IdPStatelessClustering

Ordinarily the IdP is stateful in that it stores runtime information in memory to process requests, and must replicate that information across nodes in a cluster. A quick search of the support list archives will demonstrate the complexity of the clustering technology used to replicate this state, particularly for high-volume deployments.

An alternative to this approach is to forgo some features (including one that isn't even supported yet) in return for decreased complexity and increased reliablity by reducing reliance on server-side IdP state.

Use of a stateless IdP login handler does not make a deployment completely stateless. The login context is still persisted using server-side state which must be present through the process beginning with the receipt of an AuthnRequest and ending with the issuance of a response. There must be either short session stickiness to associate a client with a single node for however long the login process may take(~5 minutes), or replication of the IdP session object itself using a stateful clustering mechanism.

Feature Limitations

Eliminating server-side state (apart from caching) means eliminating any dependencies on the IdP-managed client session and on any mappings maintained in memory. This means the following features are either unsupportable, or must operate differently.

- Artifact Bindings
    - Unsupported.
    - Artifact mappings are managed in memory and have to be available across nodes because they can be created on one node (via front channel) but then dereferenced on another (via back channel). So no artifact-based bindings can be used.
- Single Logout
    - Unsupported.
    - Single logout is not supported by the IdP software at this time, but if and when it becomes available, the feature depends on the ability to dereference a user session to retrieve the SPs bound to the session. This can't be done without permanent session affinity for the lifetime of a session, so in general single logout isn't supportable.
- Replay Detection
    - Per-Node Only.
    - The replay cache is maintained in memory, so only requests sent to the same node can be detected as replayed. This isn't a critical security feature in most cases, and even limited session affinity will often outlast the freshness window anyway.
- Attribute Queries with Transient IDs
    - Requires special configuration.
    - Ordinarily, queries based on transient identifiers rely on in-memory mappings of identifiers to users. This can be fixed either by avoiding queries by pushing attributes via the front channel, or by sharing a symmetric key on each node and switching to an alternative set of resolver plugins, as described below.
- Single Sign-On
    - Requires special configuration and often custom development. This is discussed in more detail below. An example that's in production at Ohio State is listed on the Contributions page.

Single Sign-On

The main function of the IdP session manager is to facilitate single sign-on to multiple services, independent of the underlying authentication mechanism (s). This functionality has to be replaced with an alternate technology if the sessions cannot be replicated across nodes. One option is permanent session affinity, but this can cause performance problems and limits the ability of a cluster to provide fail-over without losing SSO support.

Two viable alternatives are to leverage an external SSO system, such as pubcookie, cosign, CAS, etc., or to implement a custom login handler within the IdP that provides for SSO via a cookie. The former presumes that you have a clustering solution for the external SSO system, and is most appropriate for sites with an existing solution in place before deploying the IdP.

The latter is not a great deal of work because most of the necessary functionality is available by combining the UsernamePassword login handler code with a DataSealer component to produce an encrypted cookie containing the user's identity, login time and method, and an expiration. The cookie is MAC'd to prevent forgery.

The custom login handler simply produces a cookie after successful authentication and recovers a user's session on other nodes by decoding the cookie and validating its content. Additional logic is needed to properly handle SAML 2 features like IsPassive and ForceAuthn, of course.

An example that's fairly full-featured is in use at Ohio State and is noted on the Contributions page.

Attribute Queries

Some deployers may be content to simply disable back-channel query support, but if you prefer to maintain support for this, simply convert to the Resolver CryptoTransientIDAttributeDefinition and CryptoTransientPrincipalConnector plugins within the resolver. You will need to configure a DataSealer component with a common key on each node.

There's a bug, now fixed, that causes problems with newer SPs that enforce strict subject matching on queries. The bug is corrected in versions of the IdP since 2.3.1.