Multi-Factor Authentication and Shibboleth IdPv3

New flexibility, same old questions

Etienne Dysli-Metref
etienne.dysli-metref@switch.ch

Multi-factor authentication

Authenticate with factors picked from two (or more) of the following categories.

Something you:

- know (password)
- have (token)
- are (biometrics)

Do you want MFA?

- Does your SP ask for something better than username+password?
- Do you already have a multi-factor authentication solution deployed (without Shibboleth)?

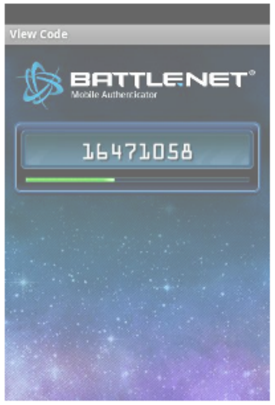
Swiss edu-ID "Processes" WG, December 2014

Many institutions wish they had MFA but no one really knows how to introduce or implement it.

Problem landscape

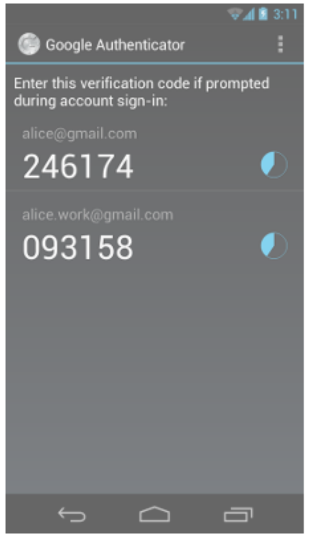
- What do you want to gain from MFA? What are the risks and expected quality levels?
- Who is going to use it?
- How much are you willing to pay? What hardware can you rely on? (mobile/smart phones)
- Is it only for the IdP or should it work with other systems too?

So many tokens!



Categories

- hardware | software
- event-based | time-based | challenge-response | out of band OTP
- standard | proprietary



Enter this verification code if prompted during account sign-in:

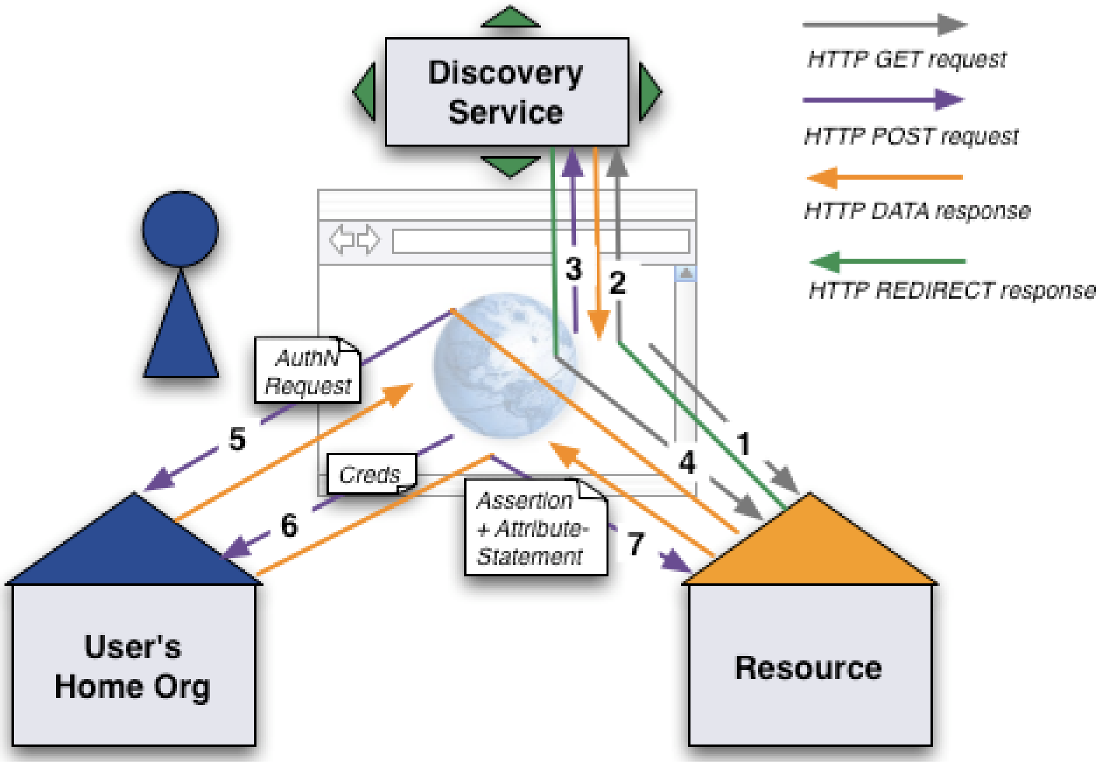
alice@gmail.com
246174

alice.work@gmail.com
093158

Standards

- OATH: HOTP, TOTP, OCRA
- X.509 certificate, smartcards

Implementation in Shibboleth



(https://www.switch.ch/aai/demo/medium/)

MFA happens in steps 5, 6 and 7

Implementation in Shibboleth

1. SP requests a specific *authentication context class*
2. IdP selects and runs a *login flow* that satisfies this class
3. IdP replies with an authentication assertion containing the class actually used

Authentication context classes

Login flows

Assembling the pieces

SAML authentication contexts

- OASIS standard (http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf) (2005)
- XML Schema to describe the authentication context
  - identification, authentication method
  - technical protection, operational protection
  - governing agreements
- Provides a list of authentication context *classes* for SAML (namespace `urn:oasis:names:tc:SAML:2.0:ac:classes`)

SAML authentication context classes

Which class to use?

