MidPoint Integrations: Partner Series

Authenticating into midPoint with SSO

Agenda

- Unicon - who we are and what we do
- Single Sign-On Introduction
- SAML2 introduction
- MidPoint Flexible Authentication
- Implementation of SSO for midPoint
- Emergency Login to midPoint
- Example configurations and resources

UNICON

Unicon - Who we are and what we do

- Professional Services company with over 30 years working with educational technology.
- Gold Partner with Evolveum.
- Numerous engagements in implementing and supporting midPoint installations for a variety of educational institutions.

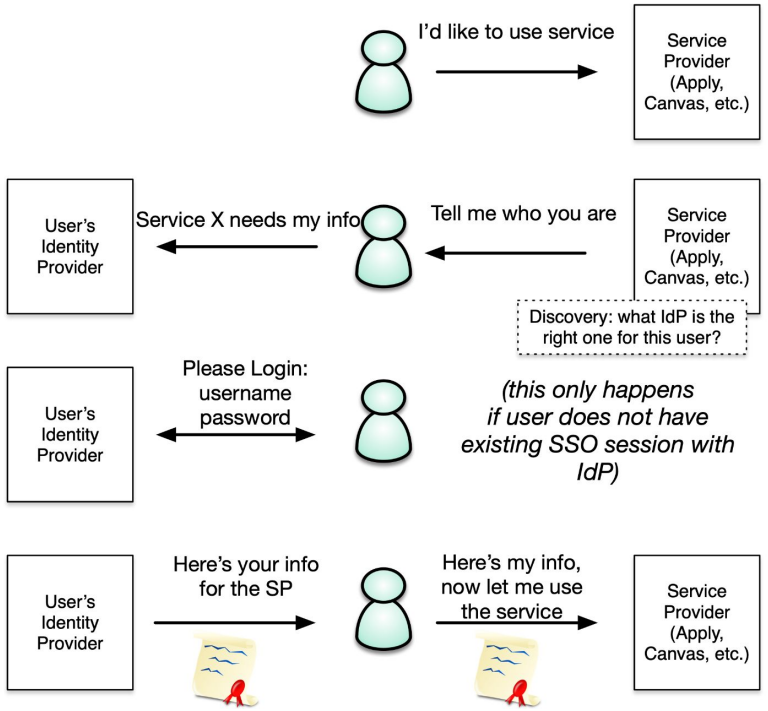
We improve the experience of learning through the use of technology because education lifts all.

UNICON

Introduction to SSO

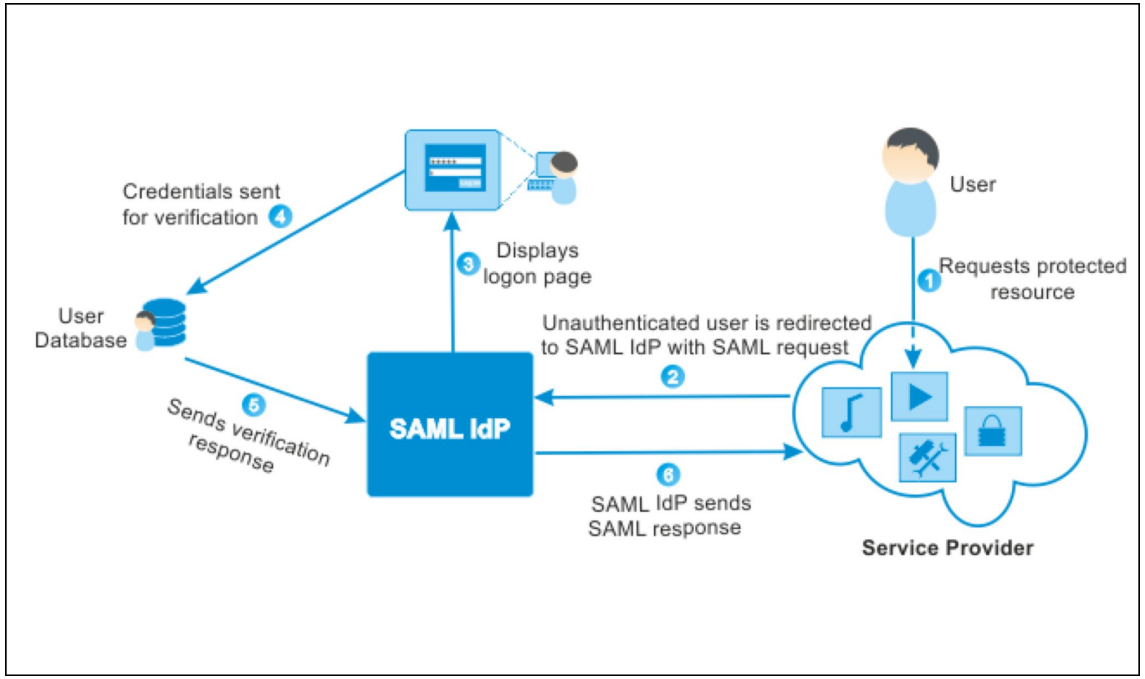
"Login once, access everything"

