# How SSO Works

Richard Frovarp

# What is SSO?

- SSO most properly stands for Single Sign On
    - Can also stand for Same Sign On
- Single Sign On allows an individual to login once, and access multiple services
- Technology can be used to simplify or outsource authentication without requirement for multiple services or organization

# Single Log Out

- SLO
- Logging out of one service logs one out of all other services
- Fairly rare
- Can have negative consequences
  - Automatic timeout causing the local logout triggering SLO

# Components

Terminology varies depending on protocol used

- IdP (Identity Provider), OP (OIDC Provider)
- SP (Service Provider), RP (Relying Party)
- Browser

May be part of the same org, or a different organization

# What is an IdP/OP?

- An Identity Provider is a system that provides authentication operations.
- Many systems can be thought of as an IdP, but the idea only starts to make sense when the IdP is separate from the SP
- Google, GitHub, and other social providers
- CAS at a university
- Shibboleth / CAF / eduGAIN at a university
- RADIUS server to eduroam via CANARIE
- Okta, Entra, and others

# What is an SP/RP?

- A Service Provider is the thing that the user wants to use
  - eduroam at another institution
  - Zoom, DocuSign, Qualtrics
  - ORCID (which can also be an IdP)
  - Gmail
  - YouTube
  - YouTube Apple TV app
  - YouTube Android App

# Why use IdPs and SPs?

- The most common reason in industry is to separate login / account logic from your services
    - Netflix can have one team handle the login code, password resets, etc
    - The the rest of the teams at Netflix can not do that work
    - This is why we use IdPs internally
        - It's easy to write login code, but it quickly becomes extra work and boring work. So just let well developed SSO libraries handle it
- Federated access is very common in higher education: eduroam, CAF, eduGAIN, etc.
    - Also somewhat popular in industry as well with the "Log in with (Google|Facebook|GitHub)" buttons
- Can put a lot of security policy in one system instead of multiple different ones
    - MFA in Google's IdP rather than in each service and client
- Consistent login experience for users

# Federated Auth Types

- Bilateral agreements
  - Two different parties agree to perform federated authentication, potentially through a contract
- Multilateral federation
  - Multiple different parties make federated authentication available to each other. This requires trust.
  - InCommon baseline expectations
  - REFEDS baseline expectations
  - REFEDS MFA profile
  - REFEDS Assurance Profile
  - REFEDS SIRTFI
  - NIH is using all of it to drive their processes and trust

# General web flow

- User hits SP (YouTube, Drive, Gmail, Keep, Google Analytics,...)
- SP redirects user over to IdP (Google's or the enterprise's)
    - In the case of going to the enterprise's, you start having proxies
- User auths to IdP
- IdP directs user's browser to send info to SP
- SP performs some validation
- User is allowed into system

# CAS (Central Authentication Service) protocol

- Three versions: 1, 2, 3
- Can be done by CAS IdP, Shibboleth IdP, simpleSAMLPHP IdP
  - plus CAS client libraries on SP side
- 1.0 is old and basically unused
- 2.0 had a nearly ubiquitous unofficial extension to do attribute return
- 3.0 adopted that extension
  - To get the attributes use the 3.0 validation URL

Also SAML 1.1 is a thing (Security Assertion Markup Language)

# CAS Login Flow

- SP redirects user to IdP with service as parameter in URL
- IdP validates service is registered
- User logs in
    - Potential for attribute release consent
- TGT (Ticket Granting Ticket) is stored in cookie
- IdP redirects user to SP with a ST (service ticket) in URL
- SP makes backchannel call to IdP with its service and the ST to the validate URL of the IdP
- IdP validates that ST belongs to that service and hasn't been used yet
- IdP releases attributes to SP
- SP establishes its own session with user

# Back channel

- Server to server communication
- Requires IdP to be reachable by SP
- Simplifies payload sent via browser

# CAS Attribute Return

- Standard ones
  - sn, givenName, email, entitlements, etc
  - Identity translation for username
- Authentication attributes
  - SSO login time
  - MFA method

# CAS Login Requests

- Standard method as described above
    - If SSO session exist, it will be used
- Renew
    - Force a new authentication
        - What does this mean with respect to MFA???
- Gateway
    - Use existing SSO if it exist
    - If it doesn't, send user back to app without logging in

# SAML 2.0

- Almost always what people mean when they just use SAML
- Most popular open source project is Shibboleth
    - Shibboleth has historical authentication use
- Can be done in the Shibboleth IdP, CAS IdP, simpleSAMLPHP IdP, and others
    - Feature support varies by IdP
- Shibboleth SP / shibd
    - daemon + integration modules for ASF HTTPD, NGINX, IIS
- A variety of WAYF (Where Are You From) methods in community
- A LOT of extensions in higher education
- Unique security challenges
    - Old from 2005 - Arab Spring 2010, Snowden 2013
    - Front channel

