication solution requires that Apache be used.

Can't connect to listener process

Appears in the browser when the server module can't communicate with `shibd` for some reason. The "duh" solution is to check whether it's running, but on Red Hat, another common cause is SELinux being enabled. That prevents socket communication between Apache and `shibd`, but doesn't really provide much feedback about it.

Feedback about the SELinux issue here can be gleaned by running SELinux in `permissive` mode - the default is `enforcing`.

What you will find, from running `audit2allow` for an initial `httpd+shibd` setup, is that you simply need a policy that permits the Apache server (`httpd`) to access `/var/run/shibboleth/shibd.sock`

Such a module can be created, once you have logged the problem by running SELinux in `permissive` mode, by running the command:

```
grep httpd_t /var/log/audit/audit.log | audit2allow -M <name_of_module>
```

after which you will have generated a `<name_of_module>.te` and `<name_of_module>.pp` file, the latter being the compiled module, along with instructions on how to load the module into the SELinux regime.

Simply giving the httpd server access to `/var/run/shibboleth/shibd.sock` should merely require a `.te` file that looks like this:

```
module name_of_module 1.0;
require {
        type var_run_t;
        type httpd_t;
        type initrc_t;
        class sock_file write;
        class unix_stream_socket connectto;
}
#============= httpd_t ==============
allow httpd_t initrc_t:unix_stream_socket connectto;
allow httpd_t var_run_t:sock_file write;
```

Your `.te` file may obviously contain extra defintions and rules depending upon what your local `httpd` is trying to access, however, if the above info is in there, then that should cure the `Can't connect to listener process` issue, even when restarting SELinux in enforcing mode once again.

Compile and install the policy file with:

```
checkmodule -m -M -o mod_shib-to-shibd.mod mod_shib-to-shibd.te
semodule_package -o mod_shib-to-shibd.pp -m mod_shib-to-shibd.mod
semodule -i mod_shib-to-shibd.pp
```