- Single Sign-On (SSO) is an authentication process that allows users to access multiple applications or systems with one set of login credentials.
- A user logs in to an identity provider (IdP).
- The IdP authenticates the user and issues a token or session.
- The user accesses multiple applications (service providers) that trust the IdP.
- Applications validate the token without requiring additional logins.
- SSO Common Protocols - SAML, OAuth 2.0, OIDC
- Simplifies Identity Management
- Enhances security posture
- Ensures compliance with industry standards



SAML2 Introduction

- Security Assertion Markup Language (SAML) 2.0 is an XML-based open standard used to securely exchange authentication and authorization data between parties.
- Enables Single Sign-On (SSO) by transferring identity information between an Identity Provider (IdP) and a Service Provider (SP).
- SAML Assertion - token containing authentication, attributes, and authorization information.
- Provides strong encryption and digital signatures for data integrity.
- Works across multiple platforms and applications.
- Common use cases include Enterprise SSO and Identity Federation.



UNICON

midPoint Flexible Authentication

Allows for midPoint authentication to be configured in a variety of ways

- midPoint is designed to handle **client-side** authentication use cases.
- midPoint is not an IdP, nor an authentication server and makes use of authentication modules that can be combined into sequences to fit the desired outcome.
- Traditional password based authentication using a directory server or the midPoint identity repository.
- midPoint can be a service provider (client) of an access management system that is using standard SSO protocols.
- midPoint can use SAML as the primary GUI authentication mechanism, and have an emergency login option for an administrator using password based authentication.

UNICON

Implementation of SSO in midPoint

XML is fun!

- Demo SSO Login
- Brief overview of SAML2 XML via SAML2 Tracer
- System configuration
  - Setting global security policy
  - Setting public url
- Security policy SSO overview and explanation of saml2 module
- Building a SAML2 Service Provider (SP) module for midPoint Flexible Authentication
  - Key generation
  - IdP Metadata
  - SP Metadata generation by midPoint
    - http://MIDPOINT_URL/midpoint/auth/default/SAML2_MODULE_ID/metadata/SERVICE_PROVIDER_ALIAS
    - For container deployments http may not be available, exec into the container and use the following midPoint location and port:
      - http://0.0.0.0:8080/midpoint/auth/default/SAML2_MODULE_IDENTIFIER/metadata/SERVICE_PROVIDER_ALIAS
  - SP Metadata manual creation
  - Combining and configuring midPoint security policy

Implementation of SSO in midPoint Continued

Always have a secure backup plan to avoid being locked out

- Demo Emergency Login
- Discussion of methods to restore security policy
- System configuration logging for troubleshooting
  - com.evolveum.midpoint.authentication
  - org.springframework.security.saml2
  - org.springframework.security.oauth2
- HTTP Header Token
  - Can be used by to proxy SSO to midPoint with a non-supported protocol or convenience in certain deployments

Generic SAML2 SSO Service Provider Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor entityID="***ENTIY_ID***" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" >
   <md:SPSSODescriptor protocolSupportEnumeration ="urn:oasis:names:tc:SAML:2.0:protocol" >
       <md:KeyDescriptor use="signing">
           <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#" >
               <ds:X509Data>
                   <ds:X509Certificate>***SIGNING_KEY***</ds:X509Certificate>
               </ds:X509Data>
           </ds:KeyInfo>
       </md:KeyDescriptor>
       <md:KeyDescriptor use="encryption">
           <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#" >
               <ds:X509Data>
                   <ds:X509Certificate>***ENCRYPTION_KEY***</ds:X509Certificate>
               </ds:X509Data>
           </ds:KeyInfo>
       </md:KeyDescriptor>
       <md:AssertionConsumerService
               Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
               Location="***ACS_URL***" index="1"/>
   </md:SPSSODescriptor>
</md:EntityDescriptor>
```

UNICON

Example SAML2 Flexible Authentication Module Configuration