- Completely define your own
  ⇒ not interoperable with other institutions
- Pick one from the OASIS list that fits your needs
  ⇒ better for interoperability
- Or one from IETF's Level of Assurance profiles registry (https://www.ietf.org/assignments/loa-profiles/)
  ⇒ for example InCommon Bronze or Silver (https://incommon.org/assurance/)

SAML authentication context classes

List of classes from the OASIS standard

Internet Protocol, Internet Protocol Password, Kerberos, Mobile One Factor Unregistered, Mobile Two Factor Unregistered, Mobile One Factor Contract, Mobile Two Factor Contract, Password, **Password Protected Transport**, Previous Session, Public Key - X.509, Public Key - PGP, Public Key - SPKI, Public Key - XML Digital Signature, Smartcard, Smartcard PKI, Software PKI, Telephony, Telephony (“Nomadic”), Telephony (Personalized), Telephony (Authenticated), Secure Remote Password, **SSL/TLS Certificate-Based Client Authentication**, Time Sync Token, **Unspecified**

Authentication context classes

Login flows

Assembling the pieces

Login flows

New technology in IdPv3

- The IdPv3 uses Spring Web Flow (http://projects.spring.io/spring-webflow/) to implement various authentication methods as login flows
- Child project of the Spring Framework
- Allows implementing the “flows” of a web application
- Sequence of steps for user interaction e.g. forms
- State machine described in XML

```xml
<!--  Examples extracted from system/flows/authn/authn-flow.xml -->
<flow xmlns="http://www.springframework.org/schema/webflow"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/webflow
  http://www.springframework.org/schema/webflow/spring-webflow.xsd"
  parent="authn.abstract">
  <action-state id="AuthenticationSetup">
    <evaluate expression="PopulateAuthenticationContext"/>
    <evaluate expression="PopulateSessionContext"/>
    <evaluate expression="'proceed'"/>
    <transition on="proceed" to="TestForSession"/>
  </action-state>
  <decision-state id="TestForSession">
  <if test="opensamlProfileRequestContext.getSubcontext(...) != null"
      then="SessionExists" else="FilterFlows"/>
  </decision-state>
  <!-- ...more states... -->
  <subflow-state id="CallAuthenticationFlow" subflow="#{currentEvent.id
    <input name="calledAsSubflow" value="true"/>
    <transition on="proceed" to="CallSubjectCanonicalization"/>
    <transition on="ReselectFlow" to="SelectAuthenticationFlow"/>
  </subflow-state>
  <!-- ...more states... -->
  <bean-import resource="authn-beans.xml"/>
</flow>
```

IdPv3 login flows

Tools you get out of the box

- Built-in login flows for:
  - Password
  - X.509 client certificate
  - IP address
- Each login flow states what authentication context classes it supports
  (configured in `conf/authn/general-authn.xml`)

IdPv3 login flows

Tools you get out of the box

- A flow may call another flow as *subflow*
- Optional *initial flow*: always runs first

  ⇒ fetch user attributes before another flow runs
  - when there is no session
  - regardless of what the SP requests
- **But** there is no generic way of chaining flows

Login flows and MFA

Combining flows (1): the big one

Write one flow to implement both first and second factors

```
Please enter your credentials
Username: [______]
Password: [______]
OTP:      [______]
```

- Completely customised to your needs
  Example: if OTP field left empty then send OTP via SMS and reprompt
- Likely to duplicate most of the password flow

Login flows and MFA

Combining flows (2): the initial glue

Write one flow for the second factor and “glue” it after an existing first factor *initial* flow

```
Please enter your credentials
Username: [_____]
Password: [_____]
```

```
Please enter your one-time password
OTP:      [_____]
```

- Can easily replace one flow with another
- SFA + SFA =? MFA

Login flows and MFA

Combining flows (3): the subflow explosion

Offer the user a choice of second factors after an *initial* password flow

```
Please enter your credentials
Username: [_____]
Password: [_____]
```

```
Please pick your authentication method
token1 | token2 | token3
token4 | token5 | token6
```

```
Please enter your one-time password
tokenX:   [_____]
```

Authentication context classes

Login flows

Assembling the pieces

Migrating version 2 extensions

- Unfortunately, IdPv2 login handlers can't be reused “as is” in v3
- New name: *login handler* → *login flow*
- Code changes are needed because the API is different
- Reimplement them as flows if possible
  ⇒ much more flexible than servlets
- Like in v2, you need someone with Spring skills

Pieces to configure

- Choose one authentication context class
- SP must request authentication with that particular class
- IdP must have a login flow for that class, enabled
- Implement that flow
- Have something to verify each authentication factor
  - inside the IdP process or external system?

Non-technical pieces

Administrative processes

- Registration of new users and distributing tokens (enrolment)
  - identity verification?
- Replacement of forgotten, lost, stolen or expired tokens
- Revocation

Summary

- Planning and deploying MFA is still as difficult as before (same old questions)
- IdPv3 offers greater flexibility thanks to flows

References

- OATH: Initiative for Open Authentication (http://openauthentication.org/)
- OASIS: Authentication Context for SAML2 (http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf)
- IETF: Level of Assurance Profiles registry (https://www.ietf.org/assignments/loa-profiles/)
- IETF: RFC 6711: An IANA Registry for Level of Assurance (LoA) Profiles (https://tools.ietf.org/html/rfc6711)

References

- InCommon Assurance Program (https://incommon.org/assurance/)
- Spring Web Flow project (http://projects.spring.io/spring-webflow/)
- Shibboleth wiki: IdPv3 Authentication Configuration (https://wiki.shibboleth.net/confluence/display/IDP30/AuthenticationConfig)