# SAML 2 Login Flow

- SP redirects user to IdP. Redirect includes SAML payload
  - Contains entityID, Assertion Consumer Service, optionally encrypted and signed
- IdP lookups SP metadata
- User authenticates
  - Optional attribute release consent
- SSO session cookie created
- SAML payload created
  - Optionally encrypted using SP's public key from metadata
  - SHOULD be signed using IdP's private key
- IdP redirects browser to POST payload to SP's ACS URL
- SP SHOULD validate signed payload with IdP's public key from metadata
- SP starts its own session with user
  - Shibboleth SP sends values through HTTP headers to app for app to use that way

# ACS Validation Failure

- AssertionConsumerService URLs are where SAML POSTs to.
- ACS is set by "SP"
- If sent ACS is malicious, attacker can get SAML payload
- IdP needs to ensure that provided ACS URL matches a ACS URL in metadata

# Burp Suite - SAML Raider

```xml
<saml:Subject>
    <saml:NameID SPNameQualifier="sitecore:sp"
                 Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
                 >richard.frovarp@ndsu.edu</saml:NameID>
    <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData NotOnOrAfter="2023-03-22T21:04:39Z"
                                      Recipient="https://members.educause.edu/Login.ashx"
                                      />
    </saml:SubjectConfirmation>
</saml:Subject>
```

# Front channel communication

- Note that the browser is transporting attributes from IdP to SP
- If browser is doing it, it can be seen and messed with
- IdP encrypting response with SP's public key a bigger deal before TLS everywhere
- Not validating signature is still a problem

# Metadata

- Contains entityID
  - This is JUST a string, but should be in YOUR namespace and something you control
- X509 certs
  - JUST a method of transporting public keys.
  - PLEASE do NOT make these TLS certs that go back to a public CA. Self sign with long lived certs
- Metadata extensions
  - Security contact information, entity categories, support contacts, error URL, requested attributes, privacy policy URLs, logo URLs

# SAML features (higher education enhancements)

- LARGE scale federation
  - One SP has thousands of IdPs available to use
- Authentication Context Request
- Force Authentication
  - What does this mean for MFA? See REFEDS MFA 1.2
- Entity Categories
  - Anonymous, Pseudonymous, Personalized (nonymous)

# Golden SAML

- For SAML 2.0
- If you control the IdP private key, you can issue any payload you want
- Since there is nothing going to the IdP in this attack, it is impossible to detect at the IdP

# IdPs in general

- Can say whatever they want to the SP
- SP puts a lot of trust in the IdP
- Releases a lot of attributes
  - Whether or not MFA happened
  - Unique IDs
  - roles
  - usernames
  - if from fresh login
  - when session began
  - etc
- IdP can lie

# IdP Capabilities

- Attribute rename
  - Can send out the attribute in any name to accommodate a variety of systems
- Attribute replacement
  - Can replace the username attribute with something else. Perhaps something less mutable
- Enforce casing
  - username will always be lower case
- Attribute modification
  - Different scoping suffix?
  - Basically can apply a Groovy script in CAS to attribute
- Attribute filtering
  - Limit release of entitlement names
- Authorization filtering
  - Can limit authentication to a SP based on user's attributes

# IdP attacks

- If system containing IdP is compromised and actor can examine memory on the system or in the IdP process, they can grab passwords out of memory
- Can use something like an unsecured / poorly secured Apache Tomcat Manager attack to be able to run as the Tomcat process
- HAR files in support systems

# MFA step up attacks

- If you have SSO enabled and don't have ubiquitous MFA, there can be configuration challenges in enforcing an MFA step up on services that require MFA.
- We just enforce MFA across the board, so we don't have this configuration issue.

# MFA assertion from IdP

- Depending on MFA configuration, there exists the possibility that the IdP wrongly asserts MFA happened when it hasn't.
- (Legacy?) Duo web integration that is configured to allow people that aren't enrolled in Duo to bypass MFA actually looks like a successful MFA via the widget, so the IdP has no idea. We make it so that if the Duo widget is displayed, you must do MFA. If you aren't configured for MFA, tough luck.

# Others

- Ad-hoc - can be scary
  - Just please don't
- WS-FED - kind of a pain and really only supported by Microsoft. Can send the suggested username from SP to IdP.
- OpenID - seems to not really be a thing any more
- OAuth 2.0 - allows the IdP to issue an authorization token to a third party to make API calls.
  - Need to secure the token and the refresh token.
- OpenID Connect (OIDC) - Sits on top of OAuth 2.0. Can send user info
- Good video on OAuth / OIDC flows: https://www.youtube.com/watch?v=8aCyojTIW6U

# Questions?