```
<saml2>
   <identifier>saml2sso</identifier>
   <description>My internal enterprise SAML-based SSO system. </description>
   <serviceProvider>
       <entityId>https://mymidpoint.local </entityId>
       <alias>idmsp</alias>
       <signRequests> false</signRequests>
       <keys>
           <activeSimpleKey>
               <privateKey>
                   <t:clearValue>-----BEGIN PRIVATE KEY-----KEY_HERE-----END PRIVATE KEY----- </t:clearValue>
               </privateKey>
               <passphrase>
                   <t:clearValue>password_here</t:clearValue>
               </passphrase>
               <!-- Public key must be UTF-8 base64 encoded including begin and end certificate portions -->
               <certificate>
                   <t:clearValue>base_64_encoded_public_key_here </t:clearValue>
               </certificate>
           </activeSimpleKey>
       </keys>
       <identityProvider>
           <entityId>https://sso.example.com </entityId>
           <metadata>
               <pathToFile>/opt/midpoint/var/metadata/idp-metadata.xml </pathToFile>
           </metadata>
           <linkText> My SAML2 SSO</linkText>
           <authenticationRequestBinding> urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST </authenticationRequestBinding>
            <!-- Maps attribute sent by SSO server in the SAML2 response/assertion XML to the midPoint User name attribute!
-->
           <nameOfUsernameAttribute> uid</nameOfUsernameAttribute>
       </identityProvider>
   </serviceProvider>
</saml2>
```

Example OIDC Flexible Authentication Module Configuration

```
<oidc>
   <identifier>oidcsso</identifier>
   <client>
       <registrationId>mymidpoint</registrationId>
       <clientId>mymidpoint</clientId>
       <clientSecret>
           <clearValue>client_secret</clearValue> <!-- Create a sufficiently good secret following OIDC/OAuth guidelines -->
       </clientSecret>
       <clientAuthenticationMethod> privateKeyJwt</clientAuthenticationMethod >
       <nameOfUsernameAttribute> sub</nameOfUsernameAttribute>
       <openIdProvider>
           <issuerUri>https://sso.example.com </issuerUri>
           <!-- If the OIDC well known endpoint doesn't work, the following are required! -->
           <!-- Note: These follow the Shibboleth IdP OIDC locations, you will need to change for other OIDC Claim Providers -->
           <authorizationUri>https://sso.example.com/idp/profile/oidc/authorize </authorizationUri>
           <tokenUri>https://sso.example.com/idp/profile/oidc/token </tokenUri>
           <userInfoUri>https://sso.example.com/idp/profile/oidc/userinfo </userInfoUri>
           <jwkSetUri>https://sso.example.com/idp/profile/oidc/keyset </jwkSetUri>
       </openIdProvider>
       <!-- The key and keystore must be created beforehand and exist otherwise module will not load!!! -->
       <keyStoreProofKey>
           <keyStorePath>/opt/midpoint/var/oidcssokeystore.jks </keyStorePath>
           <keyStorePassword>
               <clearValue>password</clearValue>
           </keyStorePassword>
           <keyAlias>midpoint</keyAlias>
           <keyPassword>
               <clearValue>password</clearValue>
           </keyPassword>
       </keyStoreProofKey>
   </client>
</oidc>
```

Example Administrator GUI Emergency Login Sequence

```
<sequence>
   <!-- Admin GUI Emergency Login URL: https://your_midpoint_url/midpoint/auth/emergency
         Note this will not work if you break config in *any* of the security policy modules!
          Use REST API/midPoint Studio or Ninja to upload a working security policy XML object in that case.
           If those don't work, use the database! -->
   <identifier>admin-gui-emergency</identifier>
   <description>Allows only users with role Superuser to Login via https://your_midpoint_url/midpoint/auth/emergency URL.
        Useful if SSO is broken. Uses default login form, passwords for the admin(s) must be set in midPoint.  </description>
   <channel>
       <channelId>http://midpoint.evolveum.com/xml/ns/public/common/channels-3#user </channelId>
       <default>false</default>
       <urlSuffix>emergency</urlSuffix>
   </channel>
   <requireAssignmentTarget oid="00000000-0000-0000-0000-000000000004" relation="org:default" type="c:RoleType">
       <!-- Superuser -->
   </requireAssignmentTarget>
   <module>
       <identifier>internalLoginForm</identifier>
       <order>30</order>
       <necessity>sufficient</necessity>
   </module>
</sequence>
```

UNICON

Links and Documentation

- Flexible Authentication Basic Concepts
- Flexible Authentication Configuration
- Evolveum Samples: Security Policy with SAML2 SSO
- SAML2 Key and Key Generation Guide

UNICON

Thank you for your attention

https://www.unicon.net

midPoint

MidPoint Integrations: Partner Series

2025