InCommon

Integrating Multi-Factor Authentication into Your Campus Identity Management System

Mike Grady, Unicon
David Walker, Internet2
(both associated with the Internet2 Scalable Privacy Project)

InCommon

Agenda

- Multi-Context Broker (MCB) Model
- MCB Model & Shibboleth IdP v3
- CAS MFA
- Some Broader MFA Operational Topics

The Multi-Context Broker (MCB) Model

Definition: Authentication Context

- What an IdP tells an SP about an authentication event and what led up to it
  - May include identity proofing, registration, Authentication Method, operational practices, *etc*.
- Establish common standards, potentially federation-wide
- A Context can *satisfy* another Context
- Examples
  - A specific type of Authentication Method (*e.g.*, Password, X.509, or MFA)
  - An assurance profile (*e.g.*, InCommon Bronze or Silver

More Definitions

- **Authentication Method.** A software module that authenticates a user using a specific authentication service
- **User Certification.** An Authentication Context that may be asserted for a particular user

What a Multi-Context Broker Does

1. SP requests an Authentication Context
2. IdP/MCB considers...
   a. the SP's request
   b. the user's certifications (contexts that can be asserted for this user)
   c. other contexts that can satisfy the requested context
3. IdP selects a context that satisfies all criteria
4. IdP invokes an authentication method, if needed
5. IdP asserts the requested context (or returns an error)

The MCB Model and the Shibboleth IdP v3

Concepts

- Authentication Context = type of Principal
- Authentication Method = Authentication Flow
- User Certifications
  - Multi-valued attribute containing the Authentication Contexts that can be asserted for each user
  - Retrieved from IdMS using the Shibboleth attribute resolver
- Authentication Contexts can be configured to satisfy the requirements of other Contexts

Configuration - Authentication Contexts and Methods

conf/authn/general-authn.xml

```
<util:list id="shibboleth.AvailableAuthenticationFlows">
    <bean id="authn/Password" parent="shibboleth.AuthenticationFlow">
        <property name="supportedPrincipals">
            <util:list>
                <bean parent="shibboleth.SAML2AuthnContextClassRef"
                    c:classRef="urn:oasis:names:tc:SAML:2.0:ac:classes:Password" />
            </util:list>
        </property>
    </bean>
    <!-- Additional Flows...  -->
</util:list>
```

- authn/Password is a Spring Web Flow, defined separately

Configuration - User Certifications

conf/idp.properties

Set to an attribute ID to resolve prior to selecting authentication flows;

its values are used to filter the flows to allow.

idp.authn.resolveAttribute = **assurance**

Configuration - Authentication Context Hierarchy - 1

conf/authn/authn-comparison.xml

```
<bean id="shibboleth.BetterClassRefMatchFactory" parent="shibboleth.
InexactMatchFactory">
    <property name="matchingRules">
        <map>
            <entry key="http://id.incommon.org/assurance/bronze">
                <list>
                    <value>http://id.incommon.org/assurance/silver</value>
                </list>
            </entry>
        </map>
    </property>
</bean>
```

- Silver satisfies the requirements of Bronze

Configuration - Authentication Context Hierarchy - 2

conf/authn/authn-comparison.xml

```
<!-- Registry of matching rules. -->
<util:map id="shibboleth.AuthnComparisonRules">
   ...
<!-- Exact matching. -->
    <entry key-ref="shibboleth.SAMLAuthnMethodExact"
                value-ref="shibboleth.BetterClassRefMatchFactory"/>
    <entry key-ref="shibboleth.SAMLACClassRefExact"
                value-ref="shibboleth.BetterClassRefMatchFactory"/>
    <entry key-ref="shibboleth.SAMLACDeclRefExact"
                value-ref="shibboleth.BetterClassRefMatchFactory"/>
   ...
</util:map>
```

- Warning: Contexts in the SAML standard do not satisfy one another.

InCommon

Configuration - Initial Authentication Context

conf/idp.properties

Regular expression of forced "initial" methods when no session exists,

usually in conjunction with the idp.authn.resolveAttribute property below.

idp.authn.flows.initial = **authn/Password**

- **Used to ensure the current user is known prior to needing user certifications, for second-factor-only authentication like Duo, or simply to require baseline authentication before further interaction with the user.**
- **Currently looking at other, more flexible, ways to handle this.**

CAS MFA

- Enterprise scope, rather than federation
- Extension for CAS 3.5.x and 3.6.x
  - Not yet for CAS 4.x, likely to eventually be “built-in”
- Orchestrates authentication methods, rather than contexts
- Currently support for Duo, Yubikey, Toopher, Authy, Radius, “strong-two-factor (Custom)”

CAS MFA

- SPs can be configured at IdP through JSON Service Registry (e.g “relying party config”)

```
{
    "services":[
        {
            "id":1,
            "serviceId":"^(https?|imaps?)://.*",
            "name":"HTTP/HTTPS service",
            "description":"Test HTTP/HTTPS services",
            "extraAttributes": {
                "authn_method": "strong_two_factor",
                "mfa_role": {
                    "mfa_attribute_name": "memberOf",
                    "mfa_attribute_pattern": "CN=StudentsOfConcern,OU=People,DC=e
                }
            }
        }
    ]
}
```

CAS MFA

- SP requests are possible through extensions to the CAS protocol (akin to SAML requested authn context)
  - added query argument to login redirect
  - e.g. https://server/cas/login?service=http%3A%2F%2Fwww.service.com&authn_method=strong_two_factor

InCommon

CAS MFA

- Can also be configured based on User attribute and/or “roles” of the user

```
<bean id="principalAttributeMfaRequestResolver"
      class="net.unicon.cas.mfa.authentication.principal.PrincipalAttributeMultiFa
      c:mfaMethodAttributeName="${mfa.method.userAttribute:authn_method}"
      c:mfaServiceFactory-ref="mfaServiceFactory"
      c:mfaRankingConfig-ref="supportedAuthenticationMethodsConfig"
      p:authenticationMethodTranslator-ref="regexAttributeAuthnMethodTranslator"/>

<bean id="regexAttributeAuthnMethodTranslator"
      class="net.unicon.cas.mfa.authentication.RegexAuthenticationMethodTranslator
      c:translationMap-ref="regexMap" />

<util:map id="regexMap" map-class="java.util.LinkedHashMap">
      <entry key="CN=StrongTwoFactor,OU=People,DC=example,DC=org" value="strong-tw
      <entry key="CN=DuoTwoFactor,OU=People,DC=example,DC=org" value="duo-two-fact
</util:map>
```

CAS MFA

- Authentication methods are ranked according to “strength”
  - A stronger method can satisfy a weaker method
  - Use “strongest” required, taking all “signals” into account

```
[ {
  "rank" : 1,
  "name" : "duo-two-factor"
}, {
  "rank" : 2,
  "name" : "strong-two-factor"
}, {
  "rank" : 3,
  "name" : "yubikey-two-factor"
}, {
  "rank" : 4,
  "name" : "radius-two-factor"
}, {
  "rank" : 5,
  "name" : "toopher-two-factor"
}, {
   "rank" : 6,
   "name" : "authy-two-factor"
} ]
```

CAS MFA

- Methods satisfied tracked in SSO session
- Decide the “winning authentication method”; one that is held and one that is requested.
- Decision takes into account method ranking
- CAS-using applications can indicate authentication method requirement to CAS server on their ticket validation request
  - https://cas.example.org/cas/serviceValidate?service=http%3A%2F%2Fwww.example.org%2Fservice&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7&authn_method=strong_tw

CAS MFA

- Method satisfied returned in serviceValidate or proxyValidate response
- e.g.

```
<cas:serviceResponse xmlns:cas="http://www.yale.edu/tp/cas">
 <cas:authenticationSuccess>
  <cas:user>username</cas:user>
  <cas:proxyGrantingTicket>PGTIOU-84678-8a9d...</cas:proxyGrantingTicket>
  <cas:authn_method>strong_two_factor</cas:authn_method>
 </cas:authenticationSuccess>
</cas:serviceResponse>
```

Broader MFA Operational Topics

- User enrollment / management interface
  - Open source options -- U Chicago, Penn, others?
  - Capture info locally, hard tokens, general UI
- Behavior on ForceAuthn
- Is it 2FA if you allow trusted devices, trusted networks, etc.?
- Fail open or Fail closed?
- Define some additional authentication contexts that are used by the community?

Broader MFA Operational Topics

- MFA Interoperability Profile Working Group
  - define a MFA profile to allow our community and SPs to rely on a standard syntax and semantics regarding MFA
  - https://spaces.internet2.edu/x/CY5HBQ

References

- *The Multi-Context Broker Model*
  - https://spaces.internet2.edu/x/kY5HBQ
- *Configuring the Multi-Context Broker Model in Shibboleth IdPv3*
  - https://wiki.shibboleth.net/confluence/x/sIA9AQ
- David Langenberg's *Replicating Multi-Context Broker Functionality (Duo + Username/Password with user-opt-in forcing Duo)*
  - https://wiki.shibboleth.net/confluence/x/lYA9AQ

References

- *CAS MFA Extension*
  - https://github.com/Unicon/cas-mfa
- *CAS MFA Wiki (Documentation)*
  - https://github.com/Unicon/cas-mfa/wiki
- *MFA Interoperability Profile Working Group*
  - https://spaces.internet2.edu/x/CY5HBQ
- Some enrollment/mgmt interfaces for Duo
  - U Chicago: https://github.com/uchicago/duo-registration
  - Open Two Factor: https://wiki.jasig.org/display/~mchyzer/Open+Two+Factor+duo